EXC1081- Open Source Development for Google Applications

2 CREDIT COURSE

SEMESTER END PROJECT

WEATHER FORECAST APP

16BCE0944

AKSHAY ARORA

INDEX

Abstract

A weather forecast app named “Forecast” is created using a given Accuweather api which tells us the next 5 days of forecast of the city of Vellore. The app will display the forecast for the day in one activity, another activity should display the forecast of the upcoming days. The coding is done on java on Android Studio. The accuweather api gives a api key which allows us to extract the temperatures and forecasts. Also various images and libraries are imported for the complete functioning of the app.

Introduction

Weather forecasting is the application of science and technology to predict the conditions of the atmosphere for a given location and time. Human beings have attempted to predict the weather informally for millennia and formally since the 19th century. Weather forecasts are made by collecting quantitative data about the current state of the atmosphere at a given place and using meteorology to project how the atmosphere will change.

Accuweather api has been used in this project for weather forecasting. Nearly 2 billion people worldwide rely on AccuWeather to help them plan their lives, protect their businesses, and get more from their day. AccuWeather provides hour-by-hour and minute-by-minute forecasts with Superior Accuracy™ with customized content and engaging video presentations available through smart phones, tablets, free wired and mobile Internet sites via AccuWeather.com, award winning AccuWeather apps, connected TVs, wearables, smart homes, and connected cars, as well as radio, television, newspapers, and the AccuWeather Network cable channel.

When we create an account with Accuweather, an api key is provided to us. This key is then used as a reference to extract various information based on location, forecast, etc.

This api is then implemented in Android Studio Java Programming using JSON Parsing and each information is extracted into strings which is then referenced within the code. The only problem with this api is that it only allows 50 calls per day. After that no more data is displayed.

The UI is solely made in Android Studio with the help of some images and online tutorials.

Methodology

The app includes two different activities-

The First Activity contains the forecast for today's weather of Vellore which include the Minimum Temperature, Maximum Temperature, Date, Day and Night Description.

A button is then used to transfer from First Activity to the Second Activity.

The Second Activity then includes the list of next 5 days of forecast including High, low and Description of Forecast, along with the dates.

The Code implemented-

First Activity-

```
package com.example.akshay.forecast;

import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

import com.android.volley.Request;
import com.android.volley.RequestQueue;
import com.android.volley.Response;
import com.android.volley.VolleyError;
import com.android.volley.toolbox.JsonObjectRequest;
import com.android.volley.toolbox.Volley;

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

import java.text.DateFormat;
import java.util.Calendar;

public class FrontActivity extends AppCompatActivity {

    public Button button;

    TextView t1_mintmp,t2_max_tmp,t3_desc,t4_desc2;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_front);

        Calendar calendar = Calendar.getInstance();
        String CurrentDate =
DateFormat.getDateInstance(DateFormat.MEDIUM).format(calendar.getTime());
```

```
        TextView textViewDate  = findViewById(R.id.date);
        textViewDate.setText(CurrentDate);

        button = findViewById(R.id.but1);
        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                openMainActivity();
            }
        });

        t1_mintmp = (TextView)findViewById(R.id.mintmp);
        t2_max_tmp = (TextView)findViewById(R.id.maxtmp);
        t3_desc = (TextView)findViewById(R.id.desc);
        t4_desc2=(TextView)findViewById(R.id.desc2);

        find_weather();
    }

    public void openMainActivity() {

        Intent intent = new Intent(this, MainActivity.class);
        startActivity(intent);
    }

    public void find_weather(){

        String url
="http://dataservice.accuweather.com/forecasts/v1/daily/1day/190795?apikey=JUlCIjrc
GoB71Vi1RgGLwwiGPooatana&metric=true";

        JsonObjectRequest jor =new JsonObjectRequest(Request.Method.GET, url, null,
new Response.Listener<JSONObject>() {
            @Override
            public void onResponse(JSONObject response) {
                try{
                    JSONArray array = response.getJSONArray("DailyForecasts");
                    JSONObject resultsObj = array.getJSONObject(0);
                    JSONObject temp = resultsObj.getJSONObject("Temperature");
                    String minTemp =
temp.getJSONObject("Minimum").getString("Value");
                    String maxTemp =
temp.getJSONObject("Maximum").getString("Value");

                    JSONObject day = resultsObj.getJSONObject("Day");
                    String desc = day.getString("IconPhrase");

                    JSONObject night = resultsObj.getJSONObject("Night");
                    String desc2 = night.getString("IconPhrase");

                    t1_mintmp.setText(minTemp);
                    t2_max_tmp.setText(maxTemp);
                    t3_desc.setText(desc);
                    t4_desc2.setText(desc2);

                }catch (JSONException e) {
                    e.printStackTrace();
                }

            }
        }, new Response.ErrorListener() {
            @Override
            public void onErrorResponse(VolleyError error) {

            }
        }
        );
        RequestQueue queue= Volley.newRequestQueue(this);
```

```
        queue.add(jor);

    }

}
```

The second activity uses 4 Java files –

Main File-

```
package com.example.akshay.forecast;

import android.os.AsyncTask;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.widget.ListView;

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

import java.io.IOException;
import java.net.URL;
import java.util.ArrayList;
import java.util.Iterator;

public class MainActivity extends AppCompatActivity {

    private static final String TAG = MainActivity.class.getSimpleName() ;
    private ArrayList<Weather> weatherArrayList = new ArrayList<>();
    private ListView listView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main2);

        listView = findViewById(R.id.idListView);

        URL weatherUrl = NetworkUtils.buildUrlForWeather();
        new FetchWeatherDetails().execute(weatherUrl);
        Log.i(TAG, "onCreate: weatherUrl: " + weatherUrl);
    }


    private class FetchWeatherDetails extends AsyncTask<URL, Void, String>{

        @Override
        protected void onPreExecute() {
            super.onPreExecute();
        }

        @Override
        protected String doInBackground(URL... urls) {

            URL weatherUrl = urls[0];
            String weatherSearchResults = null;

            try{
                weatherSearchResults =
NetworkUtils.getResponseFromHttpUrl(weatherUrl);
            } catch (IOException e) {
                e.printStackTrace();
            }
            Log.i(TAG, "doInBackground: weatherSearchResults: " +
```

```
weatherSearchResults);
            return weatherSearchResults;
        }

        @Override
        protected void onPostExecute(String weatherSearchResults) {
            if(weatherSearchResults != null && !weatherSearchResults.equals("")) {
                weatherArrayList = parseJSON(weatherSearchResults);
                Iterator itr = weatherArrayList.iterator();
                while(itr.hasNext()){
                    Weather weatherInIterator=(Weather) itr.next();
                    Log.i(TAG, "onPostExecute: Date: " +
weatherInIterator.getDate() +
                            "Min: " + weatherInIterator.getMinTemp() +
                            "Max: " + weatherInIterator.getMaxTemp() +
                            "Desc: "+ weatherInIterator.getDesc());
                }
            }
            super.onPostExecute(weatherSearchResults);
        }
    }

    private ArrayList<Weather> parseJSON(String weatherSearchResults) {
        if(weatherArrayList != null){
            weatherArrayList.clear();
        }
        if(weatherSearchResults != null){
            try{

                JSONObject rootObject = new JSONObject(weatherSearchResults);
                JSONArray results = rootObject.getJSONArray("DailyForecasts");

                for (int i=0;i<results.length(); i++){
                    Weather weather = new Weather();

                    JSONObject resultsObj = results.getJSONObject(i);

                    String date = resultsObj.getString("Date");
                    weather.setDate(date);

                    JSONObject temperatureObj =
resultsObj.getJSONObject("Temperature");
                    String minTemperature =
temperatureObj.getJSONObject("Minimum").getString("Value");
                    weather.setMinTemp(minTemperature);

                    String maxTemperature =
temperatureObj.getJSONObject("Maximum").getString("Value");
                    weather.setMaxTemp(maxTemperature);

                    JSONObject descObj = resultsObj.getJSONObject("Day");
                    String description = descObj.getString("IconPhrase");
                    weather.setDesc(description);


                    /*Log.i(TAG, "parseJSON: date: " + date + " " +
                            "Min: " + minTemperature + " " +
                            "Max: " + maxTemperature + " " +
                            "Link: " + link);*/

                    weatherArrayList.add(weather);


                }

                if(weatherArrayList != null){
                    WeatherAdapter weatherAdapter = new
WeatherAdapter(this,weatherArrayList);
```

```
                    listView.setAdapter(weatherAdapter);
                }


                return weatherArrayList;
            } catch (JSONException e) {
                e.printStackTrace();
            }
        }
        return null;
    }
}
```

API Calling File-

```
package com.example.akshay.forecast;

import android.net.Uri;
import android.util.Log;

import java.io.IOException;
import java.io.InputStream;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.URL;
import java.util.Scanner;

/**
 * Created by Akshay on 17-03-2018.
 */

public class NetworkUtils {
    private static final String TAG = "NetworkUtils";
    private final static String WEATHERDB_BASE_URL=
            "http://dataservice.accuweather.com/forecasts/v1/daily/5day/190795";

    private final static String API_KEY="JUlCIjrcGoB71Vi1RgGLwwiGPooatana";

    private final static String METRIC_VALUE="true";

    private final static String PARAM_API_KEY="apikey";

    private final static String PARAM_METRIC= "metric";

    public static URL buildUrlForWeather(){
        Uri buildUri = Uri.parse(WEATHERDB_BASE_URL).buildUpon()
                .appendQueryParameter(PARAM_API_KEY,API_KEY)
                .appendQueryParameter(PARAM_METRIC,METRIC_VALUE)
                .build();

        URL url=null;
        try {
            url=new URL(buildUri.toString());
        } catch (MalformedURLException e) {
            e.printStackTrace();
        }

        Log.i(TAG, "buildUrlForWeather: url: "+url);
        return url;
    }

    public static String getResponseFromHttpUrl(URL url) throws IOException{
        HttpURLConnection urlConnection = (HttpURLConnection) url.openConnection();
        try {
            InputStream in = urlConnection.getInputStream();
```

```
            Scanner scanner = new Scanner(in);
            scanner.useDelimiter("\\A");

            boolean hasInput = scanner.hasNext();
            if (hasInput) {
                return scanner.next();
            } else {
                return null;
            }
        } finally {
            urlConnection.disconnect();
        }
    }
}
```

Function Calling File-

```
package com.example.akshay.forecast;

/**
 * Created by Akshay on 17-03-2018.
 */

public class Weather {
    String date;
    String minTemp;
    String maxTemp;
    String desc;

    public String getDate() {
        String[] sep = date.split("T");

        return sep[0];
    }

    public void setDate(String date) { this.date = date;

    }

    public String getMinTemp() {
        return minTemp;
    }

    public void setMinTemp(String minTemp) {
        this.minTemp = minTemp;
    }

    public String getMaxTemp() {
        return maxTemp;
    }

    public void setMaxTemp(String maxTemp) {
        this.maxTemp = maxTemp;
    }

    public String getDesc() {
        return desc;
    }

    public void setDesc(String desc) {
        this.desc = desc;
    }



    }
```

Applying Temperatures file-

```
package com.example.akshay.forecast;

import android.content.Context;
import android.support.annotation.NonNull;
import android.support.annotation.Nullable;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.TextView;

import java.util.ArrayList;

/**
 * Created by Akshay on 18-03-2018.
 */

public class WeatherAdapter extends ArrayAdapter<Weather> {
    public WeatherAdapter(@NonNull Context context, ArrayList<Weather>
weatherArrayList) {
        super(context, 0, weatherArrayList);
    }

    @NonNull
    @Override
    public View getView(int position, @Nullable View convertView, @NonNull
ViewGroup parent) {
        Weather weather = getItem(position);

        if(convertView == null){
            convertView =
LayoutInflater.from(getContext()).inflate(R.layout.list_item, parent, false);
        }

        TextView dateTextView = convertView.findViewById(R.id.tvDate);
        TextView minTextView = convertView.findViewById(R.id.tvLowTemperature);
        TextView maxTextView = convertView.findViewById(R.id.tvHighTemperature);
        TextView descTextView = convertView.findViewById(R.id.tvDesc);
        //TextView maxTextView2 = convertView.findViewById(R.id.maxtmp);
        //TextView minTextView2 = convertView.findViewById(R.id.mintmp);



        dateTextView.setText(weather.getDate());
        minTextView.setText(weather.getMinTemp());
        maxTextView.setText(weather.getMaxTemp());
        descTextView.setText(weather.getDesc());
        //minTextView2.setText(weather.getMinTemp2());
        //maxTextView2.setText(weather.getMaxTemp2());


        return convertView;
    }
}
```

The two types of APIs used are-

5 Days of Daily Forecasts

Returns an array of daily forecasts for the next 5 days for a specific location. Forecast searches require a location key. Please use the Locations API to obtain the location key for your desired location. By default, a truncated version of the hourly forecast data is returned. The full object can be obtained by passing "details=true" into the url string.

Resource URL

http://dataservice.accuweather.com/forecasts/v1/daily/5day/{190795}

1 Day of Daily Forecasts

Returns daily forecast data for a specific location. Forecast searches require a location key. Please use the Locations API to obtain the location key for your desired location. By default, a truncated version of the hourly forecast data is returned. The full object can be obtained by passing "details=true" into the url string.

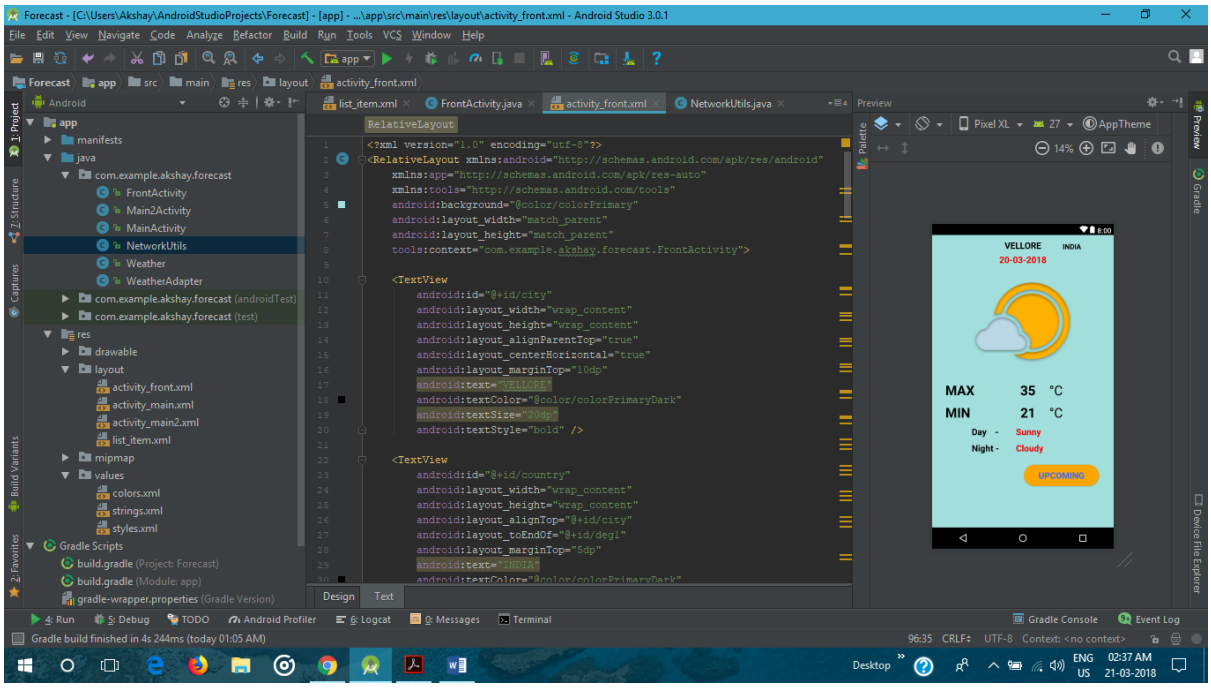
Resource URL

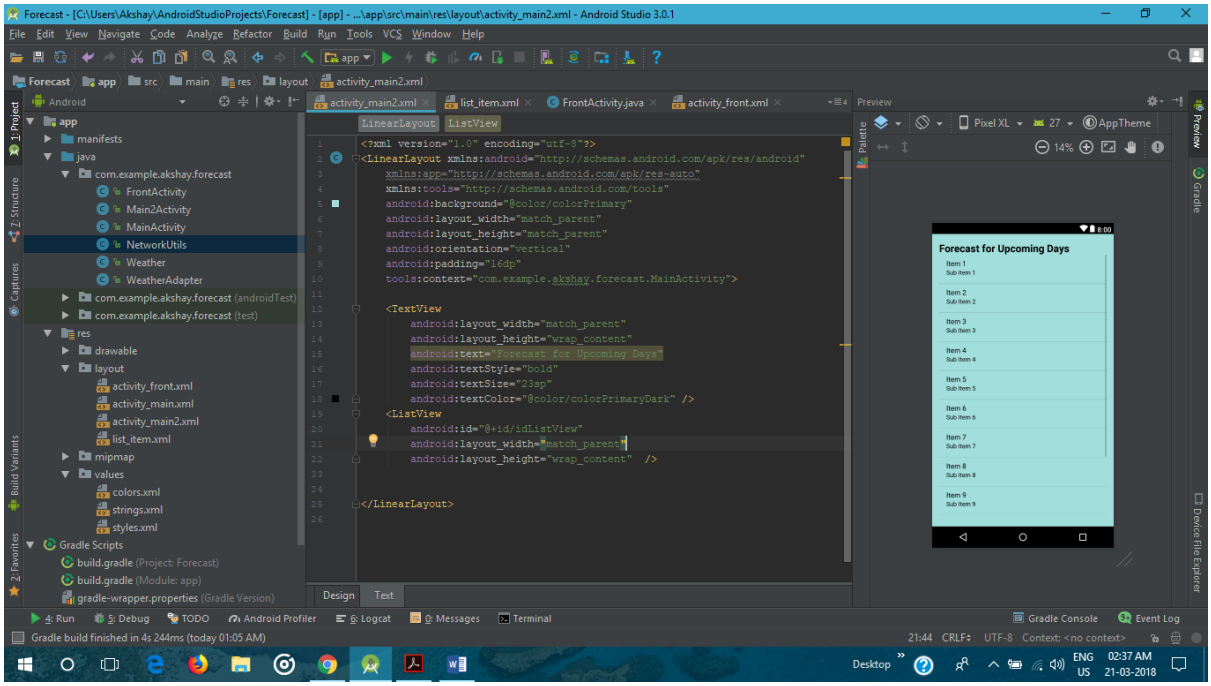
http://dataservice.accuweather.com/forecasts/v1/daily/1day/{190795}

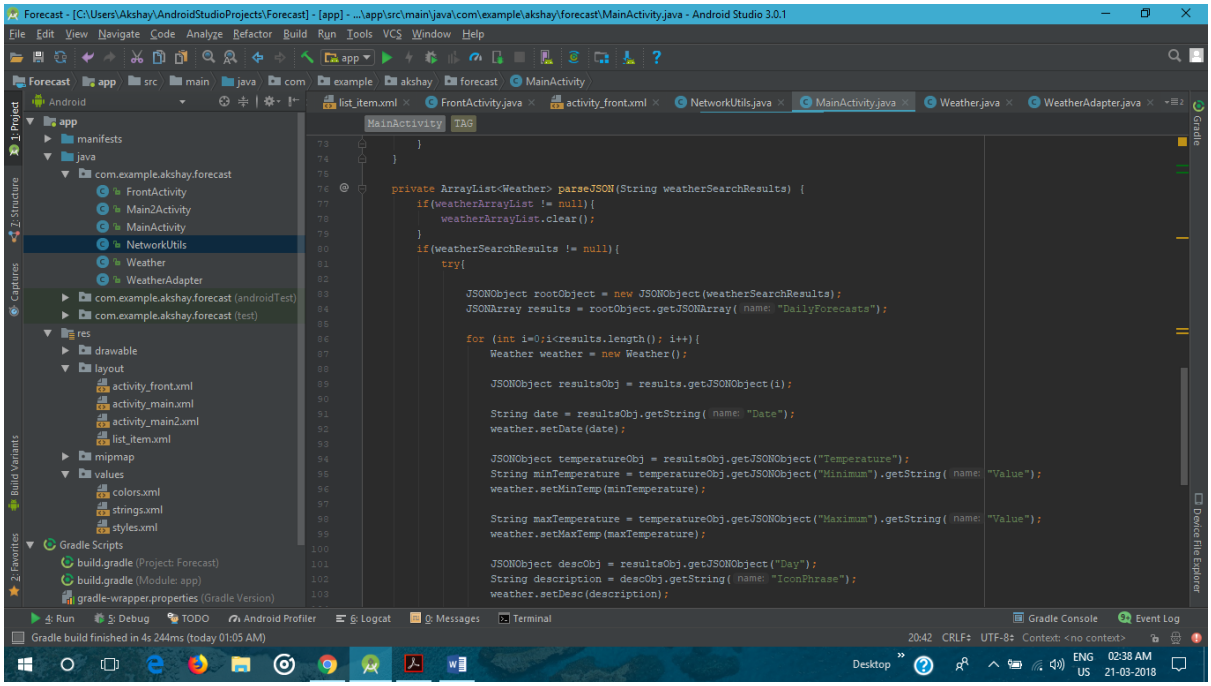
Result

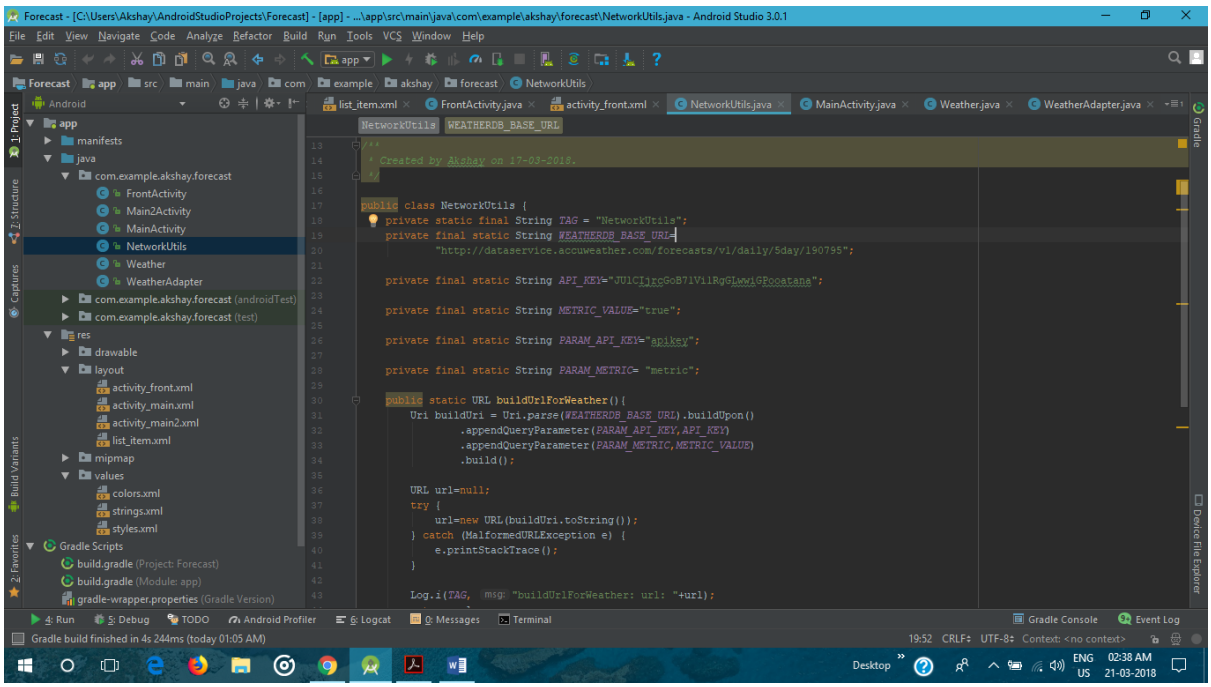
There were a lot of problems faced during the making of this project. But the biggest one was the limit on API usage i.e. 50 per day. I had to wait almost a day to check and test the app again because the limit was very easily reached when testing.

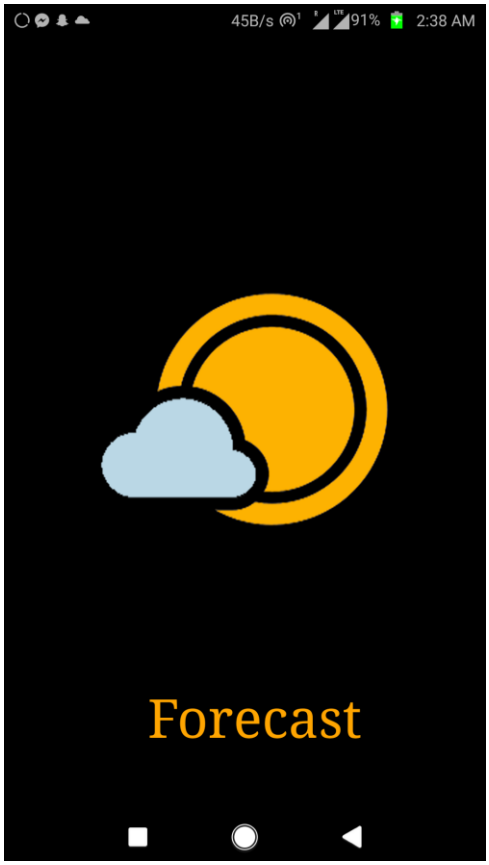
The screenshots are of Android Studio and the App are given-

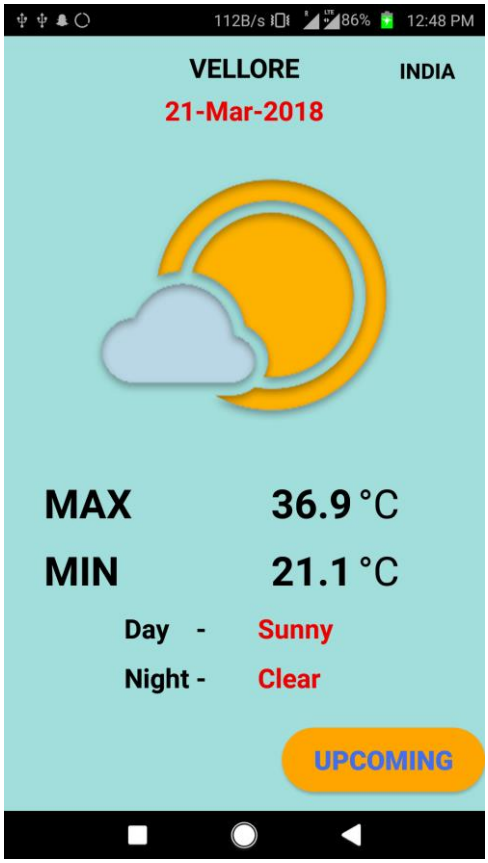


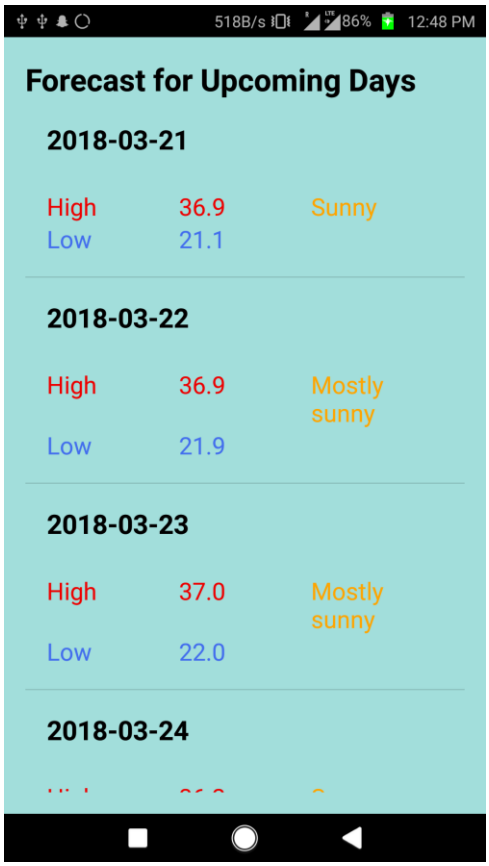


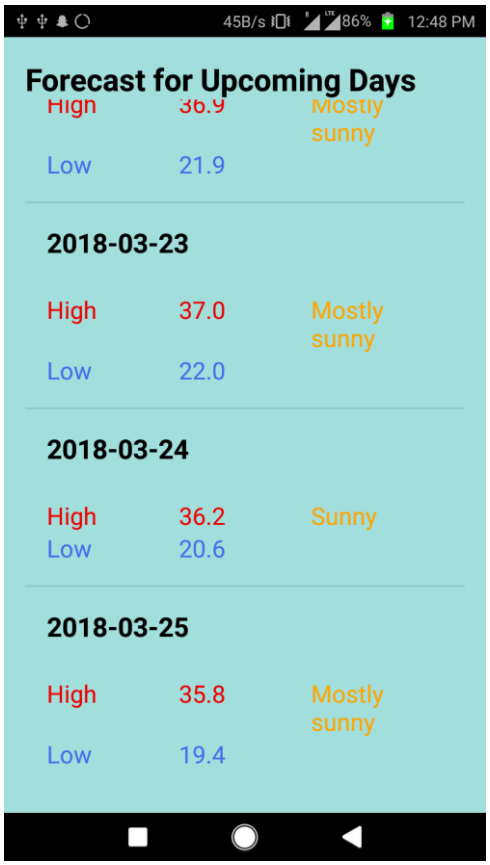

Forecast - [C:\Users\Akshay\AndroidStudioProjects\Forecast] - [app] - ...\app\src\main\java\com\example\akshay\forecast\MainActivity.java - Android Studio 3.0.1
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
list_item.xml FrontActivity.java activity_front.xml NetworkUtils.java MainActivity.java Weather.java WeatherAdapter.java
MainActivity TAG
private ArrayList<Weather> parseJSON(String weatherSearchResults) {
if(weatherArrayList != null){
weatherArrayList.clear();
}
if(weatherSearchResults != null){
try{
JSONObject rootObject = new JSONObject(weatherSearchResults);
JSONArray results = rootObject.getJSONArray(name: "DailyForecasts");
for (int i=0;i<results.length(); i++){
Weather weather = new Weather();
JSONObject resultsObj = results.getJSONObject(i);
String date = resultsObj.getString(name: "Date");
weather.setDate(date);
JSONObject temperatureObj = resultsObj.getJSONObject("Temperature");
String minTemperature = temperatureObj.getJSONObject("Minimum").getString(name: "Value");
weather.setMinTemp(minTemperature);
String maxTemperature = temperatureObj.getJSONObject("Maximum").getString(name: "Value");
weather.setMaxTemp(maxTemperature);
JSONObject descObj = resultsObj.getJSONObject("Day");
String description = descObj.getString(name: "IconPhrase");
weather.setDesc(description);
Gradle build finished in 4s 244ms (today 01:05 AM)
20:42 CRLF: UTF-8: Context: <no context>
Desktop
ENG US 02:38 AM 21-03-2018


Forecast - [C:\Users\Akshay\AndroidStudioProjects\Forecast] - [app] - ...\app\src\main\java\com\example\akshay\forecast\NetworkUtils.java - Android Studio 3.0.1
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
list_item.xml FrontActivity.java activity_front.xml NetworkUtils.java MainActivity.java Weather.java WeatherAdapter.java
NetworkUtils WEATHERDB_BASE_URL
/**
* Created by Akshay on 17-03-2018.
*/
public class NetworkUtils {
private static final String TAG = "NetworkUtils";
private final static String WEATHERDB_BASE_URL=
"http://dataservice.accuweather.com/forecasts/v1/daily/5day/190795";
private final static String API_KEY="JU1CIjrGoB71V1lRqGLwwiGPooatanq";
private final static String METRIC_VALUE="true";
private final static String PARAM_API_KEY="apikey";
private final static String PARAM_METRIC= "metric";
public static URL buildUrlForWeather(){
Uri buildUri = Uri.parse(WEATHERDB_BASE_URL).buildUpon()
.appendQueryParameter(PARAM_API_KEY,API_KEY)
.appendQueryParameter(PARAM_METRIC,METRIC_VALUE)
.build();
URL url=null;
try {
url=new URL(buildUri.toString());
} catch (MalformedURLException e) {
e.printStackTrace();
}
Log.i(TAG, msg: "buildUrlForWeather: url: "+url);
Gradle build finished in 4s 244ms (today 01:05 AM)
19:52 CRLF: UTF-8: Context: <no context>
Desktop
ENG US 02:38 AM 21-03-2018


45B/s 91% 2:38 AM
Forecast


112B/s 86% 12:48 PM
VELLORE
INDIA
21-Mar-2018
MAX
36.9 °C
MIN
21.1 °C
Day - Sunny
Night - Clear
UPCOMING


518B/s 86% 12:48 PM
Forecast for Upcoming Days
2018-03-21
High 36.9 Sunny
Low 21.1
2018-03-22
High 36.9 Mostly sunny
Low 21.9
2018-03-23
High 37.0 Mostly sunny
Low 22.0
2018-03-24


45B/s 86% 12:48 PM
Forecast for Upcoming Days
High 36.9 Mostly sunny
Low 21.9
2018-03-23
High 37.0 Mostly sunny
Low 22.0
2018-03-24
High 36.2 Sunny
Low 20.6
2018-03-25
High 35.8 Mostly sunny
Low 19.4

Conclusion

The app was completed and the apk was generated which was below a size of 5 Mb. The app is automatically synced to internet when opened and successfully showed the forecast of current as well as next 5 days. The app could also be made better by using a locations api, which can allow the user to enter any city name and the forecast will be displayed. The button was worked correctly which allowed the user to transfer from current day's forecast to next 5 days forecast. The UI design also showed the description of today's forecast.

References

1. www.developer.accuweather.com
2. www.youtube.com
3. www.images.google.com
4. www.google.com