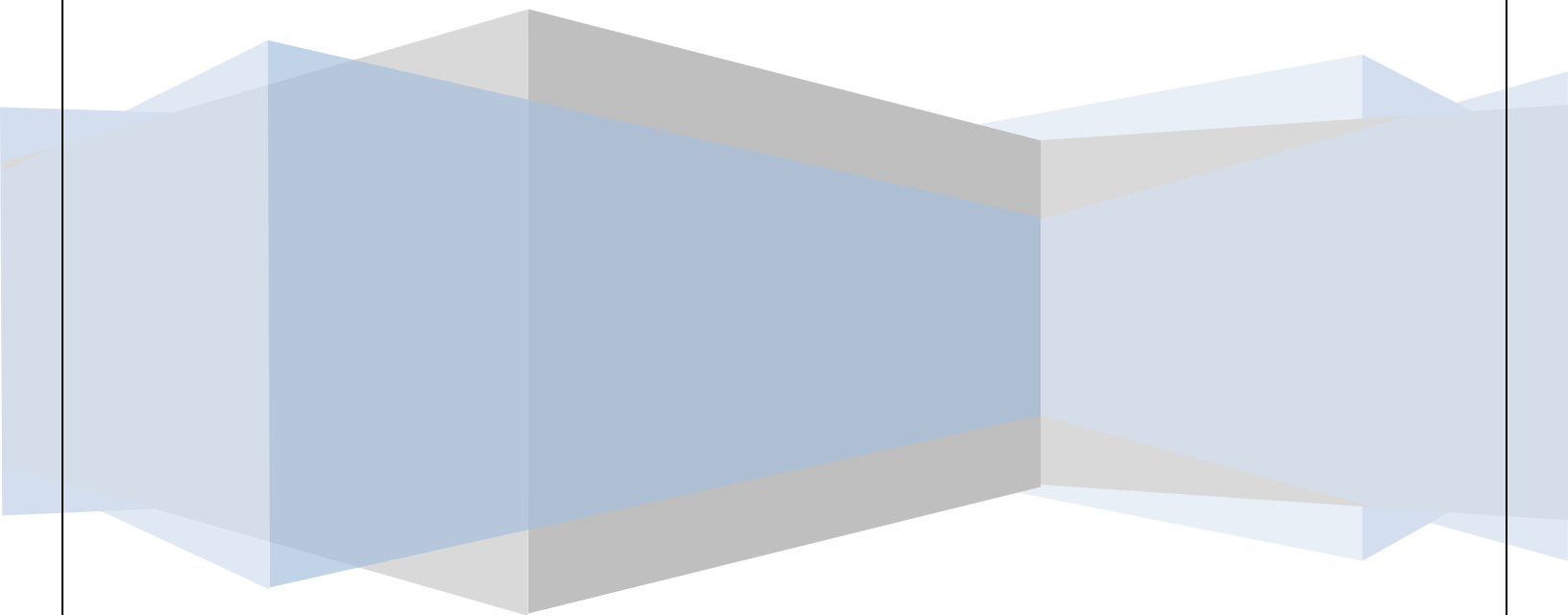


JESSICA CASTELINO – B00804805

ASSIGNMENT 3



INDEX

Sr No.	Table of Contents	Page no.
1.	Project Overview	2
2.	Notes and details	3
2.1	Pre-requisites	3
2.2	Wireframe	3
2.3	Nielson's Usability Heuristics	4
2.4	Code Organization	6
3.	Implementation problems and solutions	8
6	Testing Methodology	10
5.	References	14

1. PROJECT OVERVIEW

The main objective of this assignment is to develop a simple and intuitive android application that can help the users to view the current weather details immediately. The android app makes use of OpenWeatherMap API to fetch the current weather details in the form of a JSON Object. These details are parsed and displayed on the screen. The user must simply enter the city name and hit the “Get Weather” button to fetch the current weather details. The application has different background images as well, depending on the time of the day in the city whose current weather is requested. Thus, if a user is requesting weather details for a city, the app will load a bright morning background or a starry night background depending on the time of the day. Also, there are many icons used in the application as visuals are attractive and they improve the usability of the application. The icons also aid in faster absorption of content displayed. Another important feature is the ability of the application to display the date and time when the weather details were last updated, thus allowing the user to determine the accuracy of the results obtained.

2. NOTES AND DETAILS

2.1. Prerequisites

1. OpenWeatherMap API Key: It is important to own a key by signing up on OpenWeatherMap website. This key is used to make a REST API call and fetch current weather details.
2. Icons: Weather icons have been downloaded from OpenWeatherMap website. Apart from this, there are some icons for humidity, minimum temperature, maximum temperature, clouds, and background images which have been downloaded.
3. Application Permissions: The application requires internet connection in order to call the OpenWeatherMap API.

2.2. Wireframe

The wireframe for the weather app is as shown in Figure 1. The user interface is minimalistic and intuitive. The search button is given a different colour to improve its visibility. Owing to the limitations of the tool used to create this wireframe, icons have not been added in it. However, the actual user interface includes many icons.



Figure 1: Wireframe created for WeatherApp

2.3. Nielson's Usability Heuristics

These heuristics are a guideline for a good design. They are called “heuristics” because they are broad rules of thumb and not specific usability guidelines. [1] The application follows the Nielson's Usability Heuristics as mentioned below:

1. Visibility of system status

The application keeps the users informed about what is going on through appropriate feedback within reasonable time. Every time the user clicks a button, a feedback is provided by showing the weather details on the screen. Also, the city, country, and the time when the weather details were last updated are clearly shown on the screen.

2. Match between system and real world

The application does not use any system-oriented terms. It uses words, phrases and concepts that are familiar to the user. The error messages provided are understandable. If the user enters an invalid city name, the user is informed in plain language that it is a wrong city name instead of giving a 404 Not Found error.

3. User control and freedom

Users often choose system functions by mistake and will need a clearly marked “emergency exit” to leave the unwanted state without having to go through an extended dialogue. User has an option to re-enter city names or simply exit the application at any point.

4. Consistency and standards

The weather application’s design is consistent with other weather apps available in the market. The use of the icons, displaying the temperature in the largest font, etc are some of the ways in which this application maintains the consistency and standards.

5. Recognition rather than recall

In order to avoid overload on user’s memory, all the objects, actions, and options are kept visible. The user has can easily fetch weather details of any city at any time. Also, the user can clearly see the time when the weather details were last updated.

6. Help users recognize, diagnose and recover from errors

Error messages are expressed in plain language and they clearly indicate the problem. If the user tries to enter a blank or invalid city name, a toast message “**Please enter a valid city name!**” is shown on the screen.

7. Error prevention

the city name field is restricted to alphabets, space, and comma. Thus, the application prevents erroneous inputs from the user.

8. Aesthetic and minimalist design

The design of the application is minimalistic and uncluttered. Toast messages contain only the relevant information. No extra, confusing messages are added to confuse or distract the user.

2.3. Code Organization

Model-view-controller (MVC) architecture is followed for the development of the project. In android projects, the structure of the project forces the code to follow an MVC pattern. The activity acts as the controller, the various classes that are implemented act as the model, and the XML layouts and other resources act as the view. The interaction between these three components is clearly indicated in Figure 2.

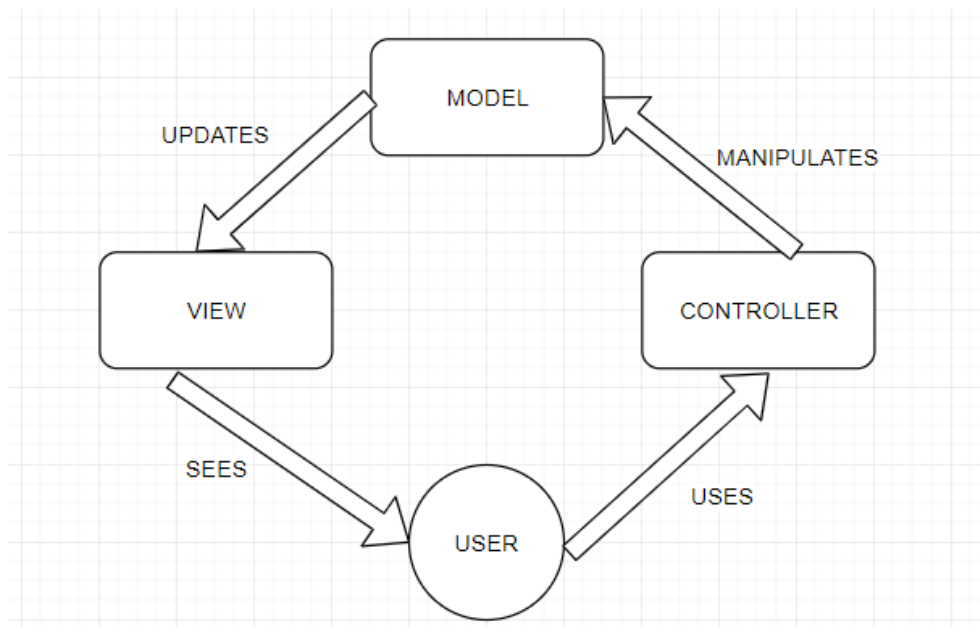


Figure 2:MVC Architecture

When the app loads for the first time, a REST call is made to OpenWeatherMap API to load the current weather details for a default city (Halifax). Thereafter, whenever the user enters a city name in the search field, the city name is used to fetch the weather details and the response is processed as an asynchronous task. The parsed JSON object is then rendered onto activity_main.xml. The sequence between the java classes is shown in Figure 3.

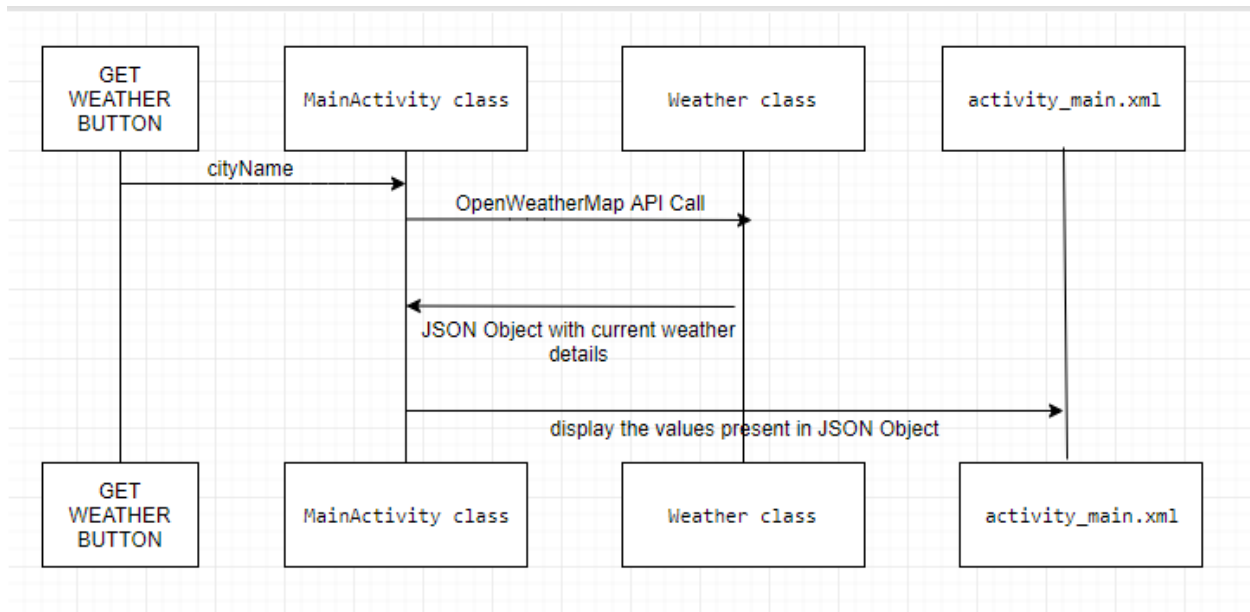


Figure 3: Sequence between java classes

3. IMPLEMENTATION PROBLEMS AND SOLUTIONS

1) Incorrect values for current weather

The JSON response received from OpenWeatherMap API returned temperature values in Fahrenheits, which is an imperial system. To resolve this issue, an extra parameter was added to the URL “**units=metric**” so that each value was in the metric system and no extra conversion was required before displaying it.

2) “Bad query response” error returned from the REST API

The OpenWeatherMap API returns a code of 200 if the city name exists and the request is successfully processed. In case any invalid city name was entered, the API returns an error code of 400. In both the cases, the JSON response received also had a different format. Thus, the application was crashing. To resolve it, the error code sent by the API is checked before processing the response.

3) Not using a library for network operations

Volley is an HTTP library that makes networking for Android apps easier and most importantly, faster. [2] As we were not allowed to use Volley for this assignment, HTTP GET request implemented as an asynchronous task was an alternative solution that I decided to explore. It was difficult to return the JSON object to the onCreate method. To resolve this, I parsed the JSON object inside the synchronous task itself. The relevant fields were fetched and sent as a JSON string to the onCreate method of the Main activity.

4) Populating images

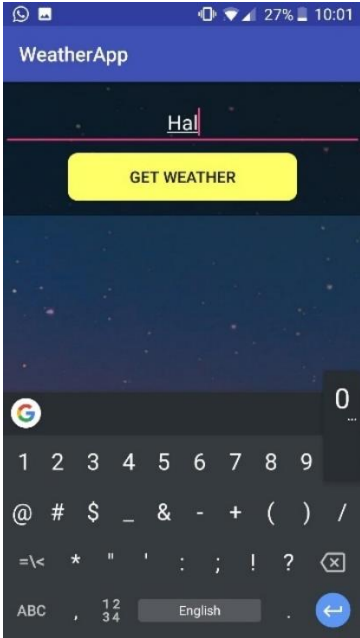

To enhance the user interface of the application, there was a need to make use of icons that can clearly reflect the current weather and support Nielson's heuristic of matching the system and the real world. However, there were problems in aligning the layout where dynamic values had to be placed within text views. For example, consider the line, (icon for minimum temperature) Min 8°C, (icon for maximum temperature) Max 9 °C. Here, the images had to be rendered with dynamic and static text views on the same line, which caused alignment issues. To resolve this problem, the images were directly added to the text view by making use of `SpannableStringBuilder` and `ImageSpan` class in java.

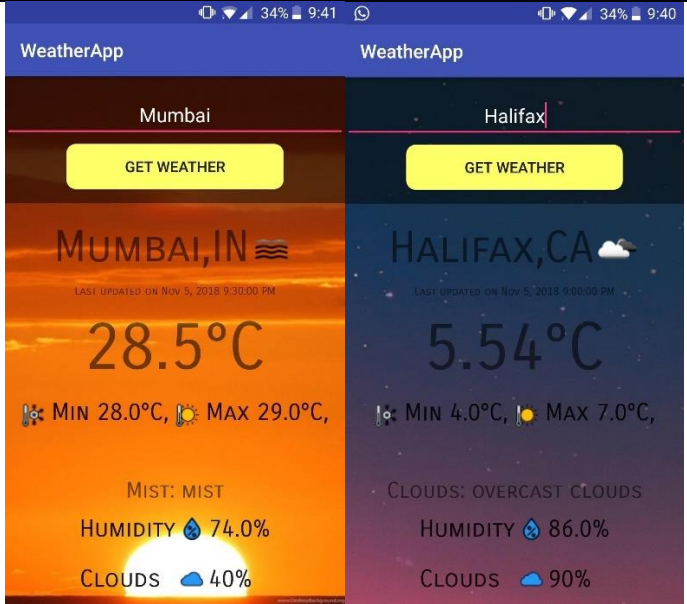

5) Changing background based on current time of the day in the queried city


An additional feature in this weather application is to know whether it is day or night in the city, whose name was entered in the search field. I tried using different API calls to find the current time, but that made the user interface lag a little. Later, I found a field called "icon", within the OpenWeatherMap API response, which gives icon names ending with the alphabet "d" if it is day time and the alphabet "n" if it is night time. By using this field value, the background of the app is changed, thus removing the need of any additional API call.

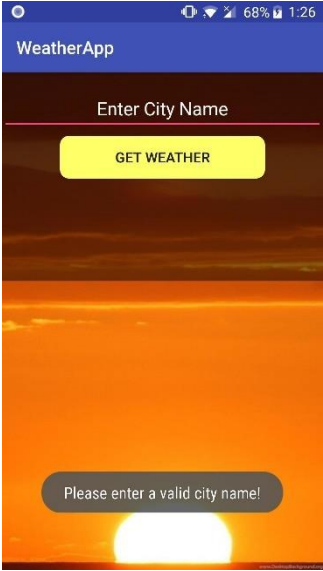
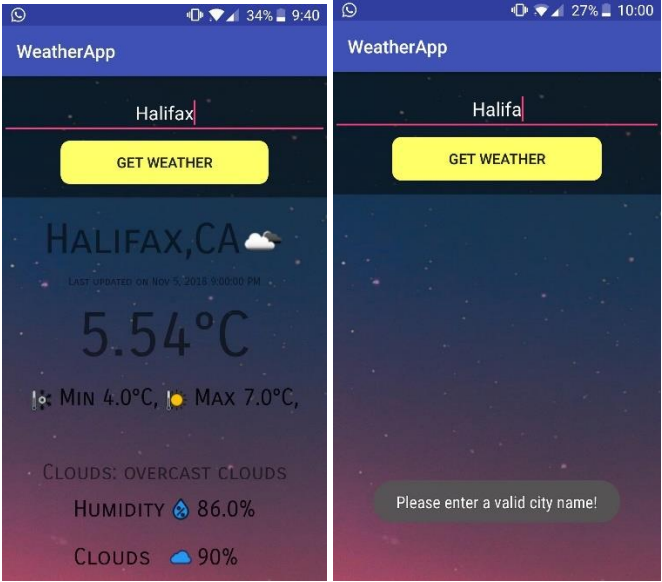
4. TESTING METHODOLOGY

The app has been tested thoroughly. Below are the test cases that have been performed.

Test Case	Result	Screenshot
Numbers should not be allowed in city name field.	Pass	 A screenshot of the WeatherApp interface. At the top, the status bar shows 27% battery and 10:01. The app title 'WeatherApp' is in a blue header. Below it, a text input field contains 'Hal'. A yellow 'GET WEATHER' button is positioned below the input field. A numeric keypad is visible at the bottom of the screen, indicating that numbers are not allowed in the city name field.
Application loads with default city name.	Pass	 A screenshot of the WeatherApp main screen. The status bar shows 67% battery and 11:29. The app title 'WeatherApp' is in a blue header. Below it, a text input field contains 'Enter City Name'. A yellow 'GET WEATHER' button is positioned below the input field. The main display area shows 'HALIFAX, CA' with a cloud icon, 'LAST UPDATED ON NOV 8, 2018 10:00:00 AM', '9.47°C', 'MIN 9.0°C, MAX 10.0°C', 'CLOUDS: SCATTERED CLOUDS', 'HUMIDITY 57.0%', and 'CLOUDS 40%'.

Background should reflect time of the day in the city for which weather details have been requested.	Pass	 <p>The screenshot shows two side-by-side views of the WeatherApp. The left view is for Mumbai, India, with an orange background and a sun icon. It displays a temperature of 28.5°C, a minimum of 28.0°C, a maximum of 29.0°C, humidity of 74.0%, and 40% clouds. The right view is for Halifax, Canada, with a dark blue background and a moon icon. It displays a temperature of 5.54°C, a minimum of 4.0°C, a maximum of 7.0°C, humidity of 86.0%, and 90% clouds. Both screens have a 'GET WEATHER' button and a 'LAST UPDATED' timestamp.</p>
The date and time when the weather details were last updated should be clearly visible.	Pass	 <p>The screenshot shows the WeatherApp interface for Halifax, Canada. It features an orange background with a sun icon. The temperature is 9.47°C, with a minimum of 9.0°C and a maximum of 10.0°C. The humidity is 57.0% and the cloud cover is 40%. The 'LAST UPDATED' timestamp, 'ON NOV 8, 2018 10:00:00 AM', is highlighted with a blue rectangular box.</p>

<p>If wrong city names are entered, toast message should be displayed</p>	<p>Pass</p>	 <p>The screenshot shows the WeatherApp interface. At the top, the status bar indicates 27% battery and 10:00. The app title 'WeatherApp' is in a blue header. Below it, a text input field contains 'Halifa'. A yellow 'GET WEATHER' button is positioned below the input field. At the bottom of the screen, a dark grey toast message box displays the text 'Please enter a valid city name!'.</p>
<p>Country name, along with the city name, should be clearly visible when the user populates.</p>	<p>Pass</p>	 <p>The screenshot shows the WeatherApp interface after a successful search. The status bar indicates 67% battery and 11:29. The app title 'WeatherApp' is in a blue header. Below it, a text input field contains 'Enter City Name'. A yellow 'GET WEATHER' button is positioned below the input field. The main display area shows the weather for 'HALIFAX, CA' with a cloud icon. Below the city name, it says 'LAST UPDATED ON NOV 8, 2018 10:00:00 AM'. The current temperature is '9.47°C'. Below that, it shows 'MIN 9.0°C' with a snowflake icon and 'MAX 10.0°C' with a sun icon. Further down, it displays 'CLOUDS: SCATTERED CLOUDS', 'HUMIDITY 57.0%' with a water drop icon, and 'CLOUDS 40%' with a cloud icon.</p>

<p>Application should not crash when user hits “Get Weather” button multiple times.</p>	<p>Pass</p>	
<p>The previous search result should be cleared every time a new request is made.</p>	<p>Pass</p>	

6. REFERENCES

[1] "10 Heuristics for User Interface Design: Article by Jakob Nielsen," *Nielsen Norman Group*. [Online]. Available: <https://www.nngroup.com/articles/ten-usability-heuristics/>. [Accessed: 10-Oct-2018].

[2] "Send a simple request | Android Developers," *Android Developers*. [Online]. Available: <https://developer.android.com/training/volley/simple>. [Accessed: 09-Nov-2018].