Department of Computer and Software Engineering- ITU

MD442L: Mobile Application Development Lab

Course Instructor: Syed Basit Ali	**Dated: 09/04/2024**
Lab Engineer: Atique Ahmad	**Semester: Spring 2024**
Session: 2020-2024	**Batch: BSCE2020**

Lab 9. Modify the Weather App to Support Multiple Languages

Name	Roll number	Obtained Marks/35
Hammad Kamran	BSCE20029	

Checked on: _____

Signature: _____

Objective

The goal of this handout is to teach students how to create, design, and develop their own Android mobile applications using Android Studio and Flutter SDK.

Equipment and Component

Component Description	Value	Quantity
Computer	Available in lab	1

Conduct of Lab

1. Students are required to perform this experiment individually.
2. In case the lab experiment is not understood, the students are advised to seek help from the course instructor, lab engineers, assigned teaching assistants (TA) and lab attendants.

Theory and Background

Android Studio is the official integrated development environment (IDE) for Google's Android operating system. It is built on JetBrains' IntelliJ IDEA software and is designed specifically for Android development. It is available for download on Windows, Mac, and Linux, and it includes a code editor, debugging and performance tools, an emulator, and a flexible build system. Android Studio also supports the Android Gradle plugin, which allows developers to manage dependencies and build their apps. Additionally, it provides features such as live layout rendering, code completion, and a visual layout editor to ease the development process. It also have integration with Firebase, which is a mobile development platform that helps you quickly develop high-quality apps, grow your user base, and earn more money.

XML layouts are used to define the user interface for an Android app. They specify the layout and structure of the app's UI using a markup language, similar to HTML. In an XML layout file, UI elements such as buttons, text fields, and images are defined as tags and attributes, and the layout of these elements is defined using a set of layout managers such as LinearLayout, RelativeLayout, and ConstraintLayout.

XML layouts are separate from the app's Java code and can be easily modified without affecting the app's functionality. This separation of concerns makes it easy to change the app's UI without having to modify the underlying code. The layout files are stored in the res/layout directory of an Android project.

Android Studio provides a visual layout editor that makes it easy to create and edit XML layouts. The editor allows you to drag and drop UI elements onto the layout and adjust their properties using the Properties window. Additionally, you can preview your layouts on different screen sizes and orientations.

In XML layouts you can also define styles and themes, which allows you to reuse common layout attributes and apply them consistently across your app. This helps to improve the overall design and maintainability of your app.

Java is the primary programming language used for developing Android applications in Android Studio. In Android Studio, you can use the Java programming language to create the logic and functionality of your app, such as handling user input, performing calculations, and updating the user interface.

A **weather app** using location allows users to view current weather conditions and forecasts for their current location. The app uses the phone's GPS or network location data to determine the user's location and then queries an external weather service API to obtain the latest weather information.

A **multilingual Android app** is an application that can display text in multiple languages. This is achieved by implementing localization in the app, which involves creating language-specific resources for the different languages that the app will support. These resources can be text strings, images, layouts, and other assets that are tailored to a particular language.

To support multiple languages, the app should have a way of detecting the user's preferred language, either by using the device's language settings or by giving the user the option to choose their preferred language within the app. Once the user's preferred language is determined, the app should load the appropriate language-specific resources and use them to display text and other UI components in the user's preferred language.

Implementing multilingual support in an Android app can greatly improve its usability and accessibility for users who speak different languages. It can also expand the app's reach to a global audience, potentially increasing its user base and engagement.

Submission Guidelines:

- Save your document as **ROLLNO_LAB_8.pdf** (e.g., BSEE22099_LAB_8.pdf).
- Clearly present **Java code and XML code** separately within the document.
- Attach a **full laptop screen screenshot** showing your app emulator in action. Ensure the **emulator** is visible in the screenshot.
- Incorrectly named files will **not be accepted.**

Lab Tasks

Task A – Design and Develop Multilangual App

In this lab, you have been tasked to modify an existing Weather App to support multiple languages. The app currently supports only English language, and you need to add support for French, Arabic, Spanish, Russian, and Urdu languages.

The following are the three main steps required to achieve this:

Step 1: Implementing Localization in the Weather App

- Add language resources for each of the supported languages (English, French, Arabic, Spanish, Urdu) in the res/values folder using the strings.xml file.
- Use string resources instead of hardcoded strings throughout the app to enable localization.

Step 2: Updating the UI to Support Multiple Languages

- Update the UI components in the app, such as text views and buttons, to use the string resources for the appropriate language.
- Use the setText() method to set the appropriate string resource for each UI component.

Step 3: Testing the Weather App with Multiple Languages

- Test the Weather App with each of the supported languages to ensure that the app changes language when the user changes the language on their phone.
- Verify that the appropriate string resources are used for each UI component.
- Verify that the app's functionality is not affected by the language change.

Note: It is important to note that the Weather API used in the app may not provide localized weather data. Therefore, the weather information displayed in the app may remain in the language of the API's response.

```
package com.example.lab9;

import android.Manifest;
import android.annotation.SuppressLint;
import android.content.Context;
import android.content.pm.PackageManager;
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
import android.os.AsyncTask;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;
```

```
import android.location.Address;
import android.location.Geocoder;
import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;

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

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.List;
import java.util.Locale;

public class MainActivity extends AppCompatActivity implements
LocationListener {
    private static final int LOCATION_PERMISSION_REQUEST_CODE = 1001;
    private static final long MIN_TIME_BW_UPDATES = 10000; // Minimum time
interval between location updates (milliseconds)
    private static final float MIN_DISTANCE_CHANGE_FOR_UPDATES = 10; //
Minimum distance between location updates (meters)

    private TextView textViewWeatherCondition;
    private TextView textViewCurrentTemp;
    private TextView textViewFeelsLike;
    private TextView textViewMaxTemp;
    private TextView textViewMinTemp;
    private TextView textViewHumidity;
    private TextView textViewVisibility;
    private TextView textViewWindSpeed;
    private TextView textViewWindDirection;
    private TextView textViewSunriseTime;
    private TextView textViewSunsetTime;
    private TextView textViewLatitude;
    private TextView textViewLongitude;
    private TextView textViewCity;
    private Button fetchWeatherButton;
    Location location ;
    private LocationManager locationManager;

    @SuppressLint("MissingInflatedId")
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        // Initialize UI components
        // Initialize UI components
        textViewWeatherCondition =
findViewById(R.id.textViewWeatherCondition);
        textViewCurrentTemp = findViewById(R.id.textViewCurrentTemp);
        textViewFeelsLike = findViewById(R.id.textViewFeelsLike);
```

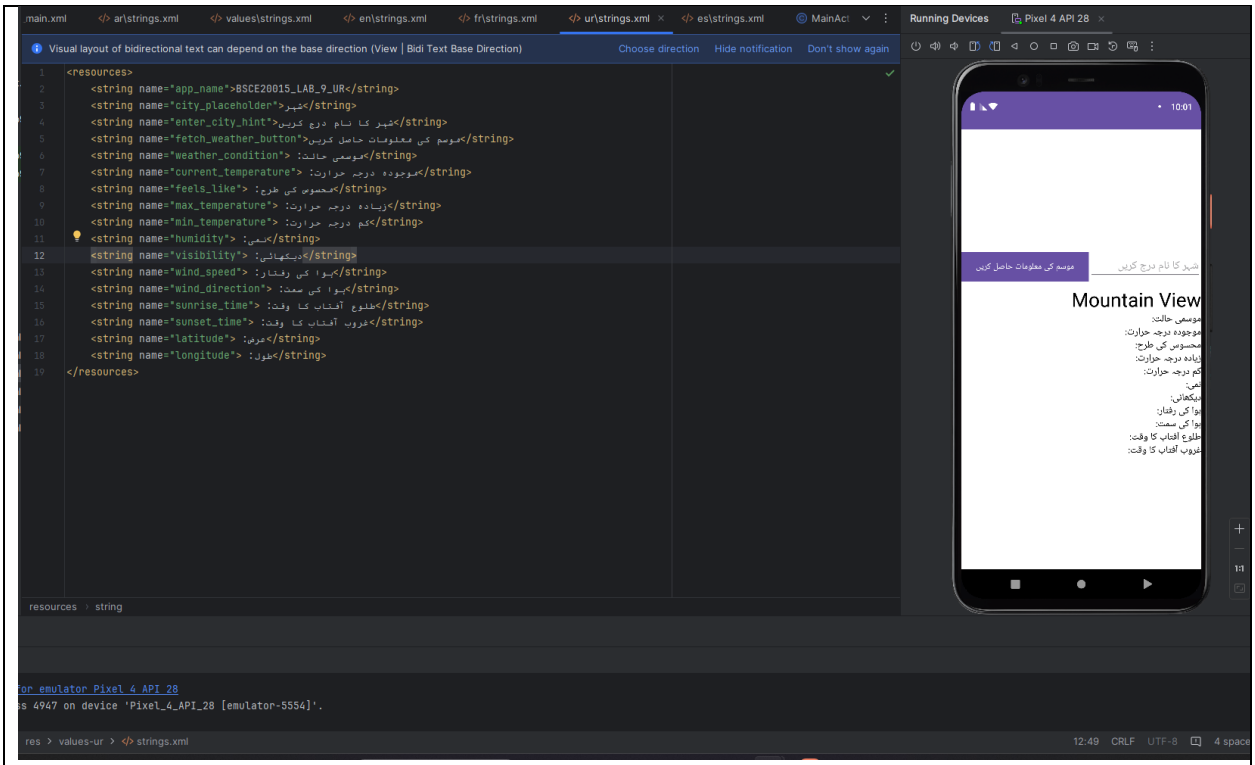
```
        textViewMaxTemp = findViewById(R.id.textViewMaxTemp);
        textViewMinTemp = findViewById(R.id.textViewMinTemp);
        textViewHumidity = findViewById(R.id.textViewHumidity);
        textViewVisibility = findViewById(R.id.textViewVisibility);
        textViewWindSpeed = findViewById(R.id.textViewWindSpeed);
        textViewWindDirection = findViewById(R.id.textViewWindDirection);
        textViewSunriseTime = findViewById(R.id.textViewSunriseTime);
        textViewSunsetTime = findViewById(R.id.textViewSunsetTime);
//        textViewLatitude = findViewById(R.id.textViewLatitude);
//        textViewLongitude = findViewById(R.id.textViewLongitude);
        fetchWeatherButton = findViewById(R.id.fetchWeatherButton);
        textViewCity = findViewById(R.id.textViewCity);

        // Initialize LocationManager
        locationManager = (LocationManager)
getSystemService(Context.LOCATION_SERVICE);

        // Check location permissions
        if (checkLocationPermissions()) {
            // If permissions granted, fetch weather using location
            fetchWeatherBasedOnLocation();
        } else {
            // Request location permissions
            requestLocationPermissions();
        }

        // Set onClickListener for the Fetch Weather button
        fetchWeatherButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // Check location permissions
                if (checkLocationPermissions()) {
                    // If permissions granted, fetch weather using location
                    fetchWeatherBasedOnLocation();
                } else {
                    // Request location permissions
                    requestLocationPermissions();
                }
            }
        });
    }

    // Check location permissions
    private boolean checkLocationPermissions() {
        return ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION) ==
PackageManager.PERMISSION_GRANTED;
    }

    // Request location permissions
    private void requestLocationPermissions() {
        ActivityCompat.requestPermissions(this, new
String[]{Manifest.permission.ACCESS_FINE_LOCATION},
LOCATION_PERMISSION_REQUEST_CODE);
    }

    // Handle location permission request result
```

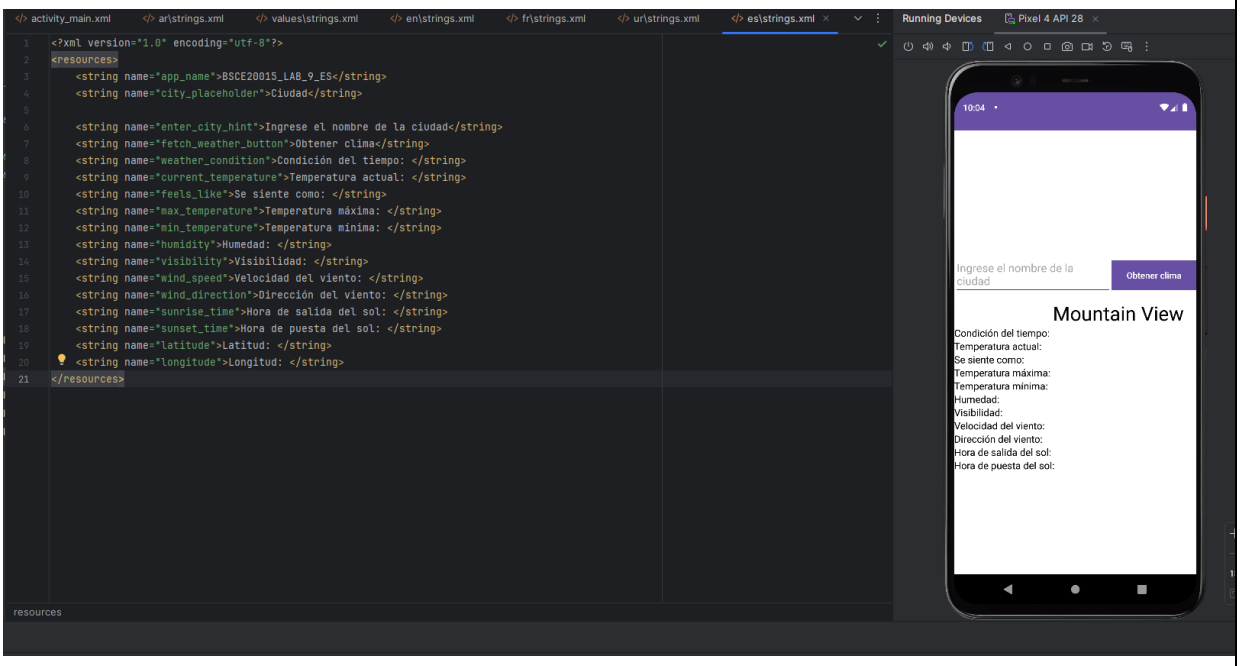
```
    @Override
    public void onRequestPermissionsResult(int requestCode, @NonNull
String[] permissions, @NonNull int[] grantResults) {
        super.onRequestPermissionsResult(requestCode, permissions,
grantResults);
        if (requestCode == LOCATION_PERMISSION_REQUEST_CODE) {
            if (grantResults.length > 0 && grantResults[0] ==
PackageManager.PERMISSION_GRANTED) {
                // Permission granted, fetch weather using location
                fetchWeatherBasedOnLocation();
            } else {
                // Permission denied, show a message or handle accordingly
                Toast.makeText(this, "Location permission denied.",
Toast.LENGTH_SHORT).show();
            }
        }
    }

    // Fetch weather using device location
    private void fetchWeatherBasedOnLocation() {
        // Check if location providers are enabled
        if (locationManager.isProviderEnabled(LocationManager.GPS_PROVIDER)
|| locationManager.isProviderEnabled(LocationManager.NETWORK_PROVIDER)) {
            // Request location updates
            if (checkLocationPermissions()) {
                if (ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED &&
ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_COARSE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {
                    // TODO: Consider calling
                    //    ActivityCompat#requestPermissions
                    // here to request the missing permissions, and then
overriding
                    //   public void onRequestPermissionsResult(int
requestCode, String[] permissions,
                    //                                          int[]
grantResults)
                    // to handle the case where the user grants the
permission. See the documentation
                    // for ActivityCompat#requestPermissions for more
details.
                    return;
                }
                locationManager.requestLocationUpdates(
                        LocationManager.NETWORK_PROVIDER,
                        MIN_TIME_BW_UPDATES,
                        MIN_DISTANCE_CHANGE_FOR_UPDATES,
                        this
                );
                locationManager.requestLocationUpdates(
                        LocationManager.GPS_PROVIDER,
                        MIN_TIME_BW_UPDATES,
                        MIN_DISTANCE_CHANGE_FOR_UPDATES,
                        this
                );
```

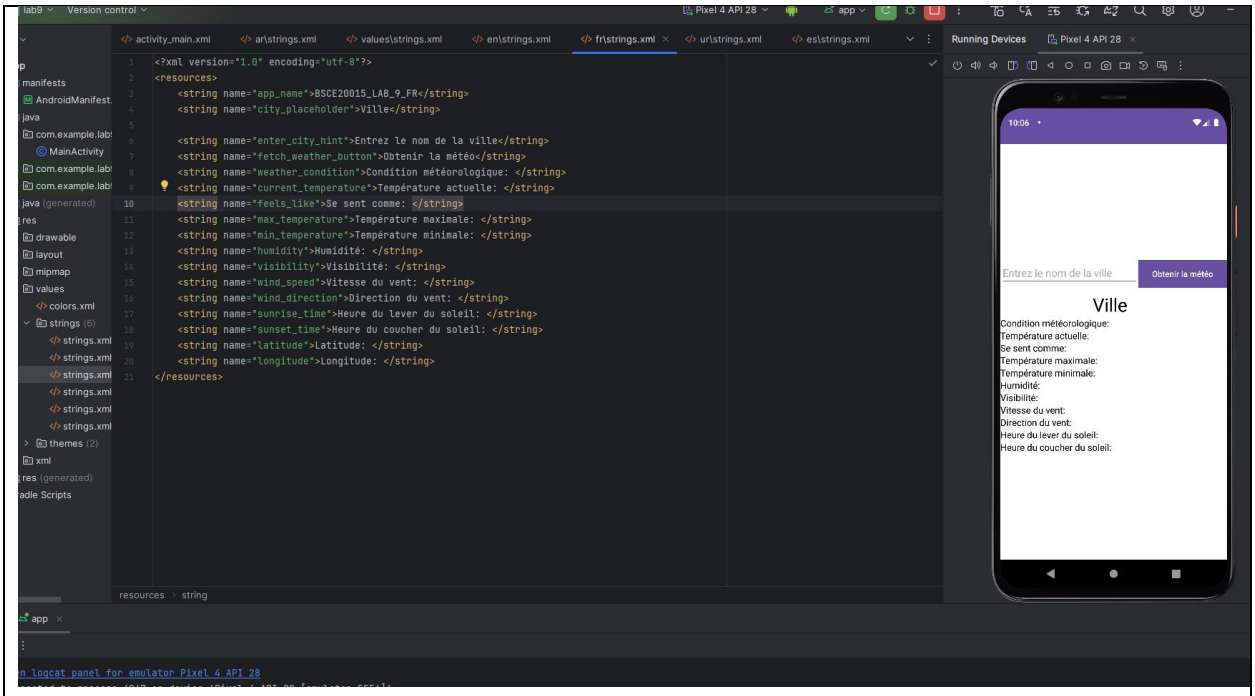
```
            } else {
                requestLocationPermissions();
            }
        } else {
            Toast.makeText(this, "Please enable location services.",
Toast.LENGTH_SHORT).show();
        }
    }

    // Stop location updates when activity is paused
    @Override
    protected void onPause() {
        super.onPause();
        locationManager.removeUpdates(this);
    }

    // LocationListener methods
    @Override
    public void onLocationChanged(@NonNull Location location) {
        // Fetch weather using the updated location
        new FetchWeatherTask().execute(location);
        // Stop location updates after fetching weather
        locationManager.removeUpdates(this);
    }

    @Override
    public void onProviderEnabled(@NonNull String provider) {}

    @Override
    public void onProviderDisabled(@NonNull String provider) {}

    @Override
    public void onStatusChanged(String provider, int status, Bundle extras)
{}

    // AsyncTask to fetch weather data in the background
    private class FetchWeatherTask extends AsyncTask<Location, Void,
JSONObject> {
        @Override
        protected JSONObject doInBackground(Location... params) {
            location = params[0];
            double latitude = location.getLatitude();
            double longitude = location.getLongitude();

            // Build the URL for weather API using latitude and longitude
            String apiKey = "412ef8af81ed389293f904eda378c651";
            String urlString =
"https://api.openweathermap.org/data/2.5/weather?lat=" + latitude + "&lon="
+ longitude + "&appid=" + apiKey;

            try {
                // Make HTTP request
                URL url = new URL(urlString);
                HttpURLConnection urlConnection = (HttpURLConnection)
url.openConnection();
                BufferedReader reader = new BufferedReader(new
InputStreamReader(urlConnection.getInputStream()));
```

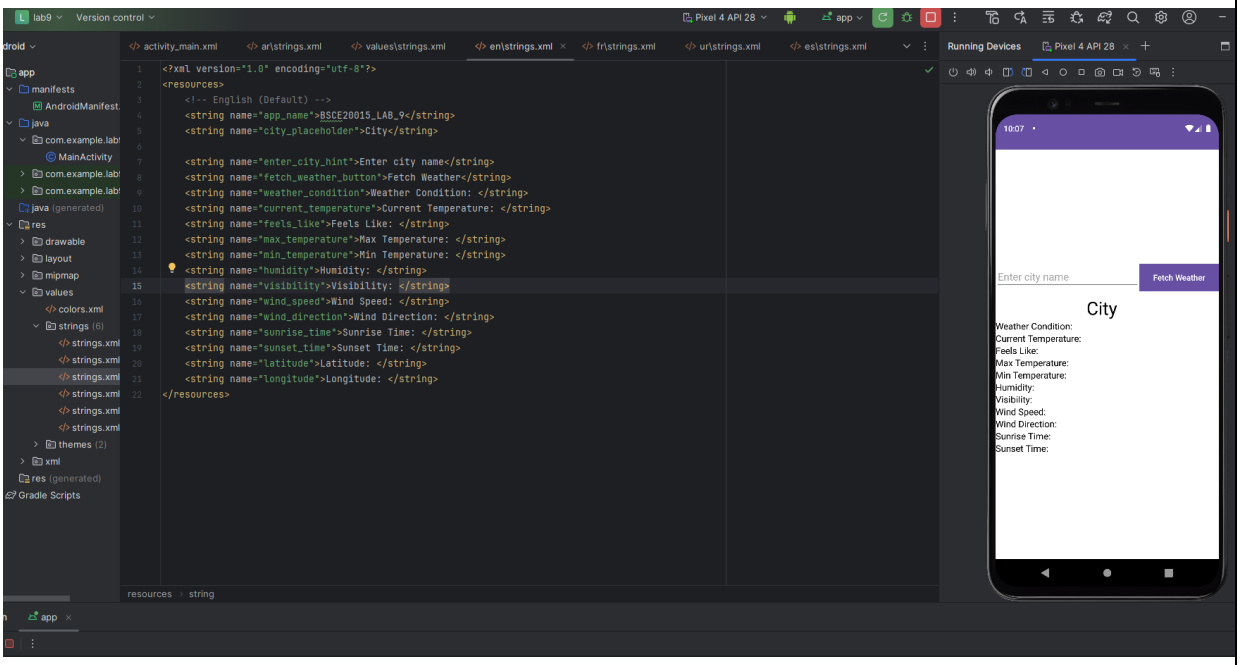
```
                StringBuilder stringBuilder = new StringBuilder();
                String line;
                while ((line = reader.readLine()) != null) {
                    stringBuilder.append(line);
                }
                reader.close();
                return new JSONObject(stringBuilder.toString());
            } catch (IOException | JSONException e) {
                e.printStackTrace();
            }
            return null;
        }

        @Override
        protected void onPostExecute(JSONObject result) {
            // Parse JSON response and display weather data
            if (result != null) {
                try {
                    // Extract weather data from JSON
                    JSONObject main = result.getJSONObject("main");
                    String weatherCondition =
result.getJSONArray("weather").getJSONObject(0).getString("description");
                    double currentTempKelvin = main.getDouble("temp");
                    double feelsLikeKelvin = main.getDouble("feels_like");
                    double maxTempKelvin = main.getDouble("temp_max");
                    double minTempKelvin = main.getDouble("temp_min");
                    int humidity = main.getInt("humidity");
                    double visibility = result.getDouble("visibility");
                    JSONObject wind = result.getJSONObject("wind");
                    double windSpeed = wind.getDouble("speed");
                    double windDirection = wind.getDouble("deg");
                    long sunriseTimestamp =
result.getJSONObject("sys").getLong("sunrise");
                    long sunsetTimestamp =
result.getJSONObject("sys").getLong("sunset");

                    // Convert temperatures from Kelvin to Celsius
                    double currentTempCelsius = currentTempKelvin - 273.15;
                    double feelsLikeCelsius = feelsLikeKelvin - 273.15;
                    double maxTempCelsius = maxTempKelvin - 273.15;
                    double minTempCelsius = minTempKelvin - 273.15;

                    // Format sunrise and sunset times
                    String sunriseTime = formatTimestamp(sunriseTimestamp);
                    String sunsetTime = formatTimestamp(sunsetTimestamp);
                    Geocoder geocoder = new Geocoder(MainActivity.this,
Locale.getDefault());
                    List<Address> addresses =
geocoder.getFromLocation(location.getLatitude(), location.getLongitude(),
1);

                    if (!addresses.isEmpty()) {
                        String cityName = addresses.get(0).getLocality();
                        if (cityName != null) {
                            textViewCity.setText(cityName);
                        } else {
                            textViewCity.setText("City not found");
```

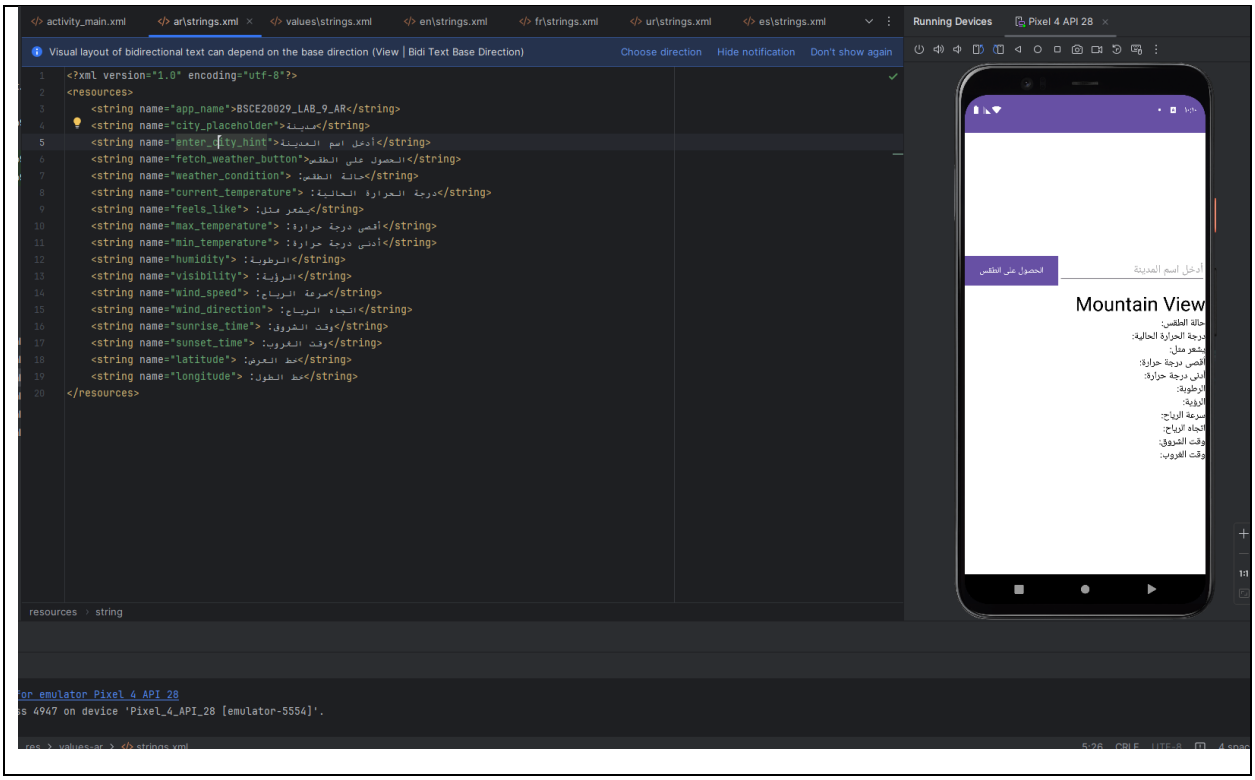
```
                        }
                    } else {
                        textViewCity.setText("City not found");
                    }
                    // Update TextViews with weather data
//                      textViewWeatherCondition.setText("Weather Condition: "
+ weatherCondition);
//                      textViewCurrentTemp.setText("Current Temperature: " +
String.format("%.2f", currentTempCelsius) + "°C");
//                      textViewFeelsLike.setText("Feels Like: " +
String.format("%.2f", feelsLikeCelsius) + "°C");
//                      textViewMaxTemp.setText("Max Temperature: " +
String.format("%.2f", maxTempCelsius) + "°C");
//                      textViewMinTemp.setText("Min Temperature: " +
String.format("%.2f", minTempCelsius) + "°C");
//                      textViewHumidity.setText("Humidity: " + humidity +
"%");
//                      textViewVisibility.setText("Visibility: " + visibility
+ " meters");
//                      textViewWindSpeed.setText("Wind Speed: " + windSpeed +
" m/s");
//                      textViewWindDirection.setText("Wind Direction: " +
windDirection + "°");
//                      textViewSunriseTime.setText("Sunrise Time: " +
sunriseTime);
//                      textViewSunsetTime.setText("Sunset Time: " +
sunsetTime);
//
//                      // Update latitude and longitude TextViews
//                      textViewLatitude.setText("Latitude: " +
location.getLatitude());
//                      textViewLongitude.setText("Longitude: " +
location.getLongitude());

                } catch (JSONException e) {
                    e.printStackTrace();
                } catch (IOException e) {
                    throw new RuntimeException(e);
                }
            }
        }

        // Method to format timestamp to human-readable time
        private String formatTimestamp(long timestamp) {
            SimpleDateFormat sdf = new SimpleDateFormat("hh:mm a",
Locale.getDefault());
            Date date = new Date(timestamp * 1000); // Convert timestamp to
milliseconds
            return sdf.format(date);
        }
    }
}
```


:main.xml
ar\strings.xml
values\strings.xml
en\strings.xml
fr\strings.xml
ur\strings.xml
es\strings.xml
MainAct
Visual layout of bidirectional text can depend on the base direction (View | Bidi Text Base Direction)
Choose direction
Hide notification
Don't show again
<resources>
<string name="app_name">BSCE20015_LAB_9_UR</string>
<string name="city_placeholder">شہر</string>
<string name="enter_city_hint">شہر کا نام درج کریں</string>
<string name="fetch_weather_button">موسم کی معلومات حاصل کریں</string>
<string name="weather_condition">موسمی حالت: </string>
<string name="current_temperature">موجودہ درجہ حرارت: </string>
<string name="feels_like">محسوس کی طرح: </string>
<string name="max_temperature">زیادہ درجہ حرارت: </string>
<string name="min_temperature">کم درجہ حرارت: </string>
<string name="humidity">نمی: </string>
<string name="visibility">دیکھنا: </string>
<string name="wind_speed">ہوا کی رفتار: </string>
<string name="wind_direction">ہوا کی سمت: </string>
<string name="sunrise_time">طلوع آفتاب کا وقت: </string>
<string name="sunset_time">غروب آفتاب کا وقت: </string>
<string name="latitude">عرض: </string>
<string name="longitude">طول: </string>
</resources>
resources
string
Running Devices
Pixel 4 API 28
10:01
موسم کی معلومات حاصل کریں
شہر کا نام درج کریں
Mountain View
موسمی حالت:
موجودہ درجہ حرارت:
محسوس کی طرح:
زیادہ درجہ حرارت:
کم درجہ حرارت:
نمی:
دیکھنا:
ہوا کی رفتار:
ہوا کی سمت:
طلوع آفتاب کا وقت:
غروب آفتاب کا وقت:
for emulator Pixel_4_API_28
s 4947 on device 'Pixel_4_API_28 [emulator-5554]'.
res > values-ur > strings.xml
12:49
CRLF
UTF-8
4 spaces


activity_main.xml
ar\strings.xml
values\strings.xml
en\strings.xml
fr\strings.xml
ur\strings.xml
es\strings.xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
<string name="app_name">BSCE20015_LAB_9_ES</string>
<string name="city_placeholder">Ciudad</string>
<string name="enter_city_hint">Ingrese el nombre de la ciudad</string>
<string name="fetch_weather_button">Obtener clima</string>
<string name="weather_condition">Condición del tiempo: </string>
<string name="current_temperature">Temperatura actual: </string>
<string name="feels_like">Se siente como: </string>
<string name="max_temperature">Temperatura máxima: </string>
<string name="min_temperature">Temperatura mínima: </string>
<string name="humidity">Humedad: </string>
<string name="visibility">Visibilidad: </string>
<string name="wind_speed">Velocidad del viento: </string>
<string name="wind_direction">Dirección del viento: </string>
<string name="sunrise_time">Hora de salida del sol: </string>
<string name="sunset_time">Hora de puesta del sol: </string>
<string name="latitude">Latitud: </string>
<string name="longitude">Longitud: </string>
</resources>
resources
Running Devices
Pixel 4 API 28
10:04
Ingrese el nombre de la ciudad
Obtener clima
Mountain View
Condición del tiempo:
Temperatura actual:
Se siente como:
Temperatura máxima:
Temperatura mínima:
Humedad:
Visibilidad:
Velocidad del viento:
Dirección del viento:
Hora de salida del sol:
Hora de puesta del sol:


lab9
Version control
Pixel 4 API 28
app
activity_main.xml
ar\strings.xml
values\strings.xml
en\strings.xml
fr\strings.xml
ur\strings.xml
es\strings.xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
<string name="app_name">BSCE20015_LAB_9_FR</string>
<string name="city_placeholder">Ville</string>
<string name="enter_city_hint">Entrez le nom de la ville</string>
<string name="fetch_weather_button">Obtenir la météo</string>
<string name="weather_condition">Condition météorologique: </string>
<string name="current_temperature">Température actuelle: </string>
<string name="feels_like">Se sent comme: </string>
<string name="max_temperature">Température maximale: </string>
<string name="min_temperature">Température minimale: </string>
<string name="humidity">Humidité: </string>
<string name="visibility">Visibilité: </string>
<string name="wind_speed">Vitesse du vent: </string>
<string name="wind_direction">Direction du vent: </string>
<string name="sunrise_time">Heure du lever du soleil: </string>
<string name="sunset_time">Heure du coucher du soleil: </string>
<string name="latitude">Latitude: </string>
<string name="longitude">Longitude: </string>
</resources>
Running Devices
Pixel 4 API 28
10:06
Entrez le nom de la ville
Obtenir la météo
Ville
Condition météorologique:
Température actuelle:
Se sent comme:
Température maximale:
Température minimale:
Humidité:
Visibilité:
Vitesse du vent:
Direction du vent:
Heure du lever du soleil:
Heure du coucher du soleil:


lab9
Version control
Pixel 4 API 28
app
activity_main.xml
ar\strings.xml
values\strings.xml
en\strings.xml
fr\strings.xml
ur\strings.xml
es\strings.xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
<!-- English (Default) -->
<string name="app_name">BSCE20015_LAB_9</string>
<string name="city_placeholder">City</string>
<string name="enter_city_hint">Enter city name</string>
<string name="fetch_weather_button">Fetch Weather</string>
<string name="weather_condition">Weather Condition: </string>
<string name="current_temperature">Current Temperature: </string>
<string name="feels_like">Feels Like: </string>
<string name="max_temperature">Max Temperature: </string>
<string name="min_temperature">Min Temperature: </string>
<string name="humidity">Humidity: </string>
<string name="visibility">Visibility: </string>
<string name="wind_speed">Wind Speed: </string>
<string name="wind_direction">Wind Direction: </string>
<string name="sunrise_time">Sunrise Time: </string>
<string name="sunset_time">Sunset Time: </string>
<string name="latitude">Latitude: </string>
<string name="longitude">Longitude: </string>
</resources>
Running Devices
Pixel 4 API 28
10:07
Enter city name
Fetch Weather
City
Weather Condition:
Current Temperature:
Feels Like:
Max Temperature:
Min Temperature:
Humidity:
Visibility:
Wind Speed:
Wind Direction:
Sunrise Time:
Sunset Time:


activity_main.xml
ar\strings.xml
values\strings.xml
en\strings.xml
fr\strings.xml
ur\strings.xml
es\strings.xml
Visual layout of bidirectional text can depend on the base direction (View | Bidi Text Base Direction)
Choose direction
Hide notification
Don't show again
<?xml version="1.0" encoding="utf-8"?>
<resources>
<string name="app_name">BSCE20029_LAB_9_AR</string>
<string name="city_placeholder">مدينة</string>
<string name="enter_city_hint">أدخل اسم المدينة</string>
<string name="fetch_weather_button">الحصول على الطقس</string>
<string name="weather_condition">حالة الطقس: </string>
<string name="current_temperature">درجة الحرارة الحالية: </string>
<string name="feels_like">يشعر مثل: </string>
<string name="max_temperature">أقصى درجة حرارة: </string>
<string name="min_temperature">أدنى درجة حرارة: </string>
<string name="humidity">الرطوبة: </string>
<string name="visibility">الرؤية: </string>
<string name="wind_speed">سرعة الرياح: </string>
<string name="wind_direction">اتجاه الرياح: </string>
<string name="sunrise_time">وقت الشروق: </string>
<string name="sunset_time">وقت الغروب: </string>
<string name="latitude">خط العرض: </string>
<string name="longitude">خط الطول: </string>
</resources>
resources › string
Running Devices
Pixel 4 API 28
الحصول على الطقس
أدخل اسم المدينة
Mountain View
حالة الطقس:
درجة الحرارة الحالية:
يشعر مثل:
أقصى درجة حرارة:
أدنى درجة حرارة:
الرطوبة:
الرؤية:
سرعة الرياح:
اتجاه الرياح:
وقت الشروق:
وقت الغروب:
or emulator Pixel_4_API_28
s 4947 on device 'Pixel_4_API_28 [emulator-5554]'.
res › values-ar › strings.xml
5:26
CRLF
UTF-8
4 spac

Assessment Rubric for Lab

Performance metric	CLO	Able to complete the task over 80% (4-5)	Able to complete the task 50-80% (2-3)	Able to complete the task below 50% (0-1)	Marks
1. Realization of experiment	1	Executes without errors excellent user prompts, good use of symbols, spacing in output. Through testing has been completed.	Executes without errors, user prompts are understandable, minimum use of symbols or spacing in output. Some testing has been completed.	Does not execute due to syntax errors, runtime errors, user prompts are misleading or non-existent. No testing has been completed.	
2. Conducting experiment	1	Able to make changes and answered all questions.	Partially able to make changes and few incorrect answers.	Unable to make changes and answer all questions.	
3. Computeruse	2	Document submission timely.	Document submission late.	Document submission not done.	
4. Teamwork	3	Actively engages and cooperates with other group member(s) in effective manner.	Cooperates with other group member(s) in a reasonable manner but conduct can be improved.	Distracts or discourages other group members from conducting the experiment	
5. Laboratorysafety and disciplinary rules	3	Code comments are added and does help the reader to understand the code.	Code comments are added and does not help the reader to understand the code.	Code comments are not added.	
6. Data collection	3	Excellent use of white space, creatively organized work, excellent use of variables and constants, correct identifiers for constants, No line-wrap.	Includes name, and assignment, white space makes the program fairly easy to read. Title, organized work, good use of variables.	Poor use of white space (indentation, blank lines) making code hard to read, disorganized and messy.	
7. Data analysis	4	Solution is efficient, easy to understand, and maintain.	A logical solution that is easy to follow but it is not the most efficient.	A difficult and inefficient solution.	
Total (out of 35):					