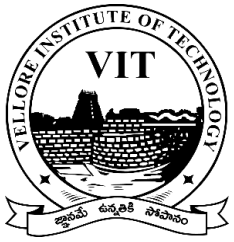


VIT®
AP

School Of Computer Science And Engineering

VIT-AP UNIVERSITY, INAVOLU,

AMARAVATI

CSE4002
MOBILE APPLICATION DEVELOPMENT

A Project Report on

MY WEATHER APP

Under The Guidance Of

Prof. Dr. Hussain Syed

Submitted By

Gudi Varaprasad - Team Lead (19BCE7048)

P.V.S.S.Ramakrishna (19BCE7011)

Nikhilesh Gunnam (19BCE7274)

P Yaswanth Naidu (19BCE7695)

INDEX

1. INTRODUCTION:

No one can imagine how the weather is going to be on a subsequent day. Definitely, the Weather forecast is a big thing that enabled many of us to stay notified about the changes in climatic conditions beforehand. It can be said that it is one of the greatest advancements of all time, mothered by innovative technologies and creative thoughts.

Having up to date information about the weather helps us to make well-read decisions. These weather apps constantly update the forecasts for a day or hour or sometimes for even a minute.

These can be simply termed as the compact weather devices, as they do not only tell about the temperature of that specific region; instead they can describe the accurate time of the sunrise and sunset, the time of the rainfall, humidity levels, etc. The furtherance of Weather Forecasting is Weather App Development.

2. LITERATURE SURVEY:

i. Problem Definition:

The main goal of this project is to develop an android application for checking weather conditions like weather description, weather forecast, temperature of a specific location through android mobile.

ii. Understand the Root Causes:

If we wish to visit a new location, we are still unable to check the weather utilizing our existing application in that location. We need to use search engines such as Google to conduct our inquiry. The current system is not portable (hard to use). We may use this suggested system to examine the weather specifics (i.e., weather forecast report).

iii. Identify the Stakeholder and Users:

Stakeholders - Admin, Developers, Target audience, Funders, Clients.

Users - Common People, Weather and Meteorological Departments.

iv. Identifying the constraints to be imposed on the solution:

a. Customers without Internet access.

b. Information Privacy.

c. Android version 7.0 & above.

3. PROBLEM STATEMENT:

i. Problems of Existing Softwares:

a. Difficult to know the weather forecast of a particular location.

b. Current apps take up a lot of space, need a stable internet connection.

c. These apps run in the background of the system which consumes more battery.

ii. Specifications of the Proposed Software:

a. Simple and Lightweight application.

b. Need for the internet and work fine online.

c. Easy to know complete weather description of any location.

d. Simple Interface and Easy Accessible.

4. OBJECTIVE:

Anyone may use the project to check the weather in their city or area. The app examines the weather and provides information such as the weather description, forecast, and temperature of a given place. It is refreshed every several hours.

Simple User Interface: It offers a simple user interface where you can quickly input a city and obtain the weather and other pertinent information.

Light Application: This is a really light app that will not take up much space on your phone when you install it and doesn't consume more battery.

Easy Accessible: You can quickly find your city by searching the name of any city. It will display the temperature and other meteorological data for all cities across the world.

5. FEASIBILITY STUDY:

The key consideration involved in the feasibility study are:

a. Technical Feasibility :

S.No.	Tools / Technology needed	Description	Type
1	Android Studio	A platform support to build Android apps	Desktop Software

2	XML	UI/UX Design Layout	Front-end Software
3	Java	A programming language for application development	Back-end Software
4	Lucidchart	A web-based proprietary platform to collaborate on drawing charts and diagrams	Design Tool Software
5	Postman	A public weather API used as REST API used to test and build powerful weather apps	API Testing
6	Weather API	Application Programming Interfaces that provide access to current & historical weather data on a global scale	API
7	An Android Phone	For Android SDK setup and application debugging	OS
8	PC / Laptop	Intel based processor-run computer system, which have keyboard and mouse as input devices.	Hardware

b. Behavioral Feasibility :

An assessment of end-user behavior that may have an impact on the system's Encirclement. People are naturally reluctant to change, and mobiles must be cognizant of this in order to support changes.

Since the proposed system's sole purpose is to meet information demands, no person would face a problem as a result of it. Because it supports the organization and its strategic objective, this android app solution is also possible for weather departments.

c. Economic Feasibility :

Type	Quantity	Figures
Human Power	3 to 4 members	$15 - $20 per day per head
Software	-	$25 - $30 approx
Hardware	2 or 3 sets	$50 - $60 approx

Total estimated budget - $1100 - $1400

d. Timeline :

Schedule	SDLC Phase
1st Month	Planning, Communication
2nd Month	Methodology, Design
3rd Month	Development, Coding
4th Month	Testing, Deployment

6.1 FUNCTIONAL REQUIREMENTS:

This section gives a high-level summary of the system requirements. The system will be able to implement a variety of functional modules, including:

6.1.0 Description:

The key consideration involved in functional requirements are the needs that the end user expresses as essential features that the software should provide.

6.1.1 User Module: Smartphone, Internet Connection, Installed App.

6.1.2 Mobile Module: Android version, Hardware requirements, Installed App.

6.1.3 Weather App Module: User Service, Touch Feature, Request Information, Verify Location.

6.1.4 API Module: Request Location, Search Location, Verify Location, Weather Data.

6.2 NON - FUNCTIONAL REQUIREMENTS:

In the insurance to the current features, the following Non-Functional Requirements will be present:

(i) Proper application termination and usage of limited space in memory.

(ii) Availability 24 hours a day, 7 days a week.

(iii) Improved component design to improve peak performance.

(iv) Live weather updates and refresh within a fraction of minutes.

(v) For future expansion, a flexible service-based architecture will be particularly desired.

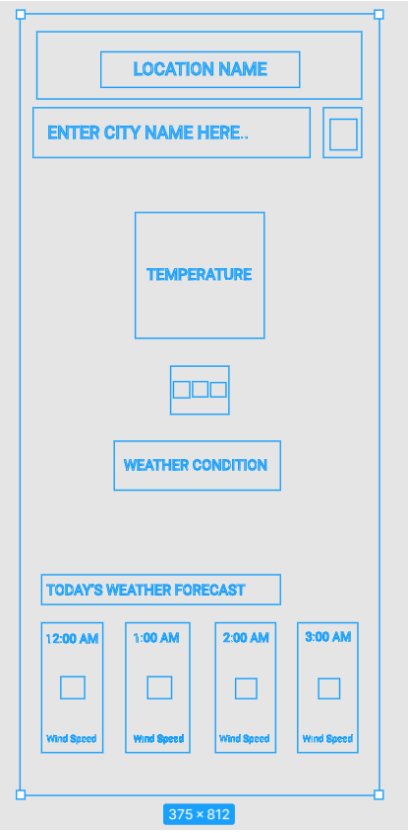
(vi) System properties and limitations are defined by Non-Functional Requirements: **Security, Reliability, Maintainability, Portability, Extensibility, Reusability, Compatibility, and Resource Utilization** are some of the additional non-functional requirements.

7. METHODOLOGY & DESIGN:

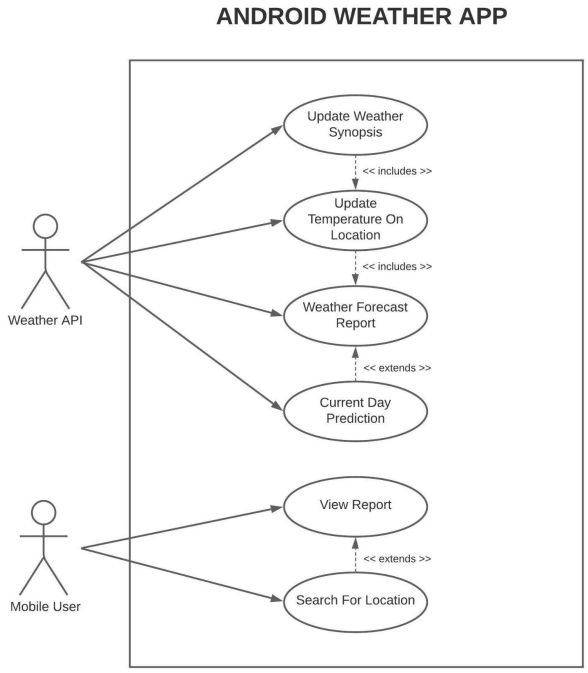
When you launch the weather app, it requests permission to access your current location. If access is denied, the application will display a toast message before terminating and closing. When a user grants current

location access, the weather forecast, temperature, and climatic state of that current place are displayed. There is also a search box where you may look for any location in the world. A toast message displaying "Please Enter City Name" validates the empty entries. If an invalid city is submitted, a toast message with the text "Invalid City name entered" appears. If a valid city name is provided, it will locate the device's coordinates (longitude and latitude). The data will then be sent to the API using an API key. The Weather API (WeatherBit API Keys) will provide us a JSON file from which we will extract the necessary data, which is the weather data, temperature, and city of the location. It displays the current temperature, weather description, climatic conditions, and forecasts the entire day's weather.

7.1 App Layout/Design :

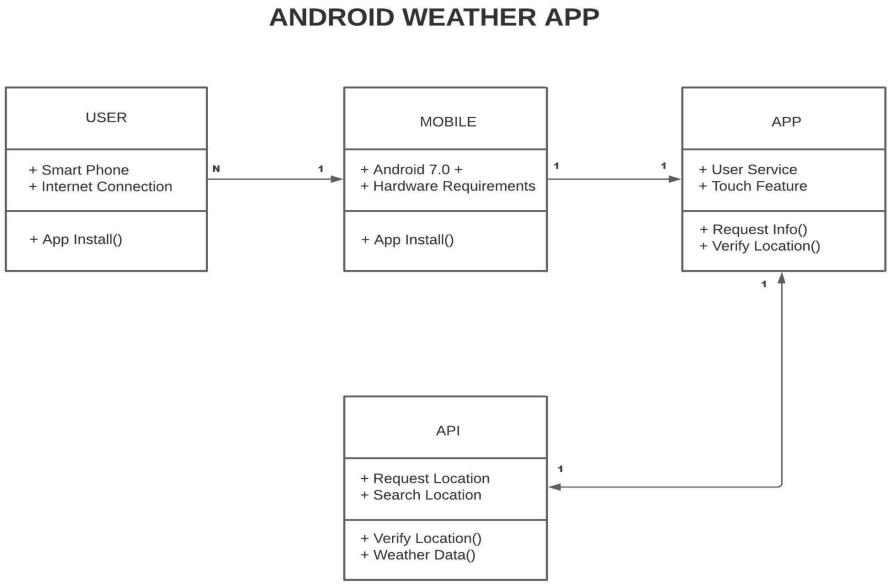


7.2 Use case Diagram :



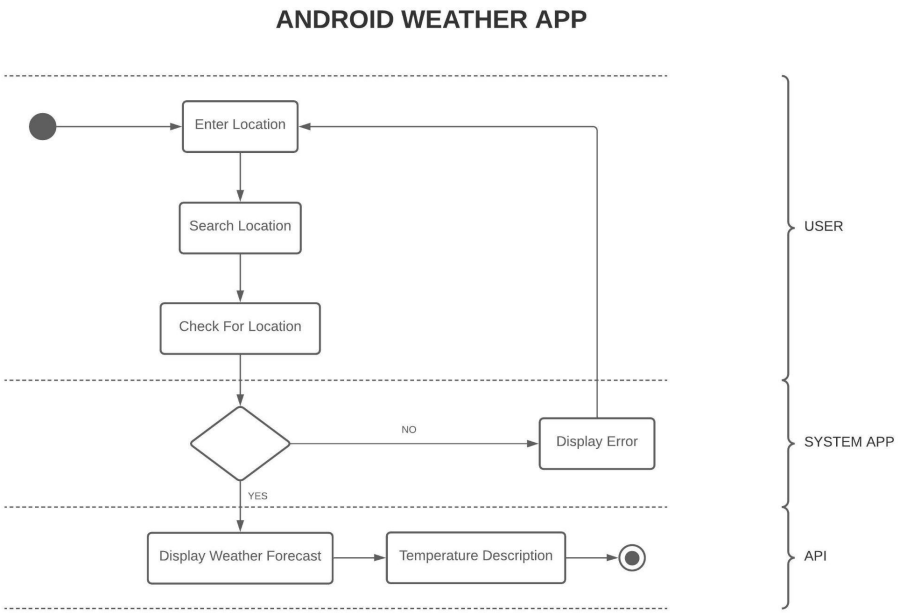
USE CASE DIAGRAM

7.3 Class Diagram :



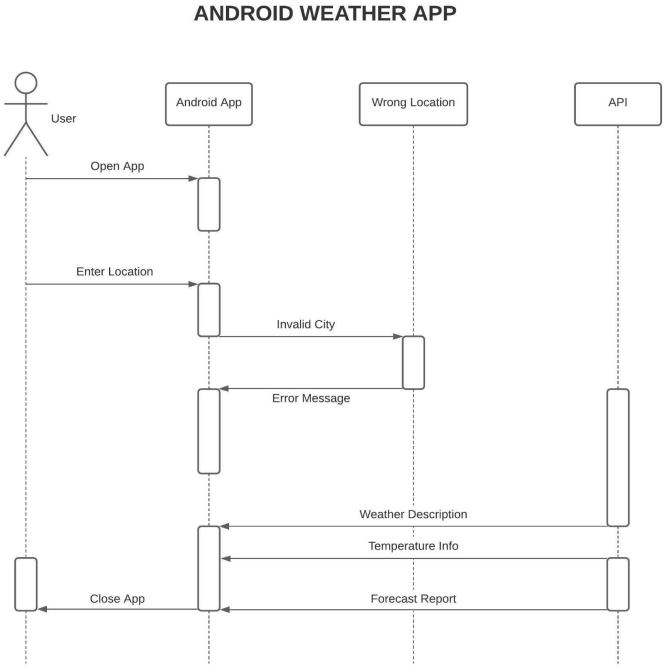
CLASS DIAGRAM

7.4 Activity Diagram :



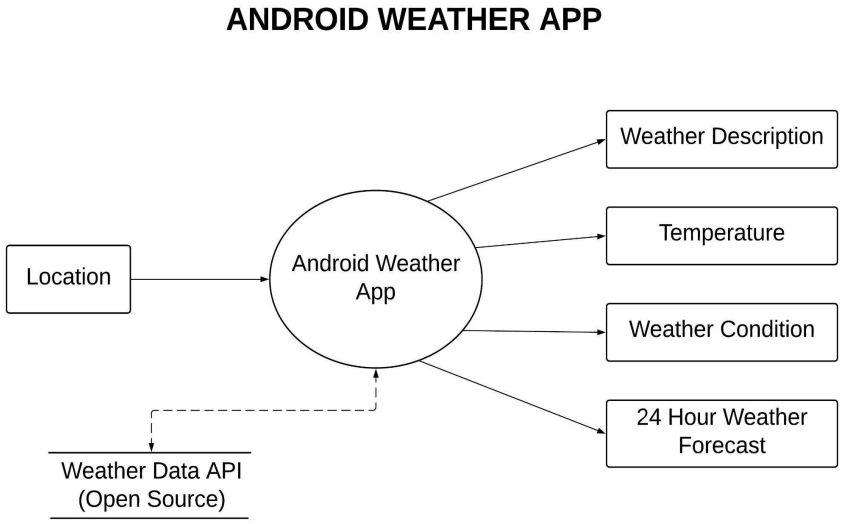
ACTIVITY DIAGRAM

7.5 Sequence Diagram :



SEQUENCE DIAGRAM

7.6 Data Flow Diagram :



Data Flow Diagram

8. CODE SNAPSHOTS:

activity_main.xml :

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/black_shade_1"
    tools:context=".MainActivity">

    <ProgressBar
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/idPBLoading"
        android:visibility="visible"
        android:layout_centerInParent="true"
        />
```

```
    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:id="@+id/idRLHome"
        android:visibility="gone"
        >

        <ImageView
            android:id="@+id/idIVBack"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:scaleType="centerCrop"
            android:src="@color/black_shade_1" />

        <TextView
            android:id="@+id/idTVCityName"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="30dp"
            android:gravity="center"
            android:padding="20dp"
            android:text="City Name"
            android:textAlignment="center"
            android:textColor="@color/white"
            android:textSize="18sp"
            />

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="horizontal"
            android:id="@+id/idLLEdt"
            android:layout_below="@id/idTVCityName"
            android:weightSum="5"
            >
            <com.google.android.material.textfield.TextInputLayout
                android:layout_width="0dp"

style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"
                android:layout_height="wrap_content"
                android:id="@+id/idTILCity"
                android:layout_margin="10dp"
```

```
            android:layout_weight="4.5"
            android:background="@android:color/transparent"
            android:hint="Enter City Name"
            android:padding="5dp"
            app:boxStrokeColor="@android:color/transparent"
            app:hintTextColor="@color/white"
            android:textColorHint="@color/white"
            >
            <com.google.android.material.textfield.TextInputEditText
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:id="@+id/idEdtCity"
                android:importantForAutofill="no"
                android:inputType="text"
                android:singleLine="true"
                android:textColor="@color/white"
                android:textSize="14sp"
                android:background="@android:color/transparent"
                />


        </com.google.android.material.textfield.TextInputLayout>

        <ImageView
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="0.5"
            android:id="@+id/idIVSearch"
            android:layout_margin="10dp"
            android:layout_gravity="center"
            android:background="@android:color/transparent"
            android:src="@drawable/ic_search"
            app:tint="@color/white" />


    </LinearLayout>

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/idTVTemperature"
        android:layout_below="@id/idLLEdt"
        android:layout_margin="10dp"
```

```
        android:gravity="center_horizontal"
        android:padding="5dp"
        android:text="23"
        android:textColor="@color/white"
        android:textSize="70dp"
        />

    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/idIVIcon"
        android:layout_centerHorizontal="true"
        android:layout_below="@id/idTVTemperature"
        android:layout_margin="10dp"
        android:src="@mipmap/ic_launcher"
        />

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/idTVCondition"
        android:layout_margin="10dp"
        android:gravity="center"
        android:textAlignment="center"
        android:text="Condition"
        android:textColor="@color/white"
        android:layout_below="@id/idIVIcon"
        />

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="8dp"
        android:layout_marginBottom="10dp"
        android:text="Today's Weather Forecast"
        android:textColor="@color/white"
        android:layout_above="@id/idRVWeather"
        android:textStyle="bold"
        />

    <androidx.recyclerview.widget.RecyclerView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
```

```
            android:id="@+id/idRVWeather"
            android:layout_alignParentBottom="true"
            android:orientation="horizontal"

app:layoutManager="androidx.recyclerview.widget.LinearLayoutManager"
            />
    </RelativeLayout>
</RelativeLayout>
```

weather_rv_item.xml :

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.cardview.widget.CardView
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="100dp"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_gravity="center"
    android:layout_margin="4dp"
    app:cardElevation="6dp"
    app:cardCornerRadius="10dp"
    android:background="@android:color/transparent"
    android:layout_height="wrap_content">

    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="@drawable/card_back">
        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:id="@+id/idTVTime"
            android:gravity="center"
            android:padding="4dp"
            android:text="time"
            android:textColor="@color/white"
            android:textAlignment="center" />

        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:id="@+id/idTVTemperature"
```

```
            android:gravity="center"
            android:text="20"
            android:textAlignment="center"
            android:textSize="20sp"
            android:layout_below="@id/idTVTime"
            android:textColor="@color/white"/>

        <ImageView
            android:layout_width="40dp"
            android:layout_height="40dp"
            android:layout_below="@id/idTVTemperature"
            android:id="@+id/idIVCondition"
            android:layout_centerHorizontal="true"
            android:layout_margin="5dp"
            android:padding="4dp"
            android:src="@mipmap/ic_launcher"/>

        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:textAlignment="center"
            android:layout_centerHorizontal="true"
            android:gravity="center"
            android:layout_below="@id/idIVCondition"
            android:id="@+id/idTVWindSpeed"
            android:textColor="@color/white"
            android:text="13"
            android:padding="3dp"
            android:layout_margin="4dp"/>
    </RelativeLayout>
</androidx.cardview.widget.CardView>
```

MainActivity.java :

```
package com.example.weatherapp;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;
import androidx.recyclerview.widget.RecyclerView;
```

```
import android.Manifest;
import android.content.Context;
import android.content.pm.PackageManager;
import android.location.Address;
import android.location.Geocoder;
import android.location.Location;
import android.location.LocationManager;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.view.WindowManager;
import android.widget.ImageView;
import android.widget.ProgressBar;
import android.widget.RelativeLayout;
import android.widget.TextView;
import android.widget.Toast;

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 com.google.android.material.textfield.TextInputEditText;
import com.squareup.picasso.Picasso;

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

import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import java.util.Locale;

public class MainActivity extends AppCompatActivity {

    private RelativeLayout homeRL;
    private ProgressBar loadingPB;
    private TextView cityNameTV,temperatureTV,conditionTV;
    private RecyclerView weatherRV;
    private TextInputEditText cityEdt;
    private ImageView backIV,iconIV,searchIV;
```

```
    private ArrayList<WeatherRVModel> weatherRVModelArrayList;
    private WeatherRVAdapter weatherRVAdapter;
    private LocationManager locationManager;
    private int PERMISSION_CODE = 1;
    private String cityName;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);


getWindow().setFlags(WindowManager.LayoutParams.FLAG_LAYOUT_NO_LIMITS,Windo
wManager.LayoutParams.FLAG_LAYOUT_NO_LIMITS);

        setContentView(R.layout.activity_main);
        homeRL = findViewById(R.id.idRLHome);
        loadingPB = findViewById(R.id.idPBLoading);
        cityNameTV = findViewById(R.id.idTVCityName);
        temperatureTV = findViewById(R.id.idTVTemperature);
        conditionTV = findViewById(R.id.idTVCondition);
        weatherRV = findViewById(R.id.idRVWeather);
        cityEdt = findViewById(R.id.idEdtCity);
        backIV = findViewById(R.id.idIVBack);
        iconIV = findViewById(R.id.idIVIcon);
        searchIV = findViewById(R.id.idIVSearch);
        weatherRVModelArrayList = new ArrayList<>();
        weatherRVAdapter = new
WeatherRVAdapter(this,weatherRVModelArrayList);
        weatherRV.setAdapter(weatherRVAdapter);

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

if(ActivityCompat.checkSelfPermission(this,Manifest.permission.ACCESS_FINE_
LOCATION)!= PackageManager.PERMISSION_GRANTED &&
ActivityCompat.checkSelfPermission(this,Manifest.permission.ACCESS_COARSE_L
OCATION)!= PackageManager.PERMISSION_GRANTED){
            ActivityCompat.requestPermissions(MainActivity.this,new
String[]{Manifest.permission.ACCESS_FINE_LOCATION,Manifest.permission.ACCES
S_COARSE_LOCATION},PERMISSION_CODE);
        }

        Location location =
```

```
locationManager.getLastKnownLocation(LocationManager.NETWORK_PROVIDER);
        //cityName =
getCityName(location.getLongitude(),location.getLatitude());
        getWeatherInfo(cityName);

        searchIV.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                String city = cityEdt.getText().toString();
                if(city.isEmpty()){
                    Toast.makeText(MainActivity.this, "Please Enter City
Name", Toast.LENGTH_SHORT).show();
                } else {
                    cityNameTV.setText(cityName);
                    getWeatherInfo(city);
                }
            }
        });
    }

    @Override
    public void onRequestPermissionsResult(int requestCode, @NonNull
String[] permissions, @NonNull int[] grantResults) {
        super.onRequestPermissionsResult(requestCode, permissions,
grantResults);
        if(requestCode == PERMISSION_CODE){
            if(grantResults.length>0 &&
grantResults[0]==PackageManager.PERMISSION_GRANTED){
                Toast.makeText(this, "Permissions Granted..",
Toast.LENGTH_SHORT).show();
            } else {
                Toast.makeText(this, "Please Provide The Permissions..",
Toast.LENGTH_SHORT).show();
                finish();
            }
        }
    }

    private String getCityName(double longitude, double latitude){
        String cityName = "Not Found";
        Geocoder gcd = new Geocoder(getBaseContext(), Locale.getDefault());
        try {
            List<Address> addresses =
```

```
gcd.getFromLocation(latitude,longitude,10);

            for(Address adr : addresses){
                if(adr!=null){
                    String city = adr.getLocality();
                    if(city!=null && !city.equals("")){
                        cityName = city;
                    } else {
                        Log.d("TAG","CITY NOT FOUND");
                        Toast.makeText(this, "User City Not Found..",
Toast.LENGTH_SHORT).show();
                    }
                }
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
        return cityName;
    }

    private void getWeatherInfo(String cityName){
        String url =
"http://api.weatherapi.com/v1/forecast.json?key=8ab1b2b8e75541eeb1763412210
309&q=" + cityName + "&days=1&aqi=yes&alerts=yes";

        cityNameTV.setText(cityName);
        RequestQueue requestQueue =
Volley.newRequestQueue(MainActivity.this);

        JsonObjectRequest jsonObjectRequest = new
JsonObjectRequest(Request.Method.GET, url, null, new
Response.Listener<JSONObject>() {
            @Override
            public void onResponse(JSONObject response) {
                loadingPB.setVisibility(View.GONE);
                homeRL.setVisibility(View.VISIBLE);
                weatherRVModelArrayList.clear();

                try {
                    String temperature =
response.getJSONObject("current").getString("temp_c");
                    temperatureTV.setText(temperature+"°c");
                    int isDay =
```

```
response.getJSONObject("current").getInt("is_day");
                    String condition =
response.getJSONObject("current").getJSONObject("condition").getString("tex
t");
                    String conditionIcon =
response.getJSONObject("current").getJSONObject("condition").getString("ico
n");

Picasso.get().load("http:".concat(conditionIcon)).into(iconIV);
                    conditionTV.setText(condition);
                    if(isDay==1){
                        //morning

Picasso.get().load("https://images.unsplash.com/photo-1566228015668-4c45dbc
4e2f5?ixid=MnwxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8&ixlib=rb-1.2.1&auto
=format&fit=crop&w=334&q=80").into(backIV);
                    } else{
                        //night

Picasso.get().load("https://images.unsplash.com/photo-1532074534361-bb09a38
cf917?ixid=MnwxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8&ixlib=rb-1.2.1&auto
=format&fit=crop&w=334&q=80").into(backIV);
                    }

                    JSONObject forecastObj =
response.getJSONObject("forecast");
                    JSONObject forcastO =
forecastObj.getJSONArray("forecastday").getJSONObject(0);
                    JSONArray hourArray = forcastO.getJSONArray("hour");

                    for(int i=0;i<hourArray.length();i++){
                        JSONObject hourObj = hourArray.getJSONObject(i);
                        String time = hourObj.getString("time");
                        String temper = hourObj.getString("temp_c");
                        String img =
hourObj.getJSONObject("condition").getString("icon");
                        String wind = hourObj.getString("wind_kph");
                        weatherRVModelArrayList.add(new
WeatherRVModel(time,temper,img,wind));
                    }

                    weatherRVAdapter.notifyDataSetChanged();
```

```
                } catch (JSONException e) {
                    e.printStackTrace();
                }
            }
        }, new Response.ErrorListener() {
            @Override
            public void onErrorResponse(VolleyError error) {
                Toast.makeText(MainActivity.this, "Please Enter Valid City
Name..", Toast.LENGTH_SHORT).show();
            }
        });

        requestQueue.add(jsonObjectRequest);
    }
}
```

WeatherRVAdapter.java :

```
package com.example.weatherapp;

import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;

import com.squareup.picasso.Picasso;

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;

public class WeatherRVAdapter extends
RecyclerView.Adapter<WeatherRVAdapter.ViewHolder> {
    private Context context;
    private ArrayList<WeatherRVModel> weatherRVModelArrayList;
```

```
    public WeatherRVAdapter(Context context, ArrayList<WeatherRVModel>
weatherRVModelArrayList) {
        this.context = context;
        this.weatherRVModelArrayList = weatherRVModelArrayList;
    }

    @NonNull
    @Override
    public WeatherRVAdapter.ViewHolder onCreateViewHolder(@NonNull ViewGroup
parent, int viewType) {
        View view =
LayoutInflater.from(context).inflate(R.layout.weather_rv_item,parent,false)
;
        return new ViewHolder(view);
    }

    @Override
    public void onBindViewHolder(@NonNull WeatherRVAdapter.ViewHolder
holder, int position) {

        WeatherRVModel model = weatherRVModelArrayList.get(position);
        holder.temperatureTV.setText(model.getTemperature()+"°c");

Picasso.get().load("http:".concat(model.getIcon())).into(holder.conditionIV
);
        holder.windTV.setText(model.getWindSpeed()+"Km/h");
        SimpleDateFormat input = new SimpleDateFormat("yyyy-MM-dd hh:mm");
        SimpleDateFormat output = new SimpleDateFormat("hh:mm aa");
        try {
            Date t = input.parse(model.getTime());
            holder.timeTV.setText(output.format(t));
        } catch (ParseException e) {
            e.printStackTrace();
        }


    }

    @Override
    public int getItemCount() {
        return weatherRVModelArrayList.size();
    }
```

```
    public class ViewHolder extends RecyclerView.ViewHolder {
        private TextView windTV,temperatureTV,timeTV;
        private ImageView conditionIV;

        public ViewHolder(@NonNull View itemView) {
            super(itemView);
            windTV = itemView.findViewById(R.id.idTVWindSpeed);
            temperatureTV = itemView.findViewById(R.id.idTVTemperature);
            timeTV = itemView.findViewById(R.id.idTVTime);
            conditionIV = itemView.findViewById(R.id.idIVCondition);

        }
    }
}
```

WeatherRVModel.java :

```
package com.example.weatherapp;

public class WeatherRVModel {

    private String time;
    private String temperature;
    private String icon;
    private String windSpeed;

    public WeatherRVModel(String time, String temperature, String icon,
String windSpeed) {
        this.time = time;
        this.temperature = temperature;
        this.icon = icon;
        this.windSpeed = windSpeed;
    }

    public String getTime() {
        return time;
    }

    public void setTime(String time) {
        this.time = time;
```

```
    }

    public String getTemperature() {
        return temperature;
    }

    public void setTemperature(String temperature) {
        this.temperature = temperature;
    }

    public String getIcon() {
        return icon;
    }

    public void setIcon(String icon) {
        this.icon = icon;
    }

    public String getWindSpeed() {
        return windSpeed;
    }

    public void setWindSpeed(String windSpeed) {
        this.windSpeed = windSpeed;
    }
}
```

AndroidManifest.xml :

```
<?xml version="1.0" encoding="utf-8"?>

<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.weatherapp">

    <uses-permission android:name="android.permission.INTERNET"/>

    <uses-permission
android:name="android.permission.ACCESS_NETWORK_STATE"/>

    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
```

```
   <uses-permission
android:name="android.permission.ACCESS_COARSE_LOCATION"/>

   <uses-permission
android:name="android.permission.ACCESS_FINE_LOCATION"/>

   <application
       android:usesCleartextTraffic="true"
       android:allowBackup="true"
       android:icon="@drawable/weatherlogo"
       android:label="@string/app_name"
       android:roundIcon="@drawable/weatherlogo"
       android:supportsRtl="true"
       android:theme="@style/Theme.WeatherApp">

       <activity
           android:name=".MainActivity"
           android:exported="true">

           <intent-filter>
               <action android:name="android.intent.action.MAIN" />
               <category android:name="android.intent.category.LAUNCHER" />
           </intent-filter>

       </activity>

   </application>

</manifest>
```

Complete Application Demo and Source code can be found HERE.

9. APP TESTING AND DEBUGGING:

Since we are outlined to use the **Incremental Model** of Software Engineering, Testing and Development of the modules is done in parallel. After coding the Project, testing of the software begins with the **Unit Testing** method.

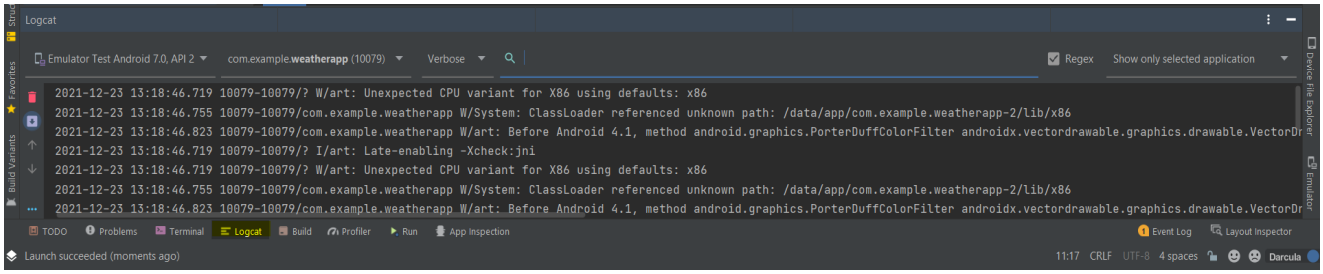


Fig : Debugging and checking for errors in Logcat window in Android Studio

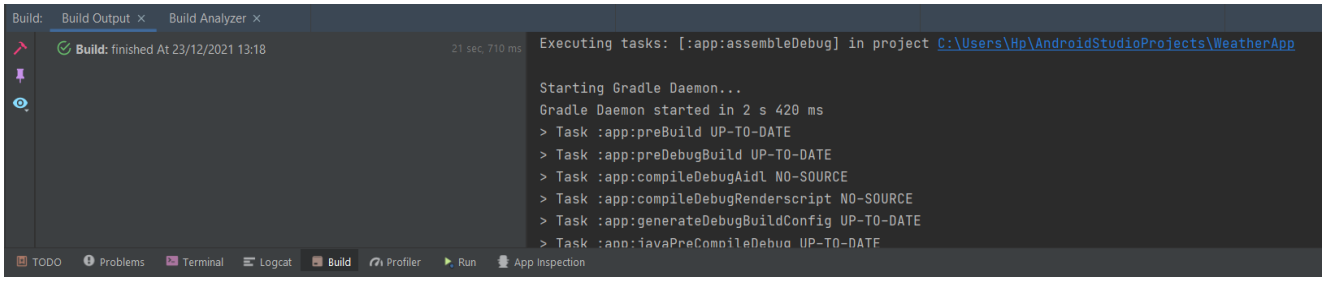


Fig : Debugging through Build Output Panel in Android Studio

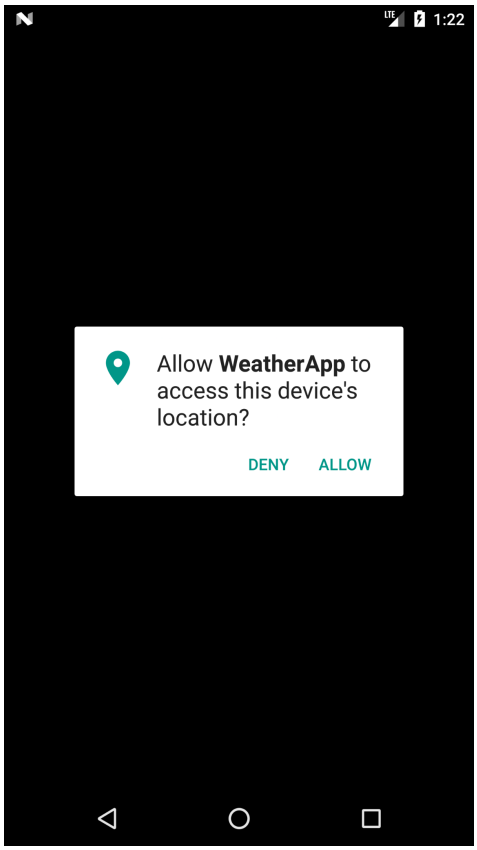


Fig : App requesting for permissions of user location

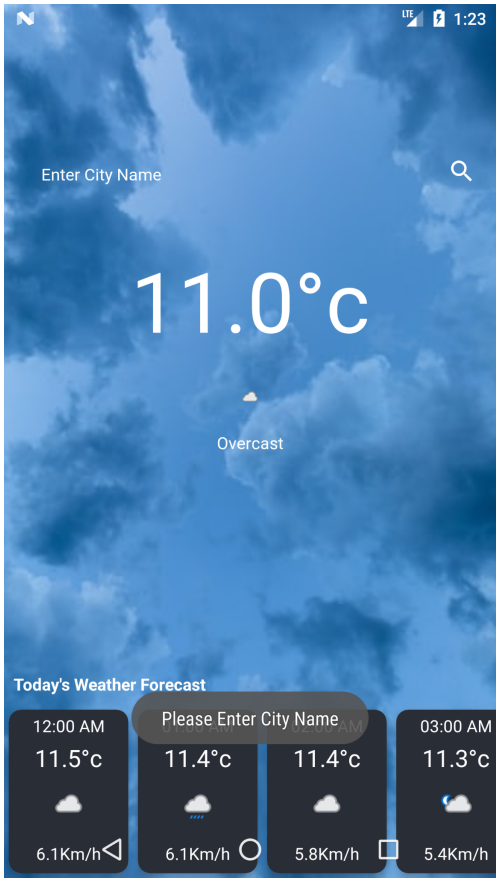


Fig : Empty User input field validation

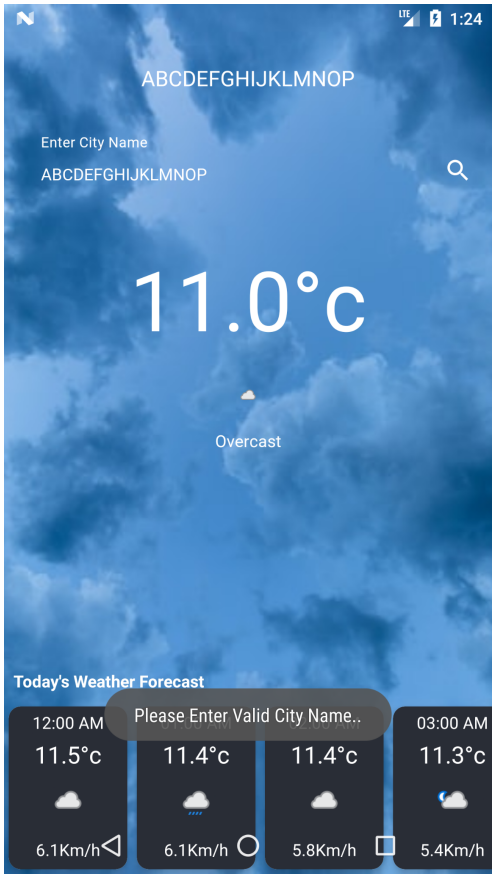


Fig : Validating user input search on invalid location

10. RESULTS DISCUSSION:

Below are the output screenshots showcasing working of the app based on user controls.

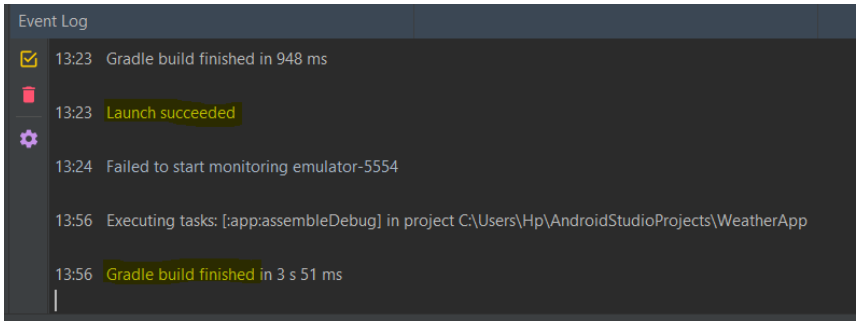


Fig : Clicking Make Project in Android Build options

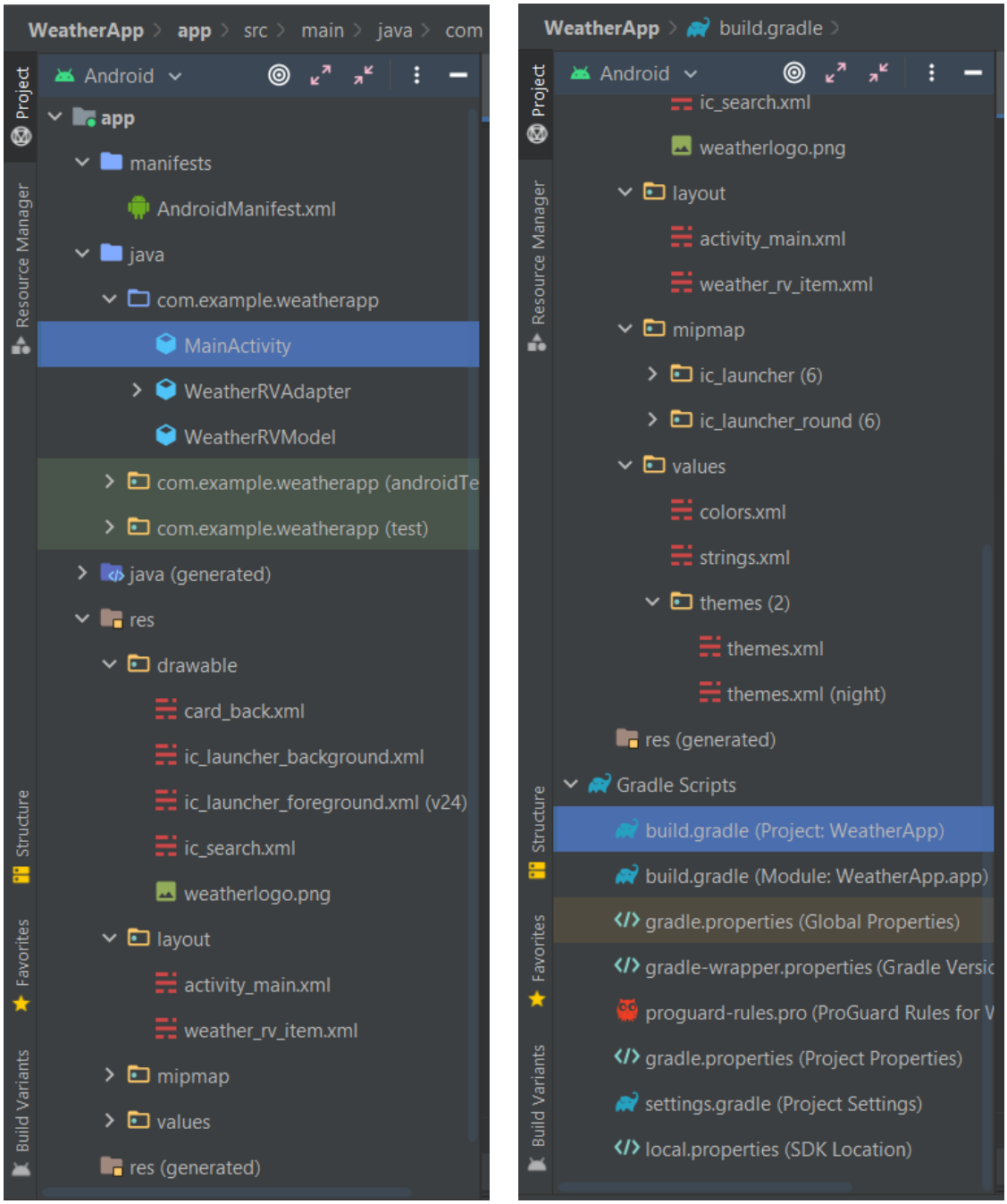


Fig : Application Directory/File hierarchy

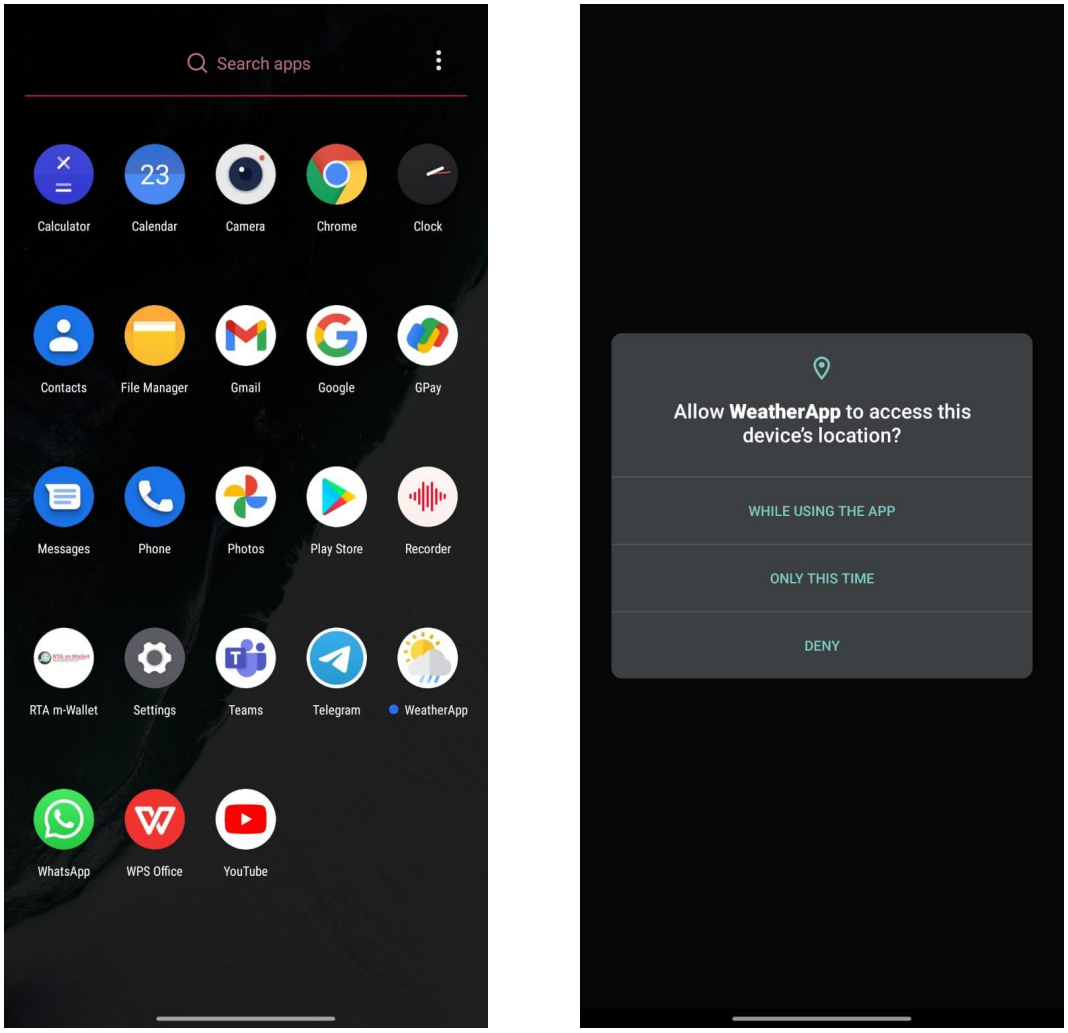


Fig : Application Logo in the Phone Menu and Application first screen on launch

Fig : Weather Description at Places - Delhi and Russia

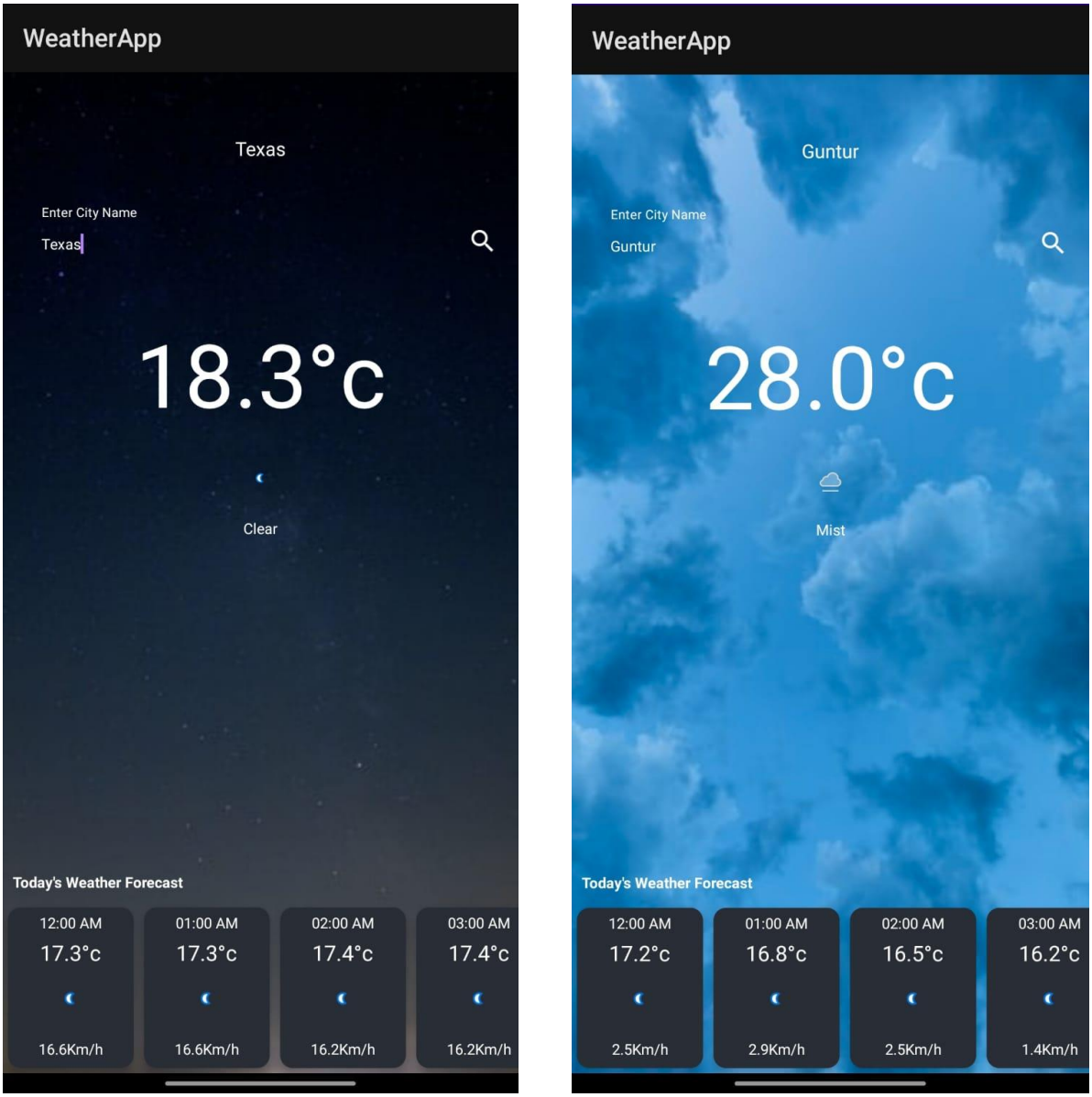


Fig : Weather Report at Places - Texas and Guntur

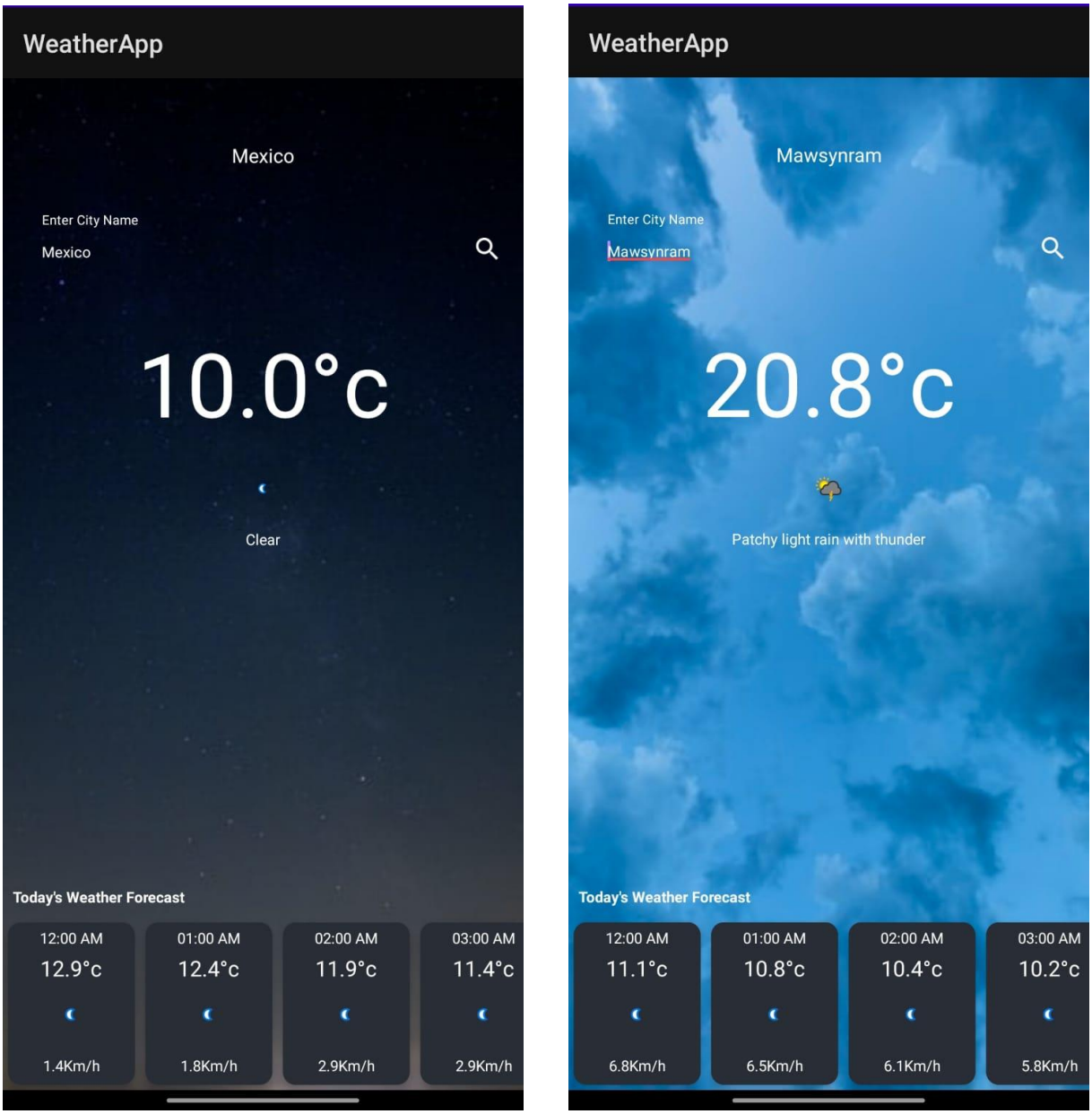


Fig : Weather Forecast at Places - Mexico and Mawsynram (Meghalaya)

Above screenshots displays weather conditions like weather description, temperature of a location, weather forecast of various cities across the globe.

11. FUTURE SCOPE:

This app helps to predict the climatic conditions beforehand and which has a major advantage to farmers to plan for which type of crop to grow. Not only that advancement in this app will also help them to analyze soil moisture and other crop details.

You never want nature to spoil your fun/mood when you have a plan for a Family/Friends Trip. Exactly, at that time this app is a lifeline for you to know your weather conditions beforehand of any particular place that you are planning to.

Similarly, like Agriculture, this app also helps fishermen to predict the situation in coastal regions as fishing is the livelihood of farmers and weather is their life savior.

Not limited to these applications, there are many practical scenarios where such apps are often used by the weather department, Meteorological Departments and many others.

12. REFERENCES:

[1] Textbook - Android for programmers, an app-driven approach developer series by Paul Deitel,Harvey Deitel, Abbey Deitel,Michael.

[2] Abhi Android Tutorials.

[3] Weather Apps UML Diagram.

[4] Open Weather Map - for free Weather data.

[5] GeeksforGeeks - Android Applications to APK.