ANNAMALAI UNIVERSITY

(Accredited with 'A+' Grade by NAAC)

FACULTY OF ENGINEERING AND TECHNOLOGY

Department of Computer Science and Engineering

Annamalai Nagar – 608 002, TAMIL NADU

The Project based on building the weather forecasting application

From the students of Bachelor in Computer Science and Engineering Students

(Pre-final Year)

DESIGNED BY:

1.2036010031	JANARTHANAN H
2.2036010048	NITHISH R
3.2036010034	JEYA PRAKASH S
4.2036010004	ARUNESWARAN A B
5.2036010026	HARISH U R
6.2036010056	RATHNAGOPAL J
7.2036010057	RAVEENDIRAN R

Submitted to

Dr. K.T. Meena Abarna

PROJECT DOCUMENTATION

Project Title: Creating a weather forecasting mobile application using android studio.

Team Members:

1.2036010031	JANARTHANAN H
2.2036010048	NITHISH R
3.2036010034	JEYA PRAKASH S
4.2036010004	ARUNESWARAN A B
5.2036010026	HARISH U R
6.2036010056	RATHNAGOPAL J
7.2036010057	RAVEENDIRAN R

Overview

In this project, we will be building a weather application. This application will show the temperature of a location. To fetch weather information we will need an **API.** An API(Application Programming Interface) is a function that allows applications to interact and share data using various components and microservices. For this project, we will be using **Weather Bit API** for fetching weather data. Weather Bit API provides a fast and elegant way to fetch weather data. Note that we are going to implement this project using the **Java** language.

Theory

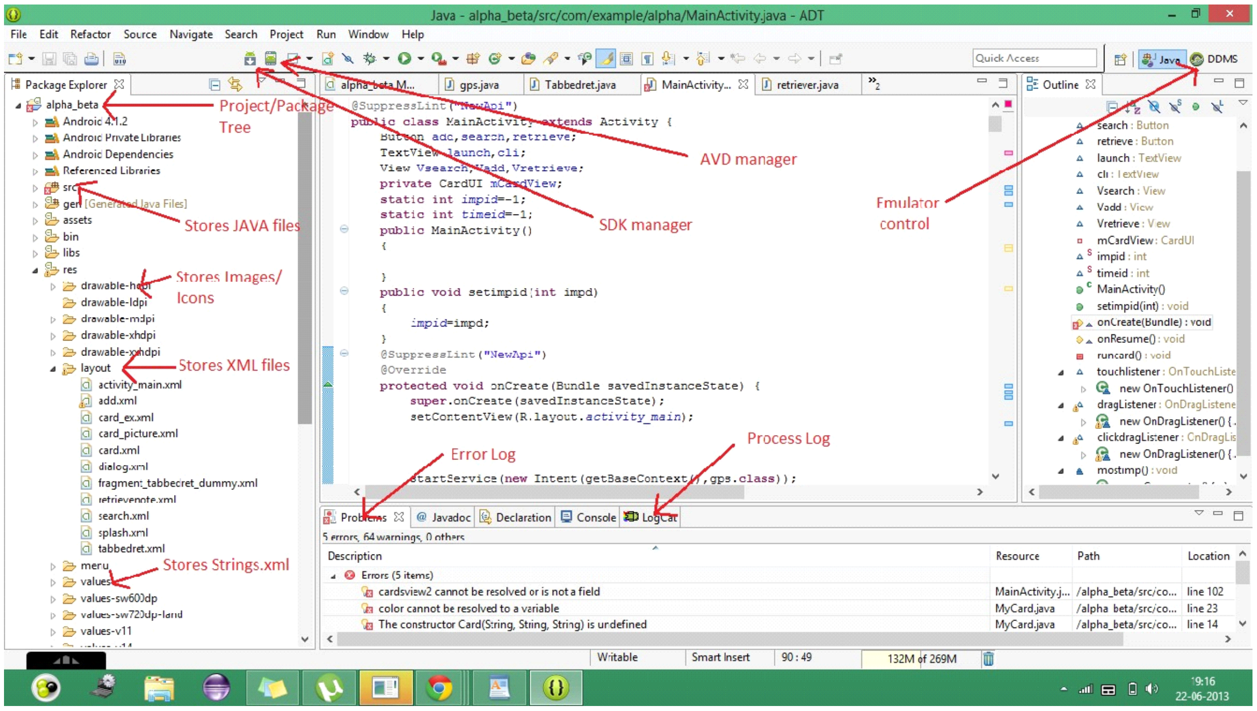
Android is a Linux-based operating system designed primarily for touchscreen mobile devices such as smartphones and tablet computers. Initially developed by Android, Inc., which Google backed financially and later bought in 2005.
Android is open source and Google releases the code under the Apache License. This open source code and permissive licensing allows the software to be freely modified and distributed by device manufacturers, wireless carriers and enthusiast developers. Additionally, Android has a large community of developers writing

applications ("apps") that extend the functionality of devices, written primarily in a customized version of the Java programming language.

Android software development is the process by which new applications are created for the Android operating system. Applications are usually developed in the Java programming language using the Android Software Development Kit.

ADT (Android Development Tools) is the software used to develop android apps. It basically encases Eclipse IDE,which is a multi-language Integrated development HYPERLINK "http://en.wikipedia.org/wiki/Integrated_development_environment" HYPERLINK "http://en.wikipedia.org/wiki/Integrated_development_environment"environment (IDE) comprising a base workspace and an extensible plug-in system for customizing the environment.. The latest version comes with ADT plugin preinstalled and bundled to the IDE.

This is how the IDE looks like with the important elements marked.



```
@SuppressLint("NewApi")
public class MainActivity extends Activity {
    Button add,search,retrieve;
    TextView launch,cli;
    View Vsearch,Vadd,Vretrieve;
    private CardUI mCardView;
    static int impid=-1;
    static int timeid=-1;
    public MainActivity()
    {

    }
    public void setimpid(int impd)
    {
        impid=impd;
    }
    @SuppressLint("NewApi")
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        startService(new Intent(getBaseContext(),gps.class));
```

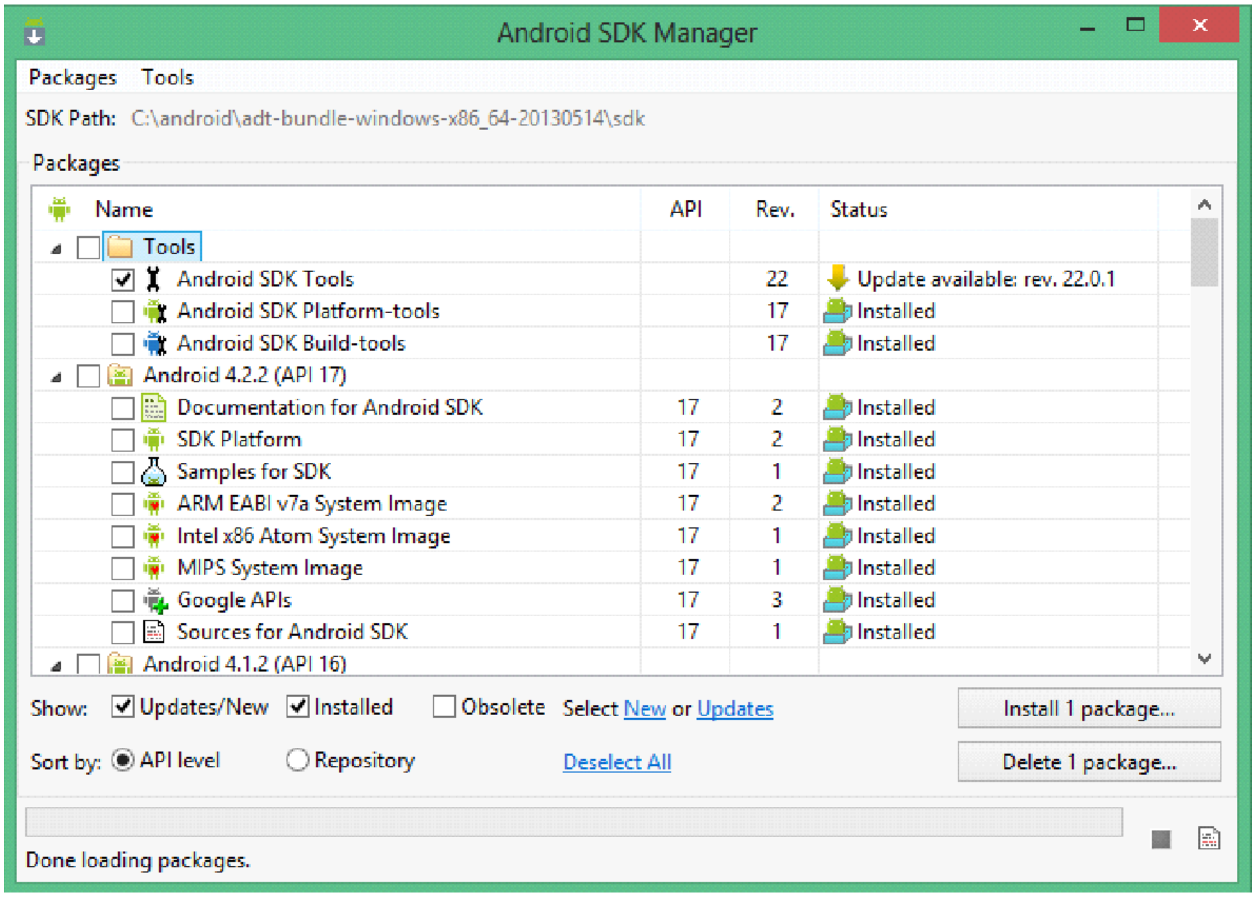
5 errors, 64 warnings, 0 others

Description	Resource	Path	Location
Errors (5 items)			
cardsview2 cannot be resolved or is not a field	MainActivity.j...	/alpha_beta/src/co...	line 102
color cannot be resolved to a variable	MyCard.java	/alpha_beta/src/co...	line 23
The constructor Card(String, String, String) is undefined	MyCard.java	/alpha_beta/src/co...	line 14

Application programming interface (API) specifies how some software components should interact with each other.
In practice in most of the cases an API is a library that usually includes specification for routines, data structures, object classes, and variables.
An API specification can take many forms, including an International Standard such as POSIX, vendor documentation such as the Microsoft Windows API, the libraries of a programming language, e.g., Standard Template Library in C++ or Java API.

Google APIs can be downloaded from Google Code, Google's site for developer tools, APIs and technical resources. The Google Data API] allows programmers to create applications that read and write data from Google services.
Currently, these include APIs for Google HYPERLINK "http://en.wikipedia.org/wiki/Google_Apps" HYPERLINK "http://en.wikipedia.org/wiki/Google_Apps"Apps, Google HYPERLINK "http://en.wikipedia.org/wiki/Google_Analytics" HYPERLINK "http://en.wikipedia.org/wiki/Google_Analytics"Analytics, Blogger, Google Base, Google Book Search, Google Calendar, Google Code

Search, Google HYPERLINK "http://en.wikipedia.org/wiki/Google_Earth" HYPERLINK "http://en.wikipedia.org/wiki/Google_Earth"Earth, Google HYPERLINK "http://en.wikipedia.org/wiki/Google_Spreadsheets" HYPERLINK "http://en.wikipedia.org/wiki/Google_Spreadsheets"Spreadsheets, Google HYPERLINK "http://en.wikipedia.org/wiki/Google_Notebook" HYPERLINK "http://en.wikipedia.org/wiki/Google_Notebook"Notebook, and Picasa Web Albums.



Android SDK Manager

Packages Tools

SDK Path: C:\android\adt-bundle-windows-x86_64-20130514\sdk

Packages

Name	API	Rev.	Status
☐ Tools			
✓ Android SDK Tools		22	Update available: rev. 22.0.1
☐ Android SDK Platform-tools		17	Installed
☐ Android SDK Build-tools		17	Installed
☐ Android 4.2.2 (API 17)			
☐ Documentation for Android SDK	17	2	Installed
☐ SDK Platform	17	2	Installed
☐ Samples for SDK	17	1	Installed
☐ ARM EABI v7a System Image	17	2	Installed
☐ Intel x86 Atom System Image	17	1	Installed
☐ MIPS System Image	17	1	Installed
☐ Google APIs	17	3	Installed
☐ Sources for Android SDK	17	1	Installed
☐ Android 4.1.2 (API 16)			

Show: ✓ Updates/New ✓ Installed ☐ Obsolete Select New or Updates

Install 1 package...

Sort by: [✓ API level] Repository Deselect All

Delete 1 package...

Done loading packages.

SDK (Software Development KIt or "devkit") is typically a set of software development tools that allows for the creation of applications for a certain software package,software framework, hardware platform, computer system, video game console, operating system, or similar development platform.
It may be something as simple as an application programming interface (API) in the form of some files to interface to a particular programming language or include sophisticated hardware to communicate with a certain embedded system. Common tools include debugging aids and other utilities often presented in an integrated development environment (IDE).
In the latest version of ADT, the android SDK adds on to the IDE automatically as soon as you unzip and load the IDE.

SDK Manager enables us to download Google APIs and use them in our code.

Android Virtual Device (AVD) manager enables us to launch virtual android devices/ emulators in our PC and run the app in the emulator,
and at the same time we can track and debug each app activity from the Logcat in our IDE.

Java Development Kit (JDK): Since, Android applications require Java programming for its backend programming; it needs a JAVA environment to support its functions, executions and syntax.

Xml: The front end design of the application involves xml statements for the Relative layouts, Radio buttons, Radio Group, buttons, text boxes and text views.

System Requirements

1. Smartphone with Android OS version 4.4 (Kit kat) or higher.
2. Minimum 512 MB of RAM.
3. A processor with speeds above 1.2 GHz (any make).
4. 16 MB of storage for the app and extra for the data stored, the size of the app increases as the number of entries are increased.
5. Android API version 19.
6. Permission to install applications over USB and installation from unknown sources from 'Developer Options'.

Our Approach

We approached this project to learn about the android development environment not merely to sneak the code from tutorials and copy-paste to build something. With this in mind, first we developed a lot of apps which implemented different aspects of android ability and which are totally unrelated to our project.

The work of our final app can be divided into the following phases:

- Step 1: Create a New Project
- Step 2: Working with the Androidmainfest.xml file
- Step 3: Working with the HomeActivity.java file
- Step 4: Working with SplashScreen.java file

- Step 5: Working with DaysAdapter.java file
- Step 6: Working with CityFinder.java file and Locationcord.java file
- Step 7: Working with InternetConnectivity.java file
- Step 8: Working with Toaster.java file
- Step 9: Working with UpdateUI.java file
- Step 10: Working with URL.java file

Source code for creating a new project

Working with the AndroidMainFest.xml file

Navigate to the **app** > **src** > **main** > **AndroidMainFest.xml** and add the below code to that file. Below is the code for the **AndroidMainFest.xml** file.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    package="com.aniketjain.weatherapp">

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

    <application
        android:allowBackup="true"
        android:hardwareAccelerated="true"
        android:icon="@mipmap/ic_main"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_main_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.WeatherApp">
        <activity
            android:name=".SplashScreen"
            android:exported="true"
            android:screenOrientation="portrait"
            android:windowSoftInputMode="stateAlwaysHidden"
            tools:ignore="LockedOrientationActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name=".HomeActivity"
            android:exported="false"
            android:screenOrientation="portrait"
            android:windowSoftInputMode="stateAlwaysHidden"
            tools:ignore="LockedOrientationActivity" />
    </application>

</manifest>
```

Working with the HomeActivity.java file

Go to HomeActivity.java Class. Below is the code for the HomeActivity.java file. Comments are added inside the code to understand the code in more detail.

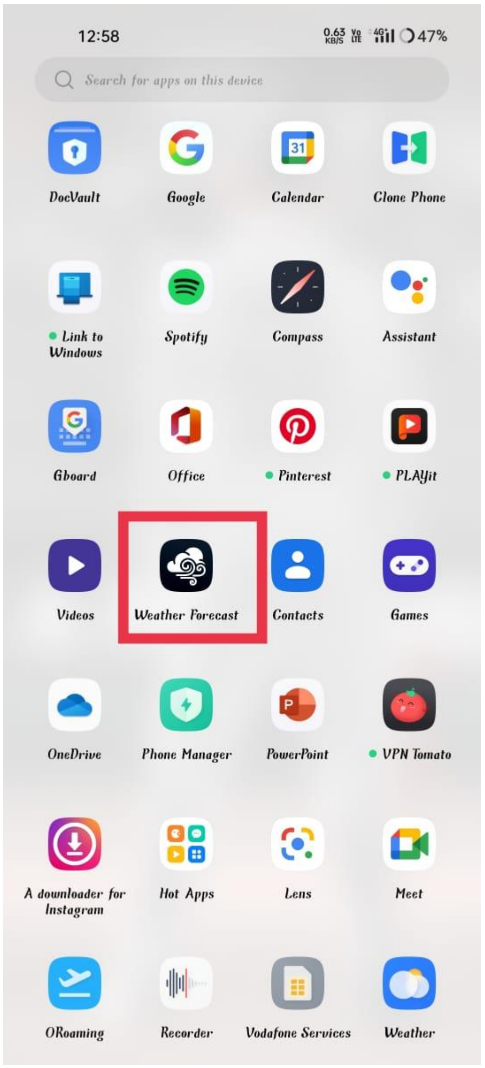
Java source code:

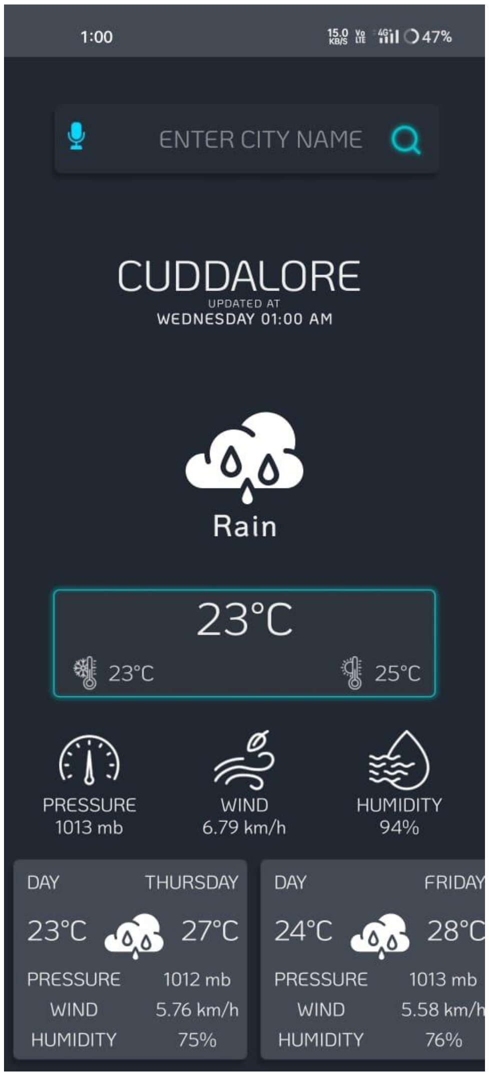
```
package com.aniketjain.weatherapp;

import static
com.aniketjain.weatherapp.location.CityFinder.getCityNameUsingNetwork;
import static com.aniketjain.weatherapp.location.CityFinder.setLongitudeLatitude;
import static
com.aniketjain.weatherapp.network.InternetConnectivity.isInternetConnected;

import android.Manifest;
import android.annotation.SuppressLint;
import android.app.Activity;
import android.content.Intent;
import android.content.IntentSender;
import android.content.pm.PackageManager;
import android.os.Build;
import android.os.Bundle;
import android.speech.RecognizerIntent;
import android.util.Log;
import android.view.View;
import android.view.inputmethod.EditorInfo;
import android.view.inputmethod.InputMethodManager;
```

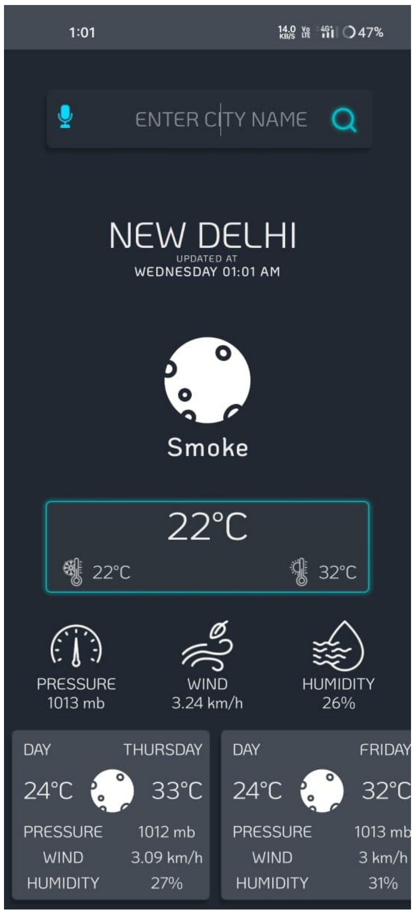
```java
import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;
import androidx.recyclerview.widget.LinearLayoutManager;

import com.android.volley.Request;
import com.android.volley.RequestQueue;
import com.android.volley.toolbox.JsonObjectRequest;
import com.android.volley.toolbox.Volley;
import com.aniketjain.weatherapp.adapter.DaysAdapter;
import com.aniketjain.weatherapp.databinding.ActivityHomeBinding;
import com.aniketjain.weatherapp.location.LocationCord;
import com.aniketjain.weatherapp.toast.Toaster;
import com.aniketjain.weatherapp.update.UpdateUI;
import com.aniketjain.weatherapp.url.URL;
import com.google.android.gms.location.FusedLocationProviderClient;
import com.google.android.gms.location.LocationServices;
import com.google.android.play.core.appupdate.AppUpdateInfo;
import com.google.android.play.core.appupdate.AppUpdateManager;
import com.google.android.play.core.appupdate.AppUpdateManagerFactory;
import com.google.android.play.core.install.model.AppUpdateType;
import com.google.android.play.core.install.model.UpdateAvailability;
import com.google.android.play.core.tasks.Task;

import org.json.JSONException;

import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.Locale;
import java.util.Objects;

public class HomeActivity extends AppCompatActivity {

    private final int WEATHER_FORECAST_APP_UPDATE_REQ_CODE = 101;   // for app
update
    private static final int PERMISSION_CODE = 1;                   // for user
location permission
    private String name, updated_at, description, temperature, min_temperature,
max_temperature, pressure, wind_speed, humidity;
    private int condition;
    private long update_time, sunset, sunrise;
    private String city = "";
    private final int REQUEST_CODE_EXTRA_INPUT = 101;
    private ActivityHomeBinding binding;

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

        // binding
        binding = ActivityHomeBinding.inflate(getLayoutInflater());
        View view = binding.getRoot();
        setContentView(view);

        // set navigation bar color
        setNavigationBarColor();

        //check for new app update
        checkUpdate();

        // set refresh color schemes
        setRefreshLayoutColor();

        // when user do search and refresh
        listeners();

        // getting data using internet connection
        getDataUsingNetwork();

    }


    @Override
    protected void onActivityResult(int requestCode, int resultCode, @Nullable
Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        if (requestCode == REQUEST_CODE_EXTRA_INPUT){
            if(resultCode == RESULT_OK && data!=null){
                ArrayList<String> arrayList =
data.getStringArrayListExtra(RecognizerIntent.EXTRA_RESULTS);
                binding.layout.cityEt.setText(Objects.requireNonNull(arrayList).g
et(0).toUpperCase());
                searchCity(binding.layout.cityEt.getText().toString());
            }
        }
    }
    private void setNavigationBarColor() {
        if (android.os.Build.VERSION.SDK_INT >= Build.VERSION_CODES.LOLLIPOP) {
            getWindow().setNavigationBarColor(getResources().getColor(R.color.nav
BarColor));
        }
    }

    private void setUpDaysRecyclerView() {
        DaysAdapter daysAdapter = new DaysAdapter(this);
        binding.dayRv.setLayoutManager(
                new LinearLayoutManager(this, LinearLayoutManager.HORIZONTAL,
false)
        );
        binding.dayRv.setAdapter(daysAdapter);
    }

    @SuppressLint("ClickableViewAccessibility")
    private void listeners() {
        binding.layout.mainLayout.setOnTouchListener((view, motionEvent) -> {
            hideKeyboard(view);
            return false;
        });
        binding.layout.searchBarIv.setOnClickListener(view ->
searchCity(binding.layout.cityEt.getText().toString()));
        binding.layout.searchBarIv.setOnTouchListener((view, motionEvent) -> {
            hideKeyboard(view);
            return false;
        });
        binding.layout.cityEt.setOnEditorActionListener((textView, i, keyEvent) -
> {
            if (i == EditorInfo.IME_ACTION_GO) {
                searchCity(binding.layout.cityEt.getText().toString());
                hideKeyboard(textView);
                return true;
            }
            return false;
        });
        binding.layout.cityEt.setOnFocusChangeListener((view, b) -> {
            if (!b) {
                hideKeyboard(view);
            }
        });
        binding.mainRefreshLayout.setOnRefreshListener(() -> {
            checkConnection();
            Log.i("refresh", "Refresh Done.");
            binding.mainRefreshLayout.setRefreshing(false);  //for the next time
        });
        //Mic Search
        binding.layout.micSearchId.setOnClickListener(new View.OnClickListener()
{
            @Override
            public void onClick(View view) {
                Intent intent = new
Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);
                intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL,RecognizerI
ntent.LANGUAGE_MODEL_FREE_FORM);
                intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL,
Locale.getDefault());
                intent.putExtra(RecognizerIntent.EXTRA_PROMPT,REQUEST_CODE_EXTRA_
INPUT);
                try {
                    //it was deprecated but still work
                    startActivityForResult(intent,REQUEST_CODE_EXTRA_INPUT);
                }catch (Exception e){
                    Log.d("Error Voice", "Mic Error:  "+e);
                }
            }
        });
    }

    private void setRefreshLayoutColor() {
        binding.mainRefreshLayout.setProgressBackgroundColorSchemeColor(
                getResources().getColor(R.color.textColor)
        );
        binding.mainRefreshLayout.setColorSchemeColors(
                getResources().getColor(R.color.navBarColor)
        );
    }

    private void searchCity(String cityName) {
        if (cityName == null || cityName.isEmpty()) {
            Toaster.errorToast(this, "Please enter the city name");
        } else {
            setLatitudeLongitudeUsingCity(cityName);
        }
    }

    private void getDataUsingNetwork() {
        FusedLocationProviderClient client =
LocationServices.getFusedLocationProviderClient(this);
```

```
        //check permission
        if (ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED &&
ActivityCompat.checkSelfPermission(this,
                Manifest.permission.ACCESS_COARSE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {
            ActivityCompat.requestPermissions(this, new
String[]{Manifest.permission.ACCESS_FINE_LOCATION,
Manifest.permission.ACCESS_COARSE_LOCATION}, PERMISSION_CODE);
        } else {
            client.getLastLocation().addOnSuccessListener(location -> {
                setLongitudeLatitude(location);
                city = getCityNameUsingNetwork(this, location);
                getTodayWeatherInfo(city);
            });
        }
    }

    private void setLatitudeLongitudeUsingCity(String cityName) {
        URL.setCity_url(cityName);
        RequestQueue requestQueue = Volley.newRequestQueue(HomeActivity.this);
        JsonObjectRequest jsonObjectRequest = new
JsonObjectRequest(Request.Method.GET, URL.getCity_url(), null, response -> {
            try {
                LocationCord.lat =
response.getJSONObject("coord").getString("lat");
                LocationCord.lon =
response.getJSONObject("coord").getString("lon");
                getTodayWeatherInfo(cityName);
                // After the successfully city search the cityEt(editText) is
Empty.
                binding.layout.cityEt.setText("");
            } catch (JSONException e) {
                e.printStackTrace();
            }
        }, error -> Toaster.errorToast(this, "Please enter the correct city
name"));
        requestQueue.add(jsonObjectRequest);
    }

    @SuppressLint("DefaultLocale")
    private void getTodayWeatherInfo(String name) {
        URL url = new URL();
        RequestQueue requestQueue = Volley.newRequestQueue(this);
        JsonObjectRequest jsonObjectRequest = new
JsonObjectRequest(Request.Method.GET, url.getLink(), null, response -> {
            try {
                this.name = name;
                update_time = response.getJSONObject("current").getLong("dt");
                updated_at = new SimpleDateFormat("EEEE hh:mm a",
Locale.ENGLISH).format(new Date(update_time * 1000));

                condition =
response.getJSONArray("daily").getJSONObject(0).getJSONArray("weather").getJSONOb
ject(0).getInt("id");
                sunrise =
response.getJSONArray("daily").getJSONObject(0).getLong("sunrise");
                sunset =
response.getJSONArray("daily").getJSONObject(0).getLong("sunset");
                description =
response.getJSONObject("current").getJSONArray("weather").getJSONObject(0).getStr
ing("main");

                temperature =
String.valueOf(Math.round(response.getJSONObject("current").getDouble("temp") -
273.15));
                min_temperature = String.format("%.0f",
response.getJSONArray("daily").getJSONObject(0).getJSONObject("temp").getDouble("
min") - 273.15);
                max_temperature = String.format("%.0f",
response.getJSONArray("daily").getJSONObject(0).getJSONObject("temp").getDouble("
max") - 273.15);
                pressure =
response.getJSONArray("daily").getJSONObject(0).getString("pressure");
                wind_speed =
response.getJSONArray("daily").getJSONObject(0).getString("wind_speed");
                humidity =
response.getJSONArray("daily").getJSONObject(0).getString("humidity");

                updateUI();
                hideProgressBar();
                setUpDaysRecyclerView();
            } catch (JSONException e) {
                e.printStackTrace();
            }
        }, null);
        requestQueue.add(jsonObjectRequest);
        Log.i("json_req", "Day 0");
    }
    @SuppressLint("SetTextI18n")
    private void updateUI() {
        binding.layout.nameTv.setText(name);
        updated_at = translate(updated_at);
        binding.layout.updatedAtTv.setText(updated_at);
        binding.layout.conditionIv.setImageResource(
                getResources().getIdentifier(
                        UpdateUI.getIconID(condition, update_time, sunrise,
sunset),
                        "drawable",
                        getPackageName()
                ));
        binding.layout.conditionDescTv.setText(description);
        binding.layout.tempTv.setText(temperature + "°C");
        binding.layout.minTempTv.setText(min_temperature + "°C");
        binding.layout.maxTempTv.setText(max_temperature + "°C");
        binding.layout.pressureTv.setText(pressure + " mb");
        binding.layout.windTv.setText(wind_speed + " km/h");
        binding.layout.humidityTv.setText(humidity + "%");
    }

    private String translate(String dayToTranslate) {
        String[] dayToTranslateSplit = dayToTranslate.split(" ");
        dayToTranslateSplit[0] =
UpdateUI.TranslateDay(dayToTranslateSplit[0].trim(), getApplicationContext());
        return dayToTranslateSplit[0].concat(" " + dayToTranslateSplit[1]);
    }

    private void hideProgressBar() {
        binding.progress.setVisibility(View.GONE);
        binding.layout.mainLayout.setVisibility(View.VISIBLE);
    }

    private void hideMainLayout() {
        binding.progress.setVisibility(View.VISIBLE);
        binding.layout.mainLayout.setVisibility(View.GONE);
    }

    private void hideKeyboard(View view) {
        InputMethodManager inputMethodManager = (InputMethodManager)
view.getContext().getSystemService(Activity.INPUT_METHOD_SERVICE);
        inputMethodManager.hideSoftInputFromWindow(view.getWindowToken(), 0);
    }
    private void checkConnection() {
        if (!isInternetConnected(this)) {
            hideMainLayout();
            Toaster.errorToast(this, "Please check your internet connection");
        } else {
            hideProgressBar();
            getDataUsingNetwork();
        }
    }

    @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) {
                Toaster.successToast(this, "Permission Granted");
                getDataUsingNetwork();
            } else {
                Toaster.errorToast(this, "Permission Denied");
                finish();
            }
        }
    }

    @Override
    protected void onResume() {
        super.onResume();
        checkConnection();
    }

    private void checkUpdate() {
        AppUpdateManager appUpdateManager =
AppUpdateManagerFactory.create(HomeActivity.this);
        Task<AppUpdateInfo> appUpdateInfoTask =
appUpdateManager.getAppUpdateInfo();
        appUpdateInfoTask.addOnSuccessListener(appUpdateInfo -> {
            if (appUpdateInfo.updateAvailability() ==
UpdateAvailability.UPDATE_AVAILABLE
                    &&
appUpdateInfo.isUpdateTypeAllowed(AppUpdateType.IMMEDIATE)) {
                try {
                    appUpdateManager.startUpdateFlowForResult(appUpdateInfo,
AppUpdateType.IMMEDIATE, HomeActivity.this,
WEATHER_FORECAST_APP_UPDATE_REQ_CODE);
                } catch (IntentSender.SendIntentException exception) {
                    Toaster.errorToast(this, "Update Failed");
                }
            }
        });
    }
```

Output:
The Output of our weather application are below:
Note: *Before running the application make sure the location in the device is turned on and the application has access to that.*







1:01 14.0 KB/S 47%

ENTER CITY NAME

NEW DELHI
UPDATED AT
WEDNESDAY 01:01 AM

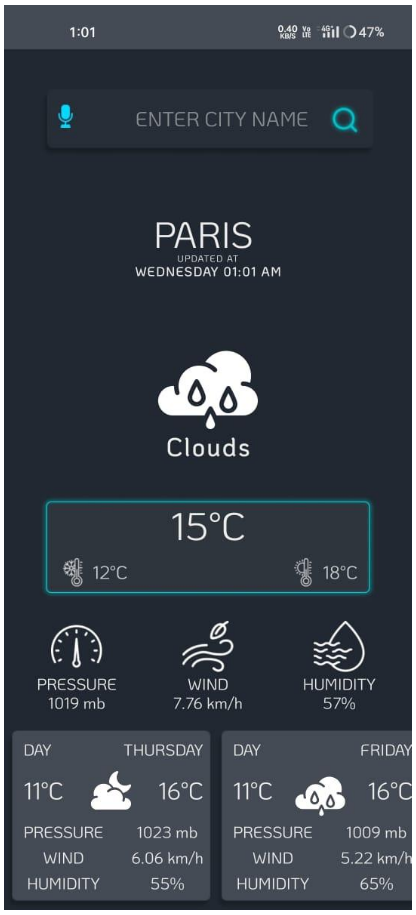
Smoke

22°C
22°C 32°C

PRESSURE 1013 mb
WIND 3.24 km/h
HUMIDITY 26%

DAY THURSDAY
24°C 33°C
PRESSURE 1012 mb
WIND 3.09 km/h
HUMIDITY 27%

DAY FRIDAY
24°C 32°C
PRESSURE 1013 mb
WIND 3 km/h
HUMIDITY 31%



1:01 0.40 KB/S 47%

ENTER CITY NAME

PARIS
UPDATED AT
WEDNESDAY 01:01 AM

Clouds

15°C
12°C 18°C

PRESSURE 1019 mb
WIND 7.76 km/h
HUMIDITY 57%

DAY THURSDAY
11°C 16°C
PRESSURE 1023 mb
WIND 6.06 km/h
HUMIDITY 55%

DAY FRIDAY
11°C 16°C
PRESSURE 1009 mb
WIND 5.22 km/h
HUMIDITY 65%

Conclusion:
In this study, we propose the design the simple weather forecasting application. The location tracking subsystem was successfully build-in Accordance with objectives, and the system will be expanded in the Accordance with goal of weather tracking.
Here by, we declare that this weather forecasting application will be great helpful application to identify the weather in the various location.