**DAILY WEATHER APP**

*Dissertation submitted in fulfilment of the requirements for the Degree of*

**BACHELOR OF TECHNOLOGY**

**in**

**COMPUTER SCIENCE AND ENGINEERING**

By

**ANUPAM SINGH**

**12108342**

CSE 225

**ANDROID APP DEVELOPMENT**



**School of Computer Science and Engineering**

Lovely Professional University

Phagwara, Punjab (India)

Month : April / Year :2024

**TABLE OF CONTENTS**

**Sl. No CONTENTS PAGE NO.**

1. Introduction 2

2. Modules or Activity Explanation & Screenshots 4

i) MainActivity.kt / activity\_main.xml

ii) FirstFragment.kt / fragment\_first.xml

iii) SecondFragment.kt / fragment\_second.xml

iv) SplashActivity.kt / activity\_splash.xml

3. Code - Module wise or Activity Wise 26

i) MainActivity.kt

ii) FirstFragment.kt

iii) SecondFragment.kt

iv) SplashActivity.kt

4. Emulator Screenshots 42

5. Conclusion & Future Scope 43

6. Project Github link 44

**INTRODUCTION**

In a world where weather patterns constantly influence our daily lives, having accurate and up-to-date weather information at your fingertips is indispensable. Welcome to "Daily Weather," your go-to application for real-time weather updates tailored to your preferences.

With its sleek design and user-friendly interface, Daily Weather offers a seamless experience for users of all levels. Whether you're planning a weekend getaway, scheduling outdoor activities, or simply staying informed about the weather in your area, Daily Weather empowers you with the information you need, when you need it.

**Key Features:**

**Real-time Updates:** Daily Weather leverages advanced weather APIs to provide you with the most current weather data for any location worldwide. Stay ahead of the curve with minute-by-minute updates on temperature, humidity, wind speed, and more.

**Customizable Locations:** Whether you're curious about the weather in your hometown or halfway across the globe, Daily Weather allows you to add and save multiple locations for easy access. Plan your travels with confidence, knowing you have accurate weather

forecasts for every destination.

**Intuitive Design:** Designed with simplicity in mind, Daily Weather offers a clean and intuitive interface that prioritizes ease of use. Navigate effortlessly between locations, view detailed weather forecasts, and stay informed with at-a-glance weather summaries.

**Interactive Maps:** Visualize weather patterns in real-time with interactive maps that provide a comprehensive overview of current weather conditions. Zoom in on specific areas, track storms, and plan your route with confidence using Daily Weather's dynamic map features.

**Personalized Alerts:** Never be caught off guard by sudden changes in the weather again. Daily Weather allows you to set personalized alerts for your preferred weather conditions, ensuring you stay informed and prepared for whatever nature has in store.

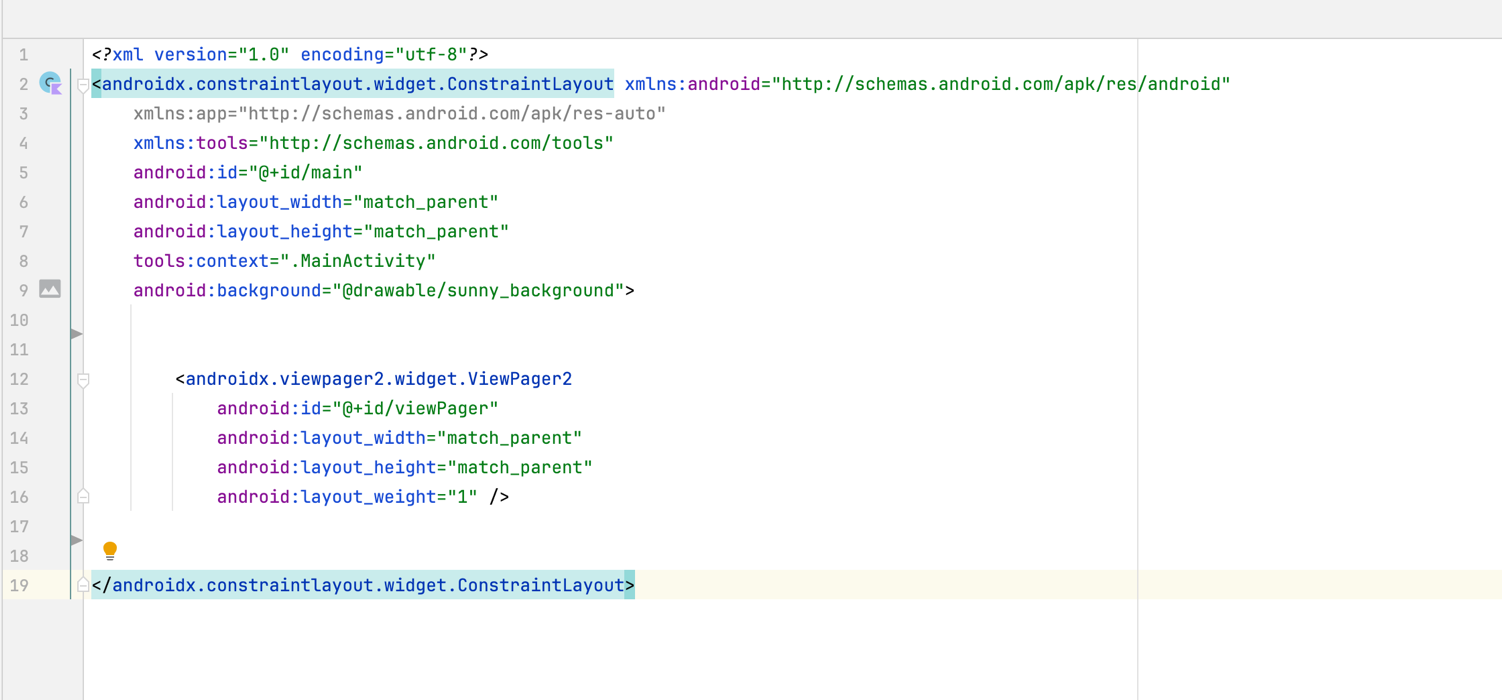
**Modules or Activity Explanation & Screenshots :**

MainActivity.kt



This Android code defines the main activity for a weather app with swipeable views. It manages two fragments (likely for current and forecast weather) and uses a ViewPager to allow users to switch between them but doesn't include the logic for fetching and displaying weather data or handling user interaction with potential UI elements like search bars or buttons.

Activity\_main.xml :

****

The provided code snippet defines the main layout (activity\_main.xml) for the weather application. This layout utilizes a ConstraintLayout for flexible element positioning and sets a background image. The key element is a ViewPager2 component, which occupies the entire screen and allows users to swipe between different views (likely fragments displaying weather information). The inclusion android:layout\_weight="1" on the ViewPager2 ensures it fills all available space within the ConstraintLayout, making it the dominant visual element on the screen.

SplashActivity.kt :

A screenshot of a computer program

Description automatically generated

The SplashActivity class manages the initial splash screen that appears when you launch the weather application. It utilizes the enableEdgeToEdge function to potentially make the content extend to the entire screen. The layout for this screen is defined in a separate file (activity\_splash.xml). After the screen displays for 3 seconds, a handler triggers the transition to the main application screen (MainActivity) using an Intent. Finally, the SplashActivity itself is closed to ensure a smooth handover to the main functionality.

Activity\_splash.xml

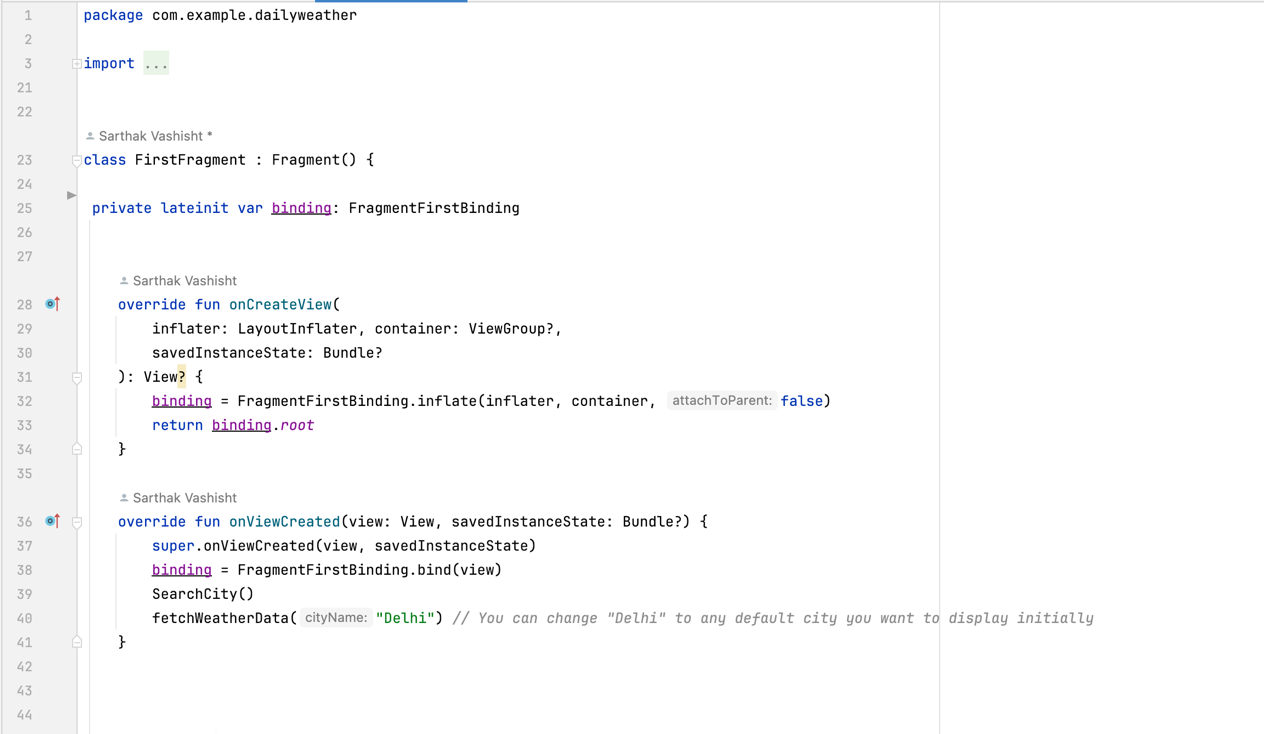


The provided code snippet defines the layout file (activity\_splash.xml) for the splash screen of the weather application. Here's a concise summary:

* ConstraintLayout: The layout utilizes a ConstraintLayout for flexible element positioning.
* Background Image: A background image depicting a weather-related scene is set using the android:background attribute.
* Cloud Image: An ImageView displays a white cloud image positioned in the center of the top screen section.
* Main Text:A TextView element with centered text ("Weather Matters, \n We've Got the Details") informs users about the app's purpose and is formatted with a specific font and size.
* Credit Text: Another TextView element displays the designer's credit ("DESIGNED BY \n Sarthak Vashisht") positioned at the bottom and formatted with a different font and white text color.

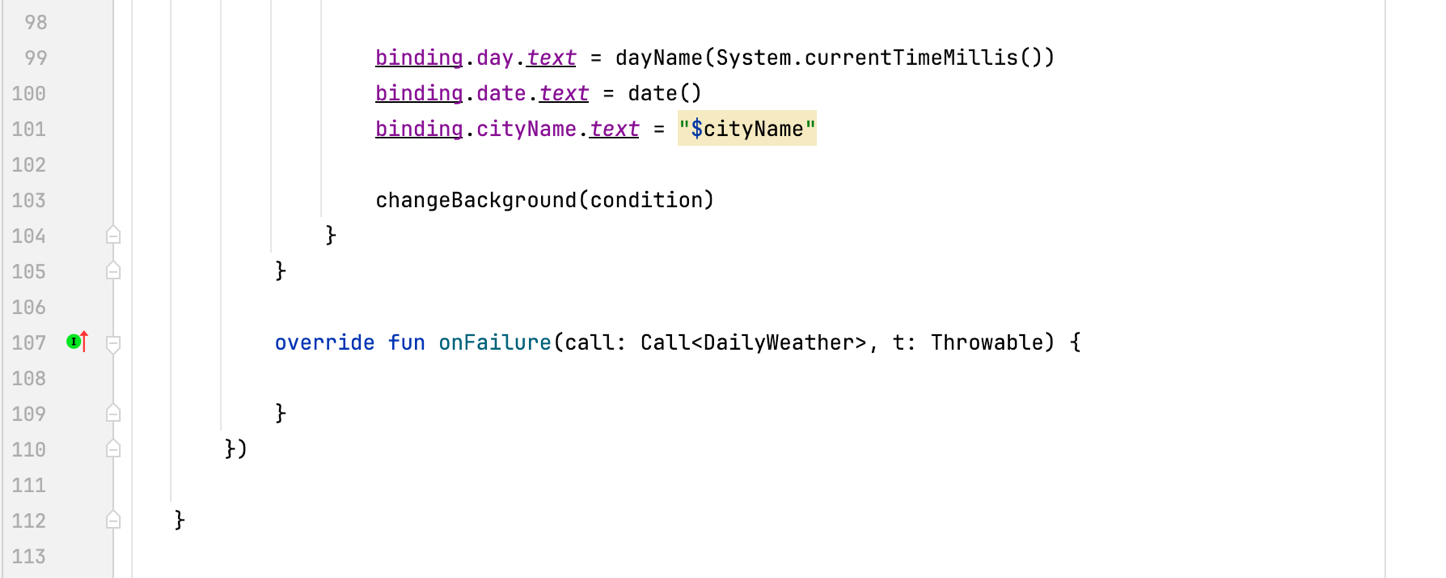
This layout creates a visually appealing and informative splash screen that users see before entering the main weather app interface.

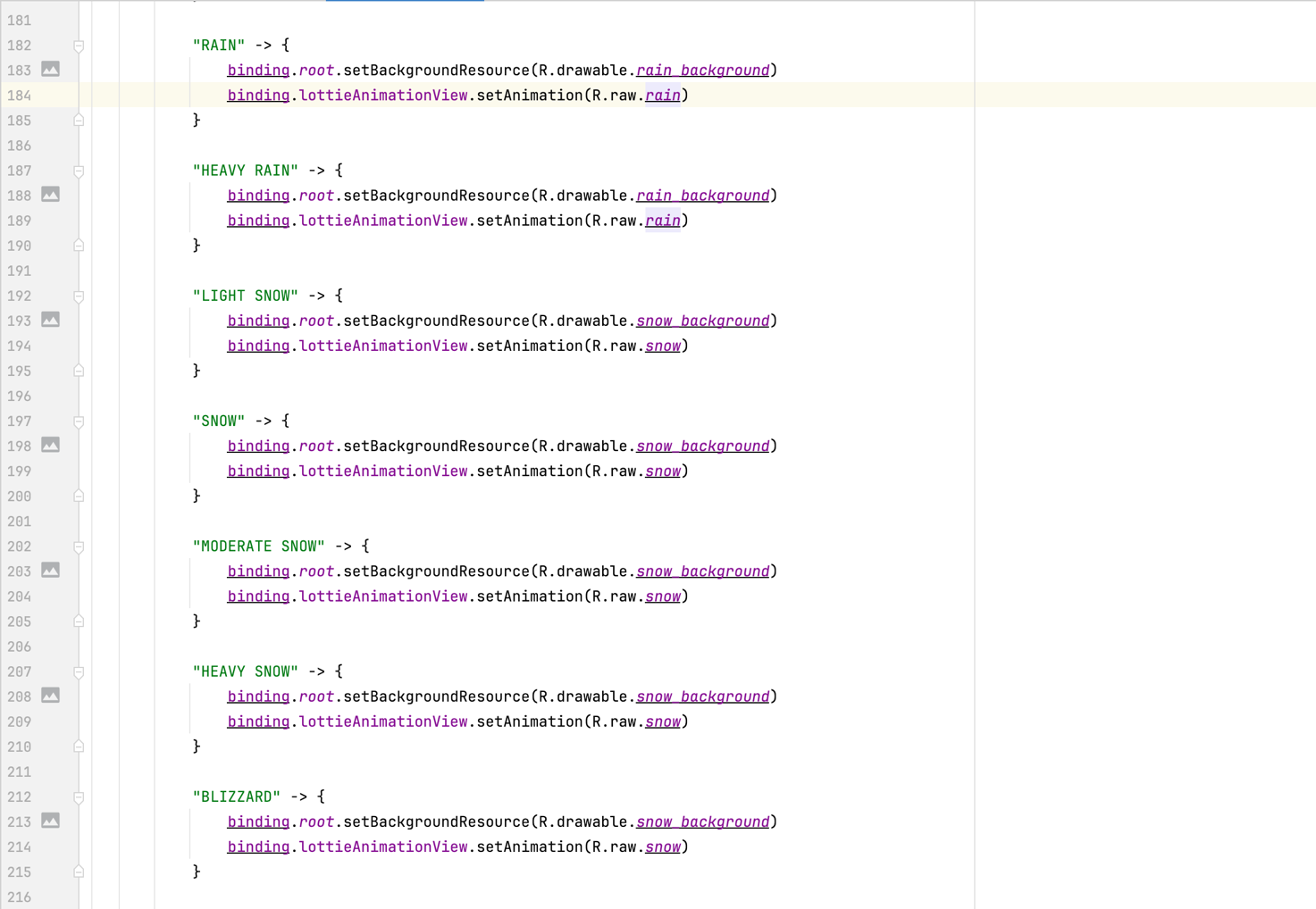
FirstFragment.kt :

****

****

****

****

****

****

Fragment Initialization:

1. The onCreateView method inflates the fragment\_first.xml layout using data binding. This establishes the connection between the UI elements defined in the layout and the code in the fragment.
2. The onViewCreated method is called after the view is created. Here, the data binding reference is re-established (binding = FragmentFirstBinding.bind(view)) to ensure proper access to UI elements.
3. The SearchCity function is called during fragment creation, likely to set up the search functionality for user interaction.
4. Finally, fetchWeatherData is called with a default city name ("Delhi" in this case) to display initial weather information upon fragment launch.

Search Functionality:

* The SearchCity function handles user interaction with the SearchView element
* A SearchView.OnQueryTextListener is set, triggering the fetchWeatherData function whenever the user submits a query (search button press) or modifies the search text.
* This allows users to dynamically retrieve weather data for their desired city.

Weather Data Fetching with Retrofit:

* The fetchWeatherData function is responsible for retrieving weather information from the OpenWeatherMap API.
* It utilizes Retrofit, a popular networking library, to build an API request.
* The function constructs the URL with the following components:
  1. City name (provided by the user's search or default value)
  2. API key (likely stored as a string literal)
  3. Desired units (metric in this case)
* Retrofit then makes an asynchronous call to the OpenWeatherMap API using the constructed URL.
* The response is handled using a Callback object, which defines methods for successful responses (onResponse) and failures (onFailure).

Data Parsing and UI Updates:

* Upon a successful response (isSuccessful), the function parses the JSON data received from the API. It likely utilizes methods of a DailyWeather class (not shown here) that holds the parsed weather data structure.
* Various weather parameters are extracted, including:
  1. Temperature
  2. Humidity
  3. Wind Speed
  4. Sunrise/Sunset timestamps
  5. Pressure
  6. Main weather condition
* Data binding simplifies updating the UI elements in the fragment's layout with these extracted values. This approach avoids the need for manual view manipulation and streamlines UI updates.

Dynamic UI Personalization**:**

* The changeBackground function personalizes the user experience by dynamically changing the fragment's background image and Lottie animation based on the retrieved weather condition (extracted from the API response).
* It uses a when statement to match the weather condition string (e.g., "Clear", "Rain") to predefined background image resources (likely stored in the drawable folder) and Lottie animation assets (potentially stored in the raw folder) representing different weather scenarios.
* This creates a visually immersive experience that reflects the current weather conditions.

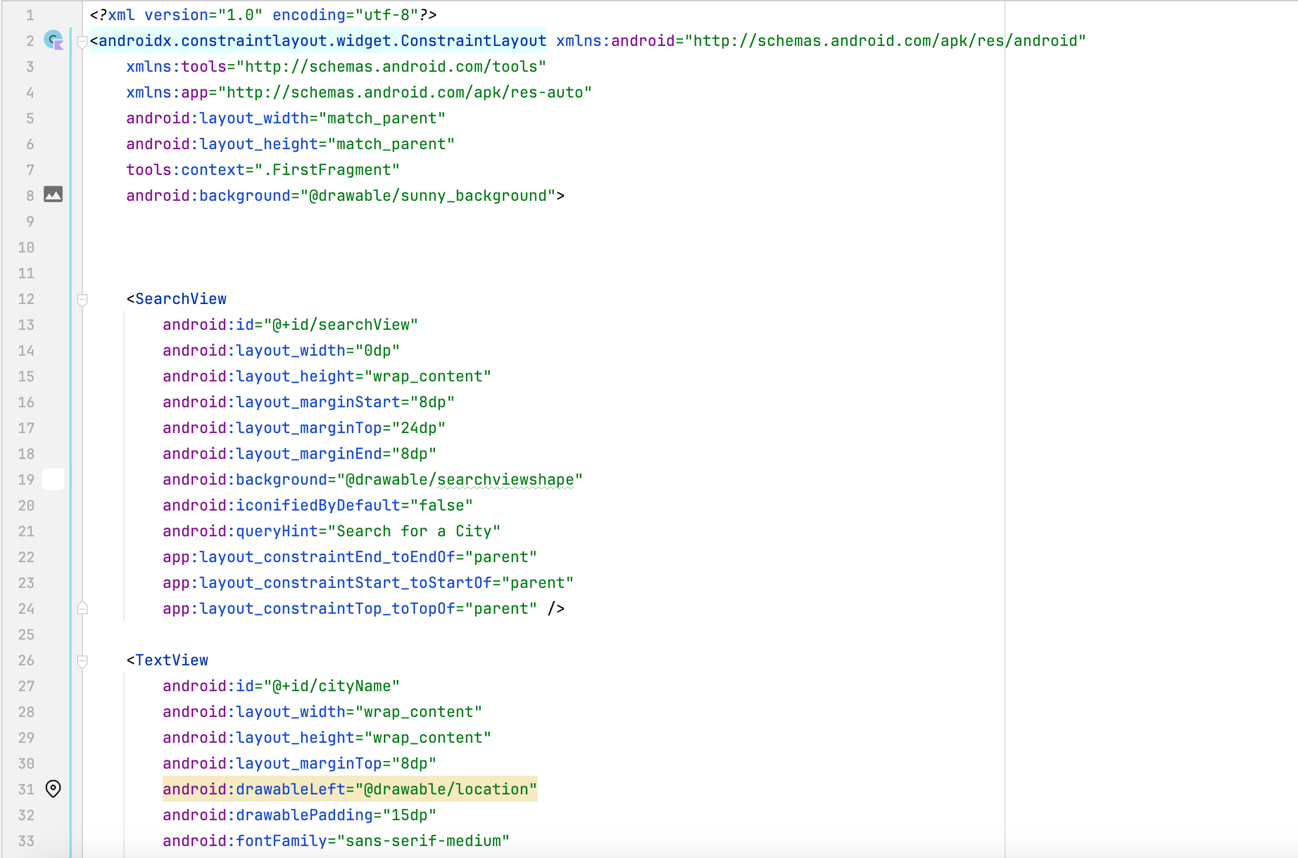
Date and Day Formatting:

* The date function formats the current date retrieved from the system using SimpleDateFormat and the device's locale.
* The time function converts sunrise/sunset timestamps (received in seconds from the API) into human-readable time formats using SimpleDateFormat and considering the device's locale.
* The dayName function likely retrieves the current day of the week based on the system timestamp.

Overall Functionality**:**

* This fragment demonstrates a well-structured approach to fetching weather data, parsing the response, and displaying the information on the user interface.
* It leverages data binding for efficient UI updates, Retrofit for robust API calls, and conditionals for dynamic UI adjustments based on weather conditions.
* The search functionality empowers users to personalize their experience by retrieving weather information for their desired location.

Fragment\_first.xml :









Layout and Background:

* ConstraintLayout serves as the root layout, allowing for flexible positioning of UI elements.
* The background uses the sunny\_background drawable resource, likely depicting a sunny scene.

Search Functionality:

* A SearchView element with the ID searchView allows users to enter a city name for weather retrieval.
* The search view has a predefined hint text "Search for a City".

City Display:

* A TextView with the ID cityName displays the current city name (initially set to "Delhi" or retrieved from user search).
* It has a location icon on the left side.

Weather Animation and Temperature:

* A LottieAnimationView with the ID lottieAnimationView displays a weather animation based on the current weather condition (e.g., sun animation for clear skies).
* A TextView with the ID temp displays the current temperature with a degree symbol (ºC) and initial value 00:00 ºC.

Weather Description:

* A TextView with the ID weather displays the main weather condition in all caps (e.g., "SUNNY").

High/Low Temperatures:

* Two TextViews with IDs max\_temp and min\_temp display the maximum and minimum temperatures for the day, respectively. Both use the format "Max : 00:00 ºC" and "Min : 00:00 ºC".

Day and Date:

* A TextView with the ID day displays the current day of the week (e.g., "Sunday").
* A TextView with the ID date displays the full date in the format "14 April 2024".

Additional Details:

* A FrameLayout with the ID frameLayout acts as a container for additional weather details.
* Inside the FrameLayout, another ConstraintLayout holds various data points:
* Wind speed with an icon and value (wind).
* Sunrise and sunset times with icons and values (sunrise, sunset).
* Current weather condition displayed again (condition).
* Humidity percentage with an icon and value (humidity).

Sea level pressure with an icon and value (sea).

SecondFragment.kt :

A screenshot of a computer program

Description automatically generated

This code defines a fragment named SecondFragment within the com.example.mytabswipe package. Here's a breakdown of its functionalities:

Fragment Creation:

The SecondFragment class extends Fragment, a core building block for modular UI components within activities in Android.

The onCreateView method inflates the fragment's layout using the fragment\_second.xml resource file. This establishes the visual elements for the fragment.

RatingBar Setup:

The fragment retrieves a reference to the RatingBar element with the ID simpleRatingBar using findViewById.

A setOnRatingBarChangeListener is set on the RatingBar. This listener is triggered whenever the user interacts with the rating bar (slides the bar to change the rating).

Rating Listener:

* 1. rating: The new rating value chosen by the user (a float value).
  2. fromUser: A boolean indicating whether the change originated from user interaction (important to avoid infinite loops).

Inside the listener, a string variable ratingText is constructed to display the current rating value.

Finally, the show method displays the toast on the screen.

Overall Functionality:

This fragment presents a RatingBar element that allows users to provide a rating.

When the rating changes, the fragment displays a toast message with the current rating value.

Fragment\_second.xml





Layout and Background:

* RelativeLayout serves as the root layout, allowing for relative positioning of UI elements.
* The background uses the splash drawable resource, likely an image used for the fragment's background.

About Us Text:

* A TextView element displays the heading "ABOUT US" with various style properties:
  + Text color: black
  + Font family: arvbold
  + Text size: 40dp

Description Text:

* A LinearLayout element groups another TextView for a multi-line description:
  + Text content describes the app's purpose and features.
  + Text color: black
  + Font family: platipibold
  + Text size: 24dp
  + Text alignment: center
  + Padding of 15dp around the text content.

Rating Bar:

* A RatingBar element with the ID simpleRatingBar allows users to provide a rating:
  + Number of stars: 5 (standard rating scale)
  + Initial rating (set to 2 in this case, can be changed programmatically)
  + Progress bar colors:
    - Background tint: #F4E0E0 (light gray)
    - Progress tint (for filled stars): #4CAF50 (green)
    - Secondary progress tint (for partially filled stars): #CDDC39
  + Step size: 0.5 (allows for half-star ratings)
  + Layout positioning:
    - Margin top: 750dp (places it near the bottom)
    - Margin left: 80dp (adds some left padding)

**Code:**

ActivityMain.kt :

package com.example.dailyweather  
  
import android.content.ContentValues.*TAG*import android.os.Bundle  
import android.util.Log  
import android.widget.Button  
import android.widget.RatingBar  
import android.widget.SearchView  
import android.widget.TextView  
import android.widget.Toast  
import androidx.activity.enableEdgeToEdge  
import androidx.appcompat.app.AppCompatActivity  
import androidx.core.view.ViewCompat  
import androidx.core.view.WindowInsetsCompat  
import androidx.viewpager2.widget.ViewPager2  
import com.example.dailyweather.databinding.ActivityMainBinding  
import com.example.mytabswipe.SecondFragment  
import com.google.android.material.tabs.TabLayout  
  
  
*//13176c001cf7c033212df14e39d59c3d*class MainActivity : AppCompatActivity() {  
  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.*activity\_main*)  
  
  
 val viewPager: ViewPager2 = findViewById(R.id.*viewPager*)  
 val fragments = *listOf*(FirstFragment(), SecondFragment()) *// Replace with your fragment instances* val adapter = ViewPagerAdapter(fragments, *supportFragmentManager*,lifecycle)  
 viewPager.*adapter* = adapter  
 }  
  
}

FirstFragment.kt

package com.example.dailyweather  
  
import android.os.Bundle  
import android.util.Log  
import android.widget.SearchView  
import androidx.fragment.app.Fragment  
import android.view.LayoutInflater  
import android.view.View  
import android.view.ViewGroup  
import com.example.dailyweather.databinding.FragmentFirstBinding  
import retrofit2.Call  
import retrofit2.Retrofit  
import retrofit2.converter.gson.GsonConverterFactory  
import retrofit2.Callback  
import retrofit2.Response  
import retrofit2.awaitResponse  
import java.text.SimpleDateFormat  
import java.util.Date  
import java.util.Locale  
import kotlin.math.log  
  
  
class FirstFragment : Fragment() {  
  
 private lateinit var binding: FragmentFirstBinding  
  
  
 override fun onCreateView(  
 inflater: LayoutInflater, container: ViewGroup?,  
 savedInstanceState: Bundle?  
 ): View? {  
 binding = FragmentFirstBinding.inflate(inflater, container, false)  
 return binding.*root* }  
  
 override fun onViewCreated(view: View, savedInstanceState: Bundle?) {  
 super.onViewCreated(view, savedInstanceState)  
 binding = FragmentFirstBinding.bind(view)  
 SearchCity()  
 fetchWeatherData("Delhi") *// You can change "Delhi" to any default city you want to display initially* }  
  
  
  
 private fun SearchCity() {  
 val searchView= binding.searchView  
 searchView.setOnQueryTextListener(object : SearchView.OnQueryTextListener{  
 override fun onQueryTextSubmit(query: String?): Boolean {  
 if (query != null) {  
 fetchWeatherData(query)  
 }  
 return true  
 }  
  
 override fun onQueryTextChange(newText: String?): Boolean {  
 if (newText != null) {  
 fetchWeatherData(newText)  
 }  
 return true  
 }  
  
 })  
 }  
  
 private fun fetchWeatherData(cityName: String){  
 val retrofit = Retrofit.Builder()  
 .addConverterFactory(GsonConverterFactory.create())  
 .baseUrl("https://api.openweathermap.org/data/2.5/")  
 .build().create(ApiInterface ::class.*java*)  
  
 val response = retrofit.getWeatherData(cityName, "13176c001cf7c033212df14e39d59c3d", "metric")  
 response.enqueue(object : Callback<DailyWeather>{  
 override fun onResponse(call: Call<DailyWeather>, response: Response<DailyWeather>) {  
 var responseBody = response.body()  
 Log.d("TAG", "onResponse: ${response.body()}")  
 if(response.*isSuccessful* && responseBody != null)  
 {  
 val temperature = responseBody.main.temp.toString()  
 val humidity = responseBody.main.humidity  
 val windSpeed = responseBody.wind.speed  
 val sunrise = responseBody.sys.sunrise.toLong()  
 val sunset = responseBody.sys.sunset.toLong()  
 val seaLevel = responseBody.main.pressure  
 val condition = responseBody.weather.*firstOrNull*()?.main?: "unknown"  
 val maxTemp = responseBody.main.temp\_max  
 val minTemp = responseBody.main.temp\_min  
  
 binding.temp.*text* = "$temperature ºC"  
 binding.weather.*text* = condition  
 binding.maxTemp.*text* = "Max Temp: $maxTemp"  
 binding.minTemp.*text* = "Min Temp: $minTemp"  
 binding.humidity.*text* = "$humidity %"  
 binding.wind.*text* = "$windSpeed m/s"  
 binding.sunrise.*text* = "${time(sunrise)}"  
 binding.sunset.*text* = "${time(sunset)}"  
 binding.sea.*text* = "$seaLevel hPa"  
 binding.condition.*text* = condition  
  
 binding.day.*text* = dayName(System.currentTimeMillis())  
 binding.date.*text* = date()  
 binding.cityName.*text* = "$cityName"  
  
 changeBackground(condition)  
 }  
 }  
  
 override fun onFailure(call: Call<DailyWeather>, t: Throwable) {  
  
 }  
 })  
  
 }  
  
 private fun changeBackground(conditions : String) {  
 Log.d("TAG", "changeBackground: $conditions")  
 when(conditions.*uppercase*()){  
  
 "CLEAR SKY" -> {  
 binding.*root*.setBackgroundResource(R.drawable.*sunny\_background*)  
 binding.lottieAnimationView.setAnimation(R.raw.*sun*)  
 }  
  
 "SUNNY"-> {  
 binding.*root*.setBackgroundResource(R.drawable.*sunny\_background*)  
 binding.lottieAnimationView.setAnimation(R.raw.*sun*)  
 }  
  
 "CLEAR" -> {  
 binding.*root*.setBackgroundResource(R.drawable.*sunny\_background*)  
 binding.lottieAnimationView.setAnimation(R.raw.*sun*)  
 }  
  
 "CLOUDS" -> {  
 binding.*root*.setBackgroundResource(R.drawable.*cloud\_background*)  
 binding.lottieAnimationView.setAnimation(R.raw.*cloud*)  
 }  
  
 "PARTLY CLOUDS" -> {  
 binding.*root*.setBackgroundResource(R.drawable.*cloud\_background*)  
 binding.lottieAnimationView.setAnimation(R.raw.*cloud*)  
 }  
  
 "OVERCAST" -> {  
 binding.*root*.setBackgroundResource(R.drawable.*cloud\_background*)  
 binding.lottieAnimationView.setAnimation(R.raw.*cloud*)  
 }  
  
 "MIST" -> {  
 binding.*root*.setBackgroundResource(R.drawable.*cloud\_background*)  
 binding.lottieAnimationView.setAnimation(R.raw.*cloud*)  
 }  
 "FOGGY"-> {  
 binding.*root*.setBackgroundResource(R.drawable.*cloud\_background*)  
 binding.lottieAnimationView.setAnimation(R.raw.*cloud*)  
 }  
  
 "HAZE" -> {  
 binding.*root*.setBackgroundResource(R.drawable.*cloud\_background*)  
 binding.lottieAnimationView.setAnimation(R.raw.*haze*)  
 }  
  
 "LIGHT RAIN" -> {  
 binding.*root*.setBackgroundResource(R.drawable.*rain\_background*)  
 binding.lottieAnimationView.setAnimation(R.raw.*rain*)  
 }  
  
 "DRIZZLE" -> {  
 binding.*root*.setBackgroundResource(R.drawable.*rain\_background*)  
 binding.lottieAnimationView.setAnimation(R.raw.*rain*)  
 }  
  
 "MODERATE RAIN"-> {  
 binding.*root*.setBackgroundResource(R.drawable.*rain\_background*)  
 binding.lottieAnimationView.setAnimation(R.raw.*rain*)  
 }  
  
 "SHOWERS" -> {  
 binding.*root*.setBackgroundResource(R.drawable.*rain\_background*)  
 binding.lottieAnimationView.setAnimation(R.raw.*rain*)  
 }  
  
 "RAIN" -> {  
 binding.*root*.setBackgroundResource(R.drawable.*rain\_background*)  
 binding.lottieAnimationView.setAnimation(R.raw.*rain*)  
 }  
  
 "HEAVY RAIN" -> {  
 binding.*root*.setBackgroundResource(R.drawable.*rain\_background*)  
 binding.lottieAnimationView.setAnimation(R.raw.*rain*)  
 }  
  
 "LIGHT SNOW" -> {  
 binding.*root*.setBackgroundResource(R.drawable.*snow\_background*)  
 binding.lottieAnimationView.setAnimation(R.raw.*snow*)  
 }  
  
 "SNOW" -> {  
 binding.*root*.setBackgroundResource(R.drawable.*snow\_background*)  
 binding.lottieAnimationView.setAnimation(R.raw.*snow*)  
 }  
  
 "MODERATE SNOW" -> {  
 binding.*root*.setBackgroundResource(R.drawable.*snow\_background*)  
 binding.lottieAnimationView.setAnimation(R.raw.*snow*)  
 }  
  
 "HEAVY SNOW" -> {  
 binding.*root*.setBackgroundResource(R.drawable.*snow\_background*)  
 binding.lottieAnimationView.setAnimation(R.raw.*snow*)  
 }  
  
 "BLIZZARD" -> {  
 binding.*root*.setBackgroundResource(R.drawable.*snow\_background*)  
 binding.lottieAnimationView.setAnimation(R.raw.*snow*)  
 }  
  
 else -> {  
 binding.*root*.setBackgroundResource(R.drawable.*sunny\_background*)  
 binding.lottieAnimationView.setAnimation(R.raw.*sun*)  
 }  
  
 }  
  
 binding.lottieAnimationView.playAnimation()  
 }  
  
 private fun date(): String {  
 val sdf = SimpleDateFormat("dd MMM yyyy", Locale.getDefault())  
 return sdf.format(Date())  
 }  
  
 private fun time(timeStamp : Long): String {  
 val sdf = SimpleDateFormat("HH:mm", Locale.getDefault())  
 return sdf.format(Date(timeStamp\*1000))  
 }  
  
 fun dayName(timeStamp : Long) : String{  
 val sdf = SimpleDateFormat("EEEE", Locale.getDefault())  
 return sdf.format(Date())  
 }  
}

SecondFragment.kt :

package com.example.mytabswipe  
  
import android.os.Bundle  
import androidx.fragment.app.Fragment  
import android.view.LayoutInflater  
import android.view.View  
import android.view.ViewGroup  
import android.widget.Button  
import android.widget.RatingBar  
import android.widget.TextView  
import android.widget.Toast  
import com.example.dailyweather.R  
  
  
class SecondFragment : Fragment() {  
  
  
  
 override fun onCreateView(  
 inflater: LayoutInflater, container: ViewGroup?,  
 savedInstanceState: Bundle?  
 ): View? {  
 *// Inflate the layout for this fragment* val view = inflater.inflate(R.layout.*fragment\_second*, container, false)  
 val rb : RatingBar = view.findViewById(R.id.*simpleRatingBar*)  
  
 rb.setOnRatingBarChangeListener **{** \_, rating, \_ **->** val ratingText = "Rating :: $rating"  
 Toast.makeText(requireContext(), ratingText, Toast.*LENGTH\_SHORT*).show()  
 **}** return view  
 }  
}

SplashScreen.kt :

package com.example.dailyweather  
  
import android.content.Intent  
import android.os.Bundle  
import android.os.Handler  
import androidx.activity.enableEdgeToEdge  
import androidx.appcompat.app.AppCompatActivity  
import androidx.core.view.ViewCompat  
import androidx.core.view.WindowInsetsCompat  
  
class SplashActivity : AppCompatActivity() {  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 *enableEdgeToEdge*()  
 setContentView(R.layout.*activity\_splash*)  
  
 Handler().postDelayed(**{** val intent = Intent(this, MainActivity::class.*java*)  
 startActivity(intent)  
 finish()  
 **}**,3000)  
 }  
 }

activity\_main.xml

*<?*xml version="1.0" encoding="utf-8"*?>*<androidx.constraintlayout.widget.ConstraintLayout 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:id="@+id/main"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 tools:context=".MainActivity"  
 android:background="@drawable/sunny\_background">  
  
  
 <androidx.viewpager2.widget.ViewPager2  
 android:id="@+id/viewPager"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:layout\_weight="1" />  
  
  
</androidx.constraintlayout.widget.ConstraintLayout>

Fragment\_first.xml

*<?*xml version="1.0" encoding="utf-8"*?>*<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:tools="http://schemas.android.com/tools"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 tools:context=".FirstFragment"  
 android:background="@drawable/sunny\_background">  
  
  
  
 <SearchView  
 android:id="@+id/searchView"  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="8dp"  
 android:layout\_marginTop="24dp"  
 android:layout\_marginEnd="8dp"  
 android:background="@drawable/searchviewshape"  
 android:iconifiedByDefault="false"  
 android:queryHint="Search for a City"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent" />  
  
 <TextView  
 android:id="@+id/cityName"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="8dp"  
 android:drawableLeft="@drawable/location"  
 android:drawablePadding="15dp"  
 android:fontFamily="sans-serif-medium"  
 android:textColor="@color/black"  
 android:textSize="20sp"  
 app:layout\_constraintEnd\_toEndOf="@+id/searchView"  
 app:layout\_constraintHorizontal\_bias="0.51"  
 app:layout\_constraintStart\_toStartOf="@+id/searchView"  
 app:layout\_constraintTop\_toBottomOf="@+id/searchView" />  
  
  
 <com.airbnb.lottie.LottieAnimationView  
 android:id="@+id/lottieAnimationView"  
 android:layout\_width="150dp"  
 android:layout\_height="150dp"  
 android:layout\_marginTop="16dp"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@+id/cityName"  
 app:lottie\_autoPlay="true"  
 app:lottie\_loop="true"  
 app:lottie\_rawRes="@raw/sun" />  
  
 <TextView  
 android:id="@+id/textView2"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="24dp"  
 android:fontFamily="@font/platypibold"  
 android:text="Today"  
 android:textColor="@color/black"  
 android:textSize="34sp"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toEndOf="@+id/lottieAnimationView"  
 app:layout\_constraintTop\_toBottomOf="@+id/cityName" />  
  
 <TextView  
 android:id="@+id/temp"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:fontFamily="@font/platypsemibold"  
 android:text="00:00 ºC "  
 android:textColor="@color/black"  
 android:textSize="50sp"  
 app:layout\_constraintEnd\_toEndOf="@+id/textView2"  
 app:layout\_constraintStart\_toStartOf="@+id/textView2"  
 app:layout\_constraintTop\_toBottomOf="@+id/textView2" />  
  
 <TextView  
 android:id="@+id/weather"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="7dp"  
 android:fontFamily="@font/platypiregular"  
 android:text="SUNNY"  
 android:textAllCaps="true"  
 android:textColor="@color/black"  
 android:textSize="20sp"  
 app:layout\_constraintEnd\_toEndOf="@+id/lottieAnimationView"  
 app:layout\_constraintStart\_toStartOf="@+id/lottieAnimationView"  
 app:layout\_constraintTop\_toBottomOf="@+id/lottieAnimationView" />  
  
 <TextView  
 android:id="@+id/max\_temp"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:fontFamily="@font/platypiregular"  
 android:text="Max : 00:00 ºC"  
 android:textColor="@color/black"  
 android:textSize="16sp"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@+id/weather" />  
  
 <TextView  
 android:id="@+id/min\_temp"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:fontFamily="@font/platypiregular"  
 android:text="Min : 00:00 ºC"  
 android:textColor="@color/black"  
 android:textSize="16sp"  
 app:layout\_constraintEnd\_toEndOf="@+id/max\_temp"  
 app:layout\_constraintStart\_toStartOf="@+id/max\_temp"  
 app:layout\_constraintTop\_toBottomOf="@+id/max\_temp" />  
  
 <TextView  
 android:id="@+id/day"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="15dp"  
 android:fontFamily="@font/platypibold"  
 android:text="Sunday"  
 android:textColor="@color/black"  
 android:textSize="20sp"  
 app:layout\_constraintEnd\_toEndOf="@+id/min\_temp"  
 app:layout\_constraintStart\_toStartOf="@+id/min\_temp"  
 app:layout\_constraintTop\_toBottomOf="@+id/min\_temp" />  
  
 <TextView  
 android:id="@+id/date"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="16dp"  
 android:fontFamily="@font/platypiregular"  
 android:text="14 April 2024"  
 android:textColor="@color/black"  
 android:textSize="16sp"  
 app:layout\_constraintEnd\_toEndOf="@+id/day"  
 app:layout\_constraintStart\_toStartOf="@+id/day"  
 app:layout\_constraintTop\_toBottomOf="@+id/day" />  
  
 <FrameLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="200dp"  
 android:layout\_marginStart="16dp"  
 android:layout\_marginEnd="16dp"  
 android:background="@drawable/backgroundshape"  
 app:layout\_constraintBottom\_toTopOf="@+id/textView11"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@+id/date">  
  
 <androidx.constraintlayout.widget.ConstraintLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent">  
  
 <androidx.constraintlayout.helper.widget.Flow  
 android:id="@+id/flow"  
 android:layout\_width="0dp"  
 android:layout\_height="0dp"  
 android:orientation="horizontal"  
 app:constraint\_referenced\_ids="linearLayout,linearLayout2,linearLayout3,linearLayout4,linearLayout5,linearLayout7"  
 app:flow\_maxElementsWrap="3"  
 app:flow\_wrapMode="chain"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintHorizontal\_bias="0.0"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"  
 app:layout\_constraintVertical\_bias="0.0" />  
  
 <LinearLayout  
 android:id="@+id/linearLayout"  
 android:layout\_width="80dp"  
 android:layout\_height="89dp"  
 android:background="@drawable/backgroundshape"  
 android:gravity="bottom|center\_vertical"  
 android:orientation="vertical">  
  
 <ImageView  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 app:srcCompat="@drawable/wind" />  
  
 <TextView  
 android:id="@+id/wind"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:fontFamily="@font/platypimedium"  
 android:gravity="center"  
 android:text="0.00 m/sec"  
 android:textColor="@color/black" />  
  
 <TextView  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginBottom="5dp"  
 android:fontFamily="@font/platypimedium"  
 android:gravity="center"  
 android:text="Wind Speed"  
 android:textColor="@color/black"  
 android:textSize="10sp" />  
 </LinearLayout>  
  
 <LinearLayout  
 android:id="@+id/linearLayout2"  
 android:layout\_width="80dp"  
 android:layout\_height="89dp"  
 android:background="@drawable/backgroundshape"  
 android:gravity="bottom|center\_vertical"  
 android:orientation="vertical">  
  
 <ImageView  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 app:srcCompat="@drawable/sunrise" />  
  
 <TextView  
 android:id="@+id/sunrise"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:fontFamily="@font/platypimedium"  
 android:gravity="center"  
 android:text="00:00"  
 android:textColor="@color/black" />  
  
 <TextView  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginBottom="5dp"  
 android:fontFamily="@font/platypimedium"  
 android:gravity="center"  
 android:text="Sunrise"  
 android:textColor="@color/black"  
 android:textSize="10sp" />  
 </LinearLayout>  
  
 <LinearLayout  
 android:id="@+id/linearLayout3"  
 android:layout\_width="80dp"  
 android:layout\_height="89dp"  
 android:background="@drawable/backgroundshape"  
 android:gravity="bottom|center\_vertical"  
 android:orientation="vertical">  
  
 <ImageView  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 app:srcCompat="@drawable/sunny" />  
  
 <TextView  
 android:id="@+id/condition"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:fontFamily="@font/platypimedium"  
 android:gravity="center"  
 android:text="SUNNY"  
 android:textColor="@color/black" />  
  
 <TextView  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginBottom="5dp"  
 android:fontFamily="@font/platypimedium"  
 android:gravity="center"  
 android:text="Condition"  
 android:textColor="@color/black"  
 android:textSize="10sp" />  
 </LinearLayout>  
  
 <LinearLayout  
 android:id="@+id/linearLayout4"  
 android:layout\_width="80dp"  
 android:layout\_height="89dp"  
 android:background="@drawable/backgroundshape"  
 android:gravity="bottom|center\_vertical"  
 android:orientation="vertical">  
  
 <ImageView  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 app:srcCompat="@drawable/humidity" />  
  
 <TextView  
 android:id="@+id/humidity"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:fontFamily="@font/platypimedium"  
 android:gravity="center"  
 android:text="00 %"  
 android:textColor="@color/black" />  
  
 <TextView  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginBottom="5dp"  
 android:fontFamily="@font/platypimedium"  
 android:gravity="center"  
 android:text="Humidity"  
 android:textColor="@color/black"  
 android:textSize="10sp" />  
 </LinearLayout>  
  
 <LinearLayout  
 android:id="@+id/linearLayout5"  
 android:layout\_width="80dp"  
 android:layout\_height="89dp"  
 android:background="@drawable/backgroundshape"  
 android:gravity="bottom|center\_vertical"  
 android:orientation="vertical">  
  
 <ImageView  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 app:srcCompat="@drawable/sunset" />  
  
 <TextView  
 android:id="@+id/sunset"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:fontFamily="@font/platypimedium"  
 android:gravity="center"  
 android:text="00:00"  
 android:textColor="@color/black" />  
  
 <TextView  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginBottom="5dp"  
 android:fontFamily="@font/platypimedium"  
 android:gravity="center"  
 android:text="Sunset"  
 android:textColor="@color/black"  
 android:textSize="10sp" />  
 </LinearLayout>  
  
 <LinearLayout  
 android:id="@+id/linearLayout7"  
 android:layout\_width="80dp"  
 android:layout\_height="89dp"  
 android:background="@drawable/backgroundshape"  
 android:gravity="bottom|center\_vertical"  
 android:orientation="vertical">  
  
 <ImageView  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 app:srcCompat="@drawable/sea" />  
  
 <TextView  
 android:id="@+id/sea"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:fontFamily="@font/platypimedium"  
 android:gravity="center"  
 android:text="0000 hPa"  
 android:textColor="@color/black" />  
  
 <TextView  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginBottom="5dp"  
 android:fontFamily="@font/platypimedium"  
 android:gravity="center"  
 android:text="Sea"  
 android:textColor="@color/black"  
 android:textSize="10sp" />  
 </LinearLayout>  
 </androidx.constraintlayout.widget.ConstraintLayout>  
  
 </FrameLayout>  
  
 <TextView  
 android:id="@+id/textView11"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text=" DESIGNED BY \n Sarthak Vashisht"  
 android:textAlignment="center"  
 android:textColor="@color/white"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 android:layout\_marginBottom="25dp"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent" />  
  
  
</androidx.constraintlayout.widget.ConstraintLayout>

Fragment\_second.xml

`*<?*xml version="1.0" encoding="utf-8"*?>*<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 tools:context=".SecondFragment"  
 android:background="@drawable/splash">  
  
 <TextView  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="20dp"  
 android:text="ABOUT US"  
 android:textColor="@color/black"  
 android:fontFamily="@font/arvobold"  
 android:textSize="40dp"  
 android:textAlignment="center"/>  
  
 <LinearLayout  
 android:layout\_marginLeft="30dp"  
 android:layout\_marginRight="30dp"  
 android:layout\_marginTop="85dp"  
 android:layout\_width="350dp"  
 android:layout\_height="wrap\_content">  
  
  
 <TextView  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:padding="15dp"  
 android:textSize="24dp"  
 android:fontFamily="@font/platypibold"  
 android:textAlignment="center"  
 android:textColor="@color/black"  
 android:text="Welcome to Daily Weather! At WeatherWise, we're dedicated to providing you  
with accurate, reliable, and user-friendly weather forecasts. Our team of meteorologists  
 and software engineers work tirelessly to ensure that our forecasts are precise and delivered  
in the most intuitive way possible. If you like our app, then please take a moment to rate it below.  
Thank you for choosing WeatherWise!" />  
</LinearLayout>  
 <RatingBar  
 android:id="@+id/simpleRatingBar"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:numStars="5"  
 android:paddingLeft="5dp"  
 android:paddingRight="1dp"  
 android:progressBackgroundTint="#F4E0E0"  
 android:progressTint="#4CAF50"  
 android:layout\_marginTop="750dp"  
 android:layout\_marginLeft="80dp"  
 android:rating="2"  
 android:secondaryProgressTint="#CDDC39"  
 android:stepSize="0.5"  
 tools:layout\_editor\_absoluteX="93dp"  
 tools:layout\_editor\_absoluteY="486dp" />  
  
</RelativeLayout>

Activity\_splash.xml

*<?*xml version="1.0" encoding="utf-8"*?>*<androidx.constraintlayout.widget.ConstraintLayout 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:id="@+id/main"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:background="@drawable/splash"  
 tools:context=".SplashActivity">  
  
 <ImageView  
 android:id="@+id/imageView"  
 android:layout\_width="211dp"  
 android:layout\_height="140dp"  
 android:layout\_marginTop="95dp"  
 android:src="@drawable/white\_cloud"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent" />  
  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="40dp"  
 android:fontFamily="@font/arvobold"  
 android:text="Weather Matters, \n We've Got the Details"  
 android:textAlignment="center"  
 android:textColor="@color/black"  
 android:textSize="25dp"  
 app:layout\_constraintEnd\_toEndOf="@+id/imageView"  
 app:layout\_constraintStart\_toStartOf="@+id/imageView"  
 app:layout\_constraintTop\_toBottomOf="@+id/imageView" />  
  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:fontFamily="@font/arvoregular"  
 android:text="DESIGNED BY \n Sarthak Vashisht"  
 android:textAlignment="center"  
 android:textColor="@color/white"  
 android:layout\_marginBottom="30dp"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent" />  
  
</androidx.constraintlayout.widget.ConstraintLayout>

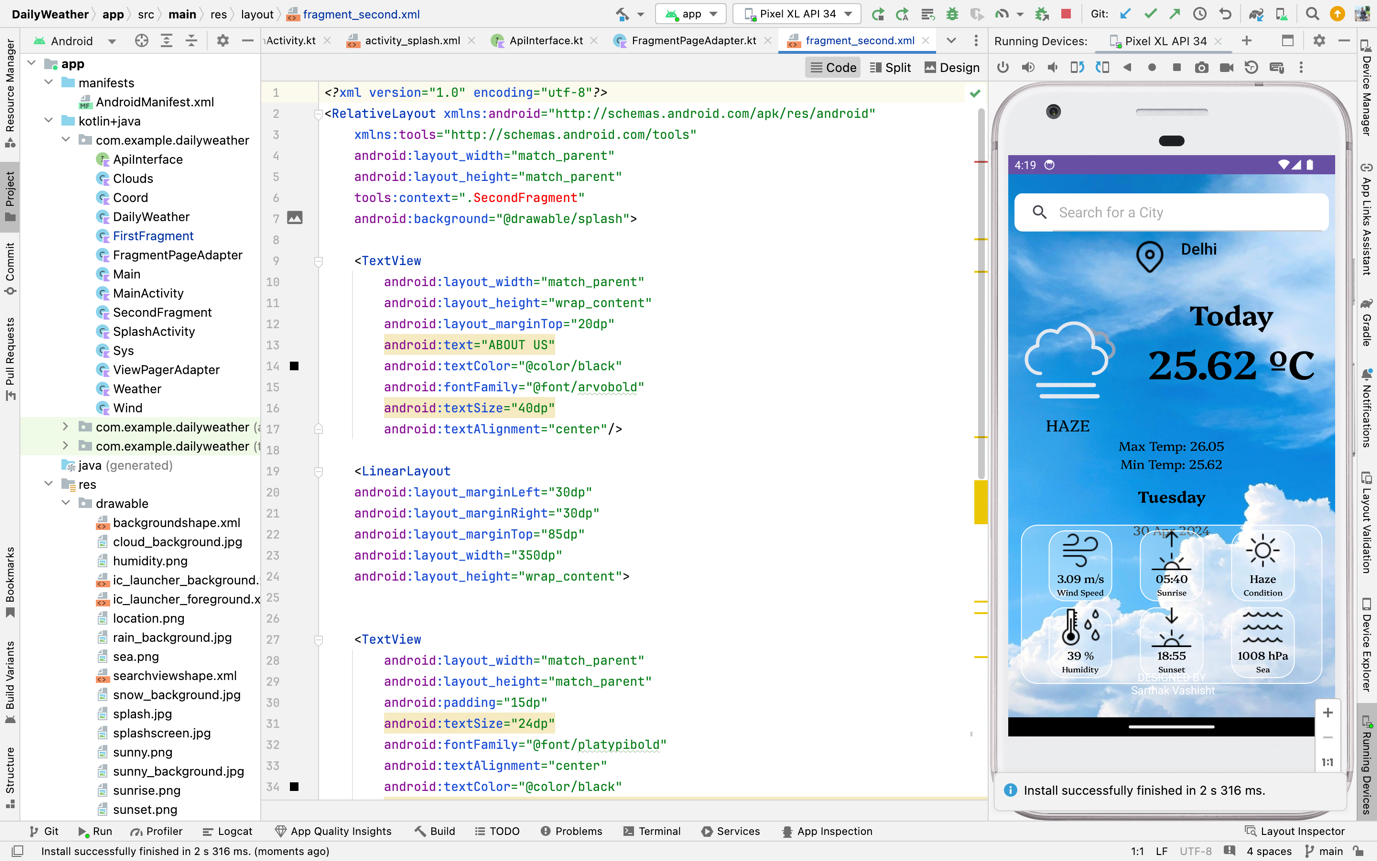
**Output :**

**A stone path with a fence and grass on the side

Description automatically generated**

****

**Emulator Screenshot :**

****

**A screenshot of a computer

Description automatically generated**

**Conclusion and Future Scope :**

**Conclusion:**

The Daily Weather app presents a user-friendly interface and robust functionality, providing users with real-time weather updates and forecasts. Through the implementation of tabbed navigation, users can easily access different sections of the app, enhancing the overall user experience.

The SecondFragment class, responsible for managing the second tab in the app, demonstrates effective handling of user interactions related to rating input. By incorporating a RatingBar component and displaying toast messages upon rating changes, the app facilitates user feedback and engagement.

Overall, the Daily Weather app shows promise in delivering reliable weather information while offering intuitive user interactions. The successful implementation of features like the SecondFragment highlights the app's potential to cater to user needs effectively.

**Future Scope:**

Moving forward, there are several areas where the Daily Weather app can further improve and expand its capabilities:

1. **Enhanced User Feedback Mechanisms:** Consider implementing additional features for user feedback, such as comments or surveys, to gather more detailed insights into user preferences and improve the app accordingly.
2. **Integration with External Services:** Explore opportunities to integrate with external services or APIs to enrich the app’s functionality, such as providing personalized recommendations based on weather preferences or integrating social media sharing options.
3. **Localization and Customization:** Incorporate localization features to support users in different regions and languages, ensuring a more inclusive user experience. Additionally, offer customization options for users to tailor the app’s interface and content to their preferences.
4. **Accessibility Improvements:** Prioritize accessibility enhancements to ensure that the app is usable by all users, including those with disabilities. This may involve implementing features such as screen reader compatibility and high-contrast modes.
5. **Performance Optimization:** Continuously optimize the app’s performance to ensure smooth navigation and minimal loading times, even under heavy usage or in low-connectivity environments.

By focusing on these areas of improvement, the Daily Weather app can further solidify its position as a reliable and user-friendly weather companion, catering to the diverse needs of its user base.