Practical No. 8

Title: Android program to work with google maps and locations

Aim: Create an application to demonstrate google maps and locations

Introduction

Android Google Map

Android provides facility to integrate Google map in our application. Google map displays your current location, navigate location direction, search location etc. We can also customize Google map according to our requirement.

Types of Google Maps

There are four different types of Google maps, as well as an optional to no map at all. Each of them gives different view on map. These maps are as follow:

1. **Normal:** This type of map displays typical road map, natural features like river and some features build by humans.
2. **Hybrid:** This type of map displays satellite photograph data with typical road maps. It also displays road and feature labels.
3. **Satellite:** Satellite type displays satellite photograph data, but doesn't display road and feature labels.
4. **Terrain:** This type displays photographic data. This includes colors, contour lines and labels and perspective shading.
5. **None:** This type displays an empty grid with no tiles loaded.

Syntax of different types of map

1. googleMap.setMapType(GoogleMap.MAP_TYPE_NORMAL);
2. googleMap.setMapType(GoogleMap.MAP_TYPE_HYBRID);
3. googleMap.setMapType(GoogleMap.MAP_TYPE_SATELLITE);

4. googleMap.setMapType(GoogleMap.MAP_TYPE_TERRAIN);

Methods of Google map

Google map API provides several methods that help to customize Google map. These methods are as following:

Methods	**Description**
addCircle(CircleOptions options)	This method add circle to map.
addPolygon(PolygonOptions options)	This method add polygon to map.
addTileOverlay(TileOverlayOptions options)	This method add tile overlay to the map.
animateCamera(CameraUpdate update)	This method moves the map according to the update with an animation.
clear()	This method removes everything from the map.
getMyLocation()	This method returns the currently displayed user location.
moveCamera(CameraUpdate update)	This method reposition the camera according to the instructions defined in the update.
setTrafficEnabled(boolean enabled)	This method set the traffic layer on or off.
snapshot(GoogleMap.SnapshotReadyCallback callback)	This method takes a snapshot of the map.
stopAnimation()	This method stops the camera animation if there is any progress.

Exercise - Create an application to demonstrate implementation of Google Maps and location-based services

Implementation:

Program:

activity_maps.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:tools="http://schemas.android.com/tools"
  android:orientation="vertical"
  android:layout_height="match_parent"
  android:layout_width="match_parent">

  <androidx.fragment.app.FragmentContainerView
    android:id="@+id/map"
    android:name="com.google.android.gms.maps.SupportMapFragment"
    android:layout_width="match_parent"
    android:layout_height="675dp"
    tools:context=".MapsActivity" />
  <LinearLayout
    android:layout_width="match_parent"
    android:layout_height="92dp"
    android:orientation="horizontal">

    <Button
      android:id="@+id/button"
      style="?android:attr/buttonBarButtonStyle"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_weight="1"
      android:onClick="onZoom"
      android:text="@string/zoom_in"
      android:background="#F3E889"
      android:textSize="16sp"
      tools:ignore="usingOnClickInXml"
      />

    <Button
      android:id="@+id/button2"
      style="?android:attr/buttonBarButtonStyle"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
```

```
        android:layout_weight="1"
        android:onClick="onZoom"
        android:text="@string/zoom_out"
        android:background="#F3E889"
        android:textSize="16sp"
        tools:ignore="usingOnClickInXml"
        />
  </LinearLayout>

</LinearLayout>
```

MapsActivity.java

```
package com.example.googlemaps;


import androidx.fragment.app.FragmentActivity;


import android.os.Bundle;

import android.view.View;


import com.google.android.gms.maps.CameraUpdateFactory;

import com.google.android.gms.maps.GoogleMap;

import com.google.android.gms.maps.OnMapReadyCallback;

import com.google.android.gms.maps.SupportMapFragment;

import com.google.android.gms.maps.model.LatLng;

import com.google.android.gms.maps.model.MarkerOptions;

import com.example.googlemaps.databinding.ActivityMapsBinding;


public class MapsActivity extends FragmentActivity implements OnMapReadyCallback {


  private GoogleMap mMap;

  private ActivityMapsBinding binding;
```

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

   binding = ActivityMapsBinding.inflate(getLayoutInflater());
   setContentView(binding.getRoot());

   // Obtain the SupportMapFragment and get notified when the map is ready to be used.
   SupportMapFragment mapFragment = (SupportMapFragment) getSupportFragmentManager()
         .findFragmentById(R.id.map);
   mapFragment.getMapAsync(this);
}

/**
 * Manipulates the map once available.
 * This callback is triggered when the map is ready to be used.
 * This is where we can add markers or lines, add listeners or move the camera. In this case,
 * we just add a marker near Sydney, Australia.
 * If Google Play services is not installed on the device, the user will be prompted to install
 * it inside the SupportMapFragment. This method will only be triggered once the user has
 * installed Google Play services and returned to the app.
 */
@Override
public void onMapReady(GoogleMap googleMap) {
   mMap = googleMap;

   // Add a marker in Sydney and move the camera
   LatLng ratnagiri = new LatLng(16.992500, 73.294197);
```

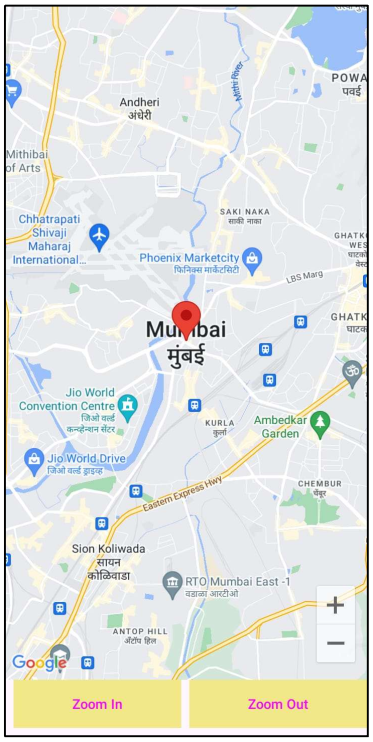
```
        mMap.addMarker(new
MarkerOptions().position(ratnagiri).title(getString(R.string.marker_in_ratnagiri)));
        mMap.moveCamera(CameraUpdateFactory.newLatLng(ratnagiri));
        mMap.animateCamera(CameraUpdateFactory.newLatLngZoom(ratnagiri,12));
        mMap.getUiSettings().setZoomControlsEnabled(true);


        LatLng mumbai = new LatLng(19.0760, 72.8777);
        mMap.addMarker(new MarkerOptions().position(mumbai).title(getString(R.string.marker_in_mumbai)));
        mMap.moveCamera(CameraUpdateFactory.newLatLng(mumbai));
        mMap.animateCamera(CameraUpdateFactory.newLatLngZoom(mumbai,12));
        mMap.getUiSettings().setZoomControlsEnabled(true);
    }


    public void onZoom(View view) {
        if(view.getId() ==R.id.button){
            mMap.animateCamera(CameraUpdateFactory.zoomIn());
        }
        if(view.getId() ==R.id.button2){
            mMap.animateCamera(CameraUpdateFactory.zoomOut());
        }
    }
}
```

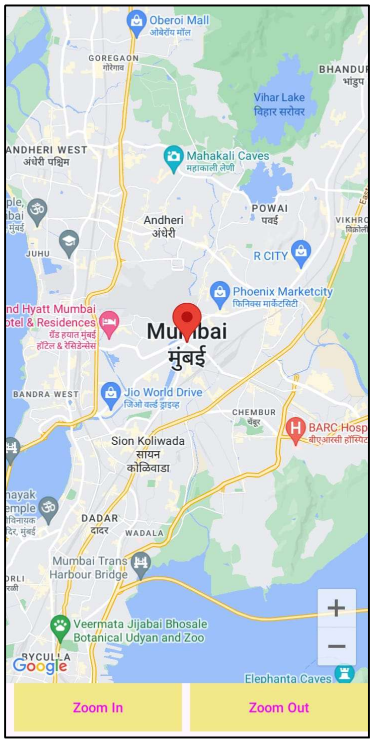
AndroidManifest.xml:

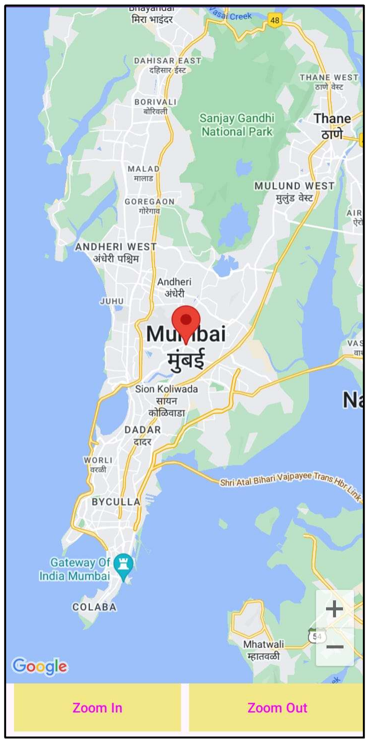
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">
```

```
<application
    android:allowBackup="true"
    android:dataExtractionRules="@xml/data_extraction_rules"
    android:fullBackupContent="@xml/backup_rules"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/Theme.GoogleMaps"
    tools:targetApi="31">

    <!--
         TODO: Before you run your application, you need a Google Maps API key.

         To get one, follow the directions here:

            https://developers.google.com/maps/documentation/android-sdk/get-api-key

         Once you have your API key (it starts with "AIza"), define a new property in your
         project's local.properties file (e.g. MAPS_API_KEY=Aiza...), and replace the
         "YOUR_API_KEY" string in this file with "${MAPS_API_KEY}".
    -->
    <meta-data
        android:name="com.google.android.geo.API_KEY"
        android:value="AIzaSyBoH-omBd54F-hd3eIvL6Nn6-jC0xiOpXQ" />
    <meta-data
        android:name="com.google.android.gms.version"
        android:value="@integer/google_play_services_version" />
    <activity
      android:name=".MapsActivity"
      android:exported="true"
      android:label="@string/title_activity_maps">
      <intent-filter>
        <action android:name="android.intent.action.MAIN" />

        <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>
    </activity>
  </application>
</manifest>
```

Output:







Original **Zoom In** **Zoom Out**

Conclusion - By effectively utilizing Google Maps APIs, developers can enrich their Android applications with location-based features, enhancing user engagement and providing valuable information. The flexibility and customization options offered by Google Maps allow for a wide range of applications, from simple navigation to complex location-based services.