BATCH AND ROLL NO:
EXPERIMENT NO.10
TITLE: Design a mobile app using Google Map and GPS to trace the location.
DATE OF PERFORMANCE:
DATE OF SUBMISSION:

Title: Design a mobile app using Google Map and GPS to trace the location.

Requirements:
1 Android studio
2.Google Play service Packages

Theory:

Introduction

In the ever-connected world of mobile applications, harnessing the power of location-based services has become essential for creating dynamic and context-aware applications. This lab focuses on designing a mobile application that integrates Google Maps and Global Positioning System (GPS) functionalities, enabling users to trace their location and visualize it on a map. The fusion of Google Maps and GPS empowers developers to craft applications that provide real-time location-based information, fostering an enriched user experience.

Objective of the Lab: The primary objective of this lab is to guide you through the process of designing a mobile application that leverages Google Maps and GPS technology. By the end of this lab, you should be adept at implementing features such as obtaining real-time location updates, displaying the user's location on a Google Map, and incorporating additional functionalities to enhance the overall location tracking experience.

Components of the Application:

1. **Google Maps Integration:**
    - The application will integrate Google Maps, allowing users to view and interact with a map interface.
    - Developers will utilize the Google Maps API to embed the map and leverage its rich features for location-based interactions.
2. **GPS Location Tracking:**
    - The application will utilize the device's GPS functionality to trace and update the user's real-time location.
    - GPS data will be used to dynamically update the user's marker on the Google Map.

Lab Prerequisites:

- Basic understanding of mobile application development concepts.
- Familiarity with the chosen development environment (e.g., Android Studio).
- Prior knowledge of programming languages such as Java (for Android)

Steps:

Step 1: Set Up Your Development Environment

- Ensure that you have Android Studio installed and configured on your machine.
- Create a new project in Android Studio.

Step 2: Obtain Google Maps API Key

- Obtain a Google Maps API key from the Google Cloud Console.
- Enable the "Maps SDK for Android" for your project.

Step 3: Add Google Maps SDK to Your Project

- Open the build.gradle file (Module: app) and add the following dependency:

  implementation 'com.google.android.gms:play-services-maps:17.0.1'

Step 4: Design the User Interface

- Open the XML layout file associated with your main activity (e.g., activity_main.xml).
- Add a SupportMapFragment or MapView element to your layout to display the Google Map.

Step 5: Implement Google Maps Integration

- Open the Java file associated with your main activity (e.g., MainActivity.java).
- Initialize the Google Map and set up its features, such as zoom controls and markers.

Step 6: Implement GPS Location Tracking

- Request permission for accessing the device's location in the AndroidManifest.xml.
- Implement a LocationListener to receive location updates.

Step 7: Test Your Application

- Run your application on an emulator or a physical device.

- Verify that the Google Map is displayed, and the user's location is updated on the map as they move.

XML Code:

Activity_main.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<fragment xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
android:id="@+id/map"
android:name="com.google.android.gms.maps.SupportMapFragment"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:layout_alignParentStart="true"
android:layout_alignParentTop="true"
android:layout_alignParentEnd="true"
android:layout_alignParentBottom="true"
tools:ignore="FragmentTagUsage" />
```

Java Code:

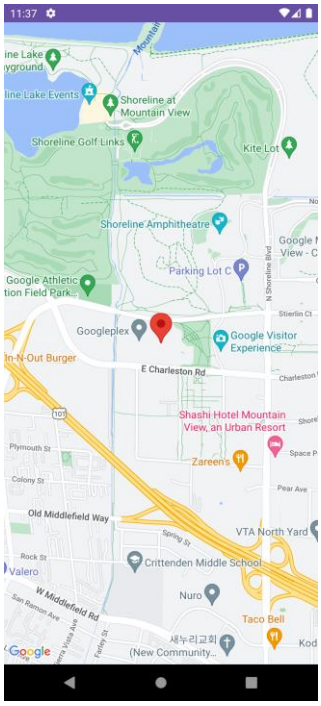
MainActivity.java:

```
package com.example.exp10;
import androidx.annotation.NonNull;
import androidx.core.app.ActivityCompat;
import androidx.fragment.app.FragmentActivity;
import android.content.pm.PackageManager;
import android.location.Location;
```

```
import android.location.LocationListener;

import android.location.LocationManager;

import android.os.Bundle;

import android.Manifest;

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;

public class MainActivity extends FragmentActivity implements OnMapReadyCallback {

private GoogleMap mMap;

private LatLng latLng;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity_main);

// Obtain the SupportMapFragment and get notified when the map is ready to be used.

SupportMapFragment mapFragment = (SupportMapFragment)
getSupportFragmentManager().findFragmentById(R.id.map);

if (mapFragment != null) {

mapFragment.getMapAsync(this);}

ActivityCompat.requestPermissions(this, new String[]{

Manifest.permission.ACCESS_FINE_LOCATION,

Manifest.permission.ACCESS_COARSE_LOCATION

}, PackageManager.PERMISSION_GRANTED);

}
```

```
@Override

public void onMapReady(@NonNull GoogleMap googleMap) {

mMap = googleMap;

LocationListener locationListener = new LocationListener() {

@Override

public void onLocationChanged(@NonNull Location location) {

latLng = new LatLng(location.getLatitude(), location.getLongitude());

mMap.clear();

mMap.addMarker(new MarkerOptions().position(latLng).title("My position"));

mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(latLng, 15));}};

LocationManager locationManager = (LocationManager) getSystemService(LOCATION_SERVICE);

try {

long MIN_DIST = 5;

long MIN_TIME = 1000;

locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, MIN_TIME, MIN_DIST,
locationListener);

} catch (SecurityException e) {

e.printStackTrace();

}}}
```

Output:



Conclusion: