

|  |
| --- |
| **BATCH AND ROLL NO: R6 - 42257** |
| **EXPERIMENT NO.10** |
| **TITLE:** Design a mobile app using Google Map and GPS to trace the location. |
| **DATE OF PERFORMANCE** |
| **DATE OF CHECKING** |

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

**Requirements:**

1 Android studio

2.Google Play service Packages

**Theory:**

Android allows us to integrate Google Maps in our application. For this Google provides us a library via Google Play Services for using maps. In order to use the Google Maps API, you must register your application on the **Google Developer Console** and enable the API.

Follow these steps to create the project in Android Studio.

**Step 1. Install the Google Play Services SDK**

Add Google Play services to Android Studio. To make the Google Play services APIs available to your app:

1. Open the build.gradle file inside your application module directory.
2. Add a new build rule under dependencies for the latest version of play-services, using one of the APIs listed below.

Implementation ‘com.google.android.gms:play-services-maps:+ googlePlayVersion’ (other Google Play API

https://developers.google.com/android/guides/setup)

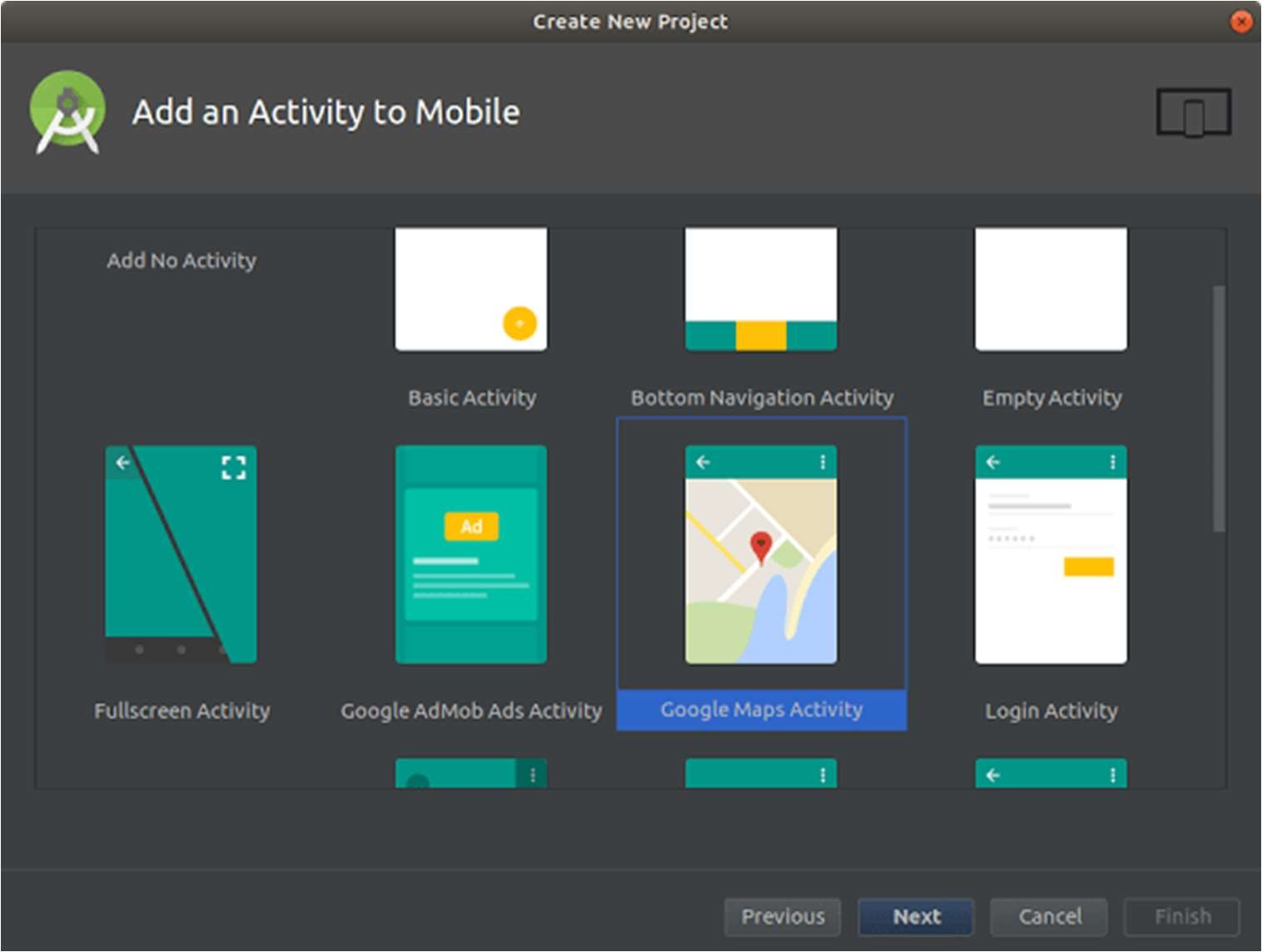
1. Save the changes and click Sync Project with Gradle Files in the toolbar.
2. If you receive an error, check that your top-level build.gradle contains a reference to the google() repo.



**Step 2. Create a new Google Maps project**

These are the steps to create a new project with Google Maps activity:

1. Start Android Studio.
2. Start a new project::
   * Click **Start a new Android Studio project** in the Welcome to Android Studio dialog
   * If you don’t see this dialog, then click **File –> New –> New Project**.
3. Fill in the app name, company domain, and location.
4. Choose form factors, for example Phone and Tablet.
5. Select the Google Maps Activity in the **Add an Activity to Mobile** dialog.





1. After filling in the activity name and title, click **Finish**. Don’t touch the default values – they’re okay.

After a few seconds, your project will be built by Gradle. Then you’ll see google\_maps\_api.xml and the MapsActivity.java files in the editor.**Step 3. Get a Google Maps API key**

The Android Maps API is used to access Google’s servers, and there are several ways you can get a key. An API key is free and supports any number of users.

* + The easiest way to get a key is to use the link provided in the google\_maps\_api.xml file that Android Studio created for you:

1. Copy the link and paste it into your browser. It will direct you to the Google API Console and you won’t have to fill in anything by yourself.
2. Select a project you’ve created before or create a new one.
3. Create an Android-restricted API key for your project.
4. Copy the API key, go back to Android Studio, and paste it into the element in the google\_maps\_api.xml file.
   * Another option for getting an API key is this:

Use the credentials provided in the google\_maps\_api.xml file that Android Studio created for you:

1. Copy the credentials provided in the google\_maps\_api.xml file.
2. Go to the Google API Console in your browser.
3. Use the copied credentials to add your app to an existing API key or to create a new API key.

**Step 4. Check your code**

Now, when everything is set up, look at the code and check these files in your project.



**Code:**

**package com.example.mapsfun;**

**import androidx.fragment.app.FragmentActivity; import android.os.Bundle;**

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

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

**public class MapsActivity extends FragmentActivity implements OnMapReadyCallback {**

**private GoogleMap mMap; @Override**

**protected void onCreate(Bundle savedInstanceState) {**

**super.onCreate(savedInstanceState); setContentView(R.layout.activity\_maps);**

**// 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;**



**mMap.setMapType(GoogleMap.MAP\_TYPE\_HYBRID);**

**// Add a marker in Sydney and move the camera LatLng pune = new LatLng(18.5245649,73.7228812);**

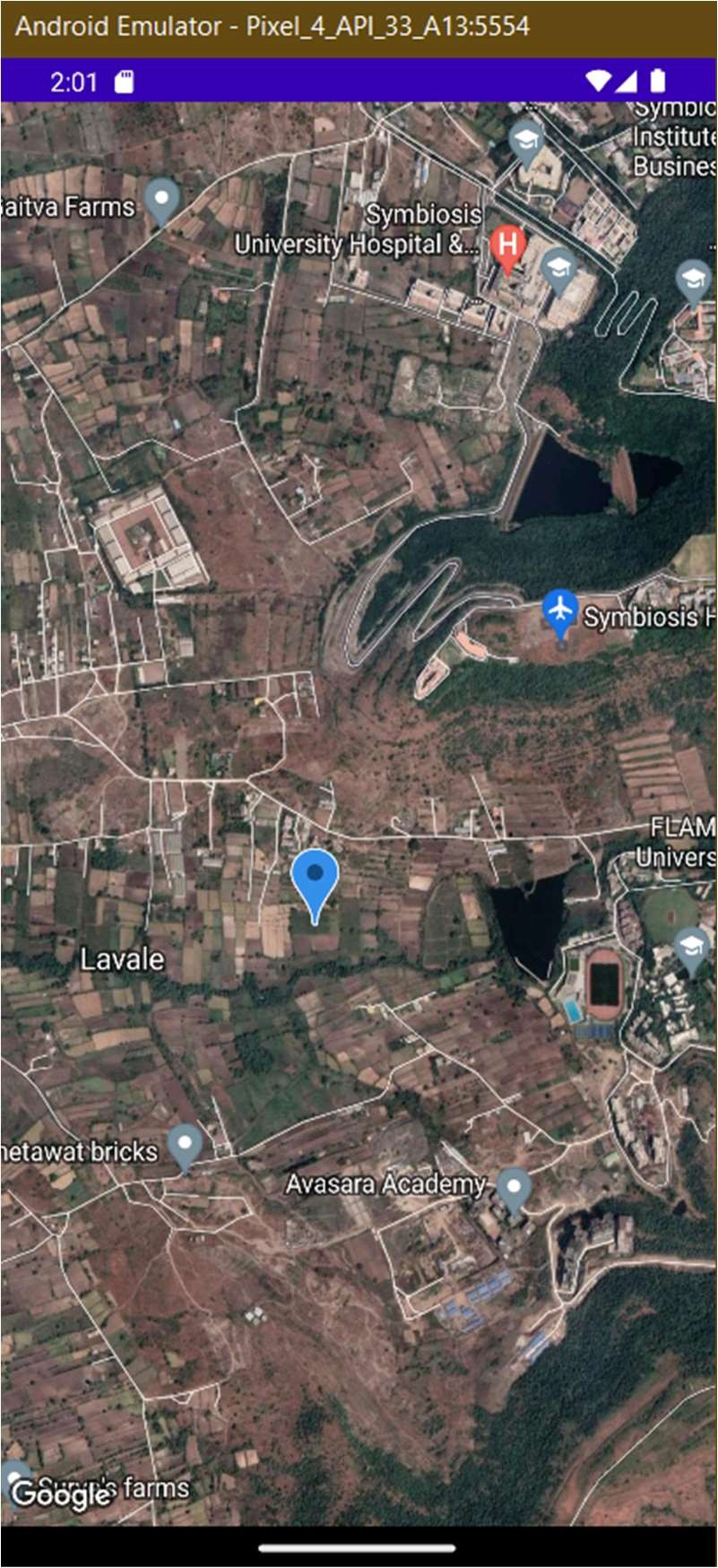
**mMap.addMarker(new MarkerOptions().position(pune).title("Puneri Pagdi").icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.H UE\_AZURE)));**

**mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(pune, 15));**

**}**

**}**

**Output:**





**CONCLUSION:**

**……………………………………………………………………………………………**

**……………………………………………………………………………………………**

**……………………………………………………………………………………………**