Security and Application Deployment

Prof. Prasad Koyande
Vidyalankar Polytechnic

SMS TELEPHONY

- Android devices can send and receive messages to or from any other phone that supports Short Message Service (SMS).

We can add code to our app to:

1. Launch an SMS messaging app from our app to handle all SMS communication.
2. Send an SMS message from within our app.
3. Receive SMS messages in our app.

Sending and Receiving SMS Messages

- App need the user's permission to directly use SMS features.
- Use an implicit Intent to launch a messaging app such as Messenger, with the ACTION_SENDTO action.
- Send the SMS message using the sendTextMessage() method or other methods of the SmsManager class.
- To receive SMS messages, the best practice is to use the onReceive() method of the BroadcastReceiver.
- Our app receives SMS messages by listening for the SMS_RECEIVED_ACTION broadcast.
- PDU (Protocol Data Unit) contains not only the SMS message, but also metadata about the SMS message, such as text encoding, the sender, SMS service center address, and much more.

SMS Example

SMS Example

- Manifest File
- Design File
- Java File

ANDROID SECURITY MODEL

- Android is a multi-process system, in which each application (and parts of the system) runs in its own process.
- Most security between applications and the system is enforced at the process level through standard Linux facilities, such as user and group IDs that are assigned to applications.
- Additional finer-grained security features are provided through a “permission” mechanism that enforces restrictions on the specific operations that a particular process can perform, and per-URI permissions for granting ad-hoc access to specific pieces of data.

- Android application has been signed with a certificate with a private key Know the owner of the application is unique. This allows the author of The application will be identified if needed
- When an application is installed in The phone is assigned a user ID, thus avoiding it from affecting it Other applications by creating a sandbox for it.

- The protection level affects whether runtime permission requests are required. There are three protection levels that affect third-party apps: **normal, signature and dangerous permissions**

Android Threat

1. Leaking Information to Logs
   - Android provides centralized logging via the Log API, which can displayed with the “logcat” command.
2. SDcard Use
   - Any application that has access to read or write data on the Sdcard
3. Wifi Sniffing
   - This may disrupt the data being transmitted from A device like many web sites and applications does not have security measures strict security

Declaring and Using Permissions

- Permission Approval
  - An app must publicize the permissions it requires by including tags in the app manifest.
- Request Prompts for Dangerous Permissions
  - Android asks the user to grant dangerous permissions depends on the version of Android running on the user's device
- Runtime Requests
  - If the device is running Android 6.0 (API level 23) or higher, and the app's targetSdkVersion is 23 or higher, the user isn't notified of any app permissions at install time.
  - Our app must ask the user to grant the dangerous permissions at runtime.

- Request prompts to Access Sensitive user Information:
- Permissions for Optional Hardware Features

Permission Enforcement

- Permissions aren't only for requesting system functionality.
- Services provided by apps can enforce custom permissions to restrict who can use them.
- Service Permission Enforcement:
  - Permissions applied using the android:permission attribute to the tag in the manifest restrict who can start or bind to the associated Service.

- Broadcast Permission Enforcement
  - Permissions applied using the android:permission attribute to the tag restrict who can send broadcasts to the associated BroadcastReceiver.
- Content Provider Permission Enforcement
  - Permissions applied using the android:permission attribute to the tag restrict who can access the data in a ContentProvider,
- URI Permissions
  - Use fine-grained permissions are declare our app's support for it with the android:grantUriPermissions attribute or tag.

Protection Levels

1. Normal Permissions
2. Signature Permissions
3. Dangerous Permissions

Permission Groups

- Permissions are organized into groups related to a device's capabilities or features.
- Under this system, permission requests are handled at the group level and a single permission group corresponds to several permission declarations in the app manifest
- For example, the SMS group includes both the READ_SMS and the RECEIVE_SMS declarations.

Using Custom Permission

- App Signing
- User IDs and File Access
- Defining and Enforcing Permissions

```
<manifest
  xmlns:android="http://schemas.android.com/apk/res/android"
  package="com.example.myapp">    <permission
      android:name="com.example.myapp.permission.DEADLY_ACTIVITY"
      android:label="@string/permlab_deadlyActivity"
      android:description="@string/permdesc_deadlyActivity"
      android:permissionGroup="android.permission-group.COST_MONEY"
      android:protectionLevel="dangerous"/>
    ...
</manifest>
```

APPLICATION DEPLOYMENT

- Publishing is the general process that makes our Android applications available to users. When we publish an Android application we perform two main tasks
  1. We prepare the application for release
  2. We release the application to users

Preparing Our App For Release

1. Configuring our Application for Release
2. Building and Signing a Release Version of Our Application
3. Testing the Release Version of Our Application
4. Updating Application Resources for Release
5. Preparing Remote Servers and Services that Our Application Depends on

Signing of Application

- Application signing allows developers to identify the author of the application and to update their application without creating complicated interfaces and permissions.
- Every application that is run on the Android platform must be signed by the developer.
- Applications that attempt to install without being signed will be rejected by either Google Play or the package installer on the Android device.

APK Signing Schemes

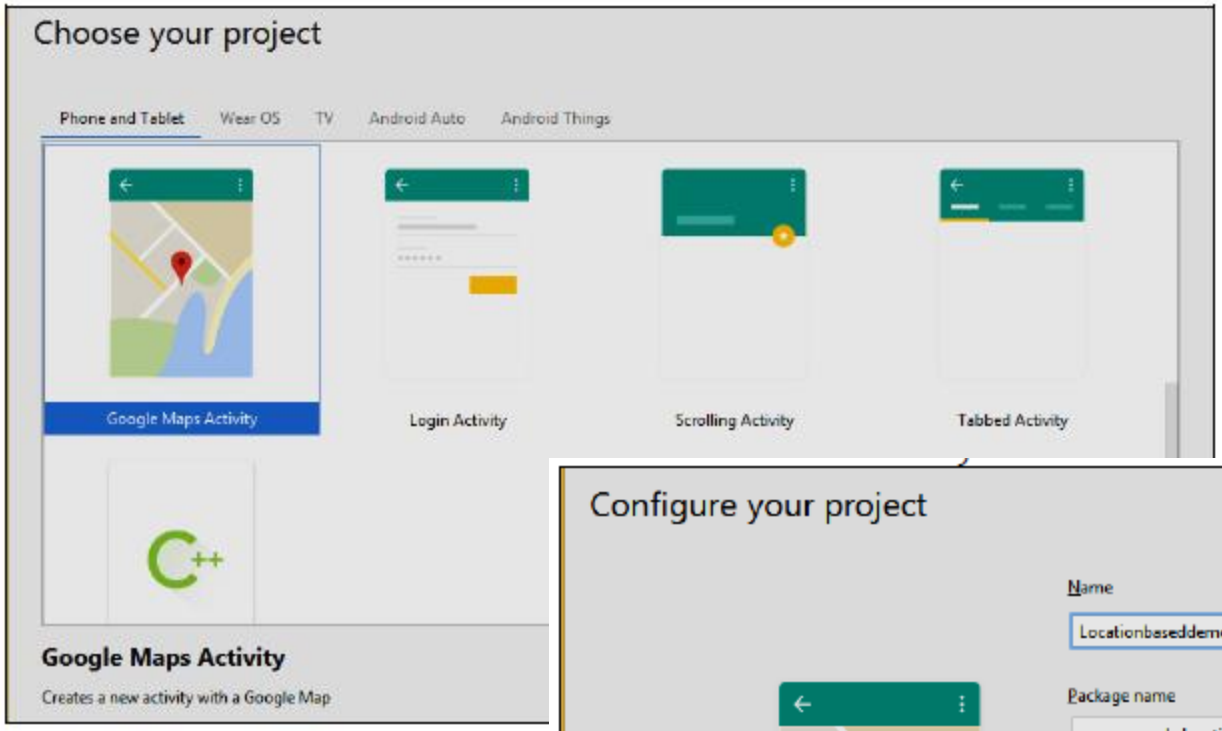
1. v1 Scheme: based on JAR signing
2. v2 Scheme: APK Signature Scheme v2, which was introduced in Android 7.0.
3. v3 Scheme: APK Signature Scheme v3, which was introduced in Android 9.

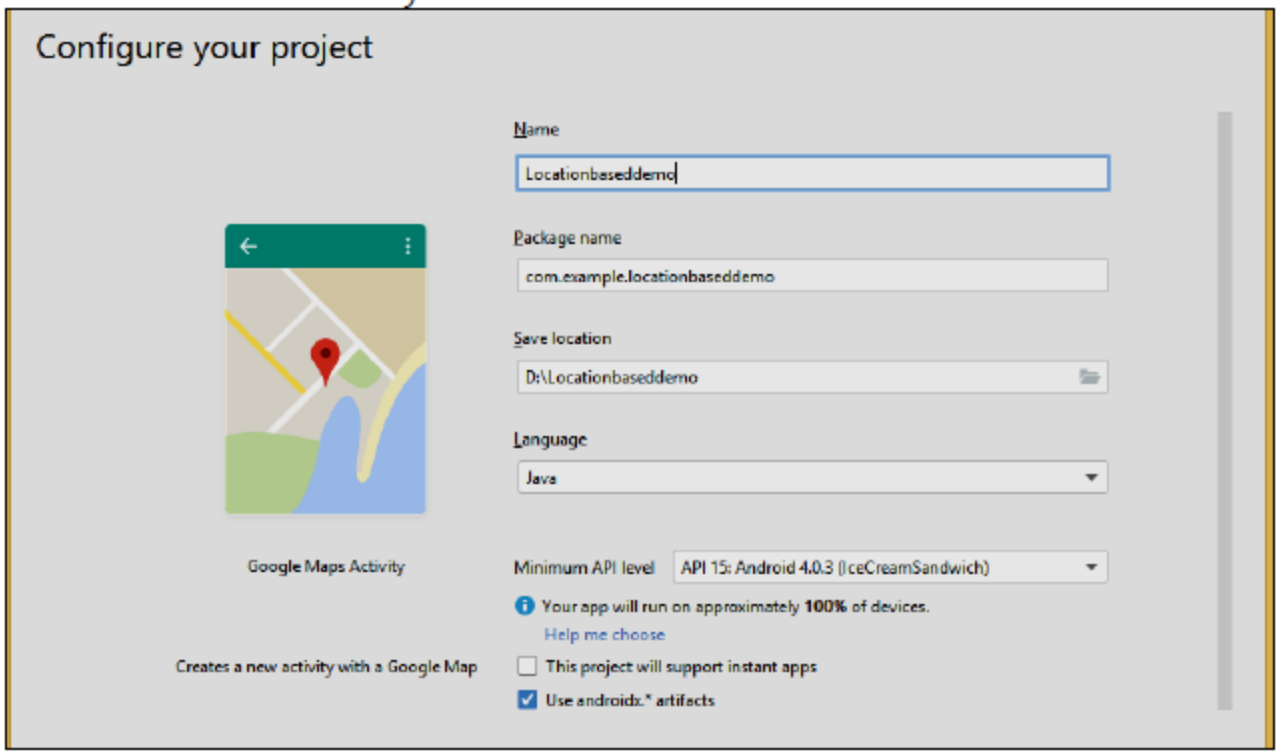
Deploying App on Google Play Store

Play Video

LOCATION BASED SERVICES (LBSs)

Creating the Project





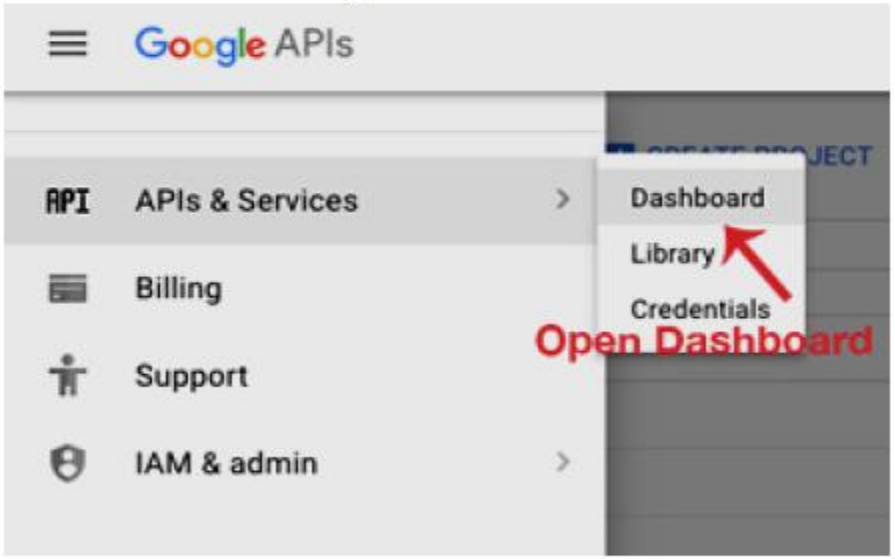
Getting the Maps API Key

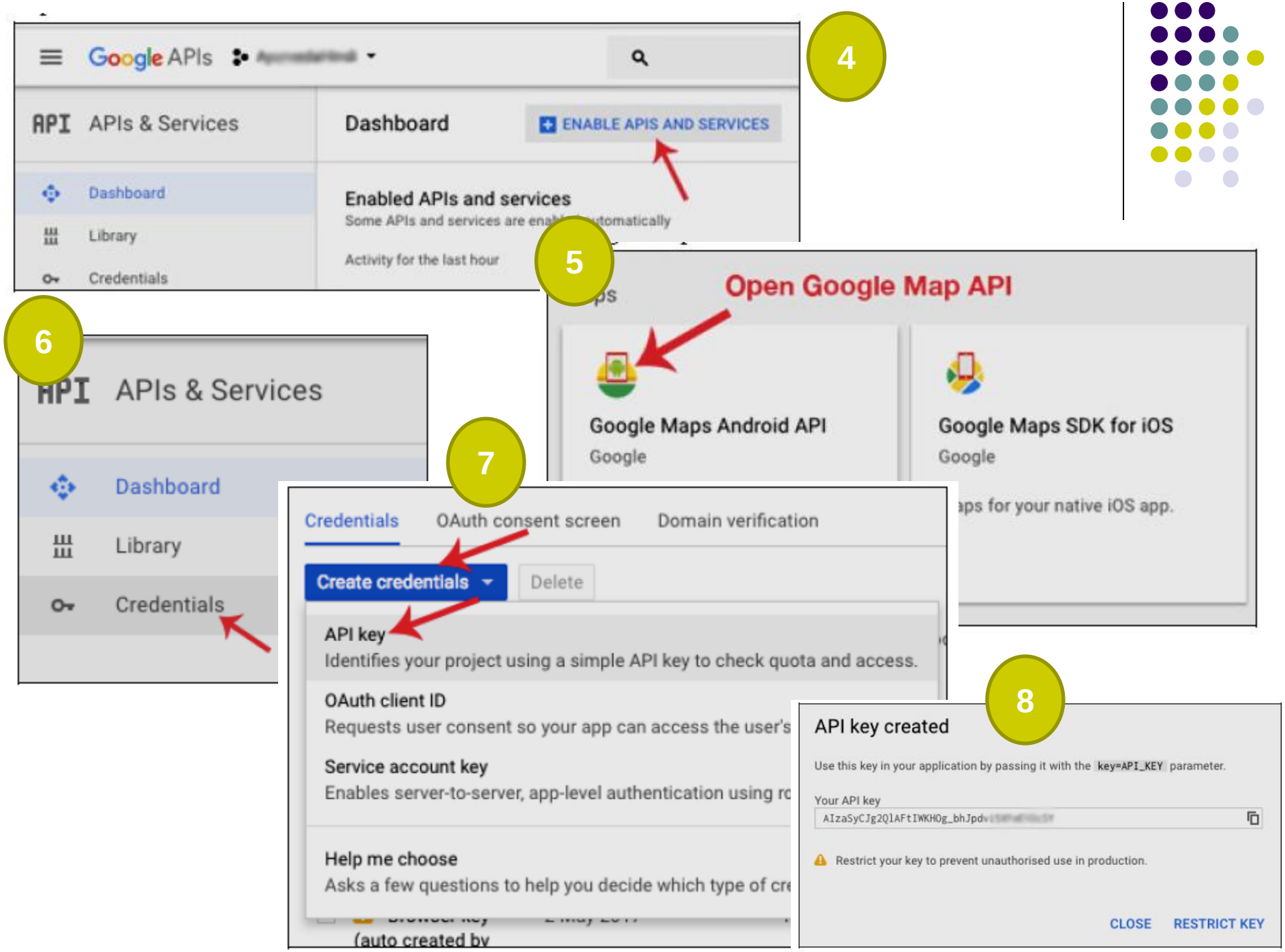
1

https://console. developers. google.com/project

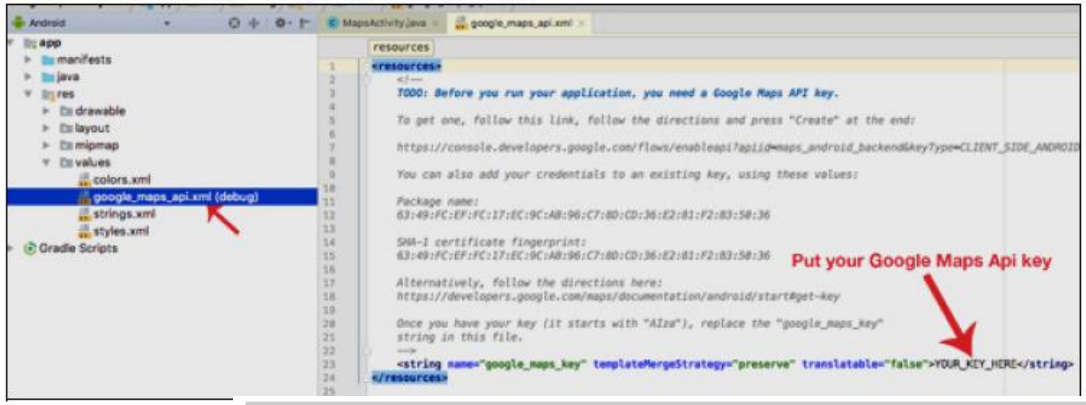
2

3




4
Google APIs
APIs & Services
Dashboard
ENABLE APIS AND SERVICES
Dashboard
Library
Credentials
Enabled APIs and services
Activity for the last hour
5
Open Google Map API
Google Maps Android API
Google
Google Maps SDK for iOS
Google
6
APIs & Services
Dashboard
Library
Credentials
7
Credentials
OAuth consent screen
Domain verification
Create credentials
Delete
API key
Identifies your project using a simple API key to check quota and access.
OAuth client ID
Requests user consent so your app can access the user's
Service account key
Enables server-to-server, app-level authentication using ro
Help me choose
Asks a few questions to help you decide which type of cre
8
API key created
Use this key in your application by passing it with the key=API_KEY parameter.
Your API key
AIzaSyCJg2QlAFtIWKHOg_bhJpd
Restrict your key to prevent unauthorised use in production.
CLOSE
RESTRICT KEY

Displaying the Maps




```
<resources>
<!--
    TODO: Before we run our application, we need a Google Maps API key.

    To get one, follow this link, follow the directions and press "Create" at the end:

https://console.developers.google.com/flows/enableapi?apiid=maps_android_backend&keyType
=CLIENT_SIDE_ANDROID&r=8B:49:70:2A:08:F2:23:14:CF:A1:FC:6F:6D:5B:60:3C:B6:85:98:F2%3Bcom
.example.abhishek.googlemaps

    We can also add our credentials to an existing key, using these values:

    Package name:
    8B:49:70:2A:08:F2:23:14:CF:A1:FC:6F:6D:5B:60:3C:B6:85:98:F2

    SHA-1 certificate fingerprint:
    8B:49:70:2A:08:F2:23:14:CF:A1:FC:6F:6D:5B:60:3C:B6:85:98:F2
    Alternatively, follow the directions here:
    https://developers.google.com/maps/documentation/android/start#get-key

    Once we have our key (it starts with "AIza"), replace the "google_maps_key"
string in this file.
    -->
<stringname="google_maps_key"templateMergeStrategy="preserve"translatable="false">AIzaSy
DV2_xy58r15K6TskZy4KWMuhUDVq67jqM</string>
</resources>
```

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

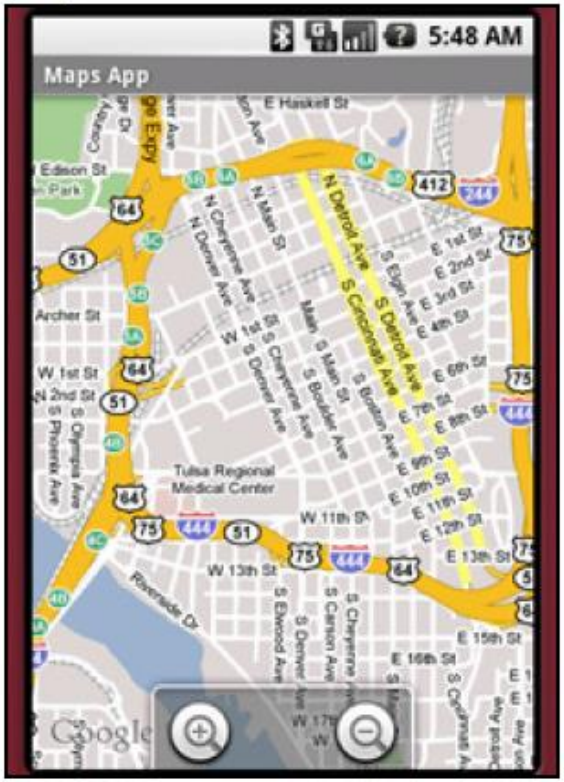
```
import androidx.fragment.app.FragmentActivity;
import android.os.Bundle;
import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.OnMapReadyCallback;
import com.google.android.gms.maps.SupportMapFragment;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.MarkerOptions;
public class MapsActivityextends FragmentActivityimplements OnMapReadyCallback {
private GoogleMapmMap;
    @Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_maps);
SupportMapFragmentmapFragment = (SupportMapFragment) getSupportFragmentManager()
                .findFragmentById(R.id.map);
mapFragment.getMapAsync(this);
    }
@Override
public void onMapReady(GoogleMapgoogleMap) {
mMap= googleMap;

LatLngsydney = new LatLng(-34, 151);
mMap.addMarker(new MarkerOptions().position(sydney).title("Marker in Sydney"));
mMap.moveCamera(CameraUpdateFactory.newLatLng(sydney));
    }
}
```

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

<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
 <application
 android:allowBackup="true"
 android:icon="@mipmap/ic_launcher"
 android:label="@string/app_name"
 android:roundIcon="@mipmap/ic_launcher_round"
 android:supportsRtl="true"
 android:theme="@style/AppTheme">

 <meta-data
 android:name="com.google.android.geo.API_KEY"
 android:value="AIzaSyABJ6evNQ52va9Rucu_sU7Tjpxvb433-9A" />

 <activity
 android:name=".MapsActivity"
 android:label="@string/title_activity_maps">
 <intent-filter>
 <action android:name="android.intent.action.MAIN" />

 <category android:name="android.intent.category.LAUNC
 </intent-filter>
 </activity>
 </application>

 </manifest>
```

Displaying the Zoom Control

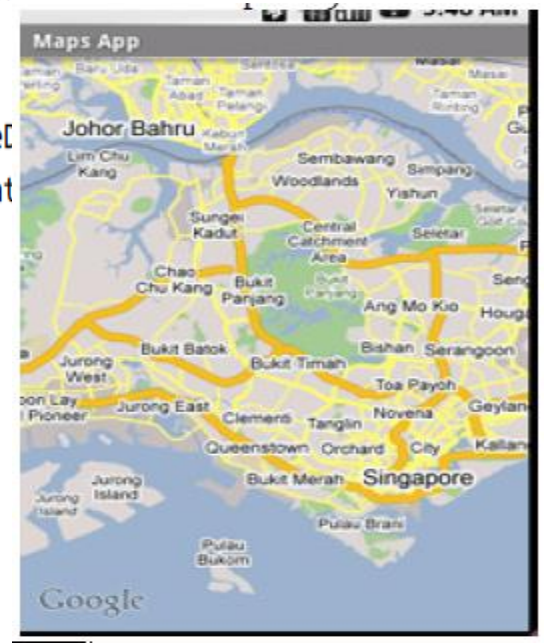
```
<RelativeLayoutxmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <com.google.android.maps.MapView
        android:id="@+id/mapView"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:enabled="true"
        android:clickable="true"
        android:apiKey="0l4sCTTyRmXTNo7k8DREHvEaLar2UmHGwnhZVHQ"
        />
    <LinearLayoutandroid:id="@+id/zoom"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_centerHorizontal="true"
        />
</RelativeLayout>
```

```
public class MapsActivity extends MapActivity
{
    MapViewmapView;
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        mapView = (MapView) findViewById(R.id.mapView);
        LinearLayoutzoomLayout = (LinearLayout)findViewById(R.id.zoom);
        View zoomView = mapView.getZoomControls();
        zoomLayout.addView(zoomView,
            newLinearLayout.LayoutParams(
                LayoutParams.WRAP_CONTENT,
                LayoutParams.WRAP_CONTENT));
        mapView.displayZoomControls(true);
    }
    @Override
    protectedbooleanisRouteDisplayed() {
        // TODO Auto-generated method stub
        return false;
    }
}
```

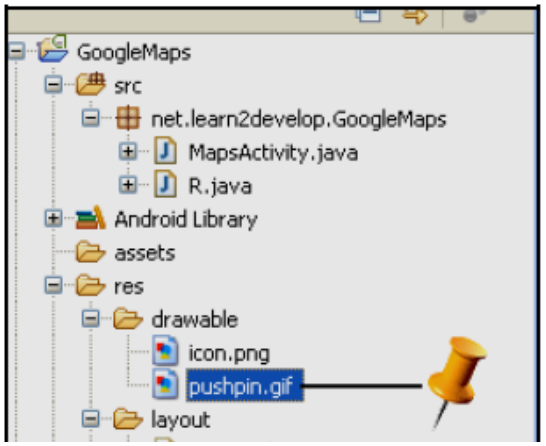


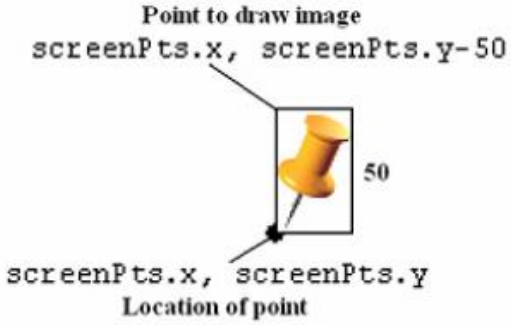
Navigating to a Specific Location

```
importcom.google.android.maps.GeoPoint;
importcom.google.android.maps.MapActivity;
importcom.google.android.maps.MapController;
importcom.google.android.maps.MapView;
importcom.google.android.maps.MapView.LayoutParams;
importandroid.os.Bundle;
importandroid.view.View;
importandroid.widget.LinearLayout;
public class MapsActivity extends MapActivity
{
    MapViewmapView;
    MapController mc;
    GeoPoint p;
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        mapView = (MapView) findViewById(R.id.mapView);
        LinearLayoutzoomLayout = (LinearLayout)findViewById(R.id.zoom);
        View zoomView = mapView.getZoomControls();
        zoomLayout.addView(zoomView,
            newLinearLayout.LayoutParams(
                LayoutParams.WRAP_CONTENT,
                LayoutParams.WRAP_CONTENT));
        mapView.displayZoomControls(true);
        mc = mapView.getController();
        String coordinates[] = {"1.352566007", "103.78921587"};
        doublelat = Double.parseDouble(coordinates[0]);
        doublelng = Double.parseDouble(coordinates[1]);
        p = new GeoPoint(
            (int) (lat * 1E6),
            (int) (lng * 1E6));
        mc.animateTo(p);
        mc.setZoom(17);
        mapView.invalidate();
    }
    @Override
    protectedbooleanisRoute[
        // TODO Auto-generat
        return false;
    }
}
```



Adding Markers

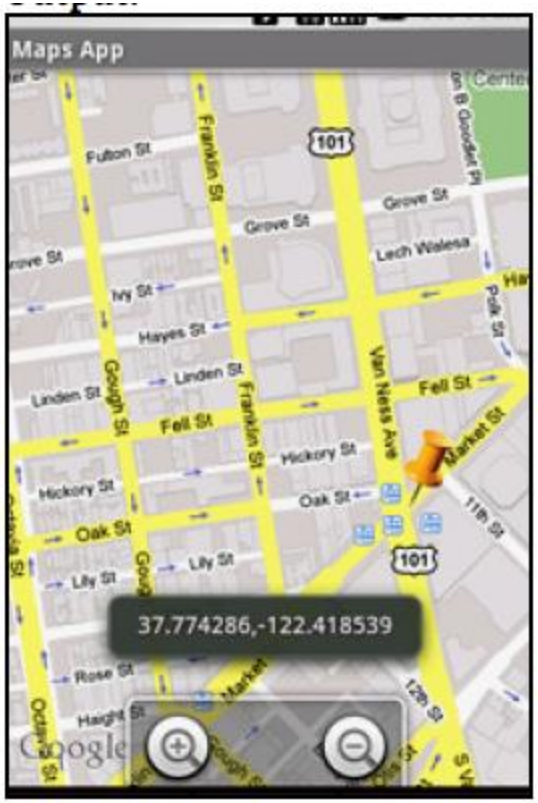





```
@Override
   public void onCreate(Bundle savedInstanceState)
   {
       super.onCreate(savedInstanceState);
       setContentView(R.layout.main);
       //...
       mc.animateTo(p);
       mc.setZoom(17);
       //---Add a location marker---
       MapOverlaymapOverlay = new MapOverlay();
       List<Overlay>listOfOverlays = mapView.getOverlays();
       listOfOverlays.clear();
       listOfOverlays.add(mapOverlay);
       mapView.invalidate();
   }
```

```
public class MapsActivity extends MapActivity
{
    MapViewmapView;
    MapController mc;
    GeoPoint p;
    classMapOverlay extends com.google.android.maps.Overlay
    {
        @Override
        publicboolean draw(Canvas canvas, MapViewmapView,
        boolean shadow, long when)
        {
            super.draw(canvas, mapView, shadow);
            //---translate the GeoPoint to screen pixels---
            Point screenPts = new Point();
            mapView.getProjection().toPixels(p, screenPts);
            //---add the marker---
            Bitmap bmp = BitmapFactory.decodeResource(
                getResources(), R.drawable.pushpin);
            canvas.drawBitmap(bmp, screenPts.x, screenPts.y-50, null);
            return true;
        }
```

Getting the Location that was Touched

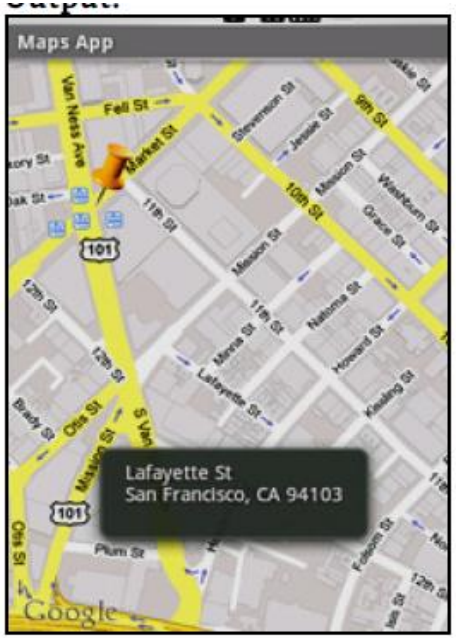



```
classMapOverlay extends com.google.android.maps.Overlay
    {
        @Override
        publicboolean draw(Canvas canvas, MapViewmapView,
        boolean shadow, long when)
        {
            //...
        }

        @Override
        publicbooleanonTouchEvent(MotionEvent event, MapViewmapView)
        {
            //---when user lifts his finger---
            if (event.getAction() == 1) {
                GeoPoint p = mapView.getProjection().fromPixels(
                    (int) event.getX(),
                    (int) event.getY());
                    Toast.makeText(getBaseContext(),
                        p.getLatitudeE6() / 1E6 + "," +
                        p.getLongitudeE6() /1E6 ,
                        Toast.LENGTH_SHORT).show();

                }
                return false;
            }
        }
```

Geocoding and Reverse Geocoding

```
classMapOverlay extends com.google.android.maps.Overlay
    {
        @Override
        publicboolean draw(Canvas canvas, MapViewmapView,
        boolean shadow, long when)
        {
          //...
        }

        @Override
        publicbooleanonTouchEvent(MotionEvent event, MapViewmapView)
        {
            //---when user lifts his finger---
            if (event.getAction() == 1) {
                GeoPoint p = mapView.getProjection().fromPixels(
                    (int) event.getX(),
                    (int) event.getY());

                Geocoder geoCoder = new Geocoder(
                    getBaseContext(), Locale.getDefault());
                try {
                    List<Address> addresses = geoCoder.getFromLocation(
```

```
                p.getLatitudeE6()  / 1E6,
                p.getLongitudeE6() / 1E6, 1);

            String add = "";
            if (addresses.size() > 0)
            {
                for (inti=0; i<addresses.get(0).getMaxAddressLineIndex();
                     i++)
                   add += addresses.get(0).getAddressLine(i) + "n";
            }

            Toast.makeText(getBaseContext(), add, Toast.LENGTH_SHORT).show();
        }
        catch (IOException e) {
            e.printStackTrace();
        }
        return true;
    }
    else
        return false;
  }
}
```



Getting Location Data

Monitoring a Location

```
importandroid.app.PendingIntent;
importandroid.content.Intent;
importandroid.net.Uri;
 //---use the LocationManager class to obtain locations data---
lm = (LocationManager)
getSystemService(Context.LOCATION_SERVICE);
 //---PendingIntent to launch activity if the user is within
 // some locations---
PendingIntentpendingIntent = PendingIntent.getActivity(
this, 0, new
Intent(android.content.Intent.ACTION_VIEW,
Uri.parse("http://www.amazon.com")), 0);
lm.addProximityAlert(37.422006, -122.084095, 5, -1, pendingIntent);
```