Mobile Programming

[week 12]

[Map and Location]

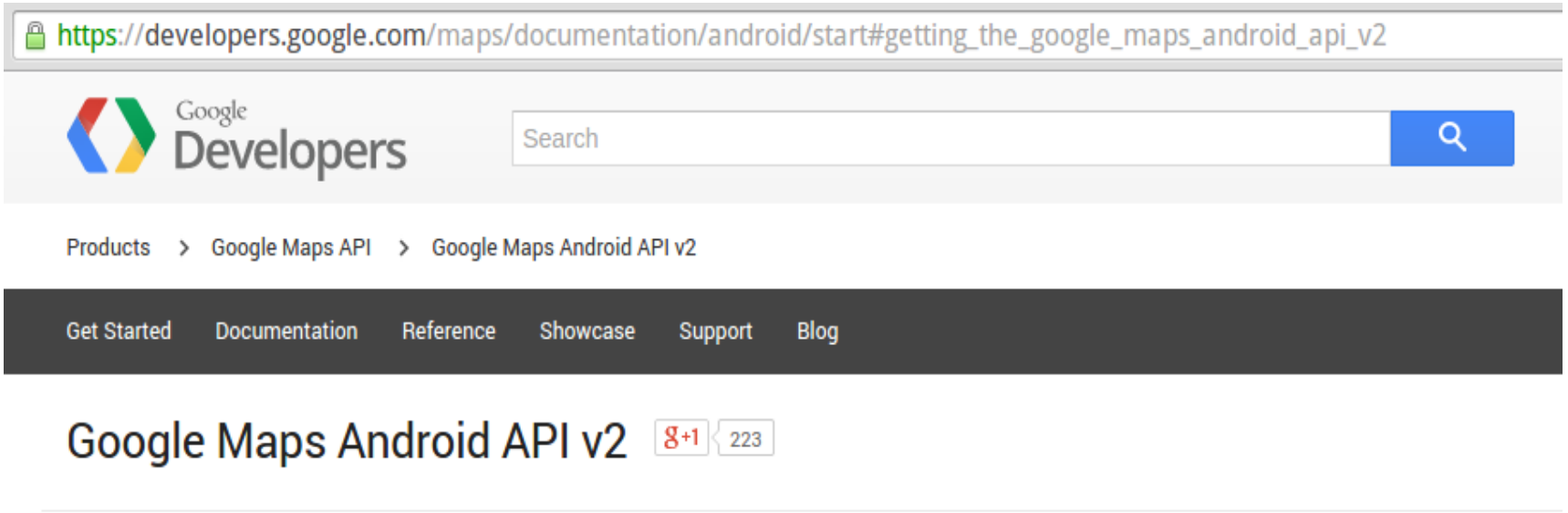
Hilmy A. T.

hilmi.tawakal@gmail.com

Google Maps Android API

Google Maps Android API

Current version of Google Maps Android API is v2



Google Maps Android API

- With the Google Maps Android API, you can add maps to your app that are based on Google Maps data.
- The API automatically handles access to Google Maps servers, data downloading, map display, and touch gestures on the map.
- You can also use API calls to add markers, polygons and overlays, and to change the user's view of a particular map area.

Google Maps Android API

- The key class in the Google Maps Android API is MapView.
- A MapView displays a map with data obtained from the Google Maps service.
- When the MapView has focus, it will capture keypresses and touch gestures to pan and zoom the map automatically, including handling network requests for additional maps tiles.

Google Maps Android API

- It also provides all of the UI elements necessary for users to control the map.
- Your application can also use MapView class methods to control the map programmatically and draw a number of overlays on top of the map.

Getting Started

- Creating a new Android application that uses the Google Maps Android API v2 requires several steps.
- Many of the steps outlined in here will only have to be performed once, but some of the information will be a handy reference for future applications.

Getting Started

The overall process of adding a map to an Android application is as follows:

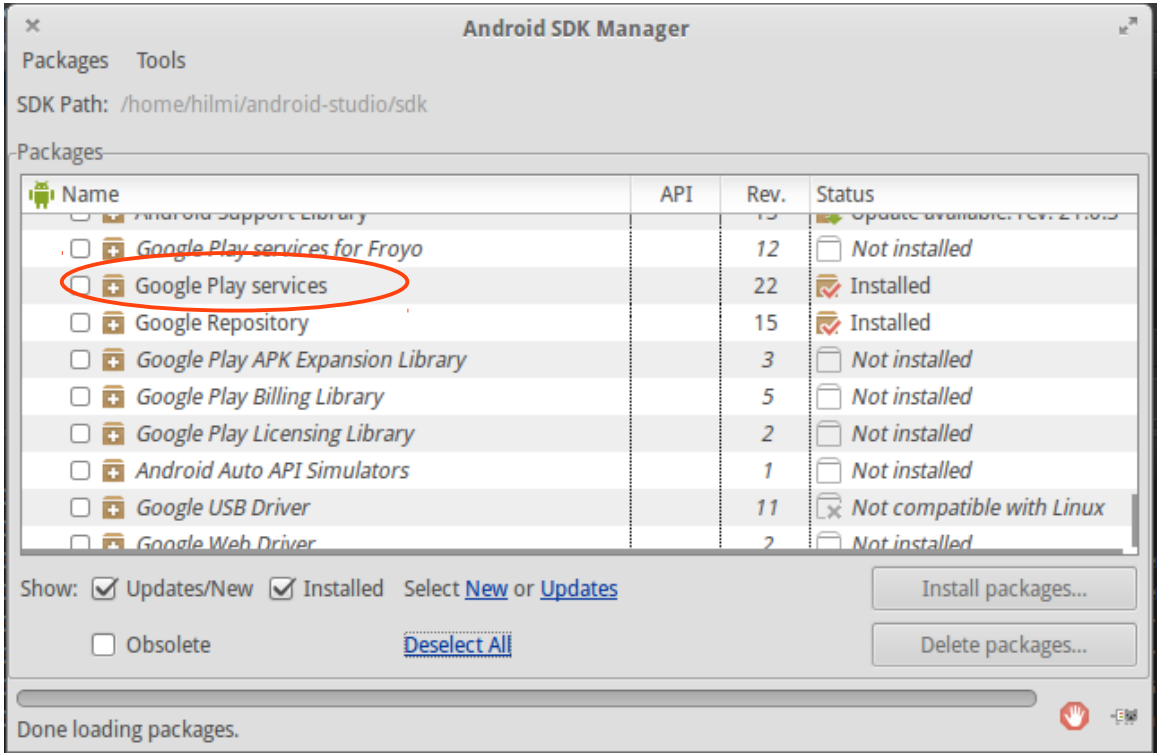
1. Install the android SDK
2. Download and configure the Google Play services SDK, which includes the Google Maps Android API.
3. Obtain an API key
4. Add the required settings in your application's manifest.
5. Add a map to your application.

Configure the Google Play services

- The Google Maps Android APIs are not included in the Android platform, but are available on any device with the Google Play Store running Android 2.2 or higher, through Google Play services.
- To integrate Google Maps into your app, you need to install the Google Play services libraries for your Android SDK.

Configure the Google Play services

You can download the Google Play services SDK via the Android SDK Manager.



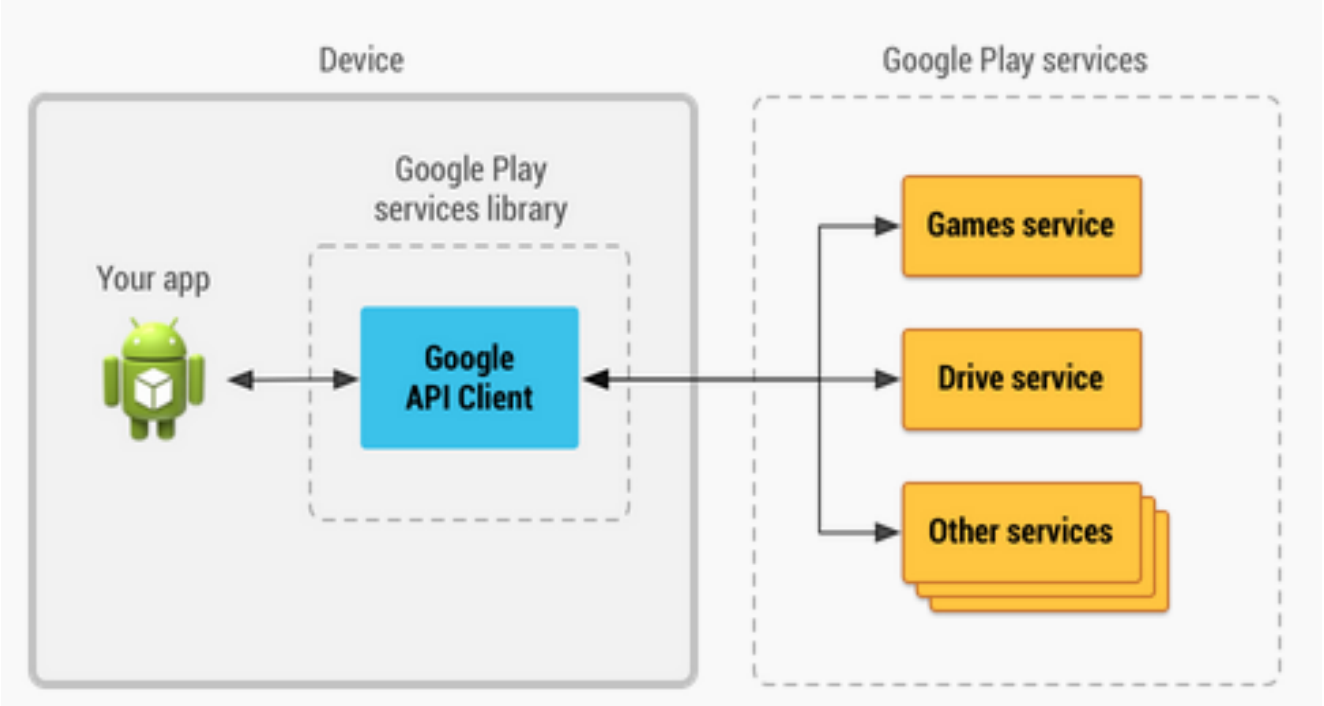
Configure the Google Play services

Edit your application's AndroidManifest.xml file, and add the following declaration within the <application> element. This embeds the version of Google Play services that the app was compiled with.

```
<meta-data
    android:name="com.google.android.gms.version"
    android:value="@integer/google_play_services_version" />
```

Configure the Google Play services

Using Google Play services



Obtain API Key

- The Google Maps Android API v2 uses a new system of managing keys. Existing keys from a Google Maps Android v1 application, commonly known as MapView, will not work with the v2 API.
- To access the Google Maps servers with the Maps API, you have to add a Maps API key to your application.

Obtain API Key

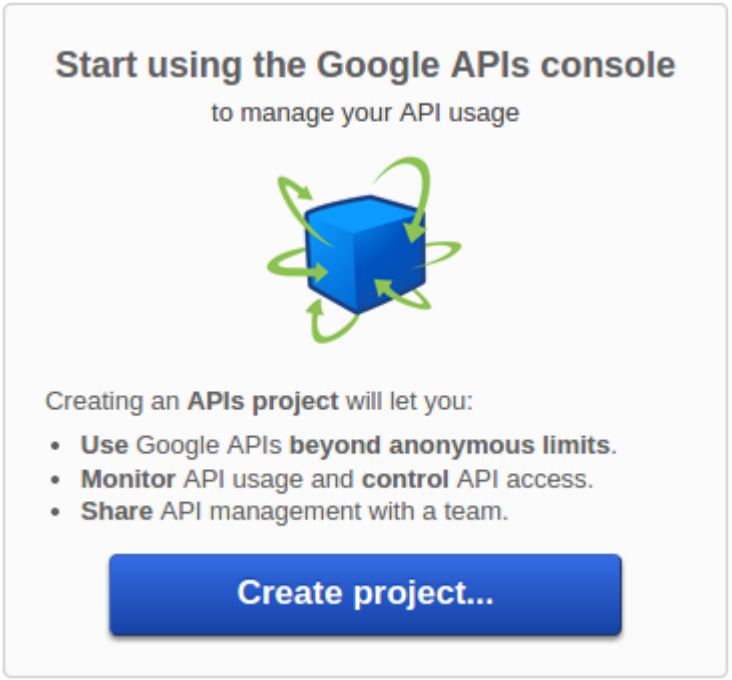
- The key is free, you can use it with any of your applications that call the Maps API, and it supports an unlimited number of users.
- You obtain a Maps API key from the Google APIs Console by providing your application's signing certificate and its package name.

Obtain API Key

1. Navigate to your project in the Google APIs Console.
2. In the Services page, verify that the "Google Maps Android API v2" is enabled.
3. In the left navigation bar, click API Access.
4. In the resulting page, click Create New Android Key....
5. In the resulting dialog, enter the SHA-1 fingerprint, then a semicolon, then your application's package name.
6. The Google APIs Console responds by displaying Key for Android apps (with certificates) followed by a forty-character API key.

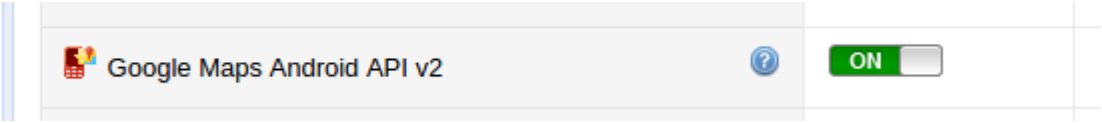
Obtain API Key

Go to google API console (https://code.google.com/apis/console)

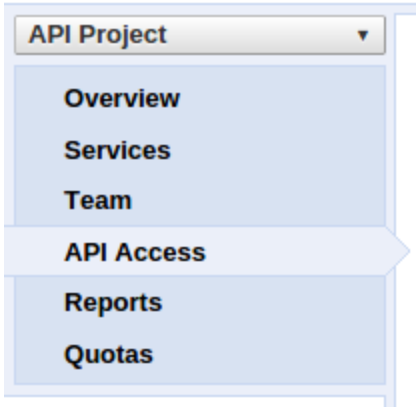


Obtain API Key

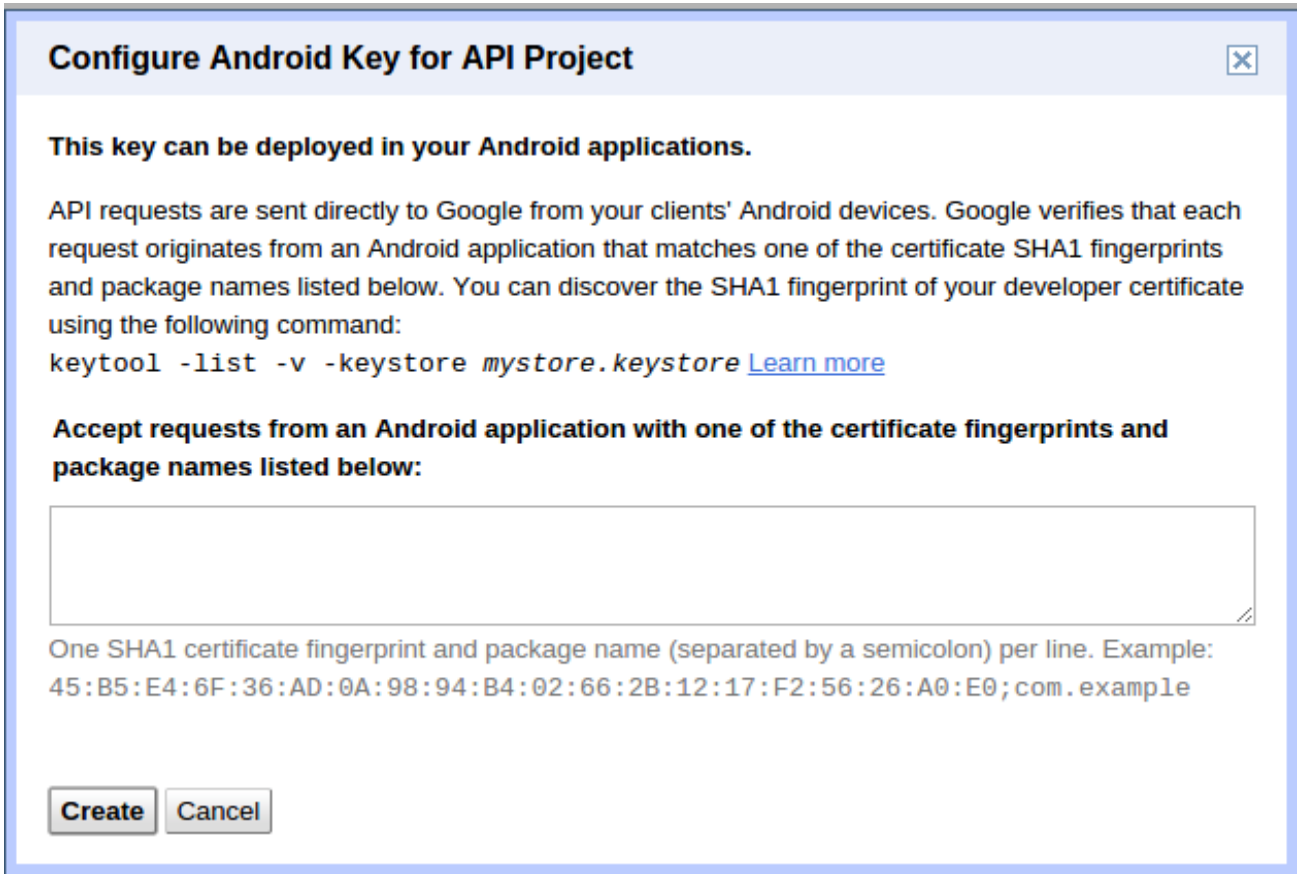
- Enable Google Maps Android API v2



- Go to API Access menu



Obtain API Key



Obtain API Key

Add generated API Key to your Application

```
<meta-data
    android:name="com.google.android.maps.v2.API_KEY"
    android:value="                                    " />
```

Specify app settings in the application manifest

Specify permissions

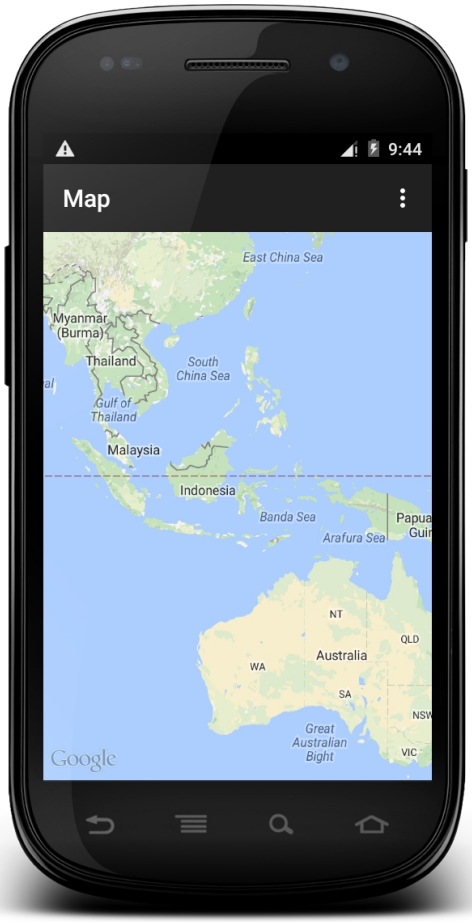
```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="com.google.android.providers.gsf.permission.READ_GSERVICES" />
<!--
The ACCESS_COARSE/FINE_LOCATION permissions are not required to use
    Google Maps Android API v2, but are recommended.
-->
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
```

Add Map to Application

Layout xml file

```
<fragment 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"
    android:id="@+id/map"
    tools:context=".MapsActivity"
    android:name="com.google.android.gms.maps.SupportMapFragment" />
```

Add Map to Application



Set initial location and zoom

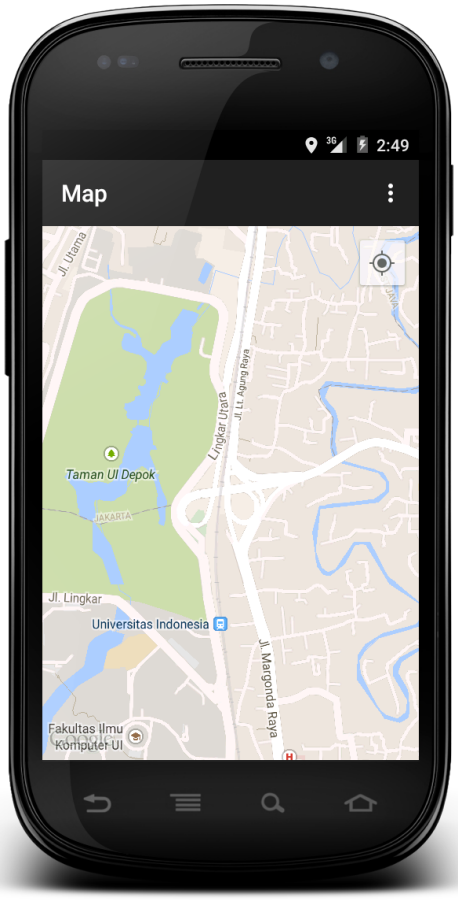
```
public class MainActivity extends ActionBarActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_maps);

        SupportMapFragment mapFragment = ((SupportMapFragment) getSupportFragmentManager()
                .findFragmentById(R.id.map));
        GoogleMap mMap = mapFragment.getMap();
        //set camera location
        LatLng depok = new LatLng(-6.355737, 106.832121);
        mMap.setMyLocationEnabled(true);
        mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(depok, 15));
    }

}
```

Set initial location and zoom



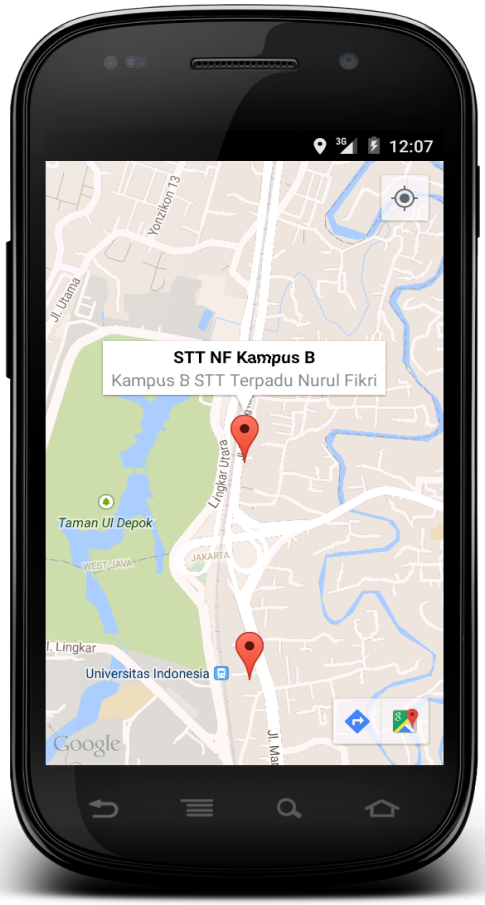
Add Marker to Map

```
LatLng kampus_a = new LatLng(-6.360732, 106.832733);
LatLng kampus_b = new LatLng(-6.352927, 106.832550);
LatLng depok = new LatLng(-6.355737, 106.832121);

mMap.setMyLocationEnabled(true);
mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(depok, 15));
mMap.addMarker(new MarkerOptions().title("STT NF Kampus A").snippet("Kampus A STT Terpadu Nurul Fikri")
        .position(kampus_a));

mMap.addMarker(new MarkerOptions().title("STT NF Kampus B").snippet("Kampus B STT Terpadu Nurul Fikri")
        .position(kampus_b));
```

Add Marker to Map



Google Location API

Making Your App Location-Aware

- One of the unique features of mobile applications is location awareness.
- Mobile users take their devices with them everywhere, and adding location awareness to your app offers users a more contextual experience.
- The location APIs available in Google Play services facilitate adding location awareness to your app with automated location tracking, geofencing, and activity recognition.

Getting the Last Known Location

- Using the Google Play services location APIs, your app can request the last known location of the user's device.
- In most cases, you are interested in the user's current location, which is usually equivalent to the last known location of the device.
- Specifically, use the fused location provider to retrieve the device's last known location.

Getting the Last Known Location

- The fused location provider is one of the location APIs in Google Play services.
- It manages the underlying location technology and provides a simple API so that you can specify requirements at a high level, like high accuracy or low power.
- It also optimizes the device's use of battery power.

Getting the Last Known Location

- Specify App Permissions

```
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
```

- Connect to Google Play Services

```
mGoogleApiClient = new GoogleApiClient.Builder(this)
        .addConnectionCallbacks(this)
        .addOnConnectionFailedListener(this)
        .addApi(LocationServices.API)
        .build();
```

Getting the Last Known Location

- Get Last Known Location

```
mLastLocation = LocationServices.FusedLocationApi.getLastLocation(
        mGoogleApiClient);
if (mLastLocation != null) {
    mLatitudeText.setText(String.valueOf(mLastLocation.getLatitude()));
    mLongitudeText.setText(String.valueOf(mLastLocation.getLongitude()));
}
```

Receiving Location Update

Implement Locationlistener and Override onLocationChange()

```
@Override
public void onLocationChanged(Location location) {
    String msg = "Updated Location: " +
            Double.toString(location.getLatitude()) + "," +
            Double.toString(location.getLongitude());
    Toast.makeText(this, msg, Toast.LENGTH_SHORT).show();
}
```

Specify Location Update Interval

```
// Create the LocationRequest object
mLocationRequest = LocationRequest.create();
// Use high accuracy
mLocationRequest.setPriority(
        LocationRequest.PRIORITY_HIGH_ACCURACY);
// Set the update interval to 5 seconds
mLocationRequest.setInterval(5000);
// Set the fastest update interval to 1 second
mLocationRequest.setFastestInterval(1000);
```

Question?