GEOFENCING

SOURCE CODE

MainActivity.java

```
package com.eebax.geofencing;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

ActivityMain.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
```

```
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Activity_maps.xml

```
<?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" />
```

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.eebax.geofencing">
    <!--
        The ACCESS_COARSE/FINE_LOCATION permissions are not required
to use
        Google Maps Android API v2, but you must specify either coarse or fine
        location permissions for the "MyLocation" functionality.
    -->
    <uses-permission
android:name="android.permission.ACCESS_COARSE_LOCATION" />
    <uses-permission
android:name="android.permission.ACCESS_FINE_LOCATION" />
```

```
    <uses-permission
android:name="android.permission.ACCESS_BACKGROUND_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/Theme.Geofencing">
        <receiver
            android:name=".GeofenceBroadcastReceiver"
            android:enabled="true"
            android:exported="true"></receiver>
        <!--
             The API key for Google Maps-based APIs is defined as a string
resource.
             (See the file "res/values/google_maps_api.xml").
             Note that the API key is linked to the encryption key used to sign the
APK.
             You need a different API key for each encryption key, including the
release key that is used to
             sign the APK for publishing.
             You can define the keys for the debug and release targets in src/debug/
and src/release/.
        -->
        <meta-data
            android:name="com.google.android.geo.API_KEY"
```

```
        android:value="AIzaSyDJn0B-nvXOV8sog0U_3BTe4r9MzZIvWAQ"
/>

        <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>
        <activity
            android:name=".MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

GeofenceBroadcastReceiver.java

```
package com.eebax.geofencing;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.util.Log;
```

```
import android.widget.Toast;
import com.google.android.gms.location.Geofence;
import com.google.android.gms.location.GeofencingEvent;
import java.util.List;
public class GeofenceBroadcastReceiver extends BroadcastReceiver {
    private static final String TAG = "GeofenceBroadcastReceiv";
    @Override
    public void onReceive(Context context, Intent intent) {
        // TODO: This method is called when the BroadcastReceiver is receiving
        // an Intent broadcast.
        // Toast.makeText(context, "Geofence triggered",
Toast.LENGTH_SHORT).show();
        NotificationHelper notificationHelper = new NotificationHelper(context);

        GeofencingEvent geofencingEvent = GeofencingEvent.fromIntent(intent);

        if (geofencingEvent.hasError()){
            Log.d(TAG, "onReceive: Error receiving geofence event...");
            return;
        }
        List<Geofence> geofenceList =
geofencingEvent.getTriggeringGeofences();
        for (Geofence geofence: geofenceList) {
            Log.d(TAG, "onReceive: " + geofence.getRequestId());
        }
//        Location location = geofencingEvent.getTriggeringLocation();
        int transitionType = geofencingEvent.getGeofenceTransition();
        switch (transitionType) {
            case Geofence.GEOFENCE_TRANSITION_ENTER:
```

```
            Toast.makeText(context, "GEOFENCE_TRANSITION_ENTER",
Toast.LENGTH_SHORT).show();
notificationHelper.sendHighPriorityNotification("GEOFENCE_TRANSITION
_ENTER", "", MapsActivity.class);
            break;
        case Geofence.GEOFENCE_TRANSITION_DWELL:
            Toast.makeText(context, "GEOFENCE_TRANSITION_DWELL",
Toast.LENGTH_SHORT).show();
notificationHelper.sendHighPriorityNotification("GEOFENCE_TRANSITION
_DWELL", "", MapsActivity.class);
            break;
        case Geofence.GEOFENCE_TRANSITION_EXIT:
            Toast.makeText(context, "GEOFENCE_TRANSITION_EXIT",
Toast.LENGTH_SHORT).show();
notificationHelper.sendHighPriorityNotification("GEOFENCE_TRANSITION
_EXIT", "", MapsActivity.class);
            break;
      }
   }
}
```

GeofenceHelper.java

```
package com.eebax.geofencing;
import android.app.PendingIntent;
import android.content.Context;
import android.content.ContextWrapper;
import android.content.Intent;
import com.google.android.gms.common.api.ApiException;
import com.google.android.gms.location.Geofence;
```

```
import com.google.android.gms.location.GeofenceStatusCodes;
import com.google.android.gms.location.GeofencingRequest;
import com.google.android.gms.maps.model.LatLng;
public class GeofenceHelper extends ContextWrapper {
    private static final String TAG = "GeofenceHelper";
    PendingIntent pendingIntent;
    public GeofenceHelper(Context base) {
        super(base);
    }
    public GeofencingRequest getGeofencingRequest(Geofence geofence){
        return new
GeofencingRequest.Builder().addGeofence(geofence).setInitialTrigger(Geofenc
ingRequest.INITIAL_TRIGGER_ENTER).build();
    }
    public Geofence getGeofence(String ID, LatLng latLng, float radius, int
transitionTypes){
        return new Geofence.Builder().setCircularRegion(latLng.latitude,
latLng.longitude,
radius).setRequestId(ID).setTransitionTypes(transitionTypes).setLoiteringDelay
(5000).setExpirationDuration(Geofence.NEVER_EXPIRE).build();
    }
    public PendingIntent getPendingIntent() {
        if (pendingIntent != null){
            return pendingIntent;
        }
        Intent intent = new Intent(this, GeofenceBroadcastReceiver.class);
        pendingIntent = PendingIntent.getBroadcast(this, 2607, intent,
PendingIntent.FLAG_UPDATE_CURRENT);
```

```
        return pendingIntent;
    }
    public String getErrorString(Exception e) {
        if (e instanceof ApiException) {
            ApiException apiException = (ApiException) e;
            switch (apiException.getStatusCode()) {
                case GeofenceStatusCodes
                        .GEOFENCE_NOT_AVAILABLE:
                    return "GEOFENCE_NOT_AVAILABLE";
                case GeofenceStatusCodes
                        .GEOFENCE_TOO_MANY_GEOFENCES:
                    return "GEOFENCE_TOO_MANY_GEOFENCES";
                case GeofenceStatusCodes
                        .GEOFENCE_TOO_MANY_PENDING_INTENTS:
                    return "GEOFENCE_TOO_MANY_PENDING_INTENTS";
            }
        }
        return e.getLocalizedMessage();
    }
}
```

MapsActivity.java

```
package com.eebax.geofencing;
import androidx.annotation.ColorInt;
import androidx.annotation.NonNull;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;
import androidx.fragment.app.FragmentActivity;
```

```
import android.Manifest;
import android.app.PendingIntent;
import android.content.pm.PackageManager;
import android.graphics.Color;
import android.os.Build;
import android.os.Bundle;
import android.util.Log;
import android.widget.Toast;
import com.google.android.gms.location.Geofence;
import com.google.android.gms.location.GeofencingClient;
import com.google.android.gms.location.GeofencingRequest;
import com.google.android.gms.location.LocationServices;
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.Circle;
import com.google.android.gms.maps.model.CircleOptions;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.MarkerOptions;
import com.eebax.geofencing.databinding.ActivityMapsBinding;
import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.OnSuccessListener;
public class MapsActivity extends FragmentActivity implements
OnMapReadyCallback, GoogleMap.OnMapLongClickListener {
    private static final String TAG = "MapsActivity";
    private GoogleMap mMap;
    private ActivityMapsBinding binding;
```

```
    GeofencingClient geofencingClient;
    private GeofenceHelper geofenceHelper;
    private float GEOFENCE_RADIUS = 200;
    private String GEOFENCE_ID = "SOME_GEOFENCE_ID";
    private int FINE_LOCATION_ACCESS_REQUEST_CODE = 10001;
    private int BACKGROUND_LOCATION_ACCESS_REQUEST_CODE =
10002;
    @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);
        geofencingClient = LocationServices.getGeofencingClient(this);
        geofenceHelper = new GeofenceHelper(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 cuddalore = new LatLng(11.7468, 79.7558);
      mMap.addMarker(new MarkerOptions().position(cuddalore).title("Marker in cuddalore"));
      float zoomLevel = 15.0f; //This goes up to 21
      mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(cuddalore, zoomLevel
      ));
      enableUserLocation();
      mMap.setOnMapLongClickListener(this);
   }
   private void enableUserLocation() {
      if (ContextCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION) == PackageManager.PERMISSION_GRANTED) {
         mMap.setMyLocationEnabled(true);
      } else {
         //Ask for permission
         if (ActivityCompat.shouldShowRequestPermissionRationale(this, Manifest.permission.ACCESS_FINE_LOCATION)) {
            //We need to show user a dialog for displaying why the permission is needed and then ask for the permission...
```

```
            ActivityCompat.requestPermissions(this, new
String[]{Manifest.permission.ACCESS_FINE_LOCATION},
FINE_LOCATION_ACCESS_REQUEST_CODE);
        } else {
            ActivityCompat.requestPermissions(this, new
String[]{Manifest.permission.ACCESS_FINE_LOCATION},
FINE_LOCATION_ACCESS_REQUEST_CODE);
        }
    }
  }
  @Override
  public void onRequestPermissionsResult(int requestCode, @NonNull
String[] permissions, @NonNull int[] grantResults) {
    super.onRequestPermissionsResult(requestCode, permissions,
grantResults);
    if (requestCode == FINE_LOCATION_ACCESS_REQUEST_CODE) {
      if (grantResults.length > 0 && grantResults[0] ==
PackageManager.PERMISSION_GRANTED) {
        //We have the permission
        if (ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED &&
ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_COARSE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {
          // TODO: Consider calling
          //    ActivityCompat#requestPermissions
          // here to request the missing permissions, and then overriding
```

```
            //   public void onRequestPermissionsResult(int requestCode,
String[] permissions,
            //                                          int[] grantResults)
            // to handle the case where the user grants the permission. See the
documentation
            // for ActivityCompat#requestPermissions for more details.
            return;
          }
          mMap.setMyLocationEnabled(true);
        } else {
          //We do not have the permission..

        }
      }
      if (requestCode ==
BACKGROUND_LOCATION_ACCESS_REQUEST_CODE) {
        if (grantResults.length > 0 && grantResults[0] ==
PackageManager.PERMISSION_GRANTED) {
          //We have the permission
          Toast.makeText(this, "You can add geofences...",
Toast.LENGTH_SHORT).show();
        } else {
          //We do not have the permission..
          Toast.makeText(this, "Background location access is neccessary for
geofences to trigger...", Toast.LENGTH_SHORT).show();
        }
      }
    }
```

```
    @Override
    public void onMapLongClick(LatLng latLng) {
        if (Build.VERSION.SDK_INT >= 29) {
            //We need background permission
            if (ContextCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_BACKGROUND_LOCATION) ==
PackageManager.PERMISSION_GRANTED) {
                handleMapLongClick(latLng);
            } else {
                if (ActivityCompat.shouldShowRequestPermissionRationale(this,
Manifest.permission.ACCESS_BACKGROUND_LOCATION)) {
                    //We show a dialog and ask for permission
                    ActivityCompat.requestPermissions(this, new String[]
{Manifest.permission.ACCESS_BACKGROUND_LOCATION},
BACKGROUND_LOCATION_ACCESS_REQUEST_CODE);
                } else {
                    ActivityCompat.requestPermissions(this, new String[]
{Manifest.permission.ACCESS_BACKGROUND_LOCATION},
BACKGROUND_LOCATION_ACCESS_REQUEST_CODE);
                }
            }
        } else {
            handleMapLongClick(latLng);
        }

    }
    private void handleMapLongClick(LatLng latLng) {
        mMap.clear();
        addMarker(latLng);
```

```
        addCircle(latLng, GEOFENCE_RADIUS);
        addGeofence(latLng, GEOFENCE_RADIUS);
    }
    private void addGeofence(LatLng latLng, float radius) {
        Geofence geofence = geofenceHelper.getGeofence(GEOFENCE_ID,
latLng, radius, Geofence.GEOFENCE_TRANSITION_ENTER |
Geofence.GEOFENCE_TRANSITION_DWELL |
Geofence.GEOFENCE_TRANSITION_EXIT);
        GeofencingRequest geofencingRequest =
geofenceHelper.getGeofencingRequest(geofence);
        PendingIntent pendingIntent = geofenceHelper.getPendingIntent();
        if (ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {
            // TODO: Consider calling
            //    ActivityCompat#requestPermissions
            // here to request the missing permissions, and then overriding
            //   public void onRequestPermissionsResult(int requestCode, String[]
permissions,
            //                                          int[] grantResults)
            // to handle the case where the user grants the permission. See the
documentation
            // for ActivityCompat#requestPermissions for more details.
            return;
        }
        geofencingClient.addGeofences(geofencingRequest,
pendingIntent).addOnSuccessListener(new OnSuccessListener<Void>() {
            @Override
            public void onSuccess(Void unused) {
```

```
            Log.d(TAG, "onSuccess: Added...");
        }
    }).addOnFailureListener(new OnFailureListener() {
        @Override
        public void onFailure(@NonNull Exception e) {
            String errorMessage = geofenceHelper.getErrorString(e);
            Log.d(TAG, "onFailure: " + e);
        }
    });
}
private void addMarker(LatLng latLng){
    MarkerOptions markerOptions = new MarkerOptions().position(latLng);
    mMap.addMarker(markerOptions);
}
private void addCircle(LatLng latLng, float radius){
    CircleOptions circleOptions = new CircleOptions();
    circleOptions.center(latLng);
    circleOptions.radius(radius);
    circleOptions.strokeColor(Color.argb(255, 255, 0, 0));
    circleOptions.fillColor(Color.argb(64, 255, 0, 0));
    circleOptions.strokeWidth(4);
    mMap.addCircle(circleOptions);
}
}
```

NotificationHelper.java

```
package com.eebax.geofencing;
import android.app.Notification;
import android.app.NotificationChannel;
```

```
import android.app.NotificationManager;
import android.app.PendingIntent;
import android.content.Context;
import android.content.ContextWrapper;
import android.content.Intent;
import android.graphics.Color;
import android.os.Build;
import androidx.annotation.RequiresApi;
import androidx.core.app.NotificationCompat;
import androidx.core.app.NotificationManagerCompat;
import java.util.Random;
public class NotificationHelper extends ContextWrapper {
    private static final String TAG = "NotificationHelper";
    public NotificationHelper(Context base) {
        super(base);
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
            createChannels();
        }
    }
    private String CHANNEL_NAME = "High priority channel";
    private String CHANNEL_ID = "com.example.notifications" +
CHANNEL_NAME;
    @RequiresApi(api = Build.VERSION_CODES.O)
    private void createChannels() {
        NotificationChannel notificationChannel = new
NotificationChannel(CHANNEL_ID, CHANNEL_NAME,
NotificationManager.IMPORTANCE_HIGH);
        notificationChannel.enableLights(true);
        notificationChannel.enableVibration(true);
```

```
        notificationChannel.setDescription("this is the description of the
channel.");
        notificationChannel.setLightColor(Color.RED);
notificationChannel.setLockscreenVisibility(Notification.VISIBILITY_PUBLI
C);
        NotificationManager manager = (NotificationManager)
getSystemService(Context.NOTIFICATION_SERVICE);
        manager.createNotificationChannel(notificationChannel);
    }
    public void sendHighPriorityNotification(String title, String body, Class
activityName) {
        Intent intent = new Intent(this, activityName);
        PendingIntent pendingIntent = PendingIntent.getActivity(this, 267, intent,
PendingIntent.FLAG_UPDATE_CURRENT);
        Notification notification = new NotificationCompat.Builder(this,
CHANNEL_ID)
//                .setContentTitle(title)
//                .setContentText(body)
                .setSmallIcon(R.drawable.ic_launcher_background)
                .setPriority(NotificationCompat.PRIORITY_HIGH)
                .setStyle(new
NotificationCompat.BigTextStyle().setSummaryText("summary").setBigConten
tTitle(title).bigText(body))
                .setContentIntent(pendingIntent)
                .setAutoCancel(true)
                .build();
        NotificationManagerCompat.from(this).notify(new Random().nextInt(),
notification);
```

```
    }
}
```

DBHandler

```
import android.content.ContentValues;
import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
public class DBHandler extends SQLiteOpenHelper {
    // creating a constant variables for our database.
    // below variable is for our database name.
    private static final String DB_NAME = "coursedb";
    // below int is our database version
    private static final int DB_VERSION = 1;
    // below variable is for our table name.
    private static final String TABLE_NAME = "mycourses";
    // below variable is for our id column.
    private static final String ID_COL = "id";
    // below variable is for our course name column
    private static final String NAME_COL = "name";
    // below variable id for our course duration column.
    private static final String DURATION_COL = "duration";
    // below variable for our course description column.
    private static final String DESCRIPTION_COL = "description";
    // below variable is for our course tracks column.
    private static final String TRACKS_COL = "tracks";
    // creating a constructor for our database handler.
    public DBHandler(Context context) {
        super(context, DB_NAME, null, DB_VERSION);
```

```
    }
    // below method is for creating a database by running a sqlite query
    @Override
    public void onCreate(SQLiteDatabase db) {
        // on below line we are creating
        // an sqlite query and we are
        // setting our column names
        // along with their data types.
        String query = "CREATE TABLE " + TABLE_NAME + " ("
                + ID_COL + " INTEGER PRIMARY KEY AUTOINCREMENT, "
                + NAME_COL + " TEXT,"
                + DURATION_COL + " TEXT,"
                + DESCRIPTION_COL + " TEXT,"
                + TRACKS_COL + " TEXT)";
        // at last we are calling a exec sql
        // method to execute above sql query
        db.execSQL(query);
    }
    // this method is use to add new course to our sqlite database.
    public void addNewCourse(String courseName, String courseDuration,
String courseDescription, String courseTracks) {
        // on below line we are creating a variable for
        // our sqlite database and calling writable method
        // as we are writing data in our database.
        SQLiteDatabase db = this.getWritableDatabase();
        // on below line we are creating a
        // variable for content values.
        ContentValues values = new ContentValues();
```

```
        // on below line we are passing all values
        // along with its key and value pair.
        values.put(NAME_COL, courseName);
        values.put(DURATION_COL, courseDuration);
        values.put(DESCRIPTION_COL, courseDescription);
        values.put(TRACKS_COL, courseTracks);
        // after adding all values we are passing
        // content values to our table.
        db.insert(TABLE_NAME, null, values);
        // at last we are closing our
        // database after adding database.
        db.close();
    }
    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion)
{
        // this method is called to check if the table exists already.
        db.execSQL("DROP TABLE IF EXISTS " + TABLE_NAME);
        onCreate(db);
    }
}
```

google_maps_api.xml

```
<resources>
    <string name="google_maps_key" templateMergeStrategy="preserve"
translatable="false">AIzaSyDJn0B-
nvXOV8sog0U_3BTe4r9MzZIvWAQ</string>
</resources>
```