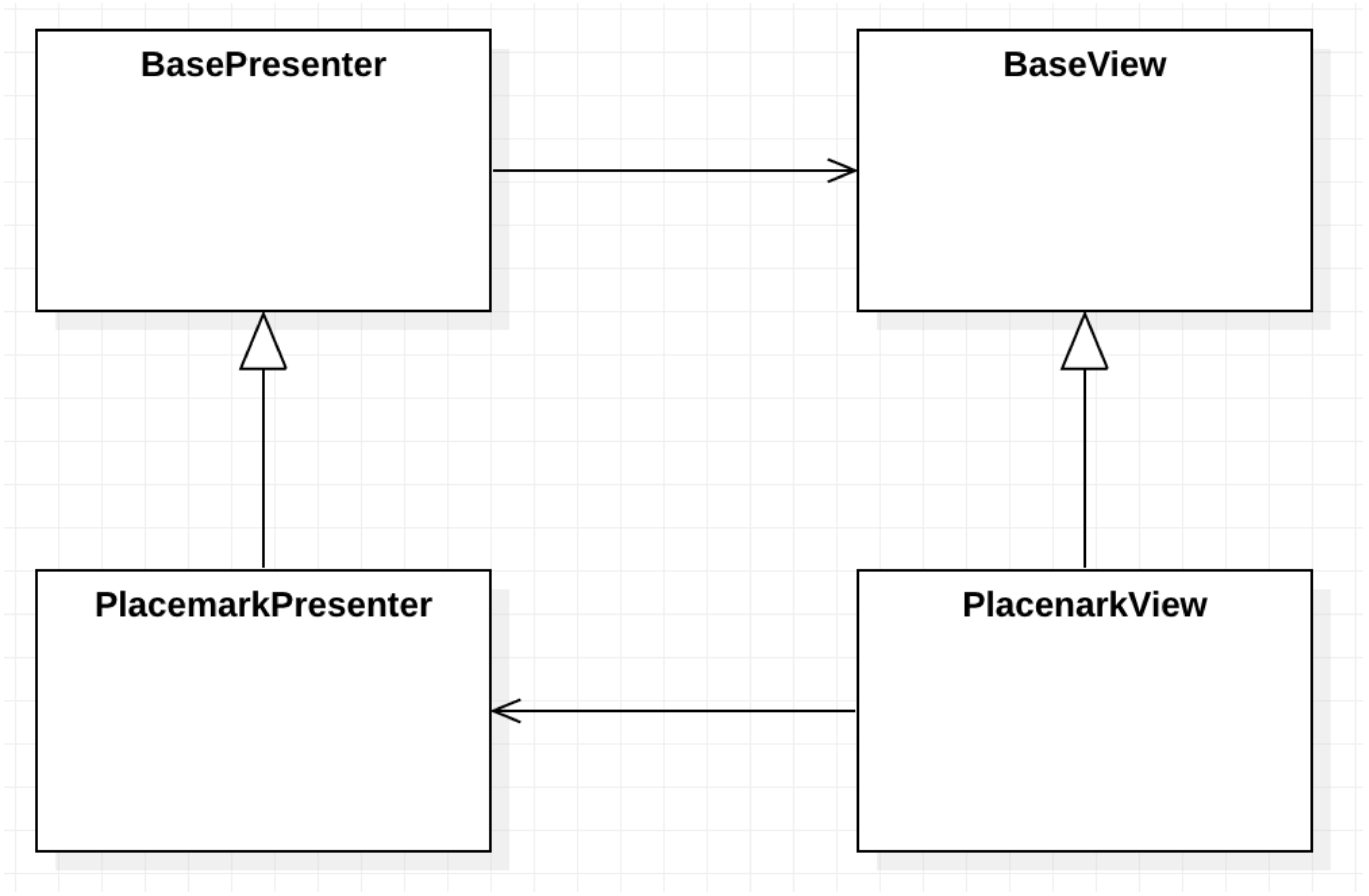
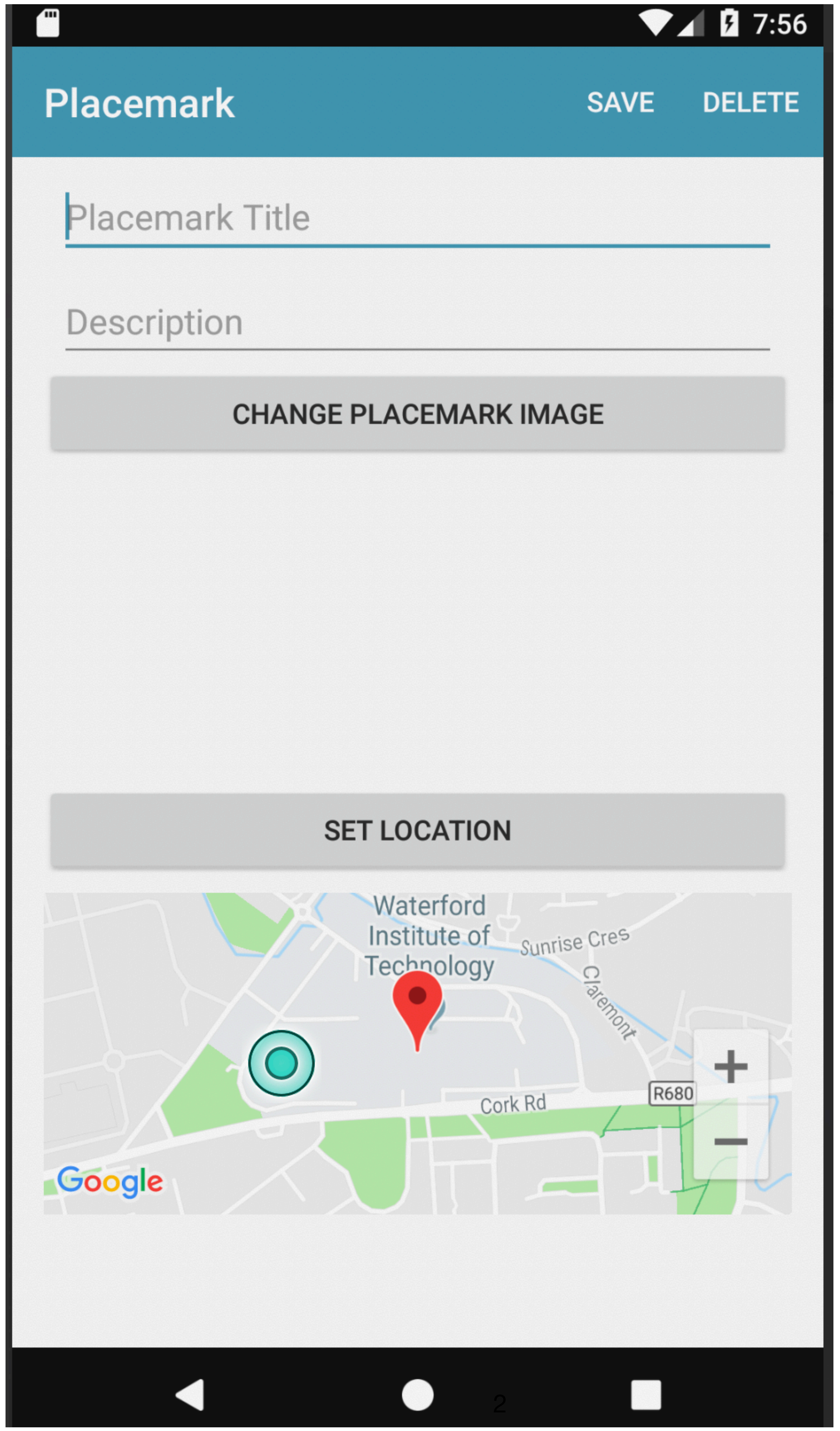
Current Location

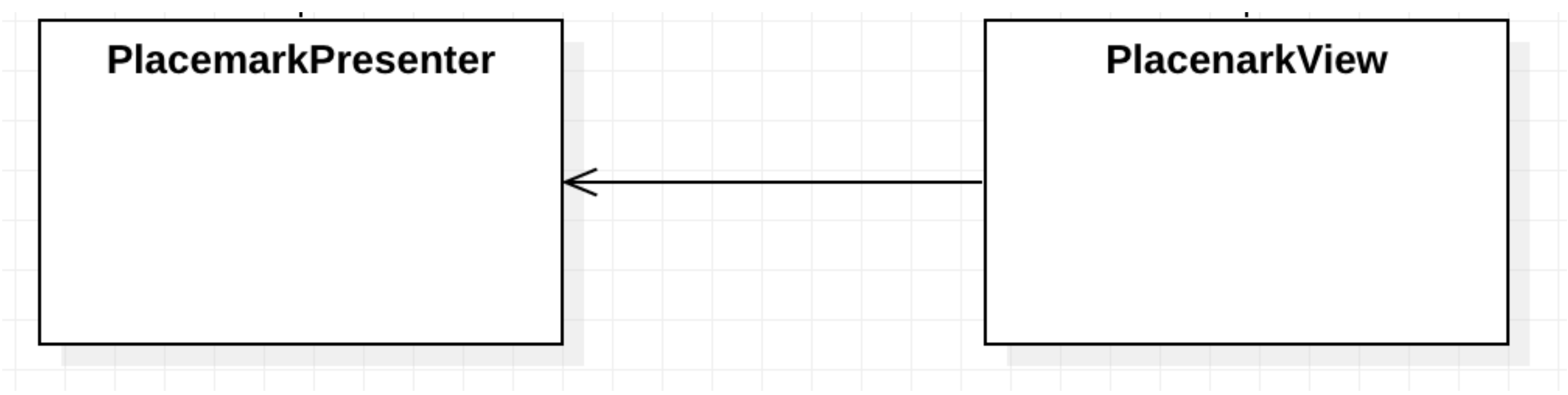
Current Location

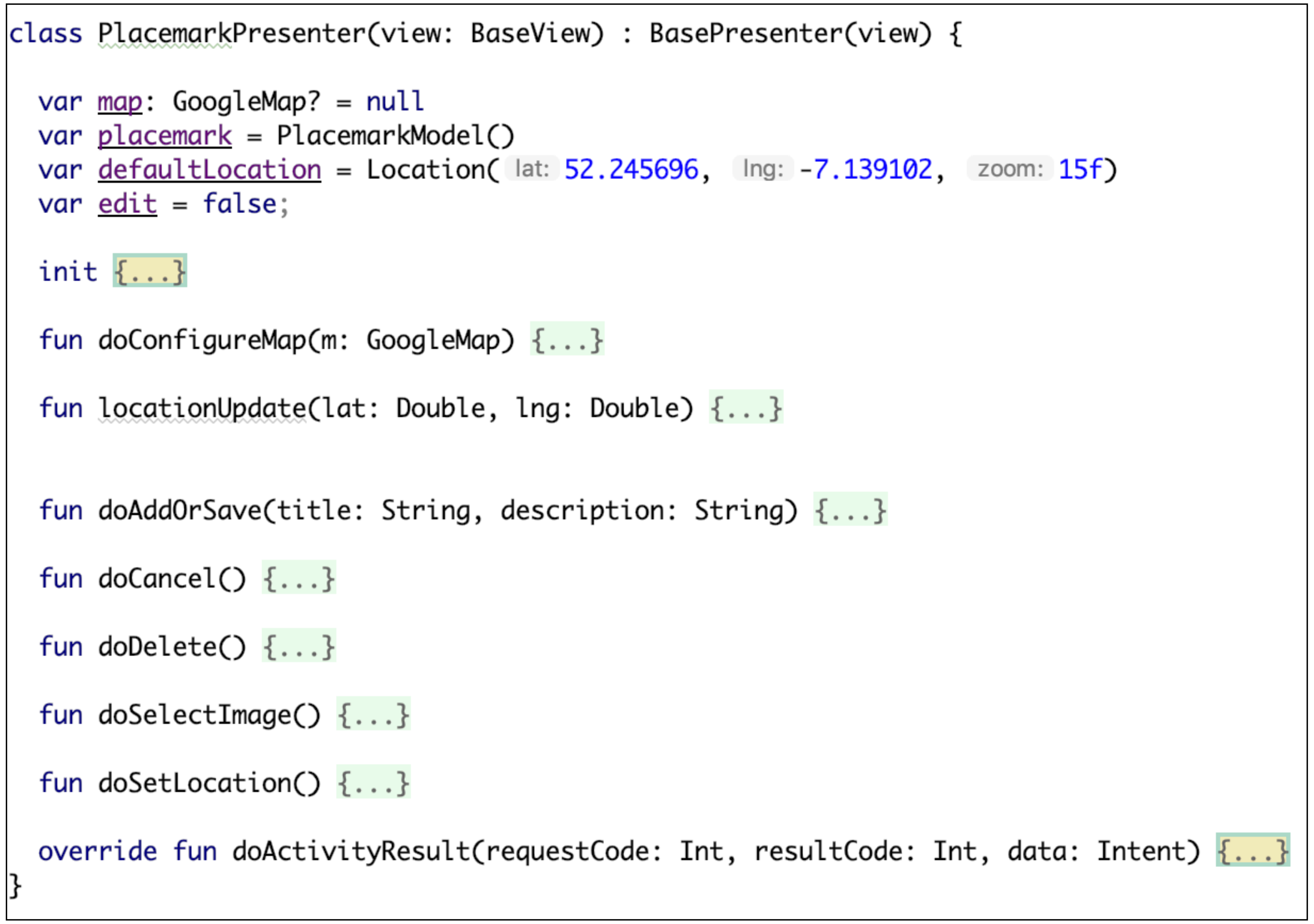
Using the Fused Location Provider to recover the last known location

PlacemarkView/Presenter

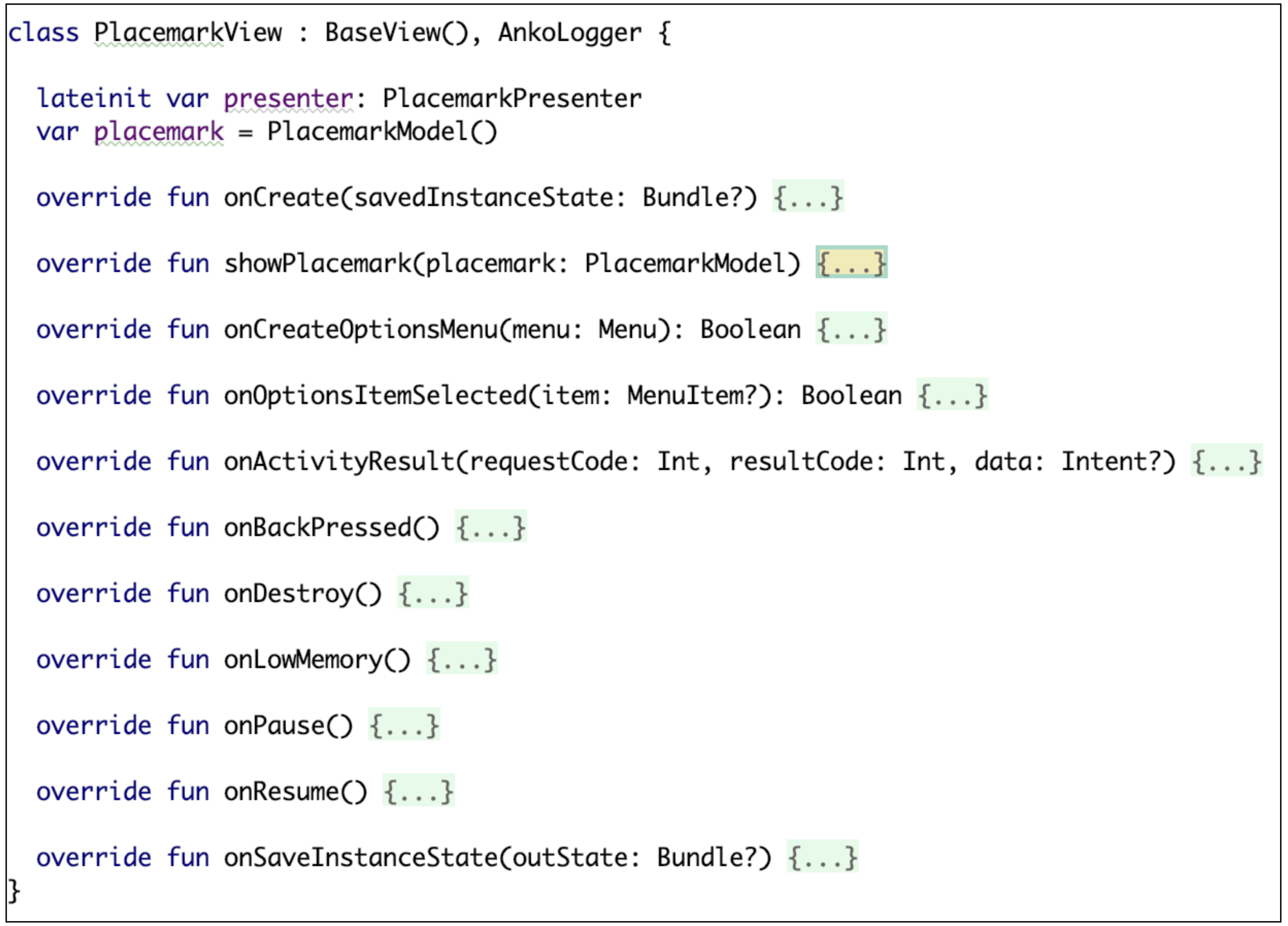








```kotlin
class PlacemarkPresenter(view: BaseView) : BasePresenter(view) {

  var map: GoogleMap? = null
  var placemark = PlacemarkModel()
  var defaultLocation = Location( lat: 52.245696,  lng: -7.139102,  zoom: 15f)
  var edit = false;

  init {...}

  fun doConfigureMap(m: GoogleMap) {...}

  fun locationUpdate(lat: Double, lng: Double) {...}


  fun doAddOrSave(title: String, description: String) {...}

  fun doCancel() {...}

  fun doDelete() {...}

  fun doSelectImage() {...}

  fun doSetLocation() {...}

  override fun doActivityResult(requestCode: Int, resultCode: Int, data: Intent) {...}
}
```



```kotlin
class PlacemarkView : BaseView(), AnkoLogger {

  lateinit var presenter: PlacemarkPresenter
  var placemark = PlacemarkModel()

  override fun onCreate(savedInstanceState: Bundle?) {...}

  override fun showPlacemark(placemark: PlacemarkModel) {...}

  override fun onCreateOptionsMenu(menu: Menu): Boolean {...}

  override fun onOptionsItemSelected(item: MenuItem?): Boolean {...}

  override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {...}

  override fun onBackPressed() {...}

  override fun onDestroy() {...}

  override fun onLowMemory() {...}

  override fun onPause() {...}

  override fun onResume() {...}

  override fun onSaveInstanceState(outState: Bundle?) {...}
}
```

```kotlin
class PlacemarkPresenter(view: BaseView) : BasePresenter(view) {

  var map: GoogleMap? = null
  var placemark = PlacemarkModel()
  var defaultLocation = Location(52.245696, -7.139102, 15f)
  var edit = false;

  init {
    if (view.intent.hasExtra("placemark_edit")) {
      edit = true
      placemark = view.intent.extras.getParcelable<PlacemarkModel>("placemark_
      view.showPlacemark(placemark)
    } else {
      placemark.lat = defaultLocation.lat
      placemark.lng = defaultLocation.lng
    }
  }

  fun doConfigureMap(m: GoogleMap) {
    map = m
    locationUpdate(placemark.lat, placemark.lng)
  }

  fun locationUpdate(lat: Double, lng: Double) {
    placemark.lat = lat
    placemark.lng = lng
    placemark.zoom = 15f
    map?.clear()
    map?.uiSettings?.setZoomControlsEnabled(true)
    val options = MarkerOptions().title(placemark.title).position(LatLng(placemark.lat, placemark.lng))
    map?.addMarker(options)
    map?.moveCamera(CameraUpdateFactory.newLatLngZoom(LatLng(placemark.lat, placemark.lng), placemark.zoom))
    view?.showPlacemark(placemark)
  }
  ...
}
```

```kotlin
class PlacemarkView : BaseView(), AnkoLogger {

  lateinit var presenter: PlacemarkPresenter
  var placemark = PlacemarkModel()

  override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_placemark)

    init(toolbarAdd)

    presenter = initPresenter (PlacemarkPresenter(this)) as PlacemarkPresenter

    mapView.onCreate(savedInstanceState);
    mapView.getMapAsync {
      presenter.doConfigureMap(it)
    }

    chooseImage.setOnClickListener { presenter.doSelectImage() }

    placemarkLocation.setOnClickListener { presenter.doSetLocation() }
  }
  ...
}
```

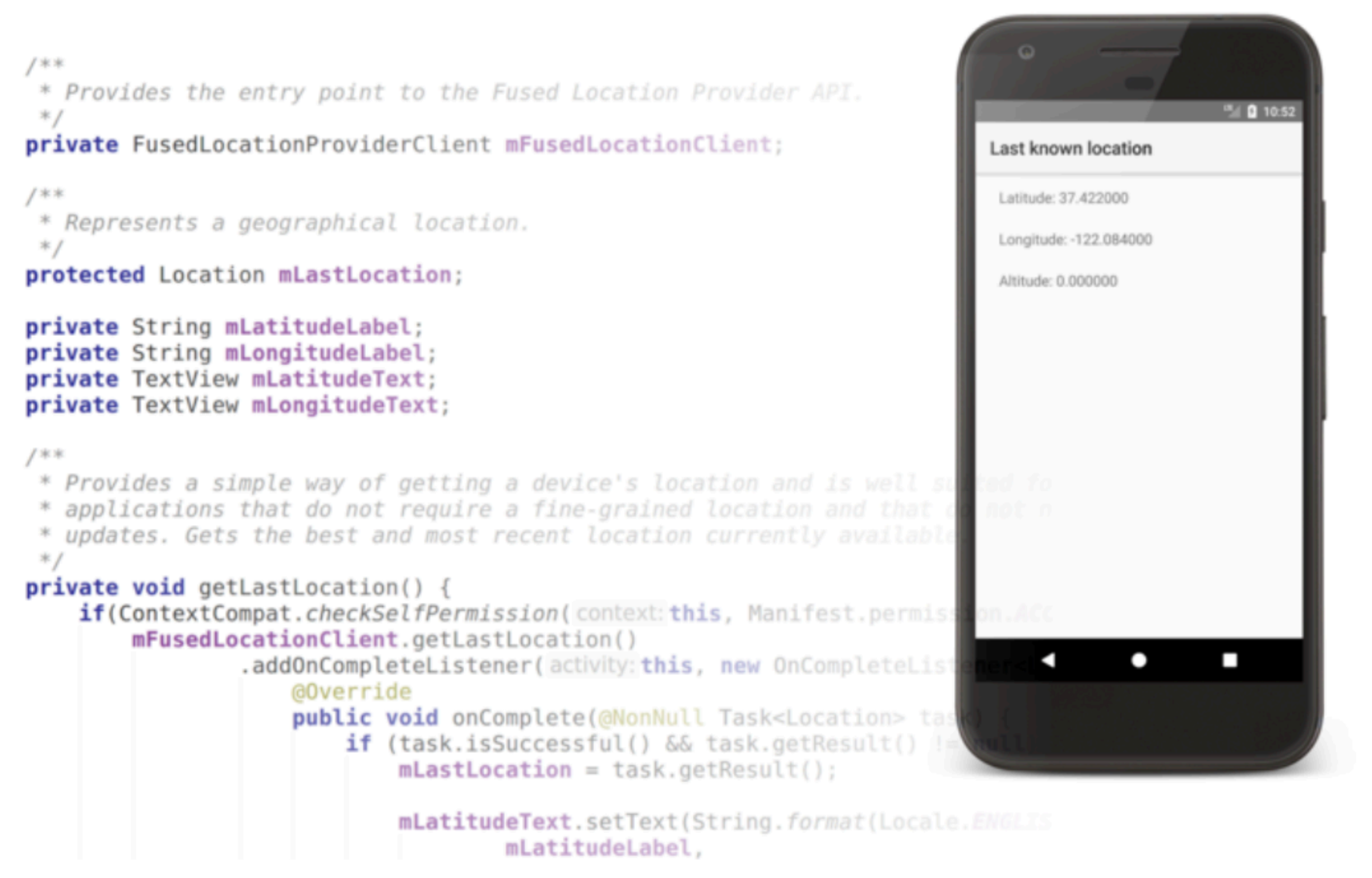
Fused Location Provider API   Search   ALL PRODUCTS

Simple, battery-efficient location API for Android

Apps can take advantage of the signals provided by multiple sensors in the device to determine device location. However, choosing the right combination of signals for a specific task in different conditions is not simple. Finding a solution that is also battery-efficient is even more complicated.

The fused location provider is a location API in Google Play services that intelligently combines different signals to provide the location information that your app needs.

The fused location provider manages the underlying location technologies, such as GPS and Wi-Fi, and provides a simple API that you can use to specify the required quality of service. For example, you can request the most accurate data available, or the best accuracy possible with no additional power consumption.



```
/**
 * Provides the entry point to the Fused Location Provider API.
 */
private FusedLocationProviderClient mFusedLocationClient;

/**
 * Represents a geographical location.
 */
protected Location mLastLocation;

private String mLatitudeLabel;
private String mLongitudeLabel;
private TextView mLatitudeText;
private TextView mLongitudeText;

/**
 * Provides a simple way of getting a device's location and is well su
 * applications that do not require a fine-grained location and that
 * updates. Gets the best and most recent location currently available
 */
private void getLastLocation() {
    if(ContextCompat.checkSelfPermission( context: this, Manifest.permission.
        mFusedLocationClient.getLastLocation()
                .addOnCompleteListener( activity: this, new OnCompleteListener<
                    @Override
                    public void onComplete(@NonNull Task<Location> task) {
                        if (task.isSuccessful() && task.getResult() != null
                            mLastLocation = task.getResult();

                            mLatitudeText.setText(String.format(Locale.
                                    mLatitudeLabel,
```

https://developers.google.com/location-context/fused-location-provider/

Support for common location scenarios

Last Known Location

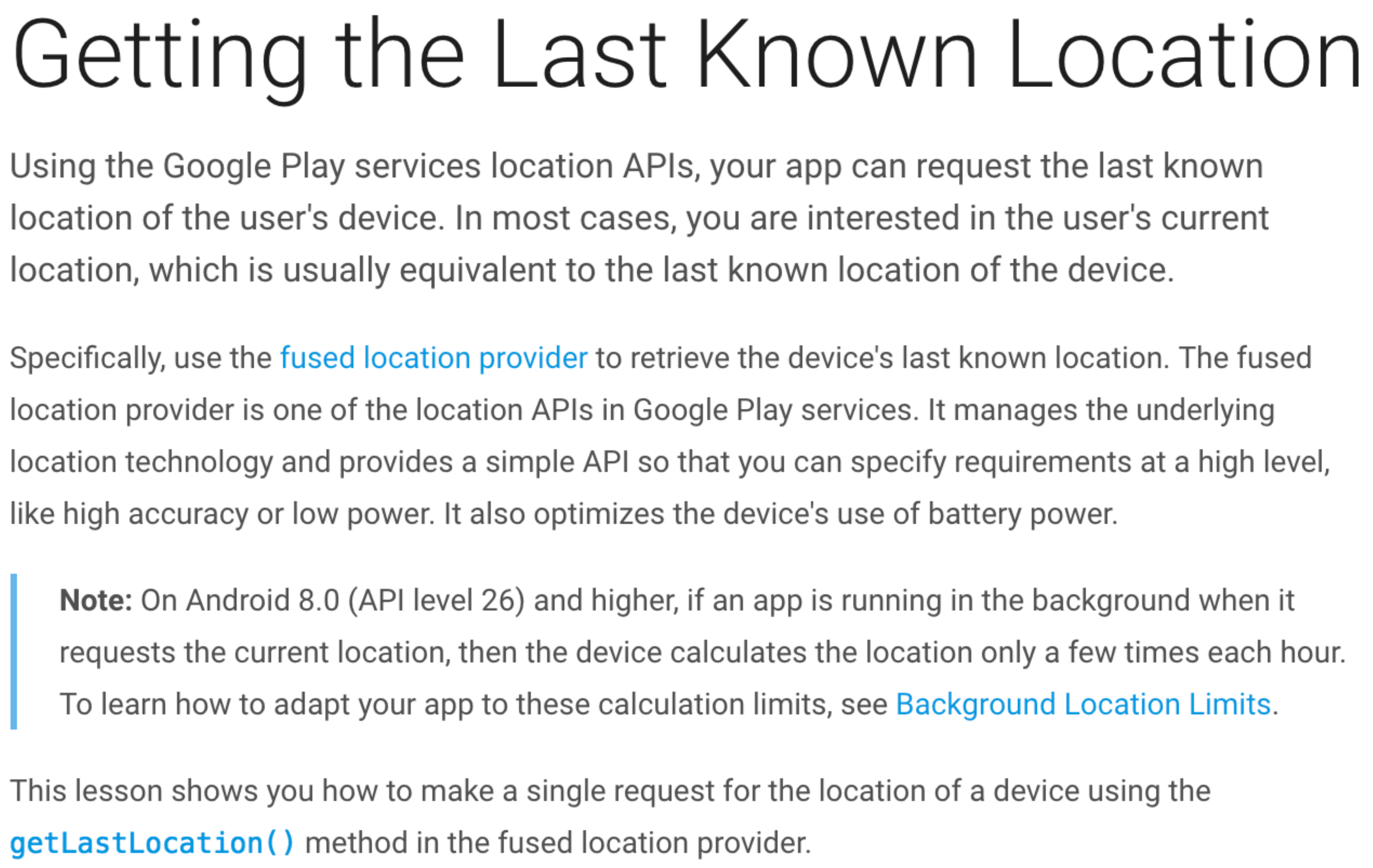
Using the fused location provider API, your app can request the last known location of the user's device. Getting the last known location is usually a good starting point for apps that require location information.

Location Settings

When requesting location information many different location sources, such as GPS and Wi-Fi, are used. Deciding which sources to use can be challenging, but the fused location provider API removes the guesswork by automatically changing the appropriate system settings. All your app must do is specify the desired level of service.

Location Updates

In addition to the last known location, the fused location provider API can deliver location updates to a callback in your app at specific intervals. You can specify the desired interval as a parameter of the quality of service. By using location updates, your app can provide additional information such as direction and velocity.



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. 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.

> **Note:** On Android 8.0 (API level 26) and higher, if an app is running in the background when it requests the current location, then the device calculates the location only a few times each hour. To learn how to adapt your app to these calculation limits, see Background Location Limits.

This lesson shows you how to make a single request for the location of a device using the `getLastLocation()` method in the fused location provider.

https://developer.android.com/training/location/retrieve-current.html

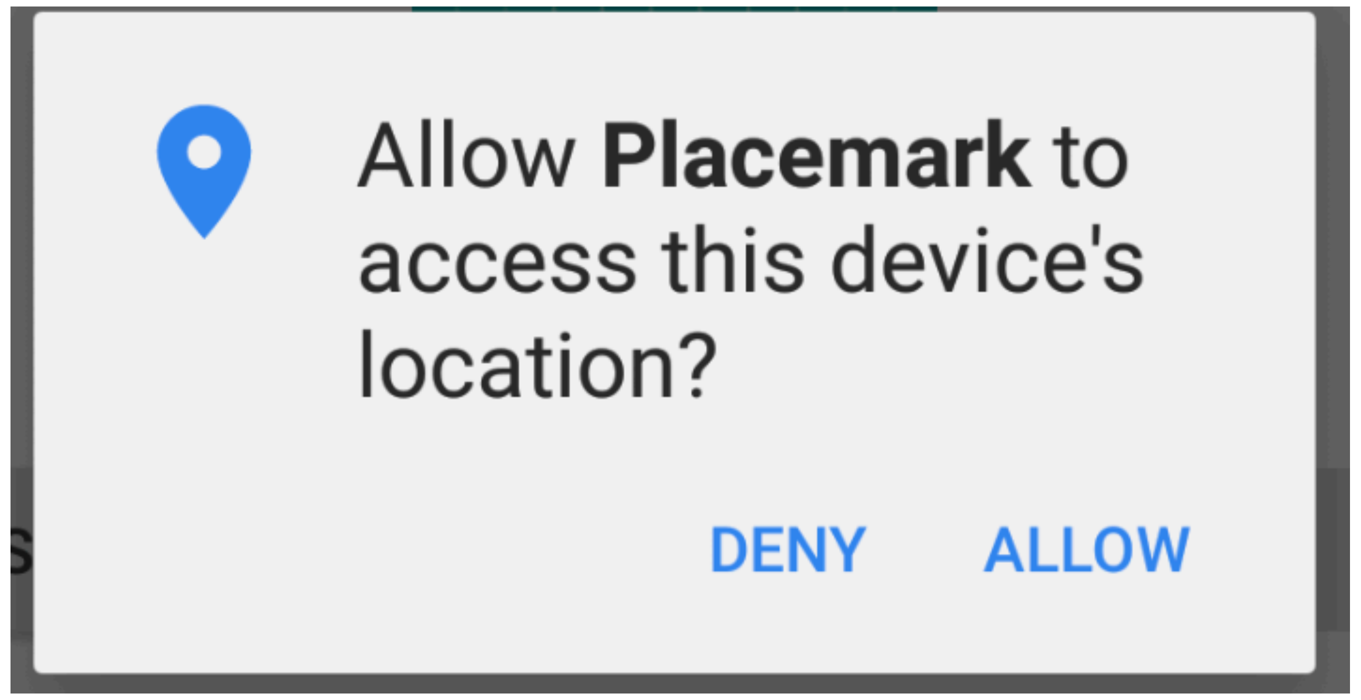
Location Permissions

If an app is to use the users location, there are 2 permission steps required

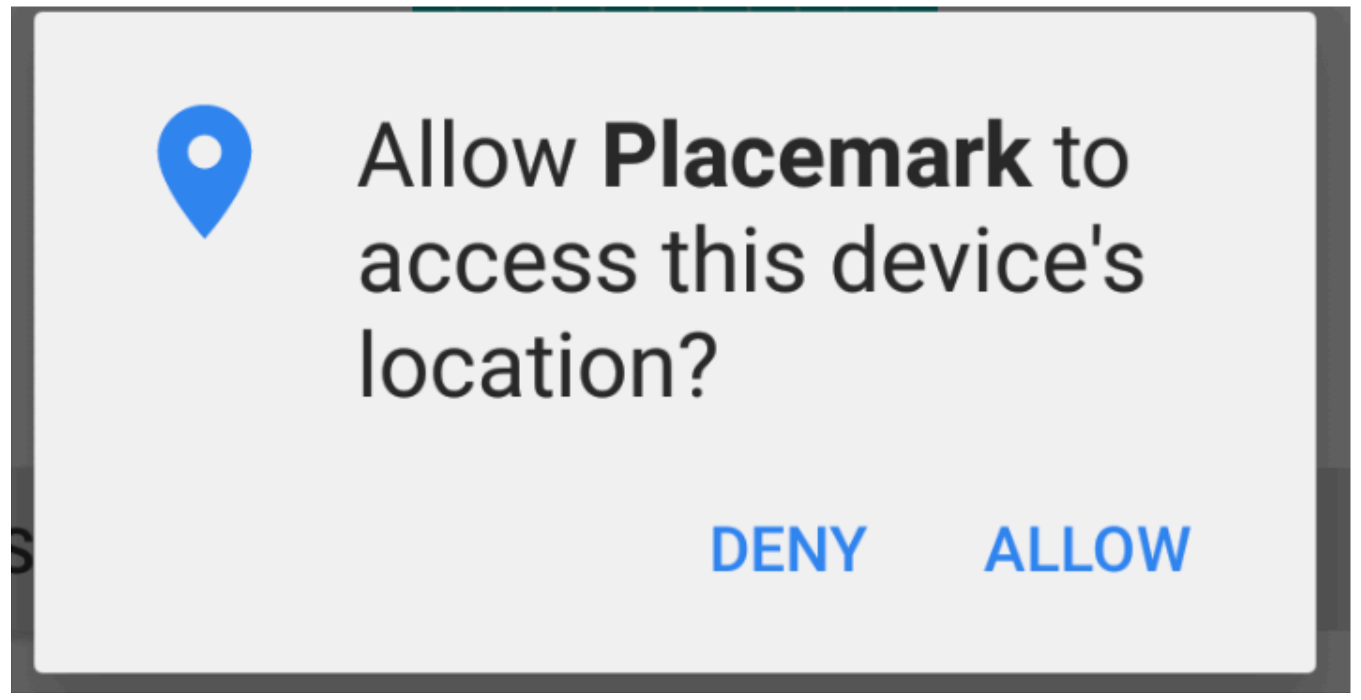
1: AndroidManifest.xml

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  package="org.wit.placemark">

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

...
```

2: Dialog Directly with the user


Allow **Placemark** to access this device's location?

2: Dialog Directly with the user


Allow **Placemark** to access this device's location?

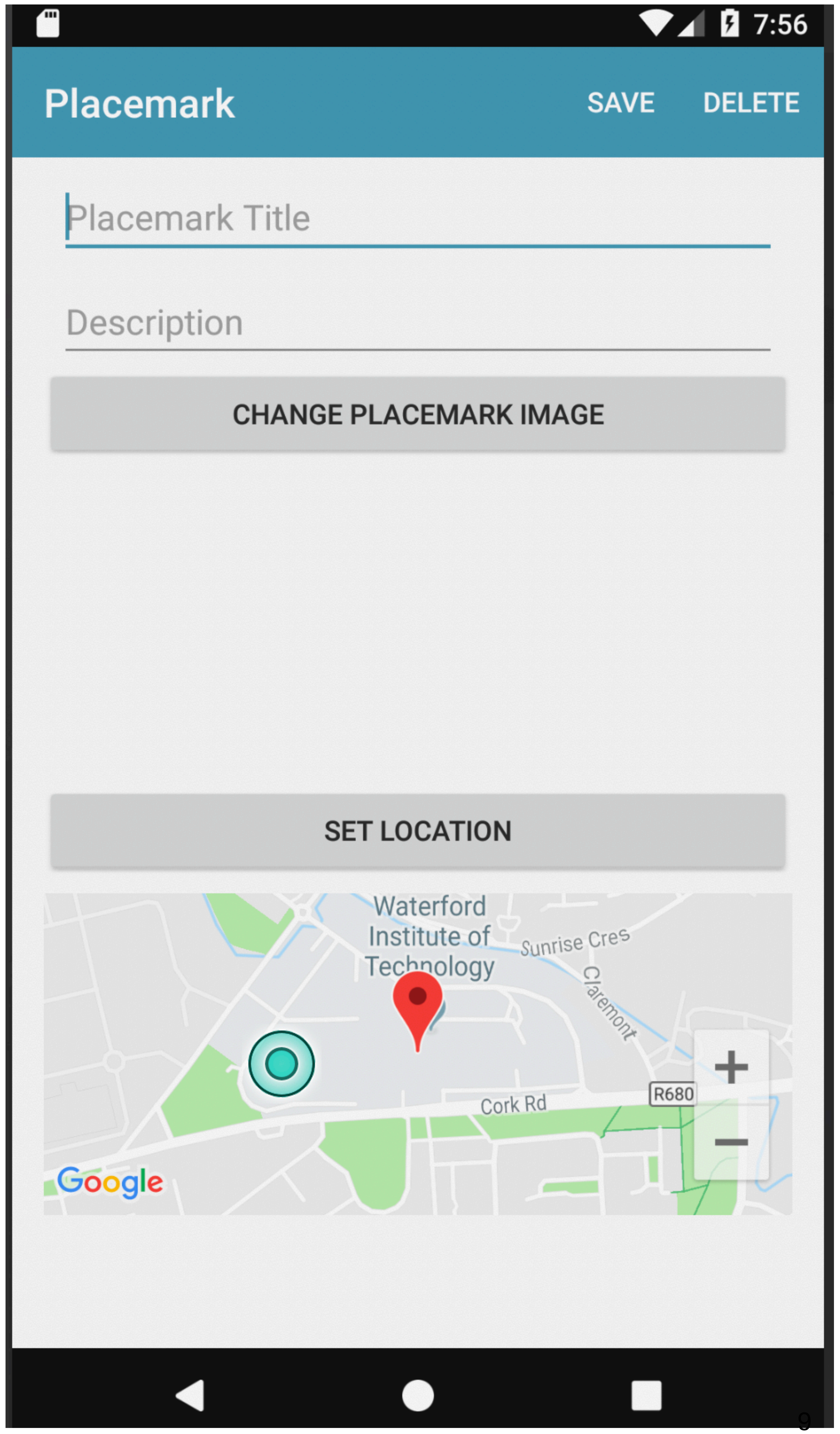
New Helper functions to support this interaction:

```
fun checkLocationPermissions(activity: Activity): Boolean {...}

fun isPermissionGranted(code: Int, grantResults: IntArray): Boolean {...}
```

LocationHelper

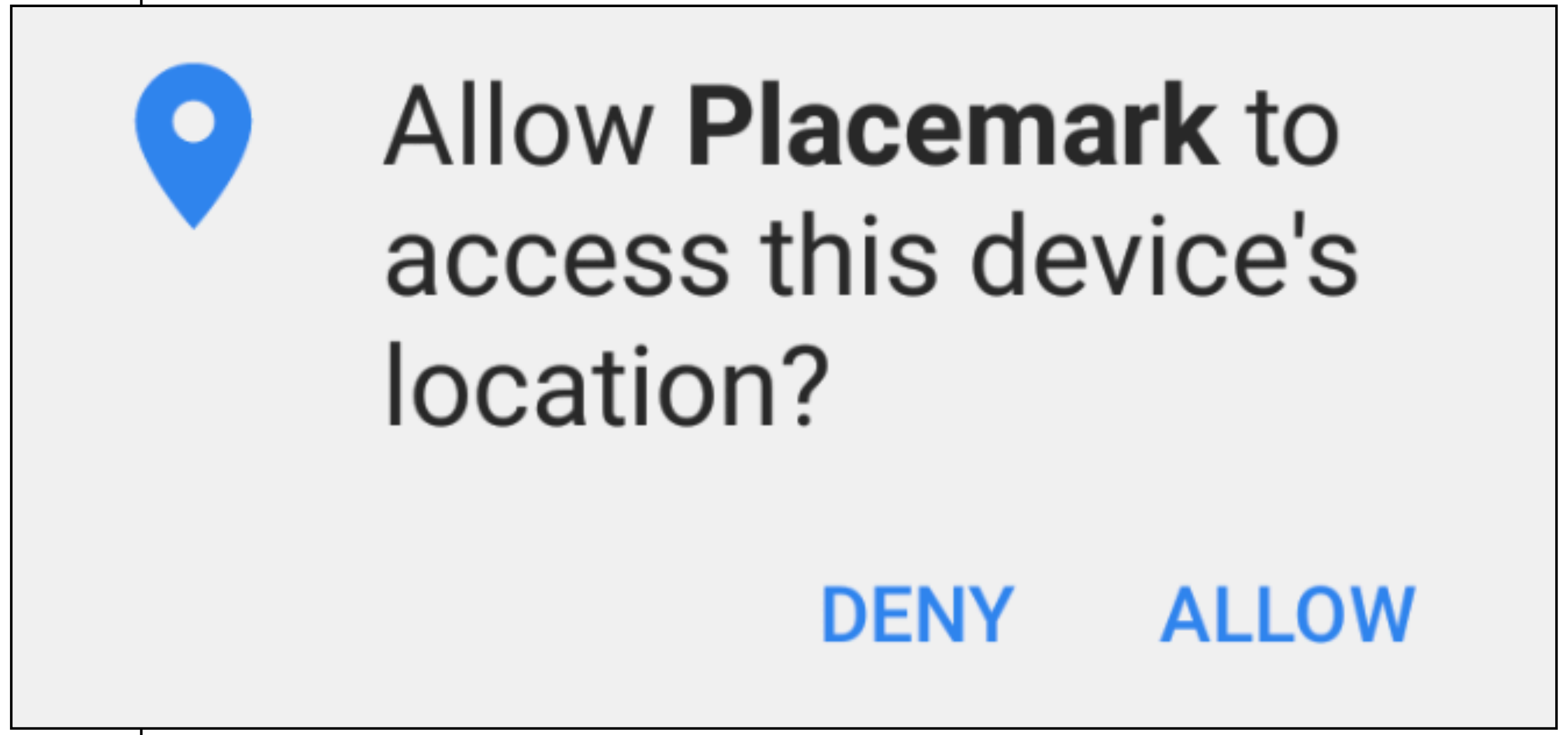
PlacemarkPresenter - Acquire Permissions

```
class PlacemarkPresenter(view: BaseView) : BasePresenter(view) {
  ...

  init {
    if (view.intent.hasExtra("placemark_edit")) {
      edit = true
      placemark = view.intent.extras.getParcelable<PlacemarkModel>("placemark_edit")
      view.showPlacemark(placemark)
    } else {
      if (checkLocationPermissions(view)) {
        // get current location
      }
    }
  }

  override fun doRequestPermissionsResult(requestCode: Int, permissions: Array<String>,
                                          grantResults: IntArray) {
    if (isPermissionGranted(requestCode, grantResults)) {
      // get current Location
    } else {
      locationUpdate(defaultLocation.lat, defaultLocation.lng)
    }
  }
…
}
```



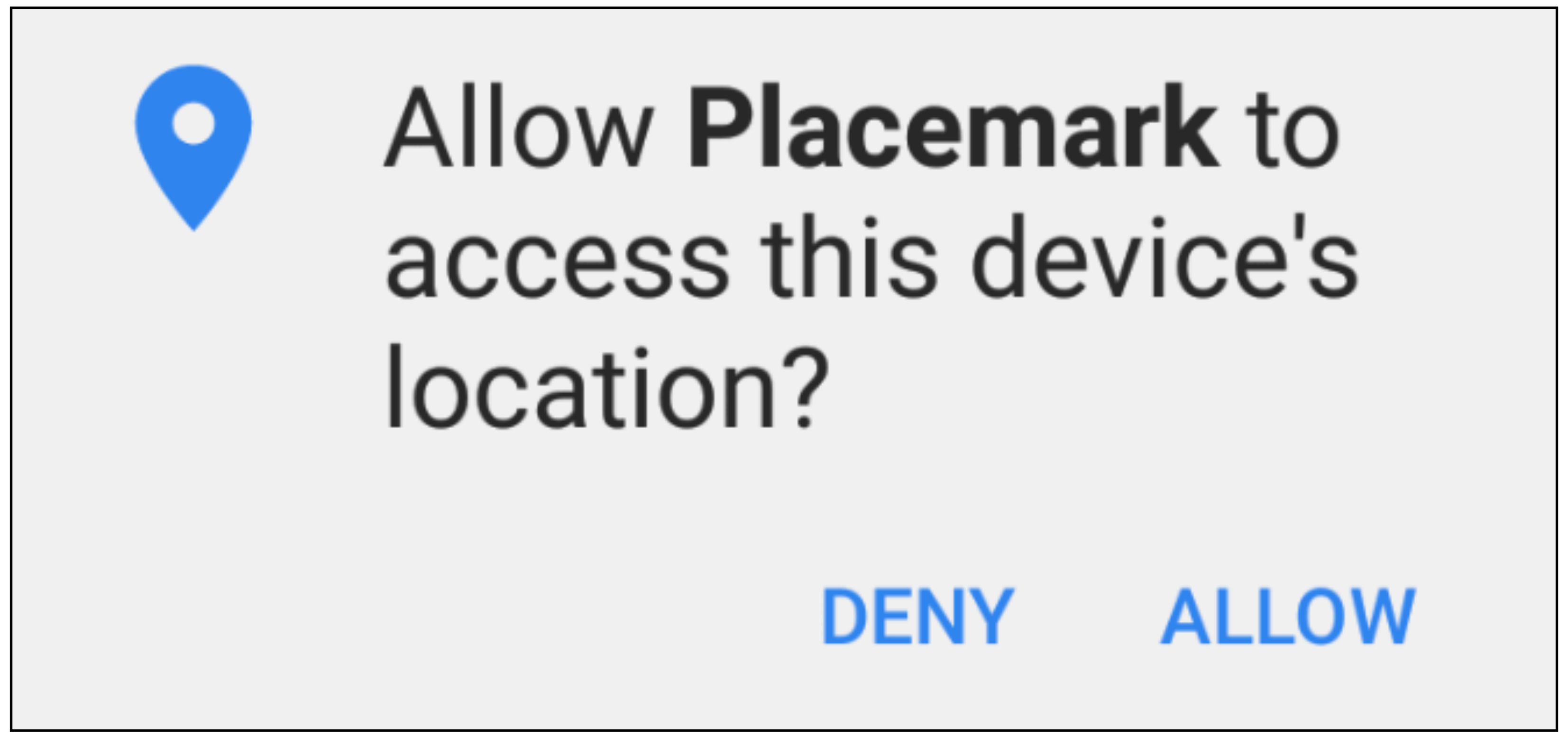
PlacemarkPresenter - Acquire Permissions

```
class PlacemarkPresenter(view: BaseView) : BasePresenter(view) {
  ...

  init {
    if (view.intent.hasExtra("placemark_edit")) {
      edit = true
      placemark = view.intent.extras.getParcelable<PlacemarkModel>("placemark_edit")
      view.showPlacemark(placemark)
    } else {
      if (checkLocationPermissions(view)) {
        // get current location
      }
    }
  }

  override fun doRequestPermissionsResult(requestCode: Int, permissions: Array<String>,
                                          grantResults: IntArray) {
    if (isPermissionGranted(requestCode, grantResults)) {
      // get current Location
    } else {
      locationUpdate(defaultLocation.lat, defaultLocation.lng)
    }
  }
…
}
```


Allow **Placemark** to access this device's location?

PlacemarkPresenter - Acquire Permissions

```
class PlacemarkPresenter(view: BaseView) : BasePresenter(view) {
  ...

  init {
    if (view.intent.hasExtra("placemark_edit")) {
      edit = true
      placemark = view.intent.extras.getParcelable<PlacemarkModel>("placemark_edit")
      view.showPlacemark(placemark)
    } else {
      if (checkLocationPermissions(view)) {
        // get current location
      }
    }
  }

  override fun doRequestPermissionsResult(requestCode: Int, permissions: Array<String>,
                                          grantResults: IntArray) {
    if (isPermissionGranted(requestCode, grantResults)) {
      // get current Location
    } else {
      locationUpdate(defaultLocation.lat, defaultLocation.lng)
    }
  }
…
}
```


Allow **Placemark** to access this device's location?

DENY ALLOW

Acquire current Location *(permissions previously granted)*

Acquire current Location *(permissions granted just now)*

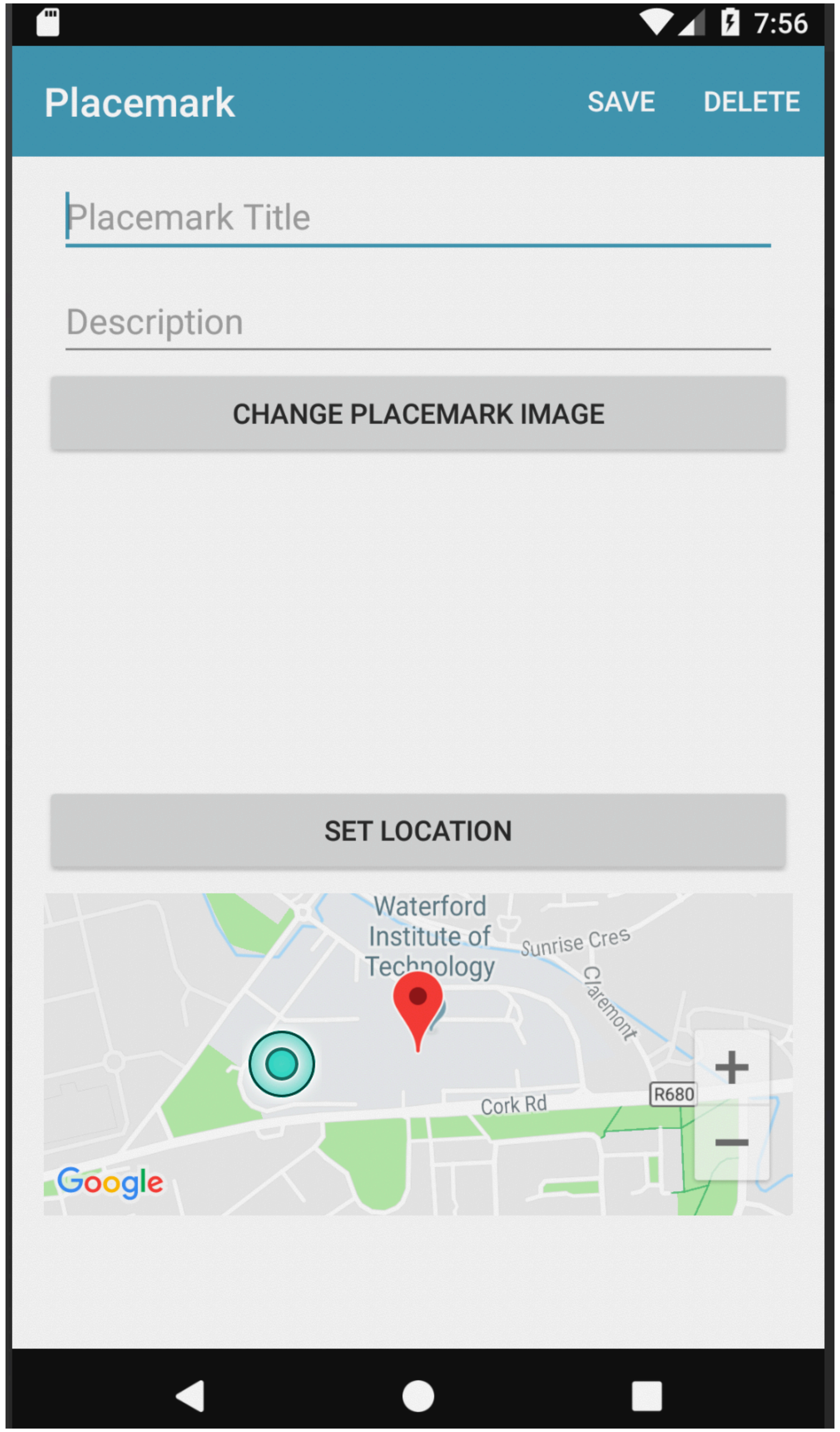
Permission Denied, show default location

Fused Location Client

Acquire a reference to the Fused location provider

```
class PlacemarkPresenter(view: BaseView) : BasePresenter(view) {

  var locationService: FusedLocationProviderClient = LocationServices.getFusedLocationProviderClient(view)

  ...

  @SuppressLint("MissingPermission")
  fun doSetCurrentLocation() {
    locationService.lastLocation.addOnSuccessListener {
      locationUpdate(Location(it.latitude, it.longitude))
    }
  }

  ...

}
```

New method to request current location



```kotlin
class PlacemarkPresenter(view: BaseView) : BasePresenter(view) {

  var map: GoogleMap? = null
  var placemark = PlacemarkModel()
  var defaultLocation = Location(52.245696, -7.139102, 15f)
  var edit = false;
  var locationService: FusedLocationProviderClient = LocationServices.getFusedLocationProviderClient(view)

  init {
    if (view.intent.hasExtra("placemark_edit")) {
      edit = true
      placemark = view.intent.extras.getParcelable<PlacemarkModel>("placemark_edit")
      view.showPlacemark(placemark)
    } else {
      if (checkLocationPermissions(view)) {
        doSetCurrentLocation()
      }
    }
  }

  @SuppressLint("MissingPermission")
  fun doSetCurrentLocation() {
    locationService.lastLocation.addOnSuccessListener {
      locationUpdate(it.latitude, it.longitude)
    }
  }

  override fun doRequestPermissionsResult(requestCode: Int, permissions: Array<String>, grantResults: IntArray) {
    if (isPermissionGranted(requestCode, grantResults)) {
      doSetCurrentLocation()
    } else {
      locationUpdate(defaultLocation.lat, defaultLocation.lng)
    }
  }

  fun locationUpdate(lat: Double, lng: Double) {
    placemark.lat = lat
    placemark.lng = lng
    placemark.zoom = 15f
    map?.clear()
    map?.uiSettings?.setZoomControlsEnabled(true)
    val options = MarkerOptions().title(placemark.title).position(LatLng(placemark.lat, placemark.lng))
    map?.addMarker(options)
    map?.moveCamera(CameraUpdateFactory.newLatLngZoom(LatLng(placemark.lat, placemark.lng), placemark.zoom))
    view?.showPlacemark(placemark)
  }
...
}
```

Recover last known location

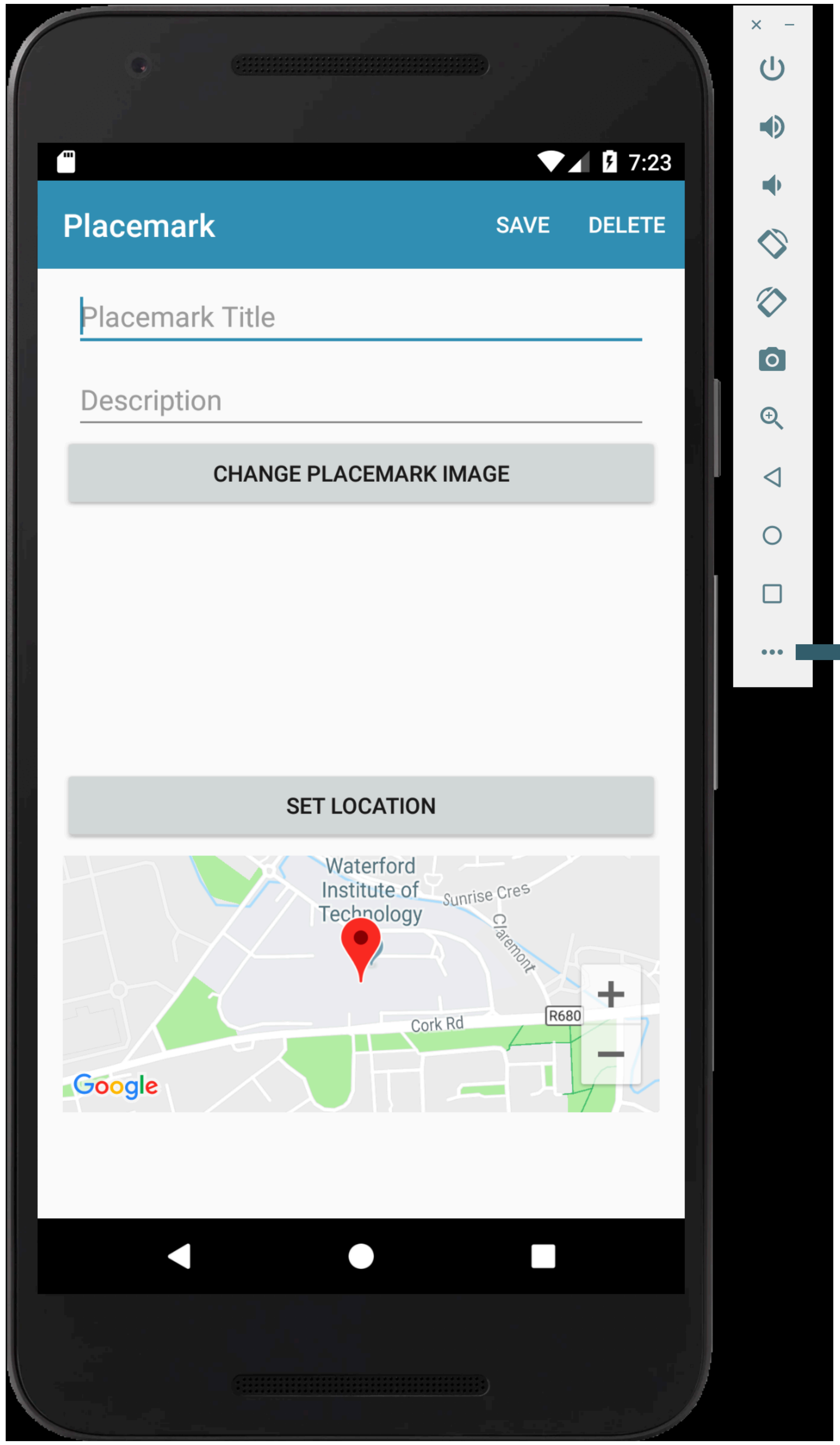
Suppress Permissions warnings from compiler

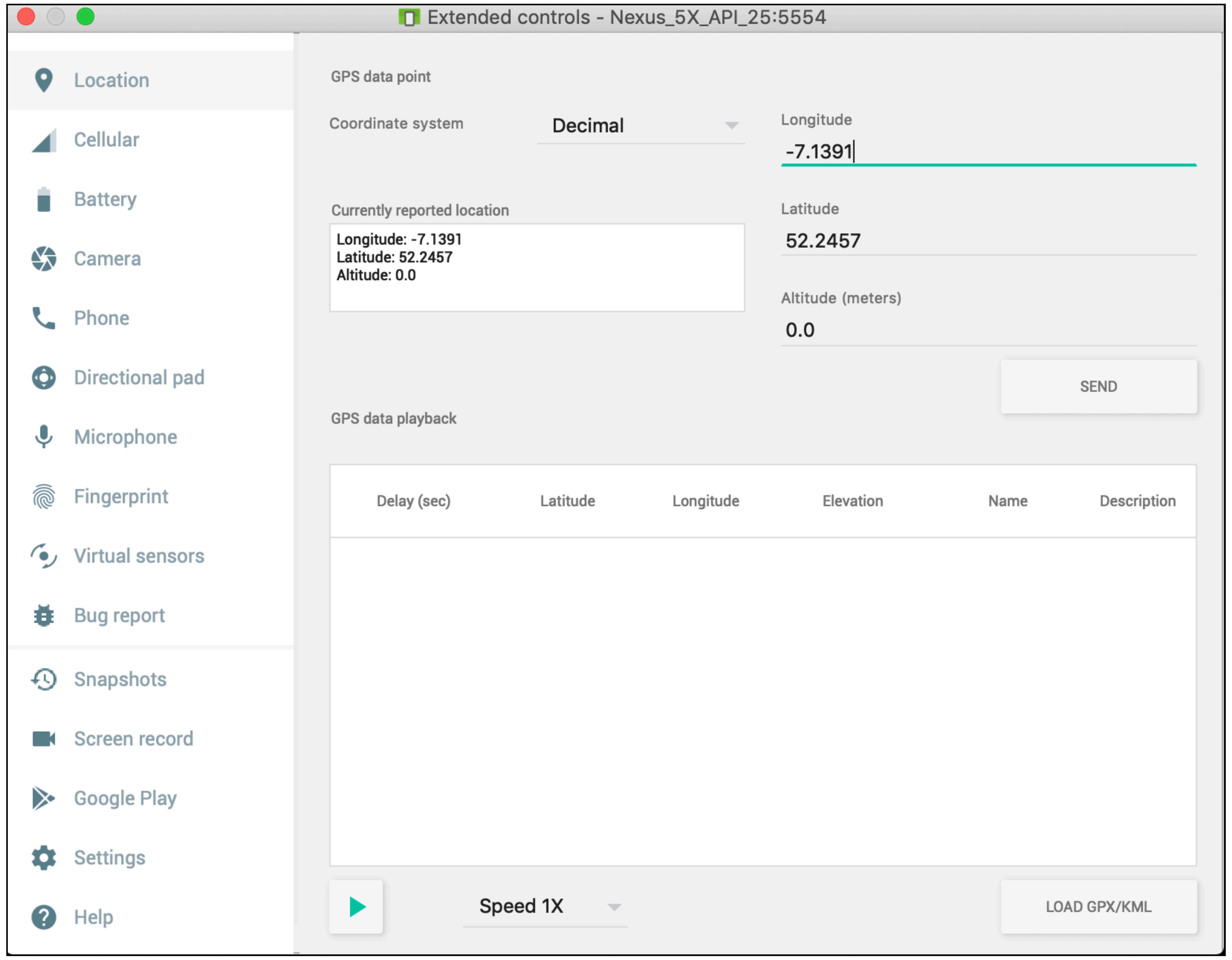
Asynchronous request to location service

```
@SuppressLint("MissingPermission")
fun doSetCurrentLocation() {
  locationService.lastLocation.addOnSuccessListener {
    locationUpdate(it.latitude, it.longitude)
  }
}
```

Location passed as default parameter **it** to callback

Recover lat/lng from **it**
Update map accordingly





Coordinate system: Decimal
Longitude: -7.1391
Latitude: 52.2457
Altitude (meters): 0.0

Currently reported location
Longitude: -7.1391
Latitude: 52.2457
Altitude: 0.0

For '**lastLocation**' there is a noticeable lag - up to several minutes - for the location changes to percolate into the simulator