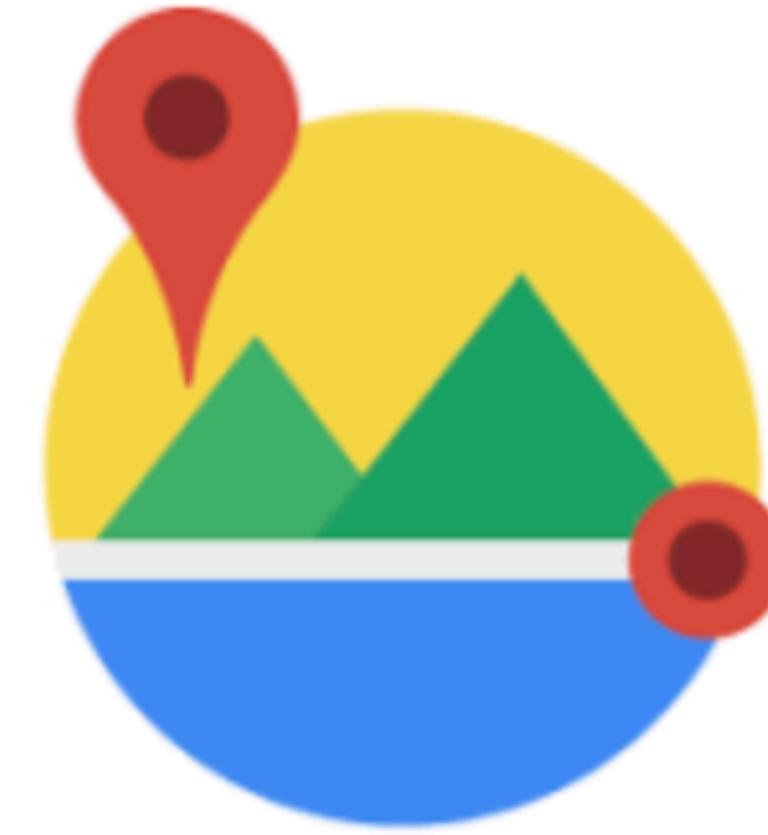


Location Service API

Location API



The location APIs available in
Google Play services

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.

- Choosing the right combination of signals for a specific task in different conditions is not simple.

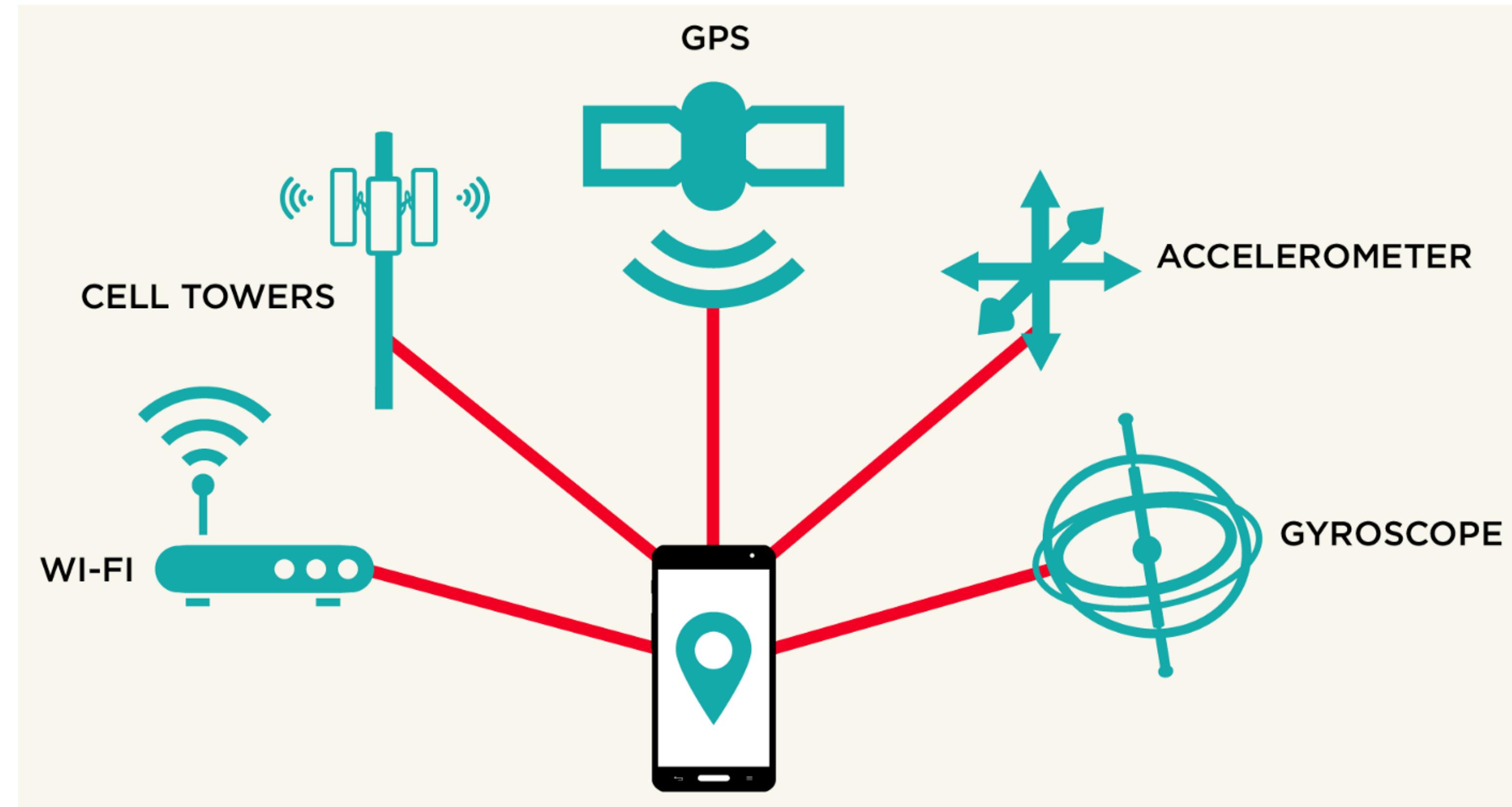
- Finding a solution that is also battery-efficient is even more complicated.

```
/**  
 * 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 suited to  
 * applications that do not require a fine-grained location and that do not  
 * need frequent updates. Gets the best and most recent location currently available.  
 */  
private void getLastLocation() {  
    if(ContextCompat.checkSelfPermission(context: this, Manifest.permission.ACCESS_FINE_LOCATION) == PackageManager.PERMISSION_GRANTED)  
        mFusedLocationClient.getLastLocation()  
            .addOnCompleteListener(activity: this, new OnCompleteListener<Location>() {  
                @Override  
                public void onComplete(@NonNull Task<Location> task) {  
                    if (task.isSuccessful() && task.getResult() != null)  
                        mLastLocation = task.getResult();  
  
                    mLatitudeText.setText(String.format(Locale.ENGLISH,  
                        "%s", mLatitudeLabel),  
                    mLongitudeText.setText(String.format(Locale.ENGLISH,  
                        "%s", mLongitudeLabel));  
                }  
            });  
}
```



Fused Location Provider

- The fused location provider is a location API in Google Play services that intelligently combines different signals to provide location information
- It manages the underlying location technologies, providing a simple API to specify the required quality of service.



Fused Location Provider

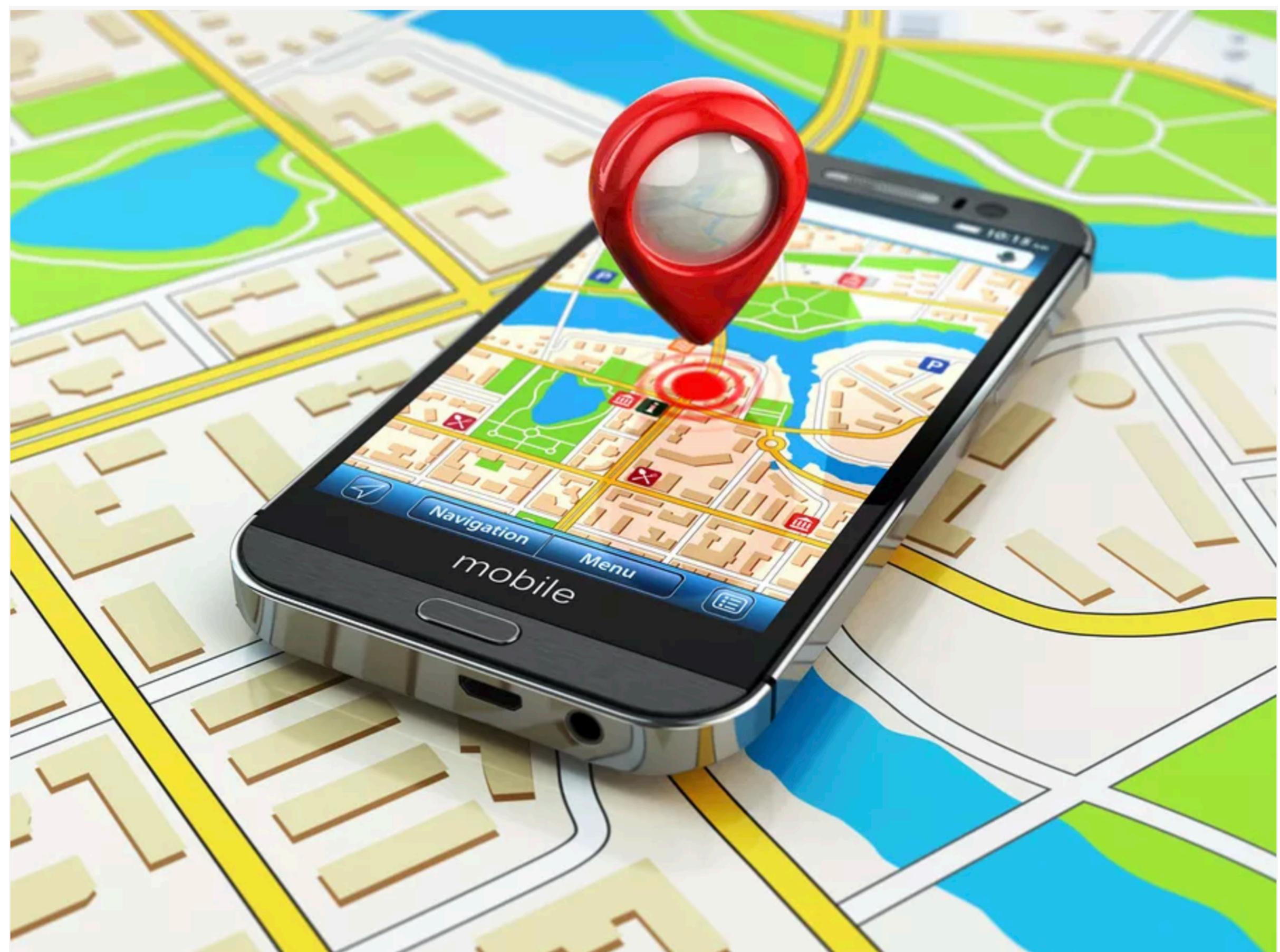
Get the last known location

Change location settings

Receive location updates

Display a location address

Create and monitor geofences



Last Known Location

```
implementation 'com.google.android.gms:play-services-location:16.0.0'
```

```
import com.google.android.gms.location.FusedLocationProviderClient  
  
var locationService: FusedLocationProviderClient = LocationServices.getFusedLocationProviderClient(view)  
  
locationService.lastLocation.addOnSuccessListener {  
  
    println(it.latitude, it.longitude)  
  
}
```

- 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

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

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

    <application
        ...
        </application>

</manifest>
```

- 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.

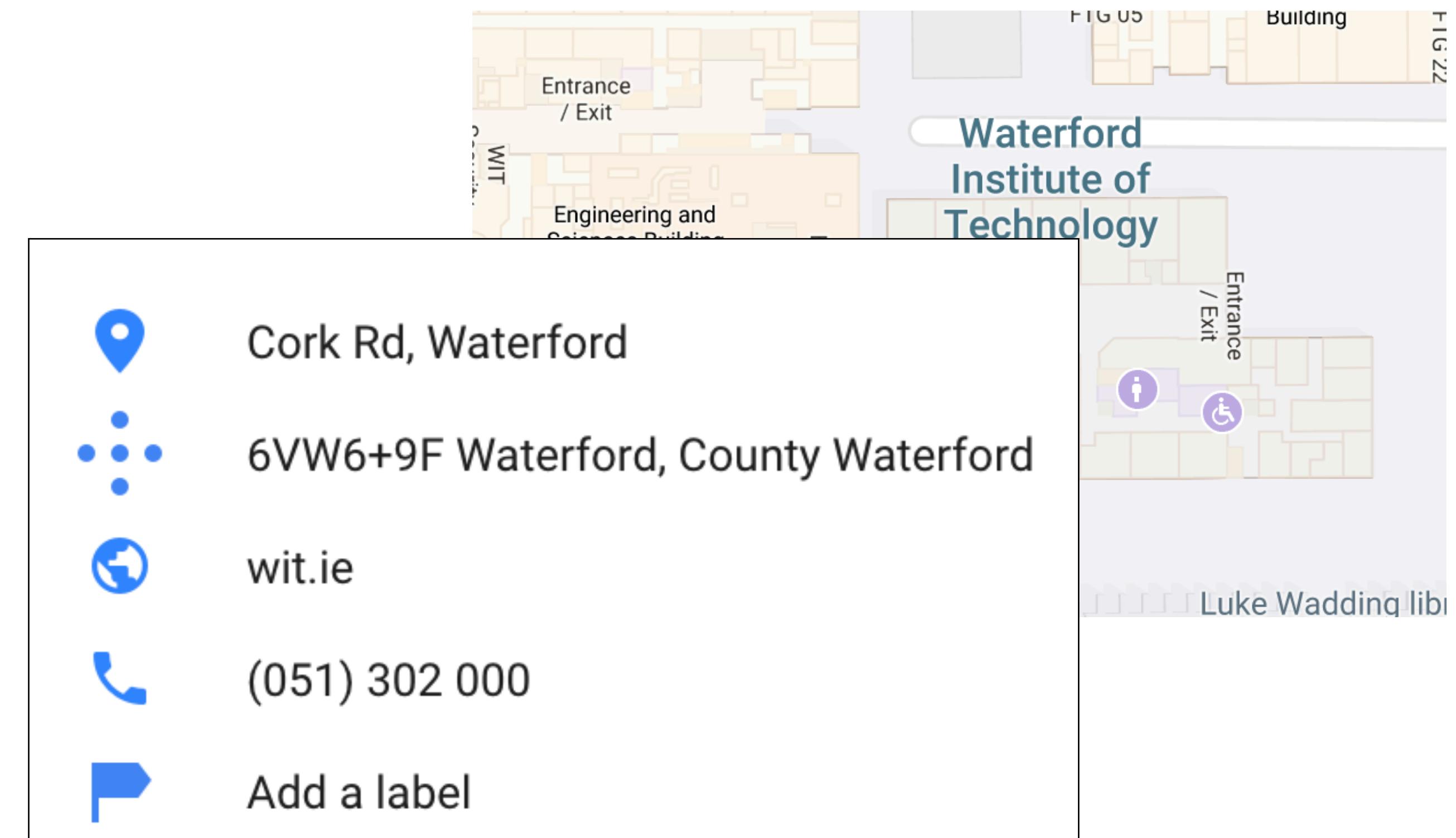
- The fused location provider API can deliver location updates to a callback in your app at specific intervals.
- Specify the desired interval as a parameter of the quality of service.
- By using location updates, your app can provide additional information such as direction and velocity.

Location Updates

```
val locationRequest = LocationRequest().apply {  
    interval = 10000  
    fastestInterval = 5000  
    priority = LocationRequest.PRIORITY_HIGH_ACCURACY  
}  
  
var locationCallback = object : LocationCallback() {  
    override fun onLocationResult(locationResult: LocationResult?) {  
        if (locationResult != null && locationResult.locations != null) {  
            val l = locationResult.locations.last()  
            println(it.latitude, it.longitude)  
        }  
    }  
}  
  
locationService.requestLocationUpdates(locationRequest, locationCallback, null)
```

Location Address

- In some cases the address of the location is more useful.
- A street address may be more meaningful than the geographic coordinates (latitude/longitude) of the location..



Geofences

- Geofencing combines awareness of the user's current location with awareness of the user's proximity to locations that may be of interest.
- To mark a location of interest, you specify its latitude and longitude.
- To adjust the proximity for the location, you add a radius. The latitude, longitude, and radius define a geofence, creating a circular area, or fence, around the location of interest.

