

Lesson 3.2: Location Awareness

ACTIVITY 3.2.1

Preparing for Google Play Services

INTRODUCTION

In this lesson, you will add a Google Play service that will be able to detect a device's location using a Global Positioning System (GPS). **Location services** are one of the many benefits of running an application on a mobile device. Google's location services requires you to create a back-end application on their Google Console and to add location management code on the front end, your TripTracker app.

location services

Services available in the Android framework that allow apps to detect the device's location.

Materials

- Computer with Android™ Studio
- Android™ tablet and USB cable, or a device emulator
- Free Backendless account per student

RESOURCES



Lesson 3.2 Reference Card for Google Play Services
Resources available online

Procedure

Part I: Create a Google Account

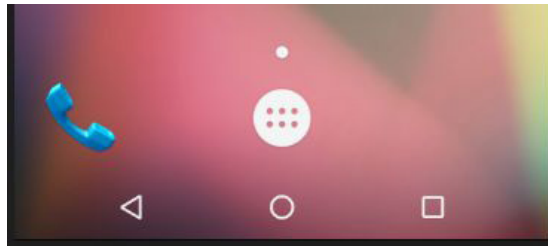
You will first create a Google account on your device and/or your emulator to allow Google Play services to run on your device. Then you will confirm and test the location detection feature.

Note that emulators do not *really* have locations, because they lack a physical GPS. In this case, you will emulate a location so that TripTracker can detect it.

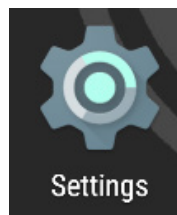
NOTE

The images in this activity reflect an Android emulator to assist you in performing the steps. Physical devices and other emulators may appear differently.

- 1 To view the apps for your device or emulator, select the **Home** button.

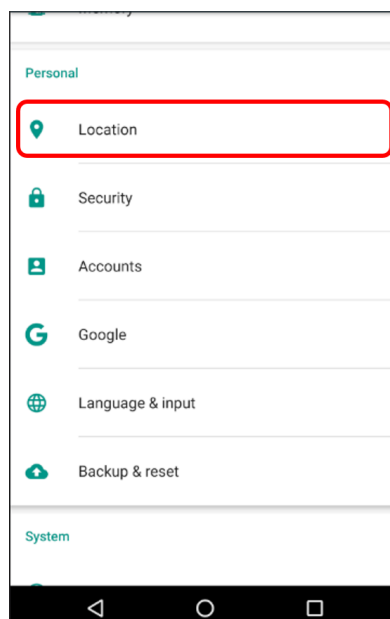


- 2 Click **Settings**.

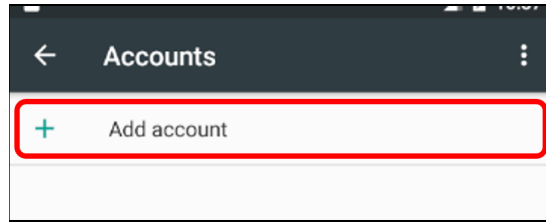


Different devices and emulators will have different organizations of their Settings. For example, you will be adding an account to the Accounts category, and you may have to select **General** or **Device** or some other Setting category to find what you need.

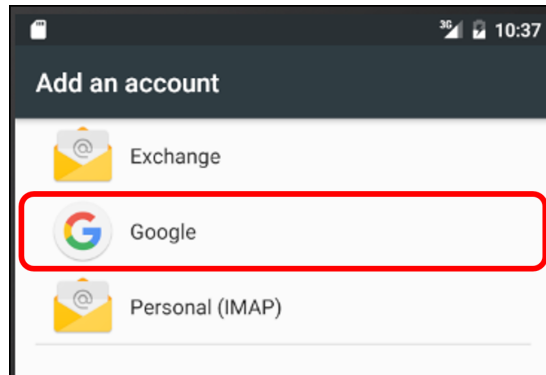
- 3 Find the **Accounts** category in **Settings**. (To scroll on the emulator, click and hold while dragging.)



- 4 Select Add account.

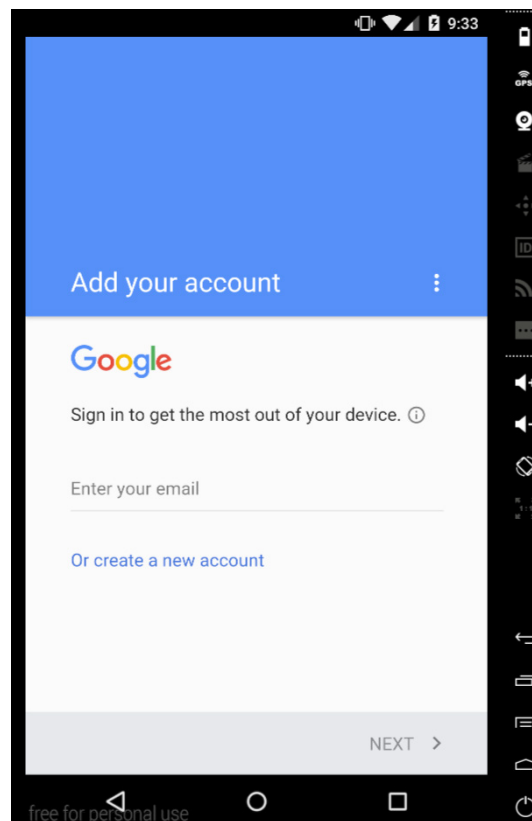


- 5 Select the **Google** account category.



On the screen that opens, do one of the following:

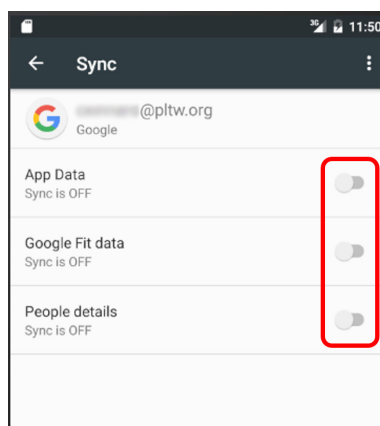
- Add an existing account, if you have a Google account
- Create a new account



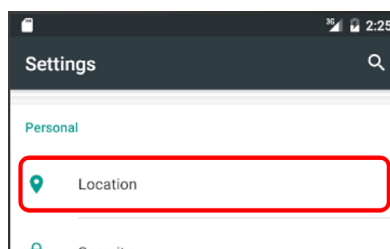
- 6 Provide your Google email address along with your password, or create a new account.
- 7 Accept the conditions and indicate whether you want to back up your device using your Google account.

Important: This interface will act as a real Android compatible device and will sync and notify you of any messages you receive in your Google account.

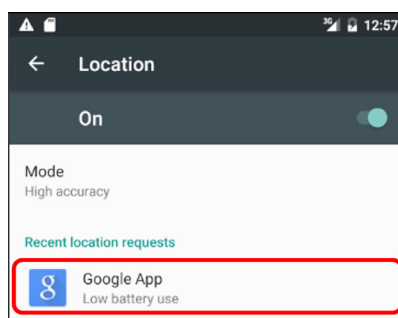
- 8 PLTW recommends that you change the Sync setting, so your device or emulator does not interfere with your Google account, as follows:
 - a. On the Accounts screen, select **Google**.
 - b. Select the account you just created.
 - c. On the Sync screen, disable all of the options.



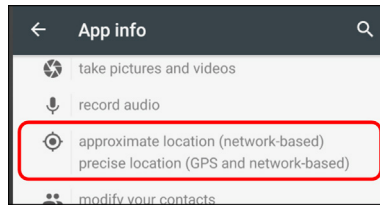
- 9 Navigate **Back** a few times until you return to the Settings screen.
- 10 Find the **Location** category.



- 11 On the Location screen, select **Google App**.



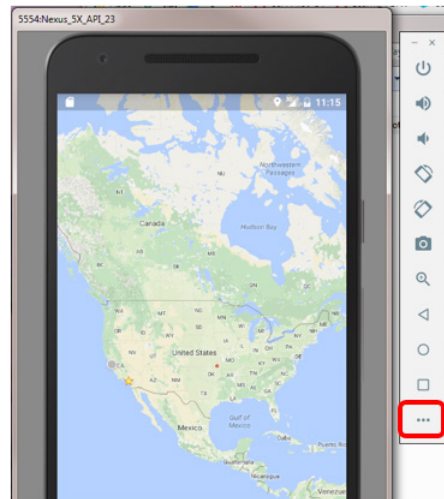
- 12 Confirm that Location is listed in the Permissions section.



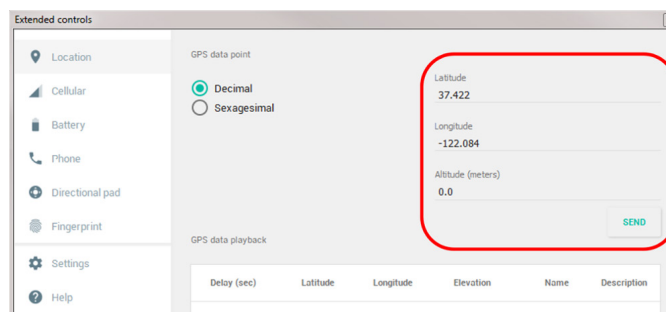
- 13 You may now quit the Settings app.

Part II: Test a Location

- 14 Start the Google Maps app from the Home screen.
- 15 If you are using a physical device, drop a pin or specify a location on the map.
- 16 If you are using an emulator, select the extended controls for your emulator:
- Select the **extended controls** icon.

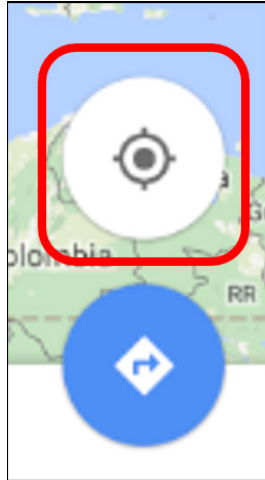


The Extended controls dialog box appears, as shown below. Here you can send a latitude and longitude to your map, emulating a real GPS. You may search for a specific lat/long in a browser with a search phrase such as “London lat long”, or you may use the default location shown, the headquarters of Google in Mountain View, CA.



- Click **Send** to send the location to the Google map app.
- Close the Extended controls window.

- 17 Back on the map, click the **My Location** icon.




The map should show you your current location. To try another location, open extended controls again and enter a new latitude and longitude and select **Send**. Each time you enter a new location, click the **My Location** icon on the map to load the new latitude and longitude.

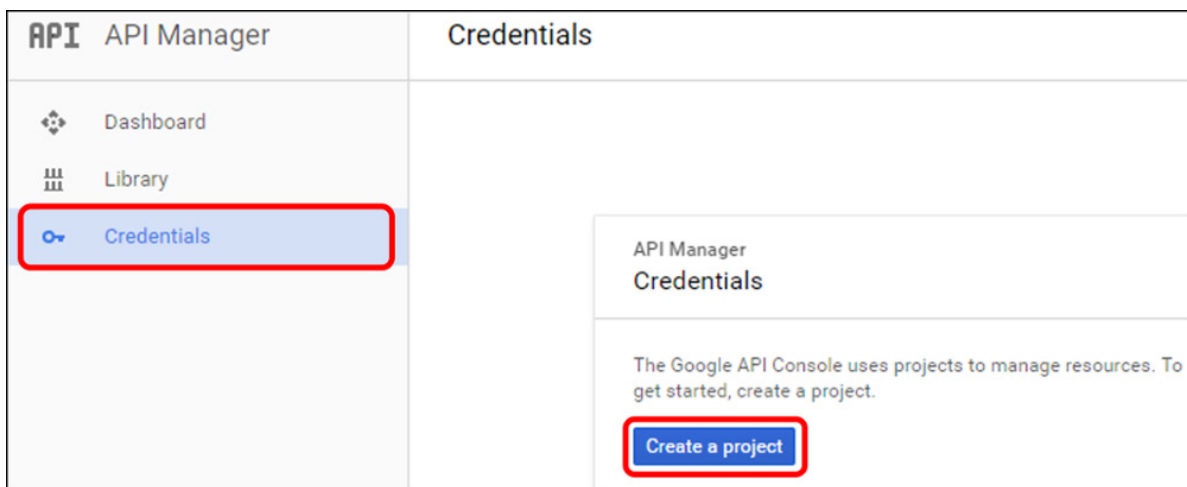
Part III: Create a Google API Project

To use Google Play services such as Maps and location detection in your app, you need to create a project in the Google API Console.

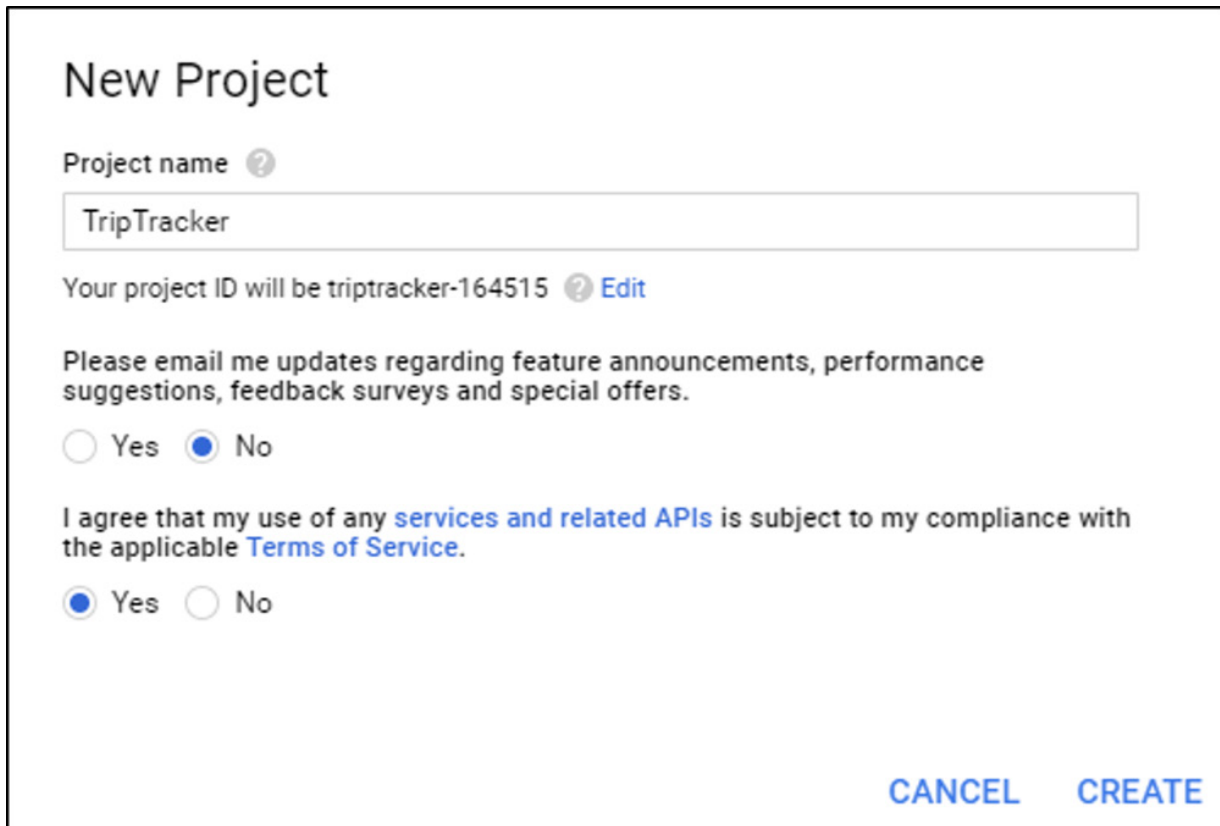
NOTE

This site changes often, so some of the images may not coincide with what you see on the website.

- 18 Visit the  [Google API Console](#).
- 19 To create a new project, select the **Credentials** link on the sidebar and click the **Create a project** button.

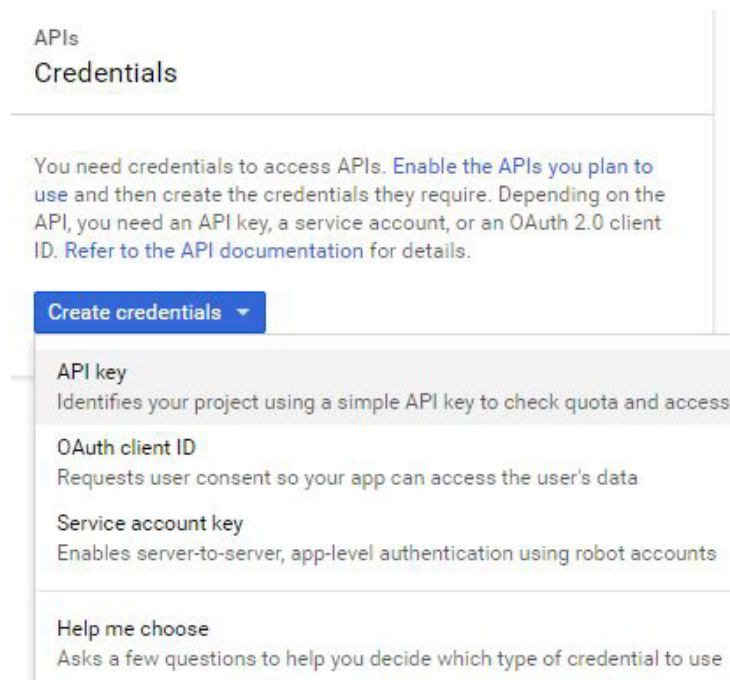


The New Project dialog box opens.



The 'New Project' dialog box has a title 'New Project' at the top. Below it is a 'Project name' label with a help icon, followed by a text input field containing 'TripTracker'. Underneath is a message: 'Your project ID will be triptracker-164515' with a help icon and an 'Edit' link. Then, there is a paragraph: 'Please email me updates regarding feature announcements, performance suggestions, feedback surveys and special offers.' followed by two radio buttons: 'Yes' (unselected) and 'No' (selected). Below that is another paragraph: 'I agree that my use of any services and related APIs is subject to my compliance with the applicable Terms of Service.' followed by two radio buttons: 'Yes' (selected) and 'No' (unselected). At the bottom right are two buttons: 'CANCEL' and 'CREATE'.

- 20 Enter **TripTracker** for the Project name, choose your email preference, and select **Yes** to agree to the Terms of Service. Click **Create**.
- 21 On the Credentials screen, select **Create credentials** > **API key**.



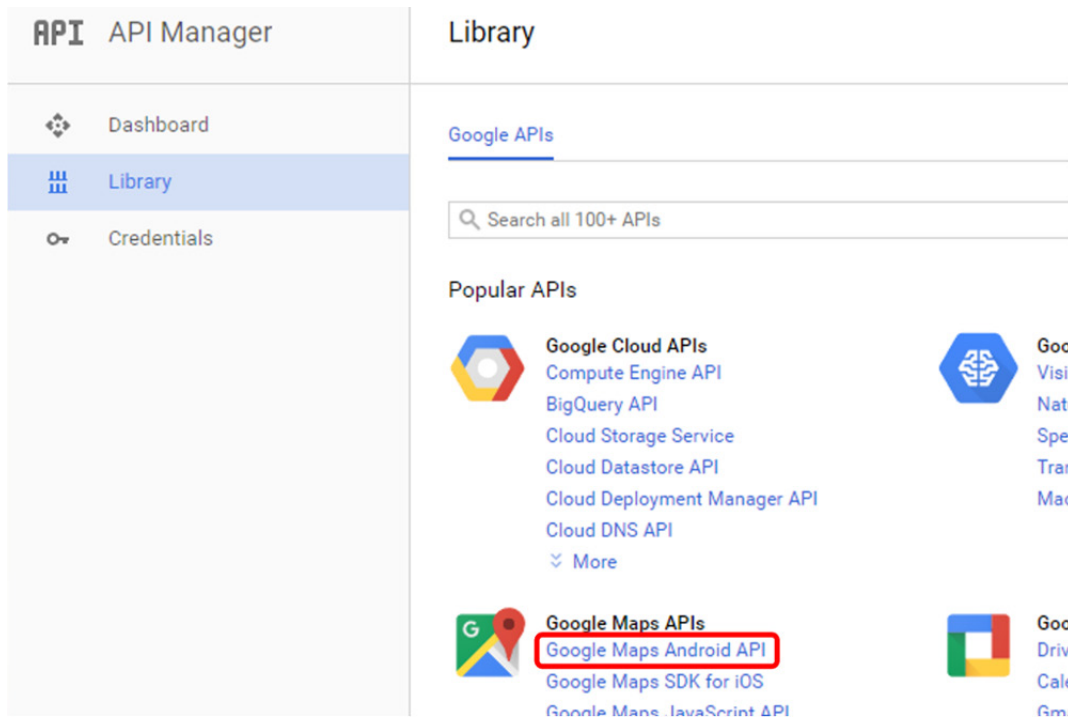
The 'APIs Credentials' screen has a header 'APIs' and 'Credentials'. Below the header is a paragraph: 'You need credentials to access APIs. Enable the APIs you plan to use and then create the credentials they require. Depending on the API, you need an API key, a service account, or an OAuth 2.0 client ID. Refer to the API documentation for details.' Below this is a blue button 'Create credentials' with a dropdown arrow. The dropdown menu is open, showing four options: 'API key' (with a description: 'Identifies your project using a simple API key to check quota and access'), 'OAuth client ID' (with a description: 'Requests user consent so your app can access the user's data'), 'Service account key' (with a description: 'Enables server-to-server, app-level authentication using robot accounts'), and 'Help me choose' (with a description: 'Asks a few questions to help you decide which type of credential to use').

- 22 In the API key created, select **CLOSE**.

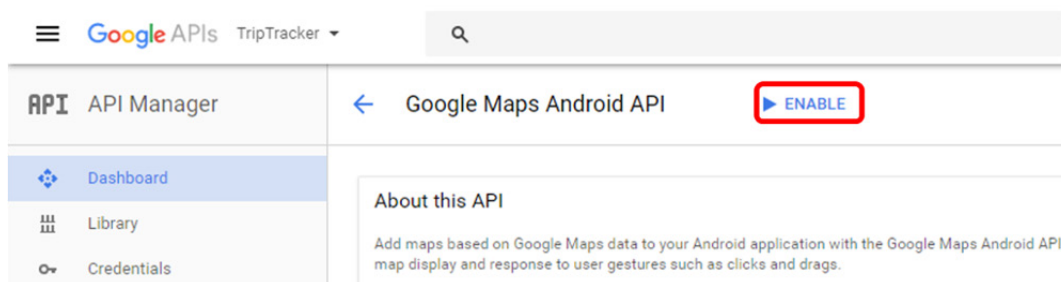
A Google Play services API key is created for you, similar to the Backendless API key. Also, your browser now shows “TripTracker” at the top of the page, next to “Google APIs”.

TripTracker will use the Google Maps feature, so you need a map API for your app:

- 23 Select the **Library** link and then **Google Maps Android API** from the list of Popular APIs.



- 24 On the next page, click **Enable** near the top of the page.



- 25 When the enabling is complete, click the **Credentials** in the sidebar. Your API Key is shown.
- In the next activity, you will add location detection using this Google Play service. You will be instructed how to return to this site to retrieve your API key.

CONCLUSION

1. What are some security risks related to a device that has a GPS and an app that can display locations?

Lesson 3.2 Reference Card for Google Play Services

RESOURCES



Lesson 3.2 Reference Card for Google Play Services
Resources available online

Section 1. Location Awareness Methods

Desired Functionality	Sample Java Code
Create a GoogleApiClient object to define the integration point to Google Play services	<pre>mGoogleApiClient = new GoogleApiClient.Builder(getActivity()) .addConnectionCallbacks(this) .addOnConnectionFailedListener(this) .addApi(LocationServices.API) .build(); //use this in onCreate(Bundle)</pre>
Start the connection to Google Play	<pre>mGoogleApiClient.connect(); //use this in onStart() or onResume()</pre>
End the connection to Google Play	<pre>mGoogleApiClient.disconnect(); //use this in onStop() or onPause()</pre>
Access last location of device	<pre>mLastLocation = LocationServices .FusedLocationApi .getLastLocation(mGoogleApiClient); //use this in onConnected(Bundle)</pre>
Get last location's latitude	<pre>mLastLocation.getLatitude();</pre>
Get last location's longitude	<pre>mLastLocation.getLongitude();</pre>

Section 2. GeoPoints and Arrays

Desired Functionality	Sample Java Code
Create a GeoPoint object to save a location in Backendless	<pre>GeoPoint geoPoint = new GeoPoint(mLastLocation.getLatitude(), mLastLocation.getLongitude());</pre>
Add a title using metadata	<pre>geoPoint.addMetadata("title", locationTitle);</pre>
Add the location to the list of locations in your Trip object	<pre>mTrip.addLocation(geoPoint);</pre>
Get the list of locations for this trip	<pre>mTripCheckIns = new ArrayList<>(trip.getLocations());</pre>
Get a single GeoPoint from a list	<pre>GeoPoint geoLocation = mTripCheckIns.get(0);</pre>

Section 3. Google Maps Methods

Desired Functionality	Sample Java Code
Access the Google Map object associated with the MapFragment UI component	<pre>mGoogleMap = ((MapFragment) getFragmentManager().findFragmentById(R. id.tripMap)).getMap();</pre>
Enable or disable the “My Location” button on a map	<pre>mGoogleMap.setMyLocationEnabled(true false);</pre>
Create a MarkerOptions object and add it to a GoogleMap object	<pre>MarkerOptions options = new MarkerOptions(); options.position(new LatLng(geoLocation. getLatitude(), geoLocation.getLongitude())); options.title(checkInTitle); mGoogleMap.addMarker(options);</pre>
Set Camera Position on a GoogleMap object	<pre>LatLng firstLatLng = new LatLng(firstCheckIn. getLatitude(), firstCheckIn.getLongitude()); CameraPosition firstPosition = new CameraPosition.Builder() .target(firstLatLng) .zoom(15) .build(); mGoogleMap.animateCamera(CameraUpdateFactory.newCameraPosition (firstPosition));</pre>