Chapter 7

Location in Android

Location Based Services

- What makes mobile different is we can build advanced services that only **mobile device with sensing can deliver**
  - **location-based search** for say supermarkets, cafe, cinema, users etc
  - Examples DarkSky(Weather App), MapMyFitness, Glympse (Tracking App), Uber etc
- **Android has two basic ways to determine a user's location.**
  - **built-in location APIs –**since its introduction
  - **Google Play Services –** new and best way
- Google Location Services API provides
  - more powerful, high-level framework that automates tasks such as **location provider choice and power management**.
  - new features such as **activity detection**
- **To use these services**
  - download the Google Play Services SDK using the SDK Manager
  - download an emulator (AVD) image that uses the Google APIs

Location Manager

- Android location manager gives location in terms of **longitude and latitude** for the location of the phone.
- Depending on the location provider selected (could be based on GPS, WiFi or Cellular) the accuracy of the location will vary
- A number of services can be built using these simple components:
  - **get the user's current location**
  - **periodically get the user location as the move around**
  - **use proximity alerts when you move in and out of a predefined area**

```
LocationManager locationManager;
String svcName = Context.LOCATION_SERVICE;
locationManager = (LocationManager)getSystemService(svcName);
```

- Modify androidmanifest to get the user's permission to track their location or get a location reading:

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

- Types of location access permission:
  - *ACCESS_COARSE_LOCATION*- support Network providers (cell towers or wifi)
  - *ACCESS_FINE_LOCATION* -support both GPS and Network providers

Location Provider

- Mobile phones can provide location from a set of providers
  - **GPS-** has good accuracy outdoors but is costly in terms of battery consumption
  - **Cellular** - cheap in terms of energy consumption but could provide very rough location information because of the lack of cell tower density but could be great in the city
- Consider the following in selecting a location provider:
  - power consumption
  - longitude/latitude accuracy
  - altitude accuracy
  - speed
  - direction information

- you can specify the location provider explicitly in the code using a number of constants:
  - LocationManager.GPS_PROVIDER
  - LocationManager.NETWORK_PROVIDER
  - LocationManager.PASSIVE_PROVIDER
- But this would be poor programming because the user might turn off GPS. So **let the Android systems match the user's needs to what providers are on offer** by using *Criteria* as shown below. The code states that the user requires:
  - coarse accuracy
  - low power consumption
  - no altitude, bearing or speed

```
    Criteria criteria = new Criteria();
    criteria.setAccuracy(Criteria.ACCURACY_FINE);
    criteria.setPowerRequirement(Criteria.POWER_LOW);
    criteria.setAltitudeRequired(false);    criteria.setBearingRequired(false);
    criteria.setSpeedRequired(false);    criteria.setCostAllowed(true);
String provider = locationManager.getBestProvider(criteria, true);
```

Geocoding

- is the process of transforming a street address or other description of a location into a (latitude, longitude) coordinate.
- Geocoder supports two services:
  - forward geocoding: from address to longitude/latitude
  - reverse geocoding: from longitude/latitude to address
    - Where latitude and longitude are points for the search
- The Geocoder class comes with the **Google Maps library**. To use the library you have to import it into the application.
- In addition, the Geocoder class **uses a server** to translate over the Internet so you need to add the following permission to the Manifest:

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

Integrating Google map with android app

- provides facility to integrate Google map in our application
- **Types of Google Maps**
  - **Normal:** displays typical road map**, natural features like river and some features** build by humans.
  - **Hybrid:** displays **satellite photograph** data with typical **road maps. It also displays road and feature labels.**
  - **Satellite:** displays **satellite photograph data, but doesn't display road and feature labels.**
  - **Terrain:** displays **photographic** data. This includes **colors, contour lines and labels and perspective shading.**
  - **None:** displays an empty grid with no tiles loaded.

```
googleMap.setMapType(GoogleMap.MAP_TYPE_NORMAL);
googleMap.setMapType(GoogleMap.MAP_TYPE_HYBRID);
googleMap.setMapType(GoogleMap.MAP_TYPE_SATELLITE);
googleMap.setMapType(GoogleMap.MAP_TYPE_TERRAIN);
```

Steps to integrate google map in android application

- install **Google Play Services** SDK in our Android Studio
  - To install Google Play Services, open **Android Studio** → Go to **Tools** menu → **Android** → click **SDK Manager**, then new window will open in that select **SDK Tools** tab → Select **Google Play Services** → click **OK.**
- Create an Android project and select Google maps activity.
- Get a Google Map API key
  - Go to your project an open **google_maps_api.xml file in res/values directory.** Copy the link provided in the **google_maps_api.xml** file
  - Paste the console URL in browser and it will take you to **Google API Console, Create new project and press Agree and Continue.**
  - Click on **Create API Key** to create an API key.
- copy the API Key, go back to android studio and paste the API key into the **<string>** element in **google_maps_api.xml** file.

  <string name="google_maps_key" templateMergeStrategy="preserve" translatable="false">
  AIzaSyCKPTaBv41DKqr9qxMPWOQAsqp0Q4NHMER</string>
- modify AndroidManifest.xml file by adding user permission like:
  - INTERNET: To determine if we are connected to the internet or not.
  - ACCESS_FINE_LOCATION: to use GPS as content provider (Wifi and mobile too)
  - ACCESS_COARSE_LOCATION: to use Wi-Fi and mobile data as content provider
- Add the following to android build gradle dependencies
  - compile 'com.google.android.gms:play-services-maps:16.1.0'