MODUL : GOOGLE MAPS API

**A. Tujuan**

Modul ini membahas mengenai contoh pengembangan aplikasi yang memanfaatkan Google Maps API dan Google Place API, yaitu aplikasi pencarian fasilitas umum yang terdekat dari pengguna, dengan tujuan agar mahasiswa:

* Mengetahui cara membuat *location based app* sederhana di Android menggunakan Google Maps API dan Google Place API
* Dapat mengimplementasikan Google Place API
* Dapat mencari berbagai fasilitas umum yang terdekat dengan lokasi pengguna menggunakan Google Maps API dan Google Place API

**B. DasarTeori**

**1. Google Maps API**

Google Maps merupakan aplikasi pemetaan yang dikembangkan oleh Google, dimana aplikasi ini dapat diimplementasikan pada desktop maupun perangkat mobile dengan berbagai platform, diantaranya Android, iOS, dan BlackBeery OS. Google menyediakan Google Maps API yang memungkinkan kita untuk mengintergrasikan peta yang berasal dari Google Maps ke dalam aplikasi mereka. API secara otomatis akan menangani akses ke server Google Maps, mengunduh data, menampilkan peta, dan merespon *map gesture*. Selain itu, kita dapat menggunakan API untuk menambahkan penanda (*marker*), poligon, dan lapisan (*overlay*) pada peta, serta mengubah user’s view pada wilayah tertentu dalam peta. Objek-objek tersebut digunakan untuk memberikan tambahan informasi lokasi pada peta, dan memungkinkan pengguna untuk berinteraksi langsung dengan peta.

Jika kita menggunakan Google Maps API pada aplikasi yang dikembangkan, kita harus menyertakan atribusi teks Google Play Service sebagai bagian dari “Legal Notices” pada aplikasi kita. Teks atribusi tersebut dapat diperoleh dengan memanggil: GoogleApiAvailability.getOpenSourceSoftwareLicenseInfo

**2. Google Places API**

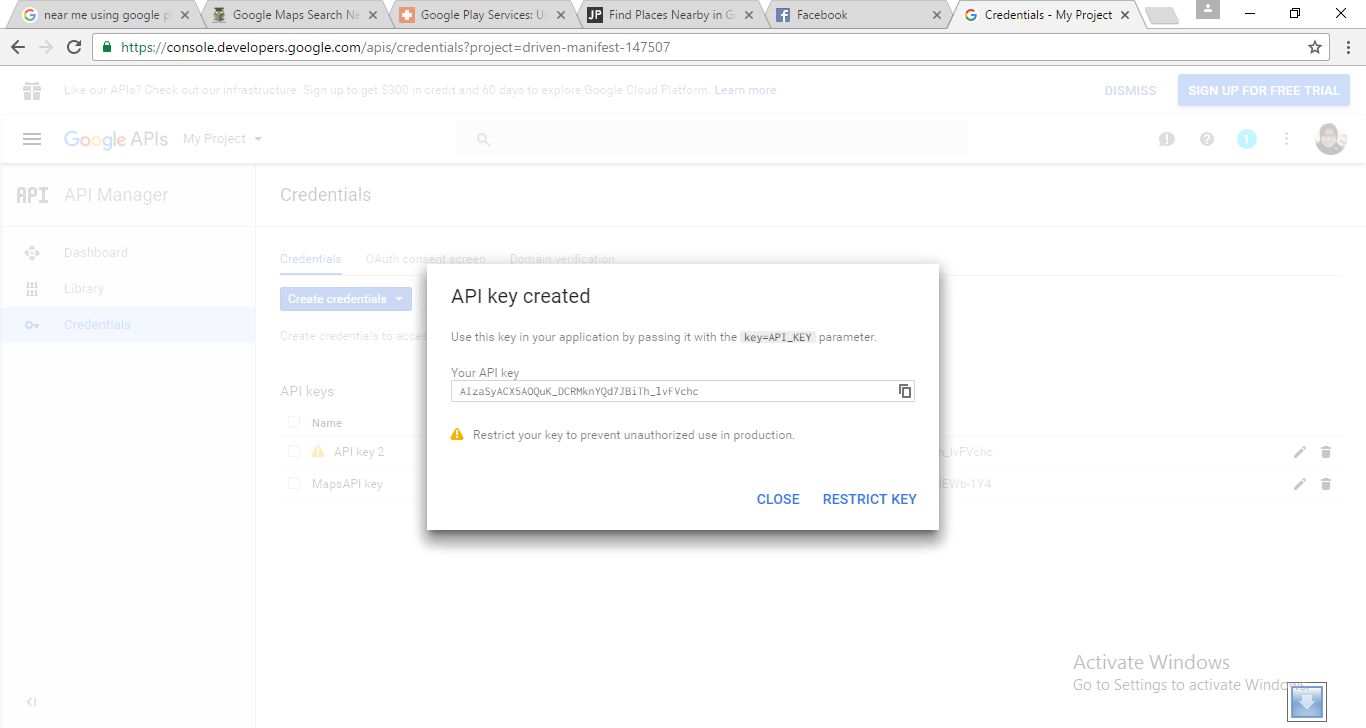
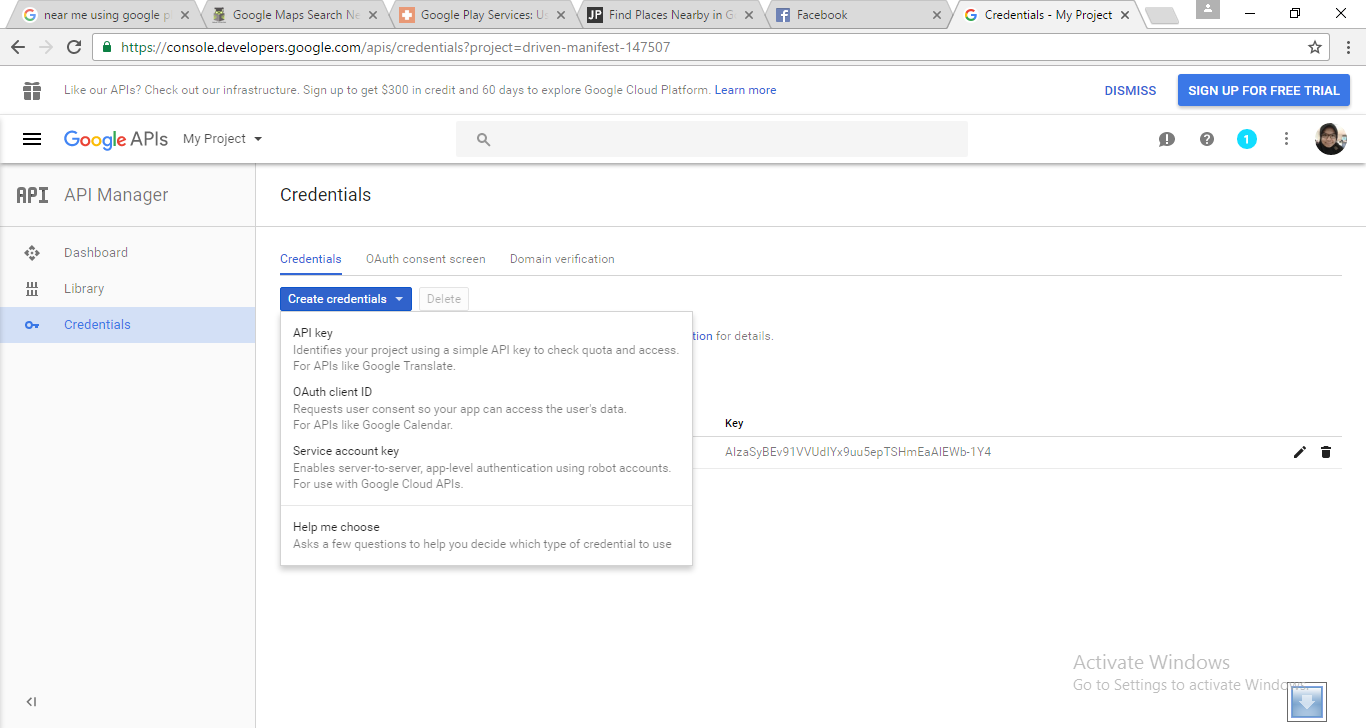
Google Places API merupakan keluarga Google Maps API berupa layanan yang akan mengembalikan informasi mengenai sebuah tempat, seperti lokasi geografis, menggunakan HTTP request. Pada saat meminta (request) informasi suatu tempat, maka lokasinya akan direpresentasikan dengan koordinat latitude dan longitude. Terdapat dua macam request yang tersedia, yaitu: request pencarian tempat dan request detail suatu tempat. Secara garis besar, request pencarian tempat digunakan untuk mengembalikan daftar tempat yang sesuai dengan pencarian, sedangkan request detail tempat akan mengembalikan informasi yang lebih spesifik mengenai suatu tempat. Google Places API dikategorikan menjadi 4 jenis, yaitu

* Google Places API for Android
* Google Places API for iOS
* Places Library in the Google Maps JavaScript API
* Google Places API Web Services

**3. Praktikum Google Maps API dan Google Places API**

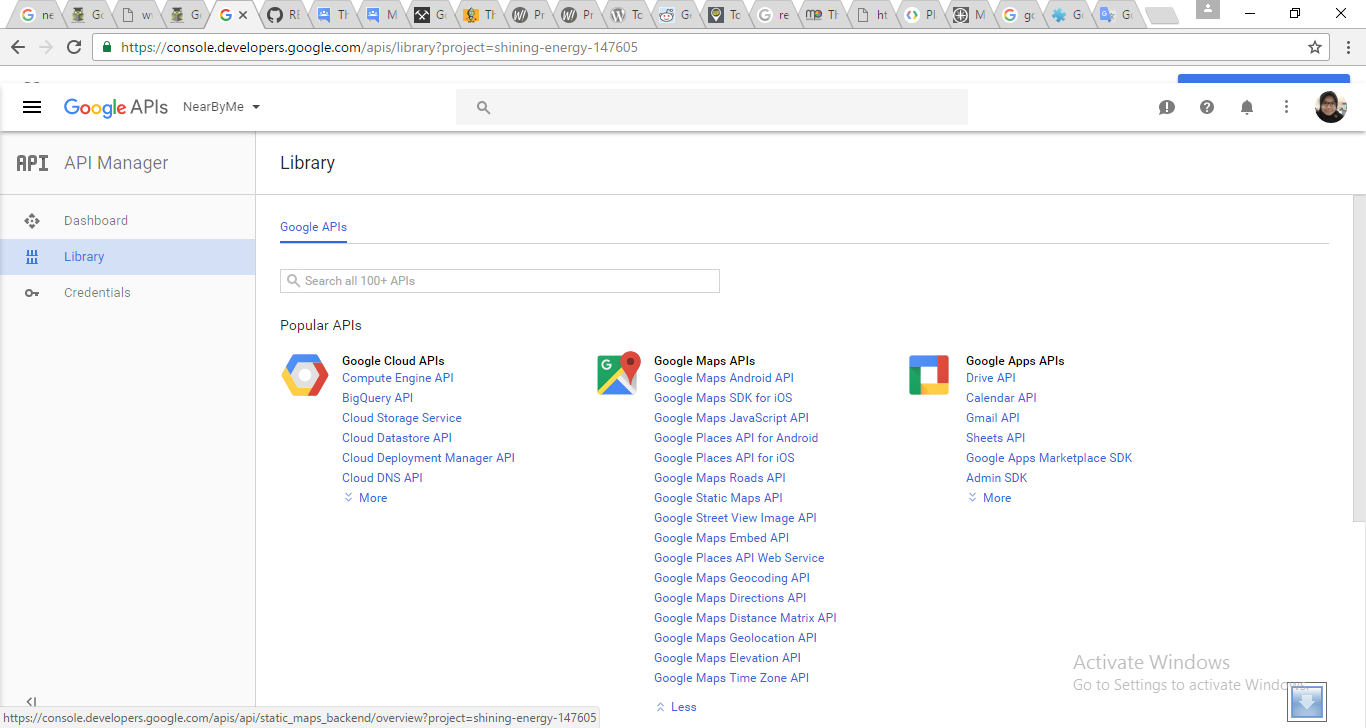
Aplikasi berbasis peta (*location based apps*) yang akan dikembangkan pada modul ini adalah aplikasi pencarian beberapa fasilitas umum, yaitu tempat makan, rumah sakit, dan sekolah, yang lokasinya paling dekat dengan lokasi pengguna saat membuka aplikasi. Tahapan-tahapan yang dilakukan adalah sebagai berikut:

1. Membuka Android Studio dan membuat project baru, kemudian pilihlah Maps Activity dan beri nama untuk Activity tersebut (nama file Activity pada modul ini adalah **activity\_fasilitas\_terdekat.xml** dan **ActivityFasilitasTerdekat.java**). Detail dari proses ini dapat dilihat pada Modul Google Maps API yang pertama.
2. Buatlah API Key pada Google Console melalui link yang terdapat pada file google\_maps\_api.xml. Pada pengembangan aplikasi ini, setelah mendapatkan API key dan melakukan pembatasan untuk aplikasi Android saja (seperti pada modul pertama), kita juga harus membuat Server Key dengan cara klik “Create credentials”, dan kemudian pilih “API Key”. Setelah mendapatkan API Key, letakkan *key* tersebut pada file google\_maps\_api.xml, dan tambahkan *user permission* pada file AndroidManifest.xml seperti pada modul 1. Key yang digunakan pada modul ini tidak perlu dibatasi hanya untuk Android (*Restrict Key*).



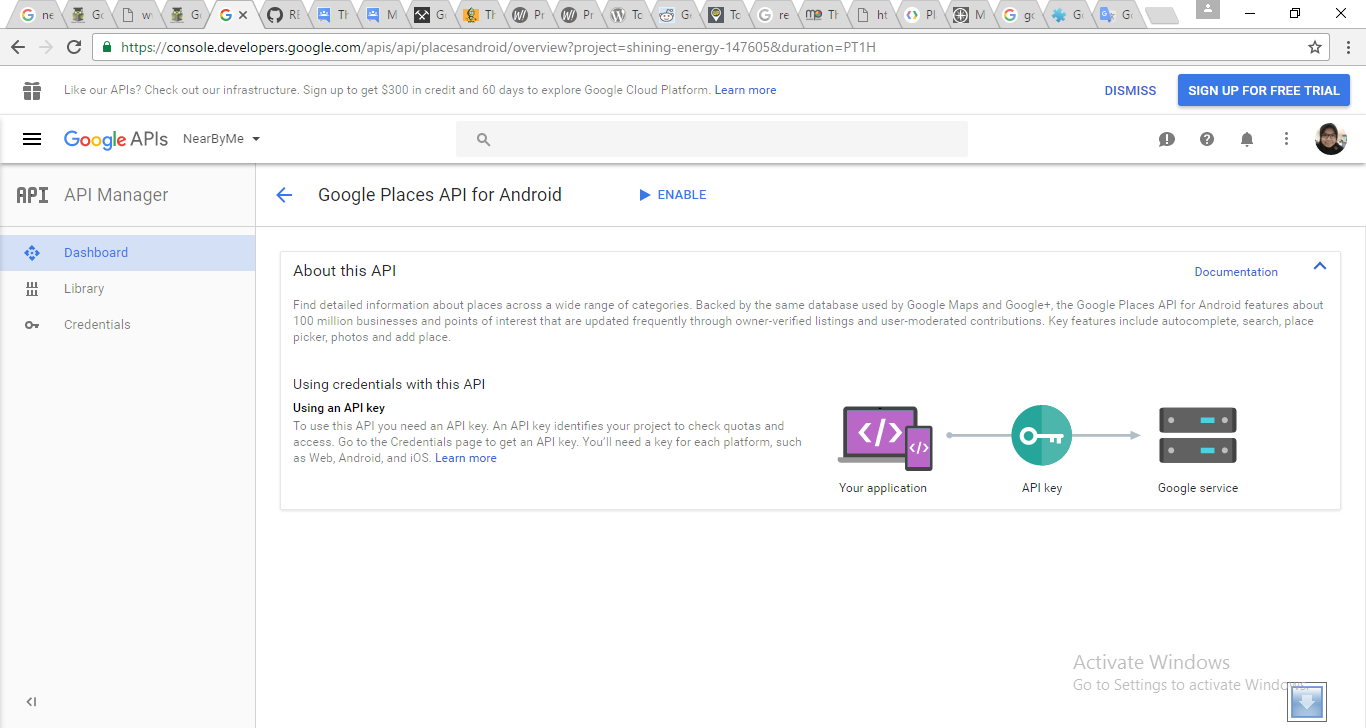
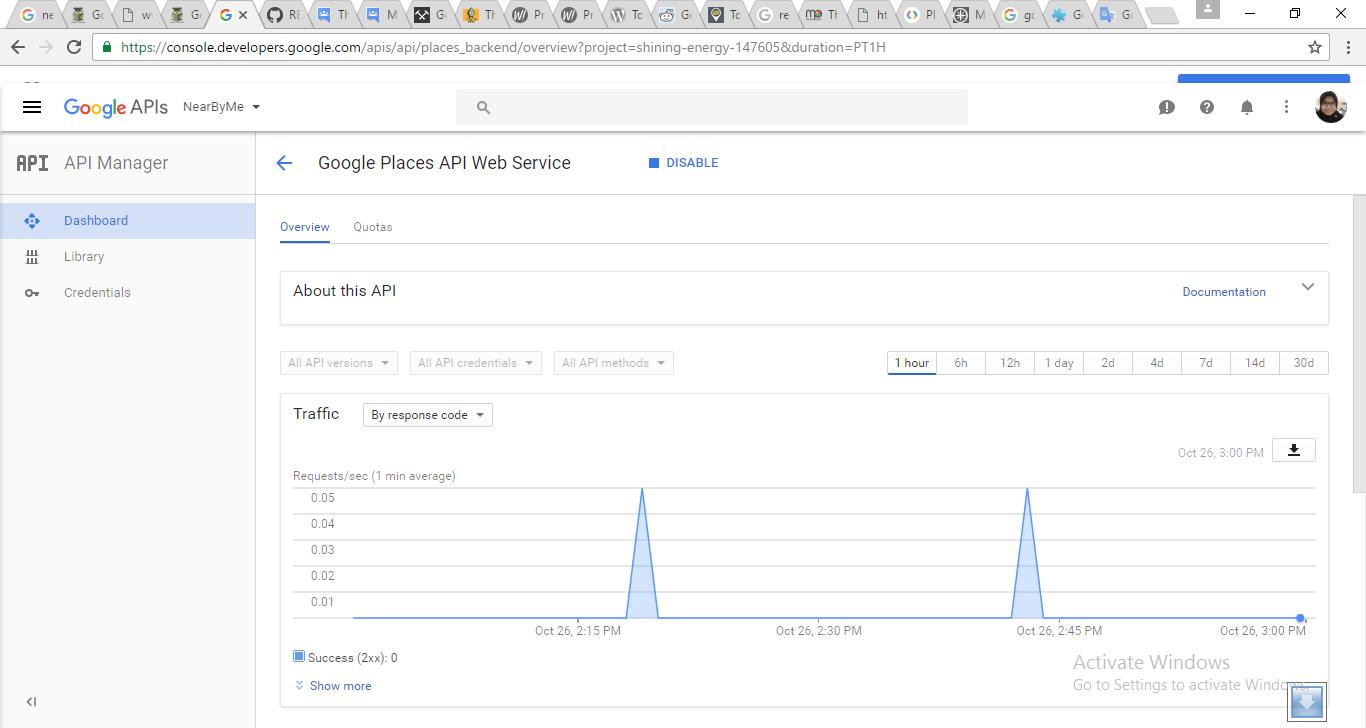
**Gambar 1**. Proses pembuatan Server Key

Kemudian pastikan bahwa Google Places API for Android dan Google Places for Web Service yang ada di halaman console kita telah berstatus “*Enable*”. Pengecekannya dilakukan dengan memilih “Library” yang terletak di sebelah kiri dan pilih Google Places API for Android sera Google Places API for Web Service.



**Gambar 2**. Halaman Library pada Google Console

Jika status saat ini menunjukkan “DISABLE”, artinya Google Places API for Android kita sudah tersedia dan bisa digunakan. Namun jika masih tertulis “ENABLE”, maka kita harus mengubah statusnya. Begitu juga untul Google Places API for Web Service.

**Gambar 3**. Contoh Library dengan status Enable dan Disable

1. Buka file build.gradle pada project kita, dan pastikan bahwa Google Play Service telah tersedia di dalam node “dependencies” seperti di bawah ini:

|  |
| --- |
| dependencies {  compile fileTree(**dir**: **'libs'**, **include**: [**'\*.jar'**])  androidTestCompile(**'com.android.support.test.espresso:espresso-core:2.2.2'**, {  exclude **group**: **'com.android.support'**, **module**: **'support-annotations'** })  compile **'com.android.support:appcompat-v7:24.2.1'** compile **'com.google.android.gms:play-services:9.6.1'** testCompile **'junit:junit:4.12'** } |

1. Tahap selanjutnya adalah menambahkan tiga tombol untuk setiap fasilitas umum. Fasilitas umum yang digunakan pada modul ini adalah tempat makan (*restaurant*), rumah sakit (*hospital*), dan sekolah (*school*). Ketika pengguna menekan tombol *restaurant*, maka akan muncul *marker* yang menunjukkan tempat makan yang paling dekat dengan pengguna, begitu pula dengan kedua tombol yang lainnya. Untuk itu kita menggunakan **FrameLayout** pada file **activity\_fasilitas\_terdekat.xml**, dimana di dalamnya juga dideklarasikan tiga tombol yang merepresentasikan fasilitas umum yang dapat dicari dari aplikasi. File activity\_fasilitas\_terdekat.xml secara keseluruhan adalah sebagai berikut:

|  |
| --- |
| **activity\_fasilitas\_terdekat.xml** |
| <**FrameLayout  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  xmlns:android="http://schemas.android.com/apk/res/android"**>  <**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="com.modul2.nearbyme.ActivityFasilitasTerdekat"** />   <**LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  xmlns:tools="http://schemas.android.com/tools"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:orientation="vertical"**>  <**Button  android:id="@+id/btnRestaurant"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:text="Restaurants"** />   <**Button  android:id="@+id/btnHospital"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:text="Hospitals"** />   <**Button  android:id="@+id/btnSchool"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:text="Schools"** />   </**LinearLayout**> </**FrameLayout**> |

1. Sekarang saatnya masuk ke bagian utama dari pengembangan aplikasi ini. Buka file ActivityFasilitasTerdekat.java dan tambahkan potongan kode pada modul 1 untuk mendapatkan lokasi pengguna pada saat membuka aplikasi (*current location*). Project pada modul ini akan dibagi ke dalam beberapa class yang berbeda untuk memudahkan dalam pengembangan aplikasi. Kita akan membuat tiga kelas baru selain ActivityFasilitasTerdekat.java, yaitu **GetDataTerdekat.java**, **DataParser.java**, dan **DownloadUrl.java**. Langkah-langkahnya adalah sebagai berikut:
2. Pertama-tama, akan dilakukan pengecekan apakah Google Play Service tersedia atau tidak pada fungsi onCreate() yang terdapat pada file ActivityFasilitasTerdekat.java. Untuk pengecekan tersebut kita membuat fungsi **CheckGooglePlayService()** sebagai berikut:

|  |
| --- |
| **private boolean** CheckGooglePlayServices() {  GoogleApiAvailability googleAPI = GoogleApiAvailability.*getInstance*();  **int** result = googleAPI.isGooglePlayServicesAvailable(**this**);  **if**(result != ConnectionResult.***SUCCESS***) {  **if**(googleAPI.isUserResolvableError(result)) {  googleAPI.getErrorDialog(**this**, result, 0).show();  }  **return false**;  }  **return true**; } |

**GoogleApiAvailability** merupakan sebuah class Helper yang digunakan untuk memverifikasi bahwa APK dari Google Play Services telah tersedia pada perangkat Android dan up-to-date. Jika hasilnya adalah **ConnectionResult.SUCCESS** maka koneksinya telah berhasil, jika tidak maka fungsi tersebut akan mengembalikan nilai **false**. Fungsi CheckGooglePlayServices() selanjutnya dipanggil pada fungsi OnCreate, dengan cara:

|  |
| --- |
| *//Cek apakah Google Play Services tersedia atau tidak*  **if** (!CheckGooglePlayServices()) {  Log.*d*(**"onCreate"**, **"Finishing test case since Google Play Services are not available"**);  finish(); } **else** {  Log.*d*(**"onCreate"**,**"Google Play Services available."**); } |

1. Membuat Google API Client dan kemudian mengatur nilai lokasi pengguna (*current user location*) agar dapat diakses. Hal ini dilakukan dengan cara menambah fungsi **buildGoogleApiClient(**) pada file ActivityFasilitasTerdekat.java, yang isinya sama dengan fungsi yang terdapat pada modul 1, yaitu:

|  |
| --- |
| **protected synchronized void** buildGoogleApiClient() {  **mGoogleApiClient** = **new** GoogleApiClient.Builder(**this**)  .addConnectionCallbacks(**this**)  .addOnConnectionFailedListener(**this**)  .addApi(LocationServices.***API***)  .build();  **mGoogleApiClient**.connect(); } |

Tahapan selanjutnya adalah mengatur bagaimana tombol fasilitas dapat bekerja dan cara untuk mendapatkan fasilitas terdekat menggunakan tombol tersebut. Kita mulai implementasinya dari fungsi setOnClickListener(), dan ketika pengguna menekan salah satu tombol fasilitas, maka perintah-perintah yang terdapat pada onClick(View) akan dieksekusi. Kode untuk ketiga tombol fasilitas (*restaurant, hospital, school*) adalah sama, perbedaannya hanya pada nilai string yang akan diberikan tergantung pada tombol yang dipilih oleh pengguna. Untuk tombol restaurant, kodenya adalah sebagai berikut:

|  |
| --- |
| Button btnRestaurant = (Button) findViewById(R.id.***btnRestaurant***); btnRestaurant.setOnClickListener(**new** View.OnClickListener() {  String **Restaurant** = **"restaurant"**;  @Override  **public void** onClick(View v) {  Log.*d*(**"onClick"**, **"Button is Clicked"**);  **mMap**.clear();  String url = getUrl(latitude, longitude, **Restaurant**);  Object[] DataTransfer = **new** Object[2];  DataTransfer[0] = **mMap**;  DataTransfer[1] = url;  Log.*d*(**"onClick"**, url);  GetNearbyPlacesData getNearbyPlacesData = **new** GetNearbyPlacesData();  getNearbyPlacesData.execute(DataTransfer);  Toast.makeText(ActivityFasilitasTerdekat.**this**,**"Nearby Restaurants"**, Toast.***LENGTH\_LONG***).show();  } }); |

Pada kode di atas, pertama-tama Google Maps akan dihapus menggunakan perintah mMap.clear() sehingga penanda (marker) yang sebelumnya ada, akan hilang. Kemudian kita membuat sebuah URL menggunakan fungsi getUrl. Fungsi ini digunakan untuk mendapatkan informasi mengenai restaurant terdekat di Google Maps. URL ini dibuat sesuai dengan Google Developer Guide untuk tempat terdekat (<https://developers.google.com/places/web-service/search>). Terakhir, marker akan ditambahkan pada restaurant terdekat menggunakan getNearbyPlacesData.execute(datatransfer).

1. Fungsi onClick(View v) pada tombol fasilitas, memanggil fungsi getNearbyPlacesData.execute untuk menambahkan marker. Untuk itu, kita perlu membuat class baru untuk mendefinisikan fungsi ini. Buatlah class dengan nama GetNearbyPlacesData.java pada package yang sama dengan ActivityFasilitasTerdekat.java. Caranya adalah: klik kanan pada nama package -> New -> Java class -> tulis nama class -> OK. Class ini merupakan extends dari AsyncTask.

Pada class ini kita menggunakan **DownloadUrl** yang merupakan class tersendiri, dan digunakan untuk mendapatkan data dari URL menggunakan **HttpURLConnection** dan metode File handling. Kita akan membuat class ini pada tahap selanjutnya. Kode yang digunakan untuk mendapatkan data dari URL adalah sebagai berikut:

|  |
| --- |
| **protected** String doInBackground(Object... params) {  **try** {  Log.*d*(**"GetNearbyPlacesData"**, **"doInBackground entered"**);  **mMap** = (GoogleMap) params[0];  **url** = (String) params[1];  DownloadUrl downloadUrl = **new** DownloadUrl();  **googlePlacesData** = downloadUrl.readUrl(**url**);  Log.*d*(**"GooglePlacesReadTask"**, **"doInBackground Exit"**);  } **catch** (Exception e) {  Log.*d*(**"GooglePlacesReadTask"**, e.toString());  }  **return googlePlacesData**;  } |

Setelah mendapatkan data dalam bentuk googlePlacesData, kita memberikannya pada fungsi onPostExecute, sebagai berikut:

|  |
| --- |
| **protected void** onPostExecute(String result) {  Log.*d*(**"GooglePlacesReadTask"**, **"onPostExecute Entered"**);  List<HashMap<String, String>> nearbyPlacesList = **null**;  DataParser dataParser = **new** DataParser();  nearbyPlacesList = dataParser.parse(result);  ShowNearbyPlaces(nearbyPlacesList);  Log.*d*(**"GooglePlacesReadTask"**, **"onPostExecute Exit"**); } |

Data yang berasal dari URL adalah data berbentuk JSON, sehingga harus kita ubah (*parse*) terlebih dahulu. Kita akan mengubah data tersebut menggunakan class bernama DataParser yang akan kita buat pada poin selanjutnya. **dataParser.parse(result)** digunakan untuk mengubah data (*parse data*) dan menyimpan hasilnya sebagai sebuah list pada **nearbyPlacesList**. Dengan demikian, nearbyPlacesList akan memiliki semua informasi mengenai fasilitas terdekat yang dapat kita akses dengan mudah dan juga dapat menambahkan penanda (marker) pada tempat yang sesuai. Marker akan ditambahkan pada Google Maps menggunakan fungsi **ShowNearbyPlaces**. Isi dari class GetNearbyPlacesData adalah sebagai berikut:

|  |
| --- |
| **GetNearbyPlacesData.java** |
| **package** com.modul2.nearbyme;  **import** android.os.AsyncTask; **import** android.util.Log;  **import** com.google.android.gms.maps.CameraUpdateFactory; **import** com.google.android.gms.maps.GoogleMap; **import** com.google.android.gms.maps.model.BitmapDescriptorFactory; **import** com.google.android.gms.maps.model.LatLng; **import** com.google.android.gms.maps.model.MarkerOptions;  **import** org.json.JSONObject;  **import** java.util.HashMap; **import** java.util.List;  **public class** GetNearbyPlacesData **extends** AsyncTask<Object, String, String> {  String **googlePlacesData**;  GoogleMap **mMap**;  String **url**;   @Override  **protected** String doInBackground(Object... params) {  **try** {  Log.*d*(**"GetNearbyPlacesData"**, **"doInBackground entered"**);  **mMap** = (GoogleMap) params[0];  **url** = (String) params[1];  DownloadUrl downloadUrl = **new** DownloadUrl();  **googlePlacesData** = downloadUrl.readUrl(**url**);  Log.*d*(**"GooglePlacesReadTask"**, **"doInBackground Exit"**);  } **catch** (Exception e) {  Log.*d*(**"GooglePlacesReadTask"**, e.toString());  }  **return googlePlacesData**;  }   @Override  **protected void** onPostExecute(String result) {  Log.*d*(**"GooglePlacesReadTask"**, **"onPostExecute Entered"**);  List<HashMap<String, String>> nearbyPlacesList = **null**;  DataParser dataParser = **new** DataParser();  nearbyPlacesList = dataParser.parse(result);  ShowNearbyPlaces(nearbyPlacesList);  Log.*d*(**"GooglePlacesReadTask"**, **"onPostExecute Exit"**);  }   **private void** ShowNearbyPlaces(List<HashMap<String, String>> nearbyPlacesList) {  **for** (**int** i = 0; i < nearbyPlacesList.size(); i++) {  Log.*d*(**"onPostExecute"**,**"Entered into showing locations"**);  MarkerOptions markerOptions = **new** MarkerOptions();  HashMap<String, String> googlePlace = nearbyPlacesList.get(i);  **double** lat = Double.*parseDouble*(googlePlace.get(**"lat"**));  **double** lng = Double.*parseDouble*(googlePlace.get(**"lng"**));  String placeName = googlePlace.get(**"place\_name"**);  String vicinity = googlePlace.get(**"vicinity"**);  LatLng latLng = **new** LatLng(lat, lng);  markerOptions.position(latLng);  markerOptions.title(placeName + **" : "** + vicinity);  **mMap**.addMarker(markerOptions);  markerOptions.icon(BitmapDescriptorFactory.*defaultMarker*(BitmapDescriptorFactory.***HUE\_RED***));  *//move map camera* **mMap**.moveCamera(CameraUpdateFactory.*newLatLng*(latLng));  **mMap**.animateCamera(CameraUpdateFactory.*zoomTo*(11));  }  } } |

1. Seperti yang telah dibahas sebelumnya, kita akan membuat class DownloadURL yang digunakan untuk mendapatkan data dari URL. Class ini harus memiliki path yang sama dengan ActivityFasilitasTerdekat.java dan diberi nama DonwloadUrl.java. Cara membuat class baru sama dengan tahap yang sebelumnya. Data yang dikembalikan dari web adalah data dengan format JSON, dimana pengguna mendapatkannya menggunakan **HttpURLConnection**. Jadi class DownloadUrl akan mengembalikan data JSON dari web. Isi dari class DownloadUrl adalah sebagai berikut

|  |
| --- |
| **DownloadUrl.java** |
| **package** com.modul2.nearbyme;  **import** android.util.Log;  **import** java.io.BufferedReader; **import** java.io.IOException; **import** java.io.InputStream; **import** java.io.InputStreamReader; **import** java.net.HttpURLConnection; **import** java.net.URL;   **public class** DownloadUrl {  **public** String readUrl(String strUrl) **throws** IOException {  String data = **""**;  InputStream iStream = **null**;  HttpURLConnection urlConnection = **null**;  **try** {  URL url = **new** URL(strUrl);   *// Membuat koneksi http to berkomunikasi dengan url* urlConnection = (HttpURLConnection) url.openConnection();   *// Konek ke URL* urlConnection.connect();   *// Membaca data dari URL* iStream = urlConnection.getInputStream();   BufferedReader br = **new** BufferedReader(**new** InputStreamReader(iStream));   StringBuffer sb = **new** StringBuffer();   String line = **""**;  **while** ((line = br.readLine()) != **null**) {  sb.append(line);  }   data = sb.toString();  Log.*d*(**"downloadUrl"**, data.toString());  br.close();   } **catch** (Exception e) {  Log.*d*(**"Exception"**, e.toString());  } **finally** {  iStream.close();  urlConnection.disconnect();  }  **return** data;  } } |

1. Class terakhir yang akan kita buat adalah DataParser, dimana class ini digunakan untuk Mengubah data dari web yang memiliki format JSON. Buatlah class baru yang lokasinya sama dengan ActivityFasilitasTerdekat.java dan beri nama DataParse.java. Isi dari DataParser.java adalah sebagai berikut:

|  |
| --- |
| **DataParser.java** |
| **package** com.modul2.nearbyme;  **import** android.util.Log;  **import** org.json.JSONArray; **import** org.json.JSONException; **import** org.json.JSONObject;  **import** java.util.ArrayList; **import** java.util.HashMap; **import** java.util.List;  **public class** DataParser {  **public** List<HashMap<String, String>> parse(String jsonData) {  JSONArray jsonArray = **null**;  JSONObject jsonObject;   **try** {  Log.*d*(**"Places"**, **"parse"**);  jsonObject = **new** JSONObject((String) jsonData);  jsonArray = jsonObject.getJSONArray(**"results"**);  } **catch** (JSONException e) {  Log.*d*(**"Places"**, **"parse error"**);  e.printStackTrace();  }  **return** getPlaces(jsonArray);  }   **private** List<HashMap<String, String>> getPlaces(JSONArray jsonArray) {  **int** placesCount = jsonArray.length();  List<HashMap<String, String>> placesList = **new** ArrayList<>();  HashMap<String, String> placeMap = **null**;  Log.*d*(**"Places"**, **"getPlaces"**);   **for** (**int** i = 0; i < placesCount; i++) {  **try** {  placeMap = getPlace((JSONObject) jsonArray.get(i));  placesList.add(placeMap);  Log.*d*(**"Places"**, **"Adding places"**);   } **catch** (JSONException e) {  Log.*d*(**"Places"**, **"Error in Adding places"**);  e.printStackTrace();  }  }  **return** placesList;  }   **private** HashMap<String, String> getPlace(JSONObject googlePlaceJson) {  HashMap<String, String> googlePlaceMap = **new** HashMap<String, String>();  String placeName = **"-NA-"**;  String vicinity = **"-NA-"**;  String latitude = **""**;  String longitude = **""**;  String reference = **""**;   Log.*d*(**"getPlace"**, **"Entered"**);   **try** {  **if** (!googlePlaceJson.isNull(**"name"**)) {  placeName = googlePlaceJson.getString(**"name"**);  }  **if** (!googlePlaceJson.isNull(**"vicinity"**)) {  vicinity = googlePlaceJson.getString(**"vicinity"**);  }  latitude = googlePlaceJson.getJSONObject(**"geometry"**).getJSONObject(**"location"**).getString(**"lat"**);  longitude = googlePlaceJson.getJSONObject(**"geometry"**).getJSONObject(**"location"**).getString(**"lng"**);  reference = googlePlaceJson.getString(**"reference"**);  googlePlaceMap.put(**"place\_name"**, placeName);  googlePlaceMap.put(**"vicinity"**, vicinity);  googlePlaceMap.put(**"lat"**, latitude);  googlePlaceMap.put(**"lng"**, longitude);  googlePlaceMap.put(**"reference"**, reference);  Log.*d*(**"getPlace"**, **"Putting Places"**);  } **catch** (JSONException e) {  Log.*d*(**"getPlace"**, **"Error"**);  e.printStackTrace();  }  **return** googlePlaceMap;  } } |

1. Source code file ActivityFasilitasTerdekat.java secara keseluruhan adalah sebagai berikut:

|  |
| --- |
| **ActivityFasilitasTerdekat.java** |
| **package** com.modul2.nearbyme;  **import** android.Manifest; **import** android.content.pm.PackageManager; **import** android.location.Location; **import** android.os.Build; **import** android.support.annotation.NonNull; **import** android.support.annotation.Nullable; **import** android.support.v4.app.ActivityCompat; **import** android.support.v4.app.FragmentActivity; **import** android.os.Bundle; **import** android.support.v4.content.ContextCompat; **import** android.util.Log; **import** android.view.View; **import** android.widget.Button; **import** android.widget.EditText; **import** android.widget.Toast;  **import** com.google.android.gms.common.ConnectionResult; **import** com.google.android.gms.common.GoogleApiAvailability; **import** com.google.android.gms.common.api.GoogleApiClient; **import** com.google.android.gms.location.LocationListener; **import** com.google.android.gms.location.LocationRequest; **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.BitmapDescriptorFactory; **import** com.google.android.gms.maps.model.LatLng; **import** com.google.android.gms.maps.model.Marker; **import** com.google.android.gms.maps.model.MarkerOptions;  **import** java.util.HashMap; **import** java.util.List;  **public class** ActivityFasilitasTerdekat **extends** FragmentActivity **implements** OnMapReadyCallback,  GoogleApiClient.ConnectionCallbacks,  GoogleApiClient.OnConnectionFailedListener,  LocationListener {   **private** GoogleMap **mMap**;  **double latitude**;  **double longitude**;  **private int PROXIMITY\_RADIUS** = 10000;  SupportMapFragment **mapFrag**;  LocationRequest **mLocationRequest**;  GoogleApiClient **mGoogleApiClient**;  Location **mLastLocation**;  Marker **mCurrLocationMarker**;   @Override  **protected void** onCreate(Bundle savedInstanceState) {  **super**.onCreate(savedInstanceState);  setContentView(R.layout.***activity\_fasilitas\_terdekat***);   **if** (android.os.Build.VERSION.***SDK\_INT*** >= Build.VERSION\_CODES.***M***) {  checkLocationPermission();  }   *// Cek apakah Google Play Services tersedia atau tidak* **if** (!CheckGooglePlayServices()) {  Log.*d*(**"onCreate"**, **"Finishing test case since Google Play Services are not available"**);  finish();  }  **else** {  Log.*d*(**"onCreate"**,**"Google Play Services available."**);  }   *// Mendapatkan SupportMapFragment and notifikasi ketika map telah siap digunakan* SupportMapFragment mapFragment = (SupportMapFragment) getSupportFragmentManager()  .findFragmentById(R.id.***map***);  mapFragment.getMapAsync(**this**);  }  *// Pengecekan terhadap ketersediaan Google Play Services*  **private boolean** CheckGooglePlayServices() {  GoogleApiAvailability googleAPI = GoogleApiAvailability.*getInstance*();  **int** result = googleAPI.isGooglePlayServicesAvailable(**this**);  **if**(result != ConnectionResult.***SUCCESS***) {  **if**(googleAPI.isUserResolvableError(result)) {  googleAPI.getErrorDialog(**this**, result, 0).show();  }  **return false**;  }  **return true**;  }   @Override  **public void** onPause() {  **super**.onPause();   *//menghentikan update lokasi ketika Activity sudah tidak aktif lagi* **if** (**mGoogleApiClient** != **null**) {  LocationServices.***FusedLocationApi***.removeLocationUpdates(**mGoogleApiClient**, **this**);  }  }   @Override  **public void** onMapReady(GoogleMap googleMap) {  **mMap** = googleMap;   *//Inisialisasi Google Play Services* **if** (android.os.Build.VERSION.***SDK\_INT*** >= Build.VERSION\_CODES.***M***) {  **if** (ContextCompat.*checkSelfPermission*(**this**,  Manifest.permission.***ACCESS\_FINE\_LOCATION***)  == PackageManager.***PERMISSION\_GRANTED***) {  buildGoogleApiClient();  **mMap**.setMyLocationEnabled(**true**);  }  }  **else** {  buildGoogleApiClient();  **mMap**.setMyLocationEnabled(**true**);  }   Button btnRestaurant = (Button) findViewById(R.id.***btnRestaurant***);  btnRestaurant.setOnClickListener(**new** View.OnClickListener() {  String **Restaurant** = **"restaurant"**;  @Override  **public void** onClick(View v) {  Log.*d*(**"onClick"**, **"Button is Clicked"**);  **mMap**.clear();  String url = getUrl(**latitude**, **longitude**, **Restaurant**);  Object[] DataTransfer = **new** Object[2];  DataTransfer[0] = **mMap**;  DataTransfer[1] = url;  Log.*d*(**"onClick"**, url);  GetNearbyPlacesData getNearbyPlacesData = **new** GetNearbyPlacesData();  getNearbyPlacesData.execute(DataTransfer);  Toast.*makeText*(ActivityFasilitasTerdekat.**this**,**"Nearby Restaurants"**, Toast.***LENGTH\_LONG***).show();  }  });   Button btnHospital = (Button) findViewById(R.id.***btnHospital***);  btnHospital.setOnClickListener(**new** View.OnClickListener() {  String **Hospital** = **"hospital"**;  @Override  **public void** onClick(View v) {  Log.*d*(**"onClick"**, **"Button is Clicked"**);  **mMap**.clear();  String url = getUrl(**latitude**, **longitude**, **Hospital**);  Object[] DataTransfer = **new** Object[2];  DataTransfer[0] = **mMap**;  DataTransfer[1] = url;  Log.*d*(**"onClick"**, url);  GetNearbyPlacesData getNearbyPlacesData = **new** GetNearbyPlacesData();  getNearbyPlacesData.execute(DataTransfer);  Toast.*makeText*(ActivityFasilitasTerdekat.**this**,**"Nearby Hospitals"**, Toast.***LENGTH\_LONG***).show();  }  });   Button btnSchool = (Button) findViewById(R.id.***btnSchool***);  btnSchool.setOnClickListener(**new** View.OnClickListener() {  String **School** = **"school"**;  @Override  **public void** onClick(View v) {  Log.*d*(**"onClick"**, **"Button is Clicked"**);  **mMap**.clear();  **if** (**mCurrLocationMarker** != **null**) {  **mCurrLocationMarker**.remove();  }  String url = getUrl(**latitude**, **longitude**, **School**);  Object[] DataTransfer = **new** Object[2];  DataTransfer[0] = **mMap**;  DataTransfer[1] = url;  Log.*d*(**"onClick"**, url);  GetNearbyPlacesData getNearbyPlacesData = **new** GetNearbyPlacesData();  getNearbyPlacesData.execute(DataTransfer);  Toast.*makeText*(ActivityFasilitasTerdekat.**this**,**"Nearby Schools"**, Toast.***LENGTH\_LONG***).show();  }  });  }   **protected synchronized void** buildGoogleApiClient() {  **mGoogleApiClient** = **new** GoogleApiClient.Builder(**this**)  .addConnectionCallbacks(**this**)  .addOnConnectionFailedListener(**this**)  .addApi(LocationServices.***API***)  .build();  **mGoogleApiClient**.connect();  }   @Override  **public void** onConnected(@Nullable Bundle bundle) {  **mLocationRequest** = **new** LocationRequest();  **mLocationRequest**.setInterval(1000);  **mLocationRequest**.setFastestInterval(1000);  **mLocationRequest**.setPriority(LocationRequest.***PRIORITY\_BALANCED\_POWER\_ACCURACY***);  **if** (ContextCompat.*checkSelfPermission*(**this**,  Manifest.permission.***ACCESS\_FINE\_LOCATION***)  == PackageManager.***PERMISSION\_GRANTED***) {  LocationServices.***FusedLocationApi***.requestLocationUpdates(**mGoogleApiClient**, **mLocationRequest**, **this**);  }  }   **private** String getUrl(**double** latitude, **double** longitude, String nearbyPlace) {   StringBuilder googlePlacesUrl = **new** StringBuilder(**"https://maps.googleapis.com/maps/api/place/nearbysearch/json?"**);  googlePlacesUrl.append(**"location="** + latitude + **","** + longitude);  googlePlacesUrl.append(**"&radius="** + **PROXIMITY\_RADIUS**);  googlePlacesUrl.append(**"&type="** + nearbyPlace);  googlePlacesUrl.append(**"&sensor=true"**);  googlePlacesUrl.append(**"&key="** + **"AIzaSyATuUiZUkEc\_UgHuqsBJa1oqaODI-3mLs0"**);  Log.*d*(**"getUrl"**, googlePlacesUrl.toString());  **return** (googlePlacesUrl.toString());  }   @Override  **public void** onConnectionSuspended(**int** i) {   }   @Override  **public void** onConnectionFailed(@NonNull ConnectionResult connectionResult) {   }   @Override  **public void** onLocationChanged(Location location) {  **mLastLocation** = location;  **if** (**mCurrLocationMarker** != **null**) {  **mCurrLocationMarker**.remove();  }   *//Memberikan marker pada current location*  **latitude** = location.getLatitude();  **longitude** = location.getLongitude();  LatLng latLng = **new** LatLng(location.getLatitude(), location.getLongitude());  MarkerOptions markerOptions = **new** MarkerOptions();  markerOptions.position(latLng);  markerOptions.title(**"Current Position"**);  markerOptions.icon(BitmapDescriptorFactory.*defaultMarker*(BitmapDescriptorFactory.***HUE\_MAGENTA***));  **mCurrLocationMarker** = **mMap**.addMarker(markerOptions);   *//memindahkan map camera* **mMap**.moveCamera(CameraUpdateFactory.*newLatLng*(latLng));  **mMap**.animateCamera(CameraUpdateFactory.*zoomTo*(15));  Toast.*makeText*(ActivityFasilitasTerdekat.**this**,**"Your Current Location"**, Toast.***LENGTH\_LONG***).show();   Log.*d*(**"onLocationChanged"**, String.*format*(**"latitude:%.3f longitude:%.3f"**,**latitude**,**longitude**));   *//menghentikan update lokasi* **if** (**mGoogleApiClient** != **null**) {  LocationServices.***FusedLocationApi***.removeLocationUpdates(**mGoogleApiClient**, **this**);  Log.*d*(**"onLocationChanged"**, **"Removing Location Updates"**);  }  Log.*d*(**"onLocationChanged"**, **"Exit"**);  }   **public static final int *MY\_PERMISSIONS\_REQUEST\_LOCATION*** = 99;  **public boolean** checkLocationPermission(){  **if** (ContextCompat.*checkSelfPermission*(**this**,  Manifest.permission.***ACCESS\_FINE\_LOCATION***)  != PackageManager.***PERMISSION\_GRANTED***) {  **if** (ActivityCompat.*shouldShowRequestPermissionRationale*(**this**,  Manifest.permission.***ACCESS\_FINE\_LOCATION***)) {  ActivityCompat.*requestPermissions*(**this**,  **new** String[]{Manifest.permission.***ACCESS\_FINE\_LOCATION***},  ***MY\_PERMISSIONS\_REQUEST\_LOCATION***);  } **else** { ActivityCompat.*requestPermissions*(**this**,  **new** String[]{Manifest.permission.***ACCESS\_FINE\_LOCATION***},  ***MY\_PERMISSIONS\_REQUEST\_LOCATION***);  }  **return false**;  } **else** {  **return true**;  }  }   @Override  **public void** onRequestPermissionsResult(**int** requestCode,  String permissions[], **int**[] grantResults) {  **switch** (requestCode) {  **case *MY\_PERMISSIONS\_REQUEST\_LOCATION***: { **if** (grantResults.**length** > 0  && grantResults[0] == PackageManager.***PERMISSION\_GRANTED***) {  **if** (ContextCompat.*checkSelfPermission*(**this**,  Manifest.permission.***ACCESS\_FINE\_LOCATION***)  == PackageManager.***PERMISSION\_GRANTED***) {   **if** (**mGoogleApiClient** == **null**) {  buildGoogleApiClient();  }  **mMap**.setMyLocationEnabled(**true**);  }   } **else** { Toast.*makeText*(**this**, **"permission denied"**, Toast.***LENGTH\_LONG***).show();  }  **return**;  } }  } } |

Hasil dari Map yang berisi tiga tombol yang merepresentaikan fasilitas berupa restaurant, hospital, dan school beserta lokasi pengguna pada saat membuka aplikasi dapat dilihat pada Gambar 4. Kemudian Gambar 5, 6, dan 7 menunjukkan peta dengan marker yang merepresentasikan masing-masing fasilitas yang paling dekat dengan lokasi pengguna. Gambar 8 menunjukkan peta yang di-zoom in. Jika kita memilih salah satu marker yang terdapat pada maps, maka akan muncul nama tempat pada marker tersebut, seperti yang terlihat pada Gambar 9. Di bagian pojok kanan bawah terdapat dua tombol, yaitu arah (direction) dan peta (maps). Apabila kita memilih direction, maka akan muncul rute perjalanan menuju lokasi tersebut beserta estimasi waktunya (Gambar 10). Jika kita memilih maps, maka akan muncul Google Maps beserta koordinat dan detail dari lokasi yang bersangkutan (Gambar 11).

|  |  |
| --- | --- |
| **Gambar 4**. Hasil Maps dengan 3 tombol fasilitas | **Gambar 5**. Hasil Maps jika menekan tombol Restaurant |
| **Gambar 6**. Hasil Maps jika menekan tombol Hospitals | **Gambar 7**. Hasil Maps jika menekan tombol Schools |
| **Gambar 8**. Hasil Maps ketika di-zoom in | **Gambar 9**. Hasil Maps jika menekan salah satu marker |
| **Gambar 10**. Hasil Maps jika menekan tombol directions | **Gambar 11**. Hasil Maps jika menekan tombol Maps |