**Mali Taxi**

**Report Implementation**

20/05/2016

4.1. DESCRIPTION COMPONENT 1: Passenger component

4.1.1. PROCESSING NARRATIVE (PSPEC) FOR COMPONENT 1

Passenger Component สร้างขึ้นเพื่อใช้จัดการในแอปพลิเคชั่นส่วนของผู้โดยสาร

Class MapsActivity เป็น Class ที่มี attribute

private GoogleMap mMap;

ใช้เก็บแผนที่จาก google map

private Location last\_location;

ใช่เก็บตำแหน่งล่าสุด

private GoogleApiClient mGoogleApiClient;

ใช้เรียกการใช้งานgoogle map api

private Marker mkSource;

ใช้เก็บ maker ของตำแหน่ง Source

private Marker mkDestination;

ใช้เก็บ maker ของตำแหน่ง Destination

int PLACE\_PICKER\_REQUEST\_SOURCE = 1; ใช้สำหรับตรวจสอบว่าเป็นการทำงานโดย set source หรือ set destination

int PLACE\_PICKER\_REQUEST\_DESTINATION = 2;

boolean first = false;

ใช้ check การทำงานรอบแรก

และมี method

protected void onCreate

public void onMapReady

protected void onStart

public void onBtnClicked

protected void onActivityResult

public static double getMeterFromLatLng

public static double calculatePriceFromMeter

protected void onStop()

private LocationRequest createLocationRequest

public void onLocationChanged

public void onConnected

class backLoadJson เป็น class ที่มี method

protected String doInBackground

protected void onPostExecute

ช่วยในการติดต่อกับฐานข้อมูลในส่วนของผุ้ขับเพื่อแสดงตำแหน่ง

4.1.2. COMPONENT 1 PROCESSING DETAIL

4.1.2.2. Restrictions/limitations for component 1

* การใช้งาน Passenger Component จะถูกใช้งานได้เมื่อใช้ผ่านแอปพลิเคชั่นของ Passenger เท่านั้น
* หน้าที่ถูกใช้งานสามารถแสดงตำแหน่งรถโดยสารโดยรอบ ที่อยู่ปัจจุบัน และ จุดรับผู้โดยสารกับจุดส่งผู้โดยสารเท่านั้น
* ไม่สามารถระบุผู้ใช้งานได้
* จำเป็นต้องใช้งานผ่านอินเตอร์เน็ต เพื่อระบุ location และใช้งานได้
* จำเป็นต้องขอ permission จาก ผู้ใช้

4.1.2.3. Performance issues for component 1

* สามารถเชื่อมต่อได้เฉพาะฐานข้อมูลของผู้ขับ
* หากไม่สามารถใช้บริการอินเตอร์เน็ตจะไม่สามารถใช้บริการแอปพลิเคชั่นได้

4.1.2.4. Design constraints for component 1

* ยังไม่สามารถใช้งานได้จริง

4.1.2.5. Processing detail for each operation of component n

**Class MapsActivity**

4.1.2.5.1.1 Processing narrative (PSPEC) for each operation

protected void onCreate(Bundle savedInstanceState) การเรียกการใช้งานครั้งแรก แสดง activity\_maps.xml เป็น แผนที่ ตำแหน่งปัจจุบัน และปุ่มที่ใช้ในการ set Source and Destination

public void onMapReady(GoogleMap googleMap) ตรวจสอบ permission ในการใช้งาน maps รวมถึงการเปิดการใช้งาน Location

protected void onStart() set การเชื่อมต่อกับ google api

public void onBtnClicked(View v) เป็น method ที่ใช้ในการรอรับการทำงานเมื่อกดปุ่ม set source, set destination และ Refresh! โดย ถ้ากด set จะเป็นการเรียกใช้ onActivityResult และกด Refresh! จะเป็นการเรียกใช้ backLoadJson

protected void onActivityResult(int requestCode, int resultCode, Intent data) ใช้เพื่อสร้างหมุดปักสำหรับตำแหน่งที่ระบุ และถ้าหากมีตำแหน่งครบทั้ง source และ destination จะทำการแสดง ราคา ตามระยะทาง

public static double getMeterFromLatLng(LatLng source, LatLng destination) ใช้เพื่อคำนวณหาระยะทางระหว่าง source และ destination

public static double calculatePriceFromMeter(double meter) ใช้เพื่อคำนวณหาราคาตามระยะทาง

protected void onStop() ยกเลิก การเชื่อมต่อกับ google api

private LocationRequest createLocationRequest() เถูกเรียกใช้เมื่อมีการสร้างและต้องการเปลี่ยนค่าlocation

public void onLocationChanged(Location location) ใช้สำหรับเปลี่ยนค่า last\_location

public void onConnected(@Nullable Bundle bundle) เป็นการเรียกใช้งานในครั้งแรก จะแสดงตำแหน่งผู้เรียกใช้งาน กับ ตำแหน่งผู้ขับในฐานข้อมูล

**Class backLoadJson**

protected String doInBackground(Void... params) เป็นการเชื่อมต่อกับฐานข้อมูล

protected void onPostExecute(String result) ทำการแสดงแผนที่และปักหมุดตำแหน่งและชื่อของผู้ขับ

4.1.2.5.1.2 Algorithmic model (e.g., PDL) for each operation

protected void onCreate(Bundle savedInstanceState)

start

เรียกการแสดงหน้า activity\_maps.xml

ใช้ google api ในการเชื่อมต่อเพื่อการแสดง map

if ยังไม่มีการเชื่อมต่อ

ให้ทำการเชื่อมต่อ

end

public void onMapReady(GoogleMap googleMap)

start

if ยังไม่มีการให้ permission

ให้ทำการร้องขอ permission จากผู้ใช้

ทำการ set ตำแหน่งปัจจุบัน

end

protected void onStart()

start

ทำการเชื่อมต่อ google api

end

public void onBtnClicked(View v)

start

if ผู้ใช้กดปุ่ม Refresh!

ทำการตรวจสอบตำแหน่งผู้ขับใหม่อีกครั้ง

else if ผู้ใช้กดปุ่ม set Source

เรียกใช้ startActivityForResult

else if ผู้ใช้กดปุ่ม set destination

เรียกใช้ startActivityForResult

end

protected void onActivityResult(int requestCode, int resultCode, Intent data)

start

if ผู้ใช้กดปุ่ม set Source

if มีปักหมุดของ source แล้ว

ลบปักหมุดอันเดิมทิ้ง

สร้างปักหมุดขึ้นใหม่พร้อมแสดงชื่อ

if มีปักหมุดของ destination แล้ว

แสดงราคาของเส้นทาง

else ผู้ใช้กดปุ่ม set Source

if มีปักหมุดของ destination แล้ว

ลบปักหมุดอันเดิมทิ้ง

สร้างปักหมุดขึ้นใหม่พร้อมแสดงชื่อ

if มีปักหมุดของ source แล้ว

แสดงราคาของเส้นทาง

end

public static double getMeterFromLatLng(LatLng source, LatLng destination)

start

นำ latitude และ longitude ของ source และ destiantion มาลบกัน

นำมาหาระยะกระจัดระหว่างสองจุด

return ระยะทาง

end

public static double calculatePriceFromMeter(double meter)

start

if meter น้อยกว่า 1 กิโลเมตร

return 35

else if น้อยกว่า 11 กิโลเมตร

return 35 + (meter-1000)\*5.5

else if น้อยกว่า 21 กิโลเมตร

return 95.5 + (meter-11000)\*6.0

else if มากกว่า 21 กิโลเมตร

return 155.5 + (meter-21000)\*6.5

else return 0

end

protected void onStop()

start

ยกเลิกการเชื่อมต่อ google api

end

private LocationRequest createLocationRequest()

start

สร้าง LocationRequest

กำหนดค่าให้มี priority สูง เพื่อให้มีการ update ค่าเป็นปัจจุบัน

return mLocationRequest

end

public void onLocationChanged(Location location)

start

ให้ last\_location = location;

end

public void onConnected(@Nullable Bundle bundle)

start

เช็ค permission

สร้าง location และทำการ update ค่า

if ไม่ใช่การแสดงผลครั้งแรก

เปลี่ยน first = true

เรียกใช้การเชื่อมต่อกับฐานข้อมูลผู้ขับ

end

class backLoadJson

protected String doInBackground(Void... params)

start

ทำการเชื่อมต่อกับฐานข้อมูล

end

protected void onPostExecute(String result)

start

นำค่า lat lng name ของผู้ขับแต่ละคนในฐานข้อมูลมา

แสดงเป็นปักหมุดตำแหน่งของผู้ขับ

end

**5. USER INTERFACE DESIGN**

5.3. SCREEN IMAGES AND DESCRIPTION

แอปพลิเคชั่น ของ Mali Taxi จะแบ่งเป็น 2 แอปพลิเคชั่น โดยแบ่งเป็น 2 ผู้ใช้งาน ได้แก่ ผู้ใช้บริการ (Passenger) มีชื่อแอปพลิเคชั่นว่า “Mali Taxi” และ ผู้ให้บริการ (Driver) มีชื่อแอปพลิเคชั่นว่า “Mali Driver”

5.3.1 แอปพลิเคชั่น Mali Taxi ของ ผู้ใช้บริการ (Passenger)

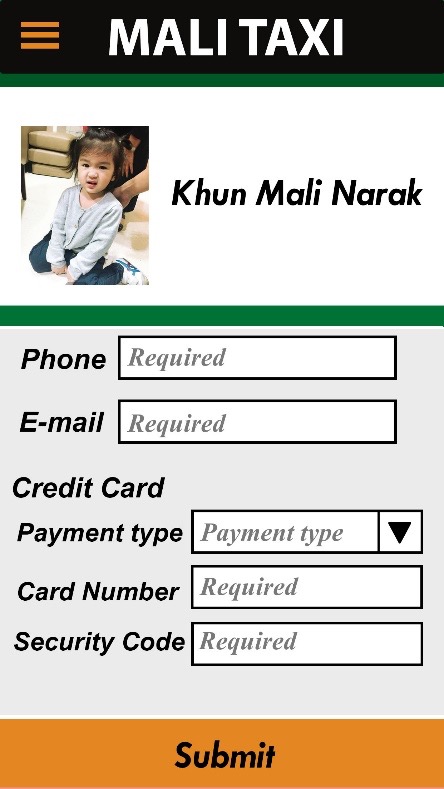
5.3.1.1 ส่วนของ Login ของผู้ใช้บริการ (Passenger)



รูป 5.3.1.1 Interface ของ Mali Taxi หน้า Login

หน้าล็อกอิน จะมีปุ่ม “Login via Facebook” เมื่อกดจะเป็นการล็อกอินผ่าน Facebook เพื่อให้ผู้ใช้งานเชื่อมต่อกับระบบผ่าน Facebook

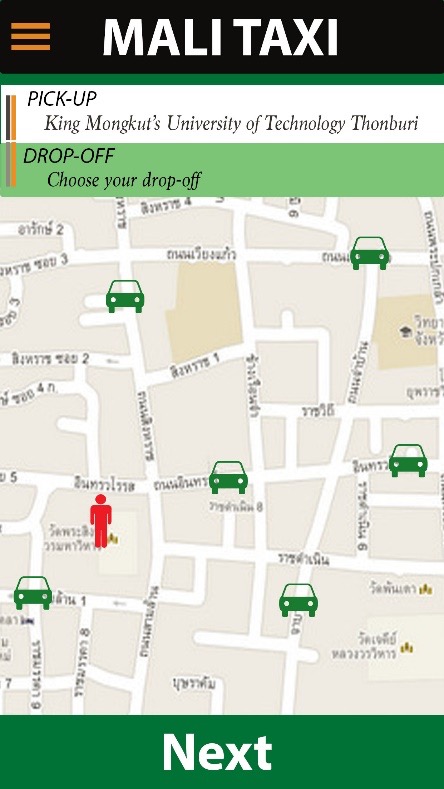
5.3.1.2 ส่วนของ Register ของผู้ใช้งาน (Passenger)



รูป 5.3.1.2 Interface ของ Mali Taxi หน้า Register

หน้าสมัครสมาชิก จะมีรูปโปรไฟล์ ซึ่งมาจาก Facebook รวมถึงชื่อของผู้ใช้งานด้วย ส่วนข้อมูลที่ต้องกรอกเพิ่มได้แก่ เบอร์โทรศัพท์ , อีเมล์ ,ข้อมูลส่วนของบัตรเครดิตที่ใช้สำหรับ ชำระค่าบริการแท็กซี่ โดยจำเป็นต้องเลือกประเภทของบัตร เช่น Visa, MasterCard, Amex เป็นต้น ซึ่งจะต้องกรอกเลขบัตรเครดิต และ Security Code ของบัตรเครดิตด้วย และกดส่งข้อมูล (Submit) ซึ่งหน้าของการจะสมัครจะขึ้นเพียงครั้งเดียว คือครั้งแรกของการใช้แอปพลิเคชั่น ครั้งต่อ ๆ ไป จะข้ามไปยังหน้าเรียกใช้แท็กซี่เลย

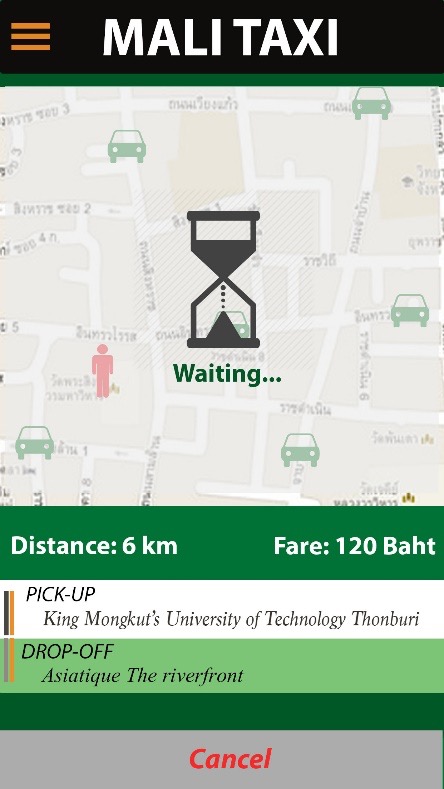
5.3.1.3 ส่วนของ การเรียกใช้งานแท็กซี่ ของผู้ใช้งาน (Passenger)



รูป 5.3.1.3 Interface ของ Mali Taxi หน้าการเรียกใช้งานแท็กซี่

ส่วนของหน้าการเรียกใช้บริการแท็กซี่ จะเป็นส่วนของผู้ใช้งาน(ผู้โดยสาร) โดยช่อง Pick-up หมายถึง จุดที่ผู้ใช้บริการต้องการให้ไปแท็กซี่ (Driver) ไปรับ และช่อง Drop-off หมายถึงจุดหมายปลายทางที่ผู้ใช้บริการต้องการจะไป รูปไอคอนรูปคน(สีแดง) บอกถึงตำแหน่งปัจจุบันของผู้ใช้งาน และรูปรถแท๊กซี่ (สีเขียว) เพื่อบอกผู้ใช้งานว่า ในบริเวณโดยรอบของผู้โดยสาร มีแท็กซี่ให้บริการ กี่คัน เมื่อทำการเลือกจุดหมายปลายทางแล้ว ให้ทำการกด Next ซึ่งจะเป็นการเรียกใช้บริการแท็กซี่

5.3.1.4 ส่วนของ การรอผู้ให้บริการมารับ ของผู้ใช้งาน (Passenger)



รูป 5.3.1.4 Interface ของ Mali Taxi หน้าการรอผู้ให้บริการมารับ

หน้ารอผู้ให้บริการ(แท็กซี่)มารับ เป็นส่วนของผู้ใช้งาน(ผู้โดยสาร) ในส่วนบนจะเป็น แผนที่ ซึ่ง ณ ขณะนั้น สถานะจะเป็น Waiting Taxi จะเป็นการบอกถึงว่ารอแท็กซี่คันใดจะมาเป็นผู้ให้บริการแก่ผู้ใช้ มีการคำนวณระยะทาง และราคาค่าโดยสาร รวมถึงแสดง จุดเริ่มต้น (Pick-Up) ของผู้โดยสาร, จุดหมายปลายทาง (Drop-Off) และ จะมีปุ่ม Cancel เพื่อยกเลิกการใช้บริการ หรือ ผู้ใช้ต้องการเปลี่ยนจุดหมายปลายทาง เมื่อกดจะมี Warning ขึ้นมาเตือน

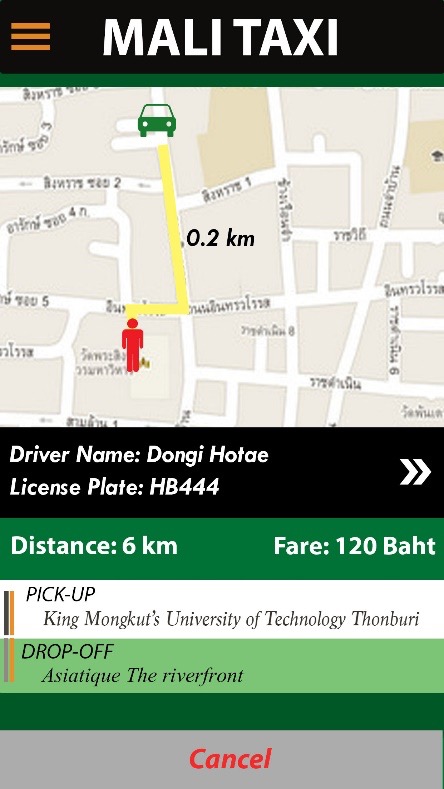
5.3.1.5 ส่วนของ Warning เมื่อทำการยกเลิกใช้บริการ ของผู้ใช้งาน (Passenger)



รูป 5.3.1.5 Interface ของ Mali Taxi หน้า Warning เมื่อทำการยกเลิกใช้บริการ

ส่วนหน้า Warning หลังจากผู้ใช้บริการ กด Cancel (ยกเลิกการใช้บริการ) เป็นการป้องกันการผิดพลาด ซึ่งผู้ใช้บริการอาจจะไม่ได้ตั้งใจไปกด โดยจะมีการเตือนว่า “Do you want to cancel?” จะมีปุ่มให้เลือก Yes หมายถึงต้องการยกเลิกการใช้บริการนี้ และ No หมายถึงไม่ต้องการยกเลิกใช้บริการนี้

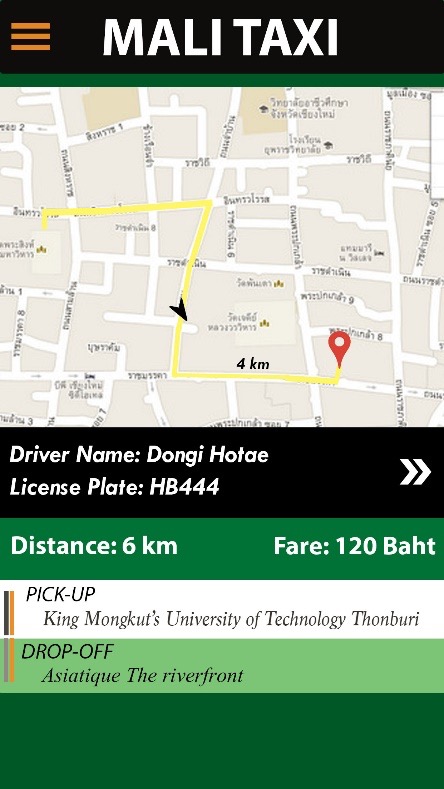
5.3.1.6 ส่วนของ การแสดงรายละเอียดผู้ให้บริการที่มารับ ของผู้ใช้งาน (Passenger)



รูป 5.3.1.6 Interface ของ Mali Taxi หน้า การแสดงรายละเอียดผู้ให้บริการที่มารับ

หน้าแสดงรายละเอียดผู้ให้บริการ(แท็กซี่) ที่จะมารับ คือส่วนของผู้โดยสาร ซึ่งจะทราบว่า แท็กซี่คันใดเป็นผู้ให้บริการ และมีรายละเอียดของคนขับ เช่น ชื่อผู้ให้บริการ และทะเบียนรถแท๊กซี่ โดยจะมี ไอคอน >> ซึ่งมีความหมายว่า ผู้ใช้บริการสามารถกดดูข้อมูลคนขับเพิ่มเติมได้ และในส่วนของแผนที่ก็จะมีลูกศร (สีเหลือง) ซึ่งแสดงเส้นทางระหว่างแท็กซี่ที่ผู้ให้บริการและผู้โดยสาร เป็นแบบ Real Time โดยบอกระยะทางว่าไกลเท่าไร ซึ่งระหว่างนั้น ผู้โดยสารสามารถกดยกเลิกใช้บริการนี้ได้

5.3.1.7 ส่วนของ การแสดงเส้นทางตั้งแต่จุดเริ่ม ถึง จุดหมายปลายทาง ของผู้ใช้งาน (Passenger)



รูป 5.3.1.7 Interface ของ Mali Taxi หน้า การแสดงเส้นทางตั้งแต่จุดเริ่ม ถึง จุดหมายปลายทาง ของผู้ใช้งาน (Passenger)

หน้าแสดงเส้นทางตั้งแต่จุดเริ่มต้น ถึง จุดหมายปลายทางของผู้โดยสาร (ผู้ใช้งาน) คือส่วนที่ผู้ใช้งาน ทราบเส้นทางและ ระยะทางระหว่างรถแท็กซี่กับจุดหมายปลายทาง แบบ Real time โดยไอคอน  แสดงตำแหน่งปัจจุบันของผู้ใช้งาน(ไอคอนสีฟ้า) และ จุดหมายปลายทาง  (ไอคอนสีแดง)

source code

package th.in.spksoft.taxi;

import android.Manifest;

import android.content.Intent;

import android.content.pm.PackageManager;

import android.location.Location;

import android.location.LocationManager;

import android.os.AsyncTask;

import android.os.Build;

import android.os.Bundle;

import android.support.annotation.NonNull;

import android.support.annotation.Nullable;

import android.support.v4.app.ActivityCompat;

import android.support.v4.app.FragmentActivity;

import android.support.v4.content.ContextCompat;

import android.util.Log;

import android.view.View;

import android.widget.TextView;

import android.widget.Toast;

import com.google.android.gms.common.ConnectionResult;

import com.google.android.gms.common.GooglePlayServicesNotAvailableException;

import com.google.android.gms.common.GooglePlayServicesRepairableException;

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.location.places.Place;

import com.google.android.gms.location.places.ui.PlacePicker;

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 org.json.JSONArray;

import org.json.JSONException;

import org.json.JSONObject;

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.net.HttpURLConnection;

import java.net.URL;

public class MapsActivity extends FragmentActivity implements OnMapReadyCallback, LocationListener, GoogleApiClient.ConnectionCallbacks, GoogleApiClient.OnConnectionFailedListener {

private GoogleMap mMap;//รับแผนที่ของ google

private Location last\_location;//ระบุตำแหน่งล่าสุด

private LocationManager locationManager;

private GoogleApiClient mGoogleApiClient;//เรียกใช้ google map api

private Marker mkSource;//save maker on map

private Marker mkDestination;//save maker on map

int PLACE\_PICKER\_REQUEST\_SOURCE = 1;

int PLACE\_PICKER\_REQUEST\_DESTINATION = 2;

boolean first = false;//to check รอบแรก

//First activity show activity\_maps.xml file.

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_maps);

// Obtain the SupportMapFragment and get notified when the map is ready to be used.

SupportMapFragment mapFragment = (SupportMapFragment) getSupportFragmentManager()

.findFragmentById(R.id.map);

mapFragment.getMapAsync(this);

if (mGoogleApiClient == null) {

mGoogleApiClient = new GoogleApiClient.Builder(this)

.addConnectionCallbacks(this)

.addOnConnectionFailedListener(this)

.addApi(LocationServices.API)

.build();

}

}

//To check permission of phone and open location

@Override

public void onMapReady(GoogleMap googleMap) {

mMap = googleMap;

if (Build.VERSION.SDK\_INT >= 23) {

if (ContextCompat.checkSelfPermission(this, Manifest.permission.ACCESS\_FINE\_LOCATION) != PackageManager.PERMISSION\_GRANTED) {

ActivityCompat.requestPermissions(this, new String[]{android.Manifest.permission.ACCESS\_FINE\_LOCATION}, 5555);

}

}

mMap.setMyLocationEnabled(true);

}

//Start Activity

protected void onStart() {

mGoogleApiClient.connect();

super.onStart();

}

//function when button was click and set latitude and longitude of source or destination.

public void onBtnClicked(View v) throws GooglePlayServicesNotAvailableException, GooglePlayServicesRepairableException {

if(v.getId() == R.id.btnRefresh) {

new backLoadJson().execute();

} else if(v.getId() == R.id.btnSource) {

PlacePicker.IntentBuilder builder = new PlacePicker.IntentBuilder();

startActivityForResult(builder.build(this), PLACE\_PICKER\_REQUEST\_SOURCE);

} else if(v.getId() == R.id.btnDest) {

PlacePicker.IntentBuilder builder = new PlacePicker.IntentBuilder();

startActivityForResult(builder.build(this), PLACE\_PICKER\_REQUEST\_DESTINATION);

}

}

//show icon and name of source and destination and price

protected void onActivityResult(int requestCode, int resultCode, Intent data) {

if (requestCode == PLACE\_PICKER\_REQUEST\_SOURCE) {

if (resultCode == RESULT\_OK) {

if (mkSource != null) mkSource.remove();

Place place = PlacePicker.getPlace(data, this);

Toast.makeText(getApplicationContext(), place.getName(), Toast.LENGTH\_LONG).show();

mkSource = mMap.addMarker(new MarkerOptions().position(place.getLatLng()).title("Source : " + place.getName().toString()).icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE\_AZURE)));

if(mkDestination != null) {

TextView tmp = (TextView) findViewById(R.id.txtPrice);

tmp.setText("Price : " + String.valueOf(calculatePriceFromMeter(getMeterFromLatLng(place.getLatLng(), mkDestination.getPosition()))));

}

}

} else if(requestCode == PLACE\_PICKER\_REQUEST\_DESTINATION) {

if (resultCode == RESULT\_OK) {

if (mkDestination != null) mkDestination.remove();

Place place = PlacePicker.getPlace(data, this);

Toast.makeText(getApplicationContext(), place.getName(), Toast.LENGTH\_LONG).show();

mkDestination = mMap.addMarker(new MarkerOptions().position(place.getLatLng()).title("Destination : " + place.getName().toString()).icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE\_AZURE)));

if(mkSource != null) {

TextView tmp = (TextView) findViewById(R.id.txtPrice);

tmp.setText("Price : " + String.valueOf(calculatePriceFromMeter(getMeterFromLatLng(mkSource.getPosition(), place.getLatLng()))));

}

}

}

}

//find distance between source and destination

public static double getMeterFromLatLng(LatLng source, LatLng destination) {

double lat1 = source.latitude;

double lng1 = source.longitude;

double lat2 = destination.latitude;

double lng2 = destination.longitude;

double earthRadius = 6371000; //meters

double dLat = Math.toRadians(lat2-lat1);

double dLng = Math.toRadians(lng2-lng1);

double a = Math.sin(dLat/2) \* Math.sin(dLat/2) +

Math.cos(Math.toRadians(lat1)) \* Math.cos(Math.toRadians(lat2)) \*

Math.sin(dLng/2) \* Math.sin(dLng/2);

double c = 2 \* Math.atan2(Math.sqrt(a), Math.sqrt(1-a));

float dist = (float) (earthRadius \* c);

return dist;

}

//function calculate price from distance

public static double calculatePriceFromMeter(double meter) {

if(meter <= 1000) return 35;

else if(meter>1000 && meter<=11000){

double temp;

temp = meter-1000;

temp = (int)temp/1000;

return 35+(temp\*(5.5));

}

else if(meter>11000 && meter<=21000){

double temp;

temp = meter-11000;

temp = (int)temp/1000;

return 95.5+(temp\*(6));

}

else if(meter>21000){

double temp;

temp = meter-21000;

temp = (int)temp/1000;

return 155.5+(temp\*(6.5));

}

else return 0;

}

//end application

protected void onStop() {

mGoogleApiClient.disconnect();

super.onStop();

}

private LocationRequest createLocationRequest() {

LocationRequest mLocationRequest = new LocationRequest();

mLocationRequest.setInterval(10000);

mLocationRequest.setFastestInterval(5000);

mLocationRequest.setPriority(LocationRequest.PRIORITY\_HIGH\_ACCURACY);

return mLocationRequest;

}

//set change location.

@Override

public void onLocationChanged(Location location) {

last\_location = location;

}

//update icon of latitude and longitude of taxi.

@Override

public void onConnected(@Nullable Bundle bundle) {

Toast.makeText(MapsActivity.this, "WTF", Toast.LENGTH\_SHORT).show();

if (Build.VERSION.SDK\_INT >= 23) {

if (ContextCompat.checkSelfPermission(this, Manifest.permission.ACCESS\_FINE\_LOCATION) != PackageManager.PERMISSION\_GRANTED) {

ActivityCompat.requestPermissions(this, new String[]{android.Manifest.permission.ACCESS\_FINE\_LOCATION}, 5555);

}

}

LocationRequest mLocationRequest = createLocationRequest();

LocationServices.FusedLocationApi.requestLocationUpdates(mGoogleApiClient, mLocationRequest, this);

last\_location = LocationServices.FusedLocationApi.getLastLocation(mGoogleApiClient);

mMap.moveCamera(CameraUpdateFactory.newLatLng(new LatLng(last\_location.getLatitude(), last\_location.getLongitude())));

mMap.animateCamera( CameraUpdateFactory.zoomTo( 12.0f ) );

if(first == false) {

first = true;

new backLoadJson().execute();

}

}

@Override

public void onConnectionSuspended(int i) {

}

@Override

public void onConnectionFailed(@NonNull ConnectionResult connectionResult) {

}

//get latitude longitude name of taxi from database.

private class backLoadJson extends AsyncTask<Void, Void, String> {

//private ProgressDialog pdia;

protected void onPreExecute() {

}

protected String doInBackground(Void... params) {

try {

String responseText = GetResponseText("192.168.202.10=" + String.valueOf(last\_location.getLatitude()) + "&lng=" + String.valueOf(last\_location.getLongitude()));

return responseText;

} catch (IOException e) {

return e.getMessage();

}

}

protected void onProgressUpdate(Integer... values) {

}

protected void onPostExecute(String result) {

Log.d("aaaaaaaaaaaaaa", result);

mMap.clear();

try {

JSONObject jsonobject = new JSONObject(result);

JSONArray list = jsonobject.getJSONArray("data");

for(int i = 0;i < list.length();i++){

JSONObject t = list.getJSONObject(i);

double lat = t.getDouble("lat");

double lng = t.getDouble("lng");

String name = t.getString("name");

LatLng pin = new LatLng(lat, lng);

mMap.addMarker(new MarkerOptions().position(pin).title(name));

}

} catch (JSONException e) {

e.printStackTrace();

Log.d("bbbbbbbbbbbb", e.getMessage());

}

}

private String GetResponseText(String stringUrl) throws IOException {

StringBuilder response = new StringBuilder();

URL url = new URL(stringUrl);

HttpURLConnection httpconn = (HttpURLConnection) url.openConnection();

if (httpconn.getResponseCode() == HttpURLConnection.HTTP\_OK) {

BufferedReader input = new BufferedReader(new InputStreamReader(httpconn.getInputStream()), 8192);

String strLine = null;

while ((strLine = input.readLine()) != null) {

response.append(strLine);

}

input.close();

} else {

return null;

}

return response.toString();

}

}

}

test case

package th.in.spksoft.taxi;

import com.google.android.gms.maps.model.LatLng;

import org.junit.Test;

import static org.junit.Assert.\*;

//Test musure distance from latitude and longitude source to latitude and longitude destiantion

public class ExampleUnitTest {

@Test

public void test\_meter\_distance\_from\_two\_latlng() throws Exception {

double distance = MapsActivity.getMeterFromLatLng(new LatLng(13.651679, 100.4869247), new LatLng(13.6472733, 100.4943752));

System.out.println(distance);

assertEquals((int)distance, 942);

}

//Test calculate price from distance

@Test

public void test\_price\_from\_meter() throws Exception {

double price = MapsActivity.calculatePriceFromMeter(942);

System.out.println(price);

assertEquals((int)price, 35);

}

}

instruction to run code

เปิดDeveloper tool ในโทรศัพท์มือถือ จากนั้นเปิด ใช้งาน USB Debugging

เปิด andriod studio

เปิดโปรเจค taxi

นำมือถือ android มาต่อกับคอมพิวเตอร์

รันโปรแกรม

Github Link : https://github.com/meawhung1/Taxi