Layout

Main========

*<?***xml version="1.0" encoding="utf-8"***?>*<**LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
xmlns:app="http://schemas.android.com/apk/res-auto"  
xmlns:tools="http://schemas.android.com/tools"  
android:layout\_width="match\_parent"  
android:layout\_height="match\_parent"  
android:orientation="vertical"  
tools:context="com.example.davidhaniel.uber.MainActivity"**>  
  
<**Button  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="@string/provider"  
 android:id="@+id/provider"**/>  
  
<**Button  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="@string/consumer"  
 android:id="@+id/consumer"**/>  
</**LinearLayout**>

Consumer Login================

*<?***xml version="1.0" encoding="utf-8"***?>*<**LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 tools:context="com.example.davidhaniel.uber.ProviderLoginActivity"  
 android:orientation="vertical"**>  
  
 <**EditText  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="email"  
 android:id="@+id/email"**/>  
  
 <**EditText  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="password"  
 android:id="@+id/password"**/>  
  
 <**Button  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Login"  
 android:id="@+id/login"**/>  
  
 <**Button  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Registration"  
 android:id="@+id/registration"**/>  
  
</**LinearLayout**>

Consumer Map=================

<**FrameLayout  
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:layout\_width="match\_parent"  
android:layout\_height="match\_parent"  
tools:context="com.example.davidhaniel.uber.ConsumerMapActivity"** >  
  
<**Button  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:id="@+id/logout"  
 android:text="@string/logout"**/>  
<**fragment  
 android:id="@+id/map"  
 android:name="com.google.android.gms.maps.SupportMapFragment"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"**/>  
<**Button  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:id="@+id/request"  
 android:text="@string/callProvider"  
 android:layout\_gravity="bottom"**/>  
  
  
</**FrameLayout**>

Provider Login=========================

*<?***xml version="1.0" encoding="utf-8"***?>*<**LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 tools:context="com.example.davidhaniel.uber.ProviderLoginActivity"  
 android:orientation="vertical"**>  
  
<**EditText  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="email"  
 android:id="@+id/email"**/>  
  
 <**EditText  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="password"  
 android:id="@+id/password"**/>  
  
 <**Button  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Login"  
 android:id="@+id/login"**/>  
  
 <**Button  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Registration"  
 android:id="@+id/registration"**/>  
  
</**LinearLayout**>

Provider Map=======================

<**FrameLayout  
 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:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 tools:context="com.example.davidhaniel.uber.ProviderMapsActivity"** >  
  
 <**Button  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:id="@+id/logout"  
 android:text="@string/logout"**/>  
 <**fragment  
 android:id="@+id/map"  
 android:name="com.google.android.gms.maps.SupportMapFragment"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"**/>  
  
</**FrameLayout**>

+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

Java=======================

Main============================

**package** com.example.davidhaniel.uber;  
  
**import** android.content.Intent;  
**import** android.support.v7.app.AppCompatActivity;  
**import** android.os.Bundle;  
**import** android.view.View;  
**import** android.widget.Button;  
  
**public class** MainActivity **extends** AppCompatActivity {  
**private** Button **nDriver**, **nCustomer**;  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_main***);  
  
 **nDriver** = (Button) findViewById(R.id.***provider***);  
 **nCustomer** = (Button) findViewById(R.id.***consumer***);  
  
 **nDriver**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View view) {  
 Intent intent = **new** Intent(MainActivity.**this**, ProviderLoginActivity.**class**);  
 startActivity(intent);  
 finish();  
 **return**;  
 }  
 });  
  
 **nCustomer**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View view) {  
 Intent intent = **new** Intent(MainActivity.**this**, ConsumerLoginActivity.**class**);  
 startActivity(intent);  
 finish();  
 **return**;  
 }  
 });  
 }  
}

ConsumerLogin====================================

**package** com.example.davidhaniel.uber;  
  
**import** android.content.Intent;  
**import** android.support.annotation.NonNull;  
**import** android.support.v7.app.AppCompatActivity;  
**import** android.os.Bundle;  
**import** android.view.View;  
**import** android.widget.Button;  
**import** android.widget.EditText;  
**import** android.widget.Toast;  
  
**import** com.google.android.gms.tasks.OnCompleteListener;  
**import** com.google.android.gms.tasks.Task;  
**import** com.google.firebase.auth.AuthResult;  
**import** com.google.firebase.auth.FirebaseAuth;  
**import** com.google.firebase.auth.FirebaseUser;  
**import** com.google.firebase.database.DatabaseReference;  
**import** com.google.firebase.database.FirebaseDatabase;  
  
**public class** ConsumerLoginActivity **extends** AppCompatActivity {  
  
 **private** EditText **nEmail**, **nPassword**;  
 **private** Button **nLogin**, **nRegistration**;  
 **private** FirebaseAuth **nAuth**;  
 **private** FirebaseAuth.AuthStateListener **firebaseAuthListener**;  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_provider\_login***);  
*//this will be called if login will fail* **nAuth** = FirebaseAuth.*getInstance*();  
 **firebaseAuthListener** = **new** FirebaseAuth.AuthStateListener() {  
 @Override  
 **public void** onAuthStateChanged(@NonNull FirebaseAuth firebaseAuth) {  
 FirebaseUser user = FirebaseAuth.*getInstance*().getCurrentUser();  
 **if** (user!=**null**){  
 Intent intent = **new** Intent(ConsumerLoginActivity.**this**, ConsumerMapActivity.**class**);  
 startActivity(intent);  
 finish();  
 **return**;  
 }  
 }  
 };  
  
 **nEmail**=(EditText) findViewById(R.id.***email***);  
 **nPassword**=(EditText) findViewById(R.id.***password***);  
  
 **nLogin** = (Button) findViewById(R.id.***login***);  
 **nRegistration** = (Button)findViewById(R.id.***registration***);  
  
 **nRegistration**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View view) {  
 **final** String email = **nEmail**.getText().toString();  
 **final** String password = **nPassword**.getText().toString();  
 *//creating or writing email and password* **nAuth**.createUserWithEmailAndPassword(email,password).addOnCompleteListener(ConsumerLoginActivity.**this**, **new** OnCompleteListener<AuthResult>() {  
 @Override  
 **public void** onComplete(@NonNull Task<AuthResult> task) {  
 **if**(!task.isSuccessful()){  
 Toast.*makeText*(ConsumerLoginActivity.**this**, **"signup error"**, Toast.***LENGTH\_SHORT***).show();  
 }**else**{  
 *//getting user id* String user\_id = **nAuth**.getCurrentUser().getUid();  
 *//specific location in the database where to write the data;* DatabaseReference current\_user\_db = FirebaseDatabase.*getInstance*().getReference().child(**"Users"**).child(**"Customers"**).child(user\_id);  
 current\_user\_db.setValue(**true**);  
 }  
  
 }  
 });  
 }  
 });  
  
 **nLogin**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View view) {  
 **final** String email = **nEmail**.getText().toString();  
 **final** String password = **nPassword**.getText().toString();  
 **nAuth**.signInWithEmailAndPassword(email,password).addOnCompleteListener(ConsumerLoginActivity.**this**, **new** OnCompleteListener<AuthResult>() {  
 @Override  
 **public void** onComplete(@NonNull Task<AuthResult> task) {  
 **if**(!task.isSuccessful()){  
 Toast.*makeText*(ConsumerLoginActivity.**this**, **"sign in error"**, Toast.***LENGTH\_SHORT***).show();  
 }  
 }  
 });  
 }  
 });  
 }  
  
 @Override  
 **protected void** onStart() {  
 **super**.onStart();  
 **nAuth**.addAuthStateListener(**firebaseAuthListener**);  
 }  
  
 @Override  
 **protected void** onStop() {  
 **super**.onStop();  
 **nAuth**.removeAuthStateListener(**firebaseAuthListener**);  
 }  
}

ConsumerMap========================================

**package** com.example.davidhaniel.uber;  
  
**import** android.content.Intent;  
**import** android.content.pm.PackageManager;  
**import** android.location.Location;  
**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.v7.app.AppCompatActivity;  
**import** android.os.Bundle;  
**import** android.view.View;  
**import** android.widget.Button;  
**import** android.widget.Toast;  
  
**import** com.firebase.geofire.GeoFire;  
**import** com.firebase.geofire.GeoLocation;  
**import** com.firebase.geofire.GeoQuery;  
**import** com.firebase.geofire.GeoQueryEventListener;  
**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.firebase.auth.FirebaseAuth;  
**import** com.google.firebase.database.DataSnapshot;  
**import** com.google.firebase.database.DatabaseError;  
**import** com.google.firebase.database.DatabaseReference;  
**import** com.google.firebase.database.FirebaseDatabase;  
  
  
**import** com.google.android.gms.common.ConnectionResult;  
**import** com.google.android.gms.common.api.GoogleApiClient;  
**import** com.google.android.gms.location.FusedLocationProviderClient;  
**import** com.google.android.gms.maps.CameraUpdateFactory;  
**import** com.google.android.gms.maps.model.MarkerOptions;  
**import** com.google.android.gms.tasks.OnSuccessListener;  
**import** com.google.firebase.database.ValueEventListener;  
  
**import** java.util.HashMap;  
**import** java.util.List;  
  
  
**public class** ConsumerMapActivity **extends** FragmentActivity **implements** OnMapReadyCallback, GoogleApiClient.ConnectionCallbacks, GoogleApiClient.OnConnectionFailedListener, com.google.android.gms.location.LocationListener {  
  
 **private** GoogleMap **nMap**;  
 GoogleApiClient **nGoogleApiClient**;  
 Location **nLastLocation**;  
 LocationRequest **nLocationRequest**;  
 *// private SupportMapFragment mapFragment;  
 // FusedLocationProviderClient mFusedLocationClient;* **private** Button **nLogout**, **nRequest**;  
 **private** LatLng **pickupLocation**;  
  
 SupportMapFragment **mapFragment** = (SupportMapFragment) getSupportFragmentManager().findFragmentById(R.id.***map***);  
  
 **private** Boolean **requestBol** = **false**; *// #12 Canceling-an-Uber-Request @@@@@@@* **private** Marker **pickupMarker**;  
  
  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_consumer\_map***);  
 *// Obtain the SupportMapFragment and get notified when the map is ready to be used.* SupportMapFragment mapFragment = (SupportMapFragment) getSupportFragmentManager()  
 .findFragmentById(R.id.***map***);  
  
 **if** (ActivityCompat.*checkSelfPermission*(**this**, android.Manifest.permission.***ACCESS\_FINE\_LOCATION***) != PackageManager.***PERMISSION\_GRANTED*** && ActivityCompat.*checkSelfPermission*(**this**, android.Manifest.permission.***ACCESS\_COARSE\_LOCATION***) != PackageManager.***PERMISSION\_GRANTED***) {  
 ActivityCompat.*requestPermissions*(ConsumerMapActivity.**this**, **new** String[]{android.Manifest.permission.***ACCESS\_FINE\_LOCATION***}, **LOCATION\_REQUEST\_CODE**);  
 *// return;* }**else** {  
 mapFragment.getMapAsync(**this**);  
 }  
  
 **nLogout** = (Button)findViewById(R.id.***logout***);  
 **nRequest** = (Button)findViewById(R.id.***request***);  
 **nLogout**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View view) {  
 *//automatic logout* FirebaseAuth.*getInstance*().signOut();  
 Intent intent = **new** Intent(ConsumerMapActivity.**this**, MainActivity.**class**);  
 startActivity(intent);  
 finish();  
 **return**;  
 }  
 });  
  
 **nRequest**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View view) {  
  
 *// #12 Canceling-an-Uber-Request @@@@@@@@@@@@@@@@@* **if** (**requestBol**){  
 **requestBol** = **false**;  
 **geoQuery**.removeAllListeners();  
 **driverLocationRef**.removeEventListener(**driverLocationRefListener**);  
  
 **if** (**providerFoundID** != **null**){  
 DatabaseReference providerRef = FirebaseDatabase.*getInstance*().getReference().child(**"Users"**).child(**"Drivers"**).child(**providerFoundID**); *//removing data or clearing data from the database* providerRef.setValue(**true**); *//removing data or clearing data from the database* **providerFoundID** = **null**; *//removing data or clearing data from the database* }  
  
 **providerFound** = **false**; *//initiating the value back again* **radius** = 1; *//initiating the value back again* String userId = FirebaseAuth.*getInstance*().getCurrentUser().getUid(); *//removing data from the database* DatabaseReference ref = FirebaseDatabase.*getInstance*().getReference(**"consumerRequest"**); *//removing data from the database* GeoFire geoFire = **new** GeoFire(ref); *//removing data from the database* geoFire.removeLocation(userId); *//removing data from the database* **if** (**pickupMarker** !=**null**){ *//removing pickup Marker* **pickupMarker**.remove(); *//removing pickup Marker* }  
  
 }**else** {  
 **requestBol** = **true**;  
 String userId = FirebaseAuth.*getInstance*().getCurrentUser().getUid();  
 DatabaseReference ref = FirebaseDatabase.*getInstance*().getReference(**"consumerRequest"**);  
  
 GeoFire geoFire = **new** GeoFire(ref);  
 geoFire.setLocation(userId, **new** GeoLocation(**nLastLocation**.getLatitude(), **nLastLocation**.getLongitude()));  
  
 **pickupLocation** = **new** LatLng(**nLastLocation**.getLatitude(), **nLastLocation**.getLongitude());  
 **pickupMarker** = **nMap**.addMarker(**new** MarkerOptions().position(**pickupLocation**).title(**"Consumer House"**).icon(BitmapDescriptorFactory.*fromResource*(R.mipmap.***ic\_house***))); *// #13 Change Marker  
 // pickupMarker = nMap.addMarker(new MarkerOptions().position(pickupLocation).title("Consumer House")); // #12 Canceling-an-Uber-Request @@@@@@@@ assigning to variable pickupMarker* **nRequest**.setText(**"Getting your Service Provider..."**);  
  
 getClosestProvider();  
  
 }  
  
  
 *// #12 Canceling-an-Uber-Request @@@@@@@@@@@@@@@@@@  
  
  
  
  
 /\* from #11 +++++++++++++++++++  
  
  
 String userId = FirebaseAuth.getInstance().getCurrentUser().getUid();  
 DatabaseReference ref = FirebaseDatabase.getInstance().getReference("consumerRequest");  
  
 GeoFire geoFire = new GeoFire(ref);  
 geoFire.setLocation(userId, new GeoLocation(nLastLocation.getLatitude(), nLastLocation.getLongitude()));  
  
 pickupLocation = new LatLng(nLastLocation.getLatitude(), nLastLocation.getLongitude());  
 nMap.addMarker(new MarkerOptions().position(pickupLocation).title("Consumer House"));  
  
 nRequest.setText("Getting your Service Provider...");  
  
 getClosestProvider(); //calling a function  
  
 from #11 +++++++++++++++++++ \*/* }  
 });  
  
 *// mFusedLocationClient = LocationServices.getFusedLocationProviderClient(this);* }  
  
 **private int radius** = 1;  
 **private** Boolean **providerFound** = **false**;  
 **private** String **providerFoundID**;  
  
 **private** GeoQuery **geoQuery**; *// #12 Canceling-an-Uber-Request @@@@@@@* **private void** getClosestProvider(){ *//creating a function* **final** DatabaseReference providerLocation = FirebaseDatabase.*getInstance*().getReference().child(**"providerAvailable"**);  
  
 GeoFire geoFire = **new** GeoFire(providerLocation);  
  
 **geoQuery** = geoFire.queryAtLocation(**new** GeoLocation(**pickupLocation**.**latitude**, **pickupLocation**.**longitude**), **radius**);  
 **geoQuery**.removeAllListeners();  
  
 **geoQuery**.addGeoQueryEventListener(**new** GeoQueryEventListener() {  
 @Override  
 **public void** onKeyEntered(String key, GeoLocation location) {  
 **if** (!**providerFound** && **requestBol**){  
 **providerFound** = **true**;  
 **providerFoundID** = key;  
  
 *//* DatabaseReference providerRef = FirebaseDatabase.*getInstance*().getReference().child(**"Users"**).child(**"Drivers"**).child(**providerFoundID**);  
 String customerId = FirebaseAuth.*getInstance*().getCurrentUser().getUid();  
 HashMap map = **new** HashMap();  
 map.put(**"customerRideId"**, customerId);  
 providerRef.updateChildren(map);  
  
 *//call function* getDriverLocation();  
 **nRequest**.setText(**"Looking for Provider Location"**);  
  
  
 }  
 }  
  
 @Override  
 **public void** onKeyExited(String key) {  
  
 }  
  
 @Override  
 **public void** onKeyMoved(String key, GeoLocation location) {  
  
 }  
  
 @Override  
 **public void** onGeoQueryReady() {  
 **if** (!**providerFound**){  
 **radius**++;  
 getClosestProvider();  
 }  
 }  
  
 @Override  
 **public void** onGeoQueryError(DatabaseError error) {  
  
 }  
 });  
  
 }  
  
 *//creating the function* **private** Marker **nDriverMarker**;  
 **private** DatabaseReference **driverLocationRef**; *// #12 Canceling-an-Uber-Request @@@@@@@* **private** ValueEventListener **driverLocationRefListener**; *// #12 Canceling-an-Uber-Request @@@@@@@ creating String Listener* **public void** getDriverLocation(){  
 **driverLocationRef** =FirebaseDatabase.*getInstance*().getReference().child(**"providerWorking"**).child(**providerFoundID**).child(**"l"**);  
  
 **driverLocationRefListener** = **driverLocationRef**.addValueEventListener(**new** ValueEventListener() {  
 @Override  
 **public void** onDataChange(DataSnapshot dataSnapshot) {  
 **if** (dataSnapshot.exists()&& **requestBol**){ *// #12 Canceling-an-Uber-Request @@@@@@@ making sure the booolean is true* List<Object> map = (List<Object>) dataSnapshot.getValue();  
 **double** locationLat = 0;  
 **double** locationLng = 0;  
 **nRequest**.setText(**"Driver Found"**);  
  
 **if**(map.get(0)!=**null**) {  
 locationLat = Double.*parseDouble*(map.get(0).toString());  
 }  
  
 **if** (map.get(1) !=**null**) {  
 locationLng = Double.*parseDouble*(map.get(1).toString());  
 }  
  
 LatLng driverLatLng = **new** LatLng(locationLat,locationLng);  
  
 **if** (**nDriverMarker** != **null**){  
 **nDriverMarker**.remove();  
 }  
  
 *//#10 distanceBetweenTwoPoints =============================  
 /\*  
 Location loc1 = new Location("");  
 loc1.setLatitude(pickupLocation.latitude);  
 loc1.setLongitude(pickupLocation.longitude);  
  
 Location loc2 = new Location("");  
 loc2.setLatitude(driverLatLng.latitude);  
 loc2.setLongitude(driverLatLng.longitude);  
  
 float distance = loc1.distanceTo(loc2);  
 nRequest.setText("Provider Found: "+ String.valueOf(distance) + " meter/s.");  
  
 //#10 distanceBetweenTwoPoints =============================  
  
  
 //#11 NoticeWhen Driver is Arrived =============================  
  
 Location loc1 = new Location("");  
 loc1.setLatitude(pickupLocation.latitude);  
 loc1.setLongitude(pickupLocation.longitude);  
  
 Location loc2 = new Location("");  
 loc2.setLatitude(driverLatLng.latitude);  
 loc2.setLongitude(driverLatLng.longitude);  
  
 float distance = loc1.distanceTo(loc2);  
  
 if (distance<100){  
 nRequest.setText("Provider is Here : "+ String.valueOf(distance) + " meter/s.");  
 }else {  
 nRequest.setText("Provider Found: "+ String.valueOf(distance) + " meter/s.");  
 }  
  
 //#11 NoticeWhen Driver is Arrived =============================  
 \*/* Location loc1 = **new** Location(**""**);  
 loc1.setLatitude(**pickupLocation**.**latitude**);  
 loc1.setLongitude(**pickupLocation**.**longitude**);  
  
 Location loc2 = **new** Location(**""**);  
 loc2.setLatitude(driverLatLng.**latitude**);  
 loc2.setLongitude(driverLatLng.**longitude**);  
  
 **float** distance = loc1.distanceTo(loc2);  
  
 **if** (distance<100){  
 **nRequest**.setText(**"Provider is Here : "**+ String.*valueOf*(distance) + **" meter/s."**);  
 }**else** {  
 **nRequest**.setText(**"Provider Found: "**+ String.*valueOf*(distance) + **" meter/s."**);  
 }  
  
  
  
 **nDriverMarker** = **nMap**.addMarker(**new** MarkerOptions().position(driverLatLng).title(**"Your Service Provider"**).icon(BitmapDescriptorFactory.*fromResource*(R.mipmap.***ic\_person***))); *// #13 Change Marker ++++++++++++  
 // nDriverMarker = nMap.addMarker(new MarkerOptions().position(driverLatLng).title("Your Service Provider"));* }  
  
  
 }  
  
 @Override  
 **public void** onCancelled(DatabaseError databaseError) {  
  
 }  
 });  
  
 }  
  
  
  
  
  
 @Override  
 **public void** onMapReady(GoogleMap googleMap) {  
 **nMap** = googleMap;  
  
*/\*  
 // Add a marker in Sydney and move the camera  
 LatLng manila = new LatLng(14.5872236, 120.9939596);  
 nMap.addMarker(new MarkerOptions().position(manila).title("Marker in Philippines"));  
 nMap.moveCamera(CameraUpdateFactory.newLatLng(manila));  
\*/* **if** (ActivityCompat.*checkSelfPermission*(**this**, android.Manifest.permission.***ACCESS\_FINE\_LOCATION***) != PackageManager.***PERMISSION\_GRANTED*** && ActivityCompat.*checkSelfPermission*(**this**, android.Manifest.permission.***ACCESS\_COARSE\_LOCATION***) != PackageManager.***PERMISSION\_GRANTED***) {  
 ActivityCompat.*requestPermissions*(ConsumerMapActivity.**this**, **new** String[]{android.Manifest.permission.***ACCESS\_FINE\_LOCATION***}, **LOCATION\_REQUEST\_CODE**);  
  
 *// return;* }  
 buildGoogleApiClient();  
 **nMap**.setMyLocationEnabled(**true**);  
  
  
 }  
  
 **protected synchronized void** buildGoogleApiClient() {  
 **nGoogleApiClient** = **new** GoogleApiClient.Builder(**this**)  
 .addConnectionCallbacks(**this**)  
 .addOnConnectionFailedListener(**this**)  
 .addApi(LocationServices.***API***)  
 .build();  
 **nGoogleApiClient**.connect();  
  
  
 }  
  
  
 @Override  
 **public void** onLocationChanged(Location location) {  
 **nLastLocation** = location;  
 LatLng latlng = **new** LatLng(location.getLatitude(), location.getLongitude());  
 **nMap**.moveCamera(CameraUpdateFactory.*newLatLng*(latlng));  
 **nMap**.moveCamera(CameraUpdateFactory.*zoomTo*(16));  
  
 *// String userId = FirebaseAuth.getInstance().getCurrentUser().getUid();  
 // DatabaseReference ref = FirebaseDatabase.getInstance().getReference("providerAvailable");  
  
 // GeoFire geoFire = new GeoFire(ref);  
 // geoFire.setLocation(userId, new GeoLocation(location.getLatitude(), location.getLongitude()));* }  
  
 */\*  
 @Override  
 public void onStatusChanged(String s, int i, Bundle bundle) {  
  
 }  
  
 @Override  
 public void onProviderEnabled(String s) {  
  
 }  
  
 @Override  
 public void onProviderDisabled(String s) {  
  
 }  
 \*/* @Override  
 **public void** onConnected(@Nullable Bundle bundle) {  
 **nLocationRequest** = **new** LocationRequest();  
 **nLocationRequest**.setInterval(1000);  
 **nLocationRequest**.setFastestInterval(1000);  
  
 **if** (ActivityCompat.*checkSelfPermission*(**this**, android.Manifest.permission.***ACCESS\_FINE\_LOCATION***) != PackageManager.***PERMISSION\_GRANTED*** && ActivityCompat.*checkSelfPermission*(**this**, android.Manifest.permission.***ACCESS\_COARSE\_LOCATION***) != PackageManager.***PERMISSION\_GRANTED***) {  
 ActivityCompat.*requestPermissions*(ConsumerMapActivity.**this**, **new** String[]{android.Manifest.permission.***ACCESS\_FINE\_LOCATION***}, **LOCATION\_REQUEST\_CODE**);  
 *// return;* }  
 LocationServices.***FusedLocationApi***.requestLocationUpdates(**nGoogleApiClient**, **nLocationRequest**, **this**);  
  
 }  
  
 @Override  
 **public void** onConnectionSuspended(**int** i) {  
  
 }  
  
 @Override  
 **public void** onConnectionFailed(@NonNull ConnectionResult connectionResult) {  
  
 }  
  
 **final int LOCATION\_REQUEST\_CODE** = 1;  
 @Override  
 **public void** onRequestPermissionsResult(**int** requestCode, @NonNull String[] permissions, @NonNull **int**[] grantResults) {  
 **super**.onRequestPermissionsResult(requestCode, permissions, grantResults);  
 **switch**(requestCode){  
 **case LOCATION\_REQUEST\_CODE**:{  
 **if** (grantResults.**length** >0 && grantResults[0] == PackageManager.***PERMISSION\_GRANTED***){  
 **mapFragment**.getMapAsync(**this**);  
 }**else**{  
 Toast.*makeText*(getApplicationContext(), **"Please provide permission"**, Toast.***LENGTH\_SHORT***).show();  
 }  
 **break**;  
 }  
 }  
  
  
 }  
  
 @Override  
 **protected void** onStop() {  
 **super**.onStop();  
 *// LocationServices.FusedLocationApi.removeLocationUpdates(nGoogleApiClient, this);  
  
 // String userId = FirebaseAuth.getInstance().getCurrentUser().getUid();  
 // DatabaseReference ref = FirebaseDatabase.getInstance().getReference("providerAvailable");  
  
 // GeoFire geoFire = new GeoFire(ref);  
 // geoFire.removeLocation(userId);* }  
  
  
}

ProviderLogin==============================

**package** com.example.davidhaniel.uber;  
  
**import** android.content.Intent;  
**import** android.content.pm.PackageManager;  
**import** android.location.Location;  
**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.v7.app.AppCompatActivity;  
**import** android.os.Bundle;  
**import** android.view.View;  
**import** android.widget.Button;  
**import** android.widget.Toast;  
  
**import** com.firebase.geofire.GeoFire;  
**import** com.firebase.geofire.GeoLocation;  
**import** com.firebase.geofire.GeoQuery;  
**import** com.firebase.geofire.GeoQueryEventListener;  
**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.firebase.auth.FirebaseAuth;  
**import** com.google.firebase.database.DataSnapshot;  
**import** com.google.firebase.database.DatabaseError;  
**import** com.google.firebase.database.DatabaseReference;  
**import** com.google.firebase.database.FirebaseDatabase;  
  
  
**import** com.google.android.gms.common.ConnectionResult;  
**import** com.google.android.gms.common.api.GoogleApiClient;  
**import** com.google.android.gms.location.FusedLocationProviderClient;  
**import** com.google.android.gms.maps.CameraUpdateFactory;  
**import** com.google.android.gms.maps.model.MarkerOptions;  
**import** com.google.android.gms.tasks.OnSuccessListener;  
**import** com.google.firebase.database.ValueEventListener;  
  
**import** java.util.HashMap;  
**import** java.util.List;  
  
  
**public class** ConsumerMapActivity **extends** FragmentActivity **implements** OnMapReadyCallback, GoogleApiClient.ConnectionCallbacks, GoogleApiClient.OnConnectionFailedListener, com.google.android.gms.location.LocationListener {  
  
 **private** GoogleMap **nMap**;  
 GoogleApiClient **nGoogleApiClient**;  
 Location **nLastLocation**;  
 LocationRequest **nLocationRequest**;  
 *// private SupportMapFragment mapFragment;  
 // FusedLocationProviderClient mFusedLocationClient;* **private** Button **nLogout**, **nRequest**;  
 **private** LatLng **pickupLocation**;  
  
 SupportMapFragment **mapFragment** = (SupportMapFragment) getSupportFragmentManager().findFragmentById(R.id.***map***);  
  
 **private** Boolean **requestBol** = **false**; *// #12 Canceling-an-Uber-Request @@@@@@@* **private** Marker **pickupMarker**;  
  
  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_consumer\_map***);  
 *// Obtain the SupportMapFragment and get notified when the map is ready to be used.* SupportMapFragment mapFragment = (SupportMapFragment) getSupportFragmentManager()  
 .findFragmentById(R.id.***map***);  
  
 **if** (ActivityCompat.*checkSelfPermission*(**this**, android.Manifest.permission.***ACCESS\_FINE\_LOCATION***) != PackageManager.***PERMISSION\_GRANTED*** && ActivityCompat.*checkSelfPermission*(**this**, android.Manifest.permission.***ACCESS\_COARSE\_LOCATION***) != PackageManager.***PERMISSION\_GRANTED***) {  
 ActivityCompat.*requestPermissions*(ConsumerMapActivity.**this**, **new** String[]{android.Manifest.permission.***ACCESS\_FINE\_LOCATION***}, **LOCATION\_REQUEST\_CODE**);  
 *// return;* }**else** {  
 mapFragment.getMapAsync(**this**);  
 }  
  
 **nLogout** = (Button)findViewById(R.id.***logout***);  
 **nRequest** = (Button)findViewById(R.id.***request***);  
 **nLogout**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View view) {  
 *//automatic logout* FirebaseAuth.*getInstance*().signOut();  
 Intent intent = **new** Intent(ConsumerMapActivity.**this**, MainActivity.**class**);  
 startActivity(intent);  
 finish();  
 **return**;  
 }  
 });  
  
 **nRequest**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View view) {  
  
 *// #12 Canceling-an-Uber-Request @@@@@@@@@@@@@@@@@* **if** (**requestBol**){  
 **requestBol** = **false**;  
 **geoQuery**.removeAllListeners();  
 **driverLocationRef**.removeEventListener(**driverLocationRefListener**);  
  
 **if** (**providerFoundID** != **null**){  
 DatabaseReference providerRef = FirebaseDatabase.*getInstance*().getReference().child(**"Users"**).child(**"Drivers"**).child(**providerFoundID**); *//removing data or clearing data from the database* providerRef.setValue(**true**); *//removing data or clearing data from the database* **providerFoundID** = **null**; *//removing data or clearing data from the database* }  
  
 **providerFound** = **false**; *//initiating the value back again* **radius** = 1; *//initiating the value back again* String userId = FirebaseAuth.*getInstance*().getCurrentUser().getUid(); *//removing data from the database* DatabaseReference ref = FirebaseDatabase.*getInstance*().getReference(**"consumerRequest"**); *//removing data from the database* GeoFire geoFire = **new** GeoFire(ref); *//removing data from the database* geoFire.removeLocation(userId); *//removing data from the database* **if** (**pickupMarker** !=**null**){ *//removing pickup Marker* **pickupMarker**.remove(); *//removing pickup Marker* }  
  
 }**else** {  
 **requestBol** = **true**;  
 String userId = FirebaseAuth.*getInstance*().getCurrentUser().getUid();  
 DatabaseReference ref = FirebaseDatabase.*getInstance*().getReference(**"consumerRequest"**);  
  
 GeoFire geoFire = **new** GeoFire(ref);  
 geoFire.setLocation(userId, **new** GeoLocation(**nLastLocation**.getLatitude(), **nLastLocation**.getLongitude()));  
  
 **pickupLocation** = **new** LatLng(**nLastLocation**.getLatitude(), **nLastLocation**.getLongitude());  
 **pickupMarker** = **nMap**.addMarker(**new** MarkerOptions().position(**pickupLocation**).title(**"Consumer House"**).icon(BitmapDescriptorFactory.*fromResource*(R.mipmap.***ic\_house***))); *// #13 Change Marker  
 // pickupMarker = nMap.addMarker(new MarkerOptions().position(pickupLocation).title("Consumer House")); // #12 Canceling-an-Uber-Request @@@@@@@@ assigning to variable pickupMarker* **nRequest**.setText(**"Getting your Service Provider..."**);  
  
 getClosestProvider();  
  
 }  
  
  
 *// #12 Canceling-an-Uber-Request @@@@@@@@@@@@@@@@@@  
  
  
  
  
 /\* from #11 +++++++++++++++++++  
  
  
 String userId = FirebaseAuth.getInstance().getCurrentUser().getUid();  
 DatabaseReference ref = FirebaseDatabase.getInstance().getReference("consumerRequest");  
  
 GeoFire geoFire = new GeoFire(ref);  
 geoFire.setLocation(userId, new GeoLocation(nLastLocation.getLatitude(), nLastLocation.getLongitude()));  
  
 pickupLocation = new LatLng(nLastLocation.getLatitude(), nLastLocation.getLongitude());  
 nMap.addMarker(new MarkerOptions().position(pickupLocation).title("Consumer House"));  
  
 nRequest.setText("Getting your Service Provider...");  
  
 getClosestProvider(); //calling a function  
  
 from #11 +++++++++++++++++++ \*/* }  
 });  
  
 *// mFusedLocationClient = LocationServices.getFusedLocationProviderClient(this);* }  
  
 **private int radius** = 1;  
 **private** Boolean **providerFound** = **false**;  
 **private** String **providerFoundID**;  
  
 **private** GeoQuery **geoQuery**; *// #12 Canceling-an-Uber-Request @@@@@@@* **private void** getClosestProvider(){ *//creating a function* **final** DatabaseReference providerLocation = FirebaseDatabase.*getInstance*().getReference().child(**"providerAvailable"**);  
  
 GeoFire geoFire = **new** GeoFire(providerLocation);  
  
 **geoQuery** = geoFire.queryAtLocation(**new** GeoLocation(**pickupLocation**.**latitude**, **pickupLocation**.**longitude**), **radius**);  
 **geoQuery**.removeAllListeners();  
  
 **geoQuery**.addGeoQueryEventListener(**new** GeoQueryEventListener() {  
 @Override  
 **public void** onKeyEntered(String key, GeoLocation location) {  
 **if** (!**providerFound** && **requestBol**){  
 **providerFound** = **true**;  
 **providerFoundID** = key;  
  
 *//* DatabaseReference providerRef = FirebaseDatabase.*getInstance*().getReference().child(**"Users"**).child(**"Drivers"**).child(**providerFoundID**);  
 String customerId = FirebaseAuth.*getInstance*().getCurrentUser().getUid();  
 HashMap map = **new** HashMap();  
 map.put(**"customerRideId"**, customerId);  
 providerRef.updateChildren(map);  
  
 *//call function* getDriverLocation();  
 **nRequest**.setText(**"Looking for Provider Location"**);  
  
  
 }  
 }  
  
 @Override  
 **public void** onKeyExited(String key) {  
  
 }  
  
 @Override  
 **public void** onKeyMoved(String key, GeoLocation location) {  
  
 }  
  
 @Override  
 **public void** onGeoQueryReady() {  
 **if** (!**providerFound**){  
 **radius**++;  
 getClosestProvider();  
 }  
 }  
  
 @Override  
 **public void** onGeoQueryError(DatabaseError error) {  
  
 }  
 });  
  
 }  
  
 *//creating the function* **private** Marker **nDriverMarker**;  
 **private** DatabaseReference **driverLocationRef**; *// #12 Canceling-an-Uber-Request @@@@@@@* **private** ValueEventListener **driverLocationRefListener**; *// #12 Canceling-an-Uber-Request @@@@@@@ creating String Listener* **public void** getDriverLocation(){  
 **driverLocationRef** =FirebaseDatabase.*getInstance*().getReference().child(**"providerWorking"**).child(**providerFoundID**).child(**"l"**);  
  
 **driverLocationRefListener** = **driverLocationRef**.addValueEventListener(**new** ValueEventListener() {  
 @Override  
 **public void** onDataChange(DataSnapshot dataSnapshot) {  
 **if** (dataSnapshot.exists()&& **requestBol**){ *// #12 Canceling-an-Uber-Request @@@@@@@ making sure the booolean is true* List<Object> map = (List<Object>) dataSnapshot.getValue();  
 **double** locationLat = 0;  
 **double** locationLng = 0;  
 **nRequest**.setText(**"Driver Found"**);  
  
 **if**(map.get(0)!=**null**) {  
 locationLat = Double.*parseDouble*(map.get(0).toString());  
 }  
  
 **if** (map.get(1) !=**null**) {  
 locationLng = Double.*parseDouble*(map.get(1).toString());  
 }  
  
 LatLng driverLatLng = **new** LatLng(locationLat,locationLng);  
  
 **if** (**nDriverMarker** != **null**){  
 **nDriverMarker**.remove();  
 }  
  
 *//#10 distanceBetweenTwoPoints =============================  
 /\*  
 Location loc1 = new Location("");  
 loc1.setLatitude(pickupLocation.latitude);  
 loc1.setLongitude(pickupLocation.longitude);  
  
 Location loc2 = new Location("");  
 loc2.setLatitude(driverLatLng.latitude);  
 loc2.setLongitude(driverLatLng.longitude);  
  
 float distance = loc1.distanceTo(loc2);  
 nRequest.setText("Provider Found: "+ String.valueOf(distance) + " meter/s.");  
  
 //#10 distanceBetweenTwoPoints =============================  
  
  
 //#11 NoticeWhen Driver is Arrived =============================  
  
 Location loc1 = new Location("");  
 loc1.setLatitude(pickupLocation.latitude);  
 loc1.setLongitude(pickupLocation.longitude);  
  
 Location loc2 = new Location("");  
 loc2.setLatitude(driverLatLng.latitude);  
 loc2.setLongitude(driverLatLng.longitude);  
  
 float distance = loc1.distanceTo(loc2);  
  
 if (distance<100){  
 nRequest.setText("Provider is Here : "+ String.valueOf(distance) + " meter/s.");  
 }else {  
 nRequest.setText("Provider Found: "+ String.valueOf(distance) + " meter/s.");  
 }  
  
 //#11 NoticeWhen Driver is Arrived =============================  
 \*/* Location loc1 = **new** Location(**""**);  
 loc1.setLatitude(**pickupLocation**.**latitude**);  
 loc1.setLongitude(**pickupLocation**.**longitude**);  
  
 Location loc2 = **new** Location(**""**);  
 loc2.setLatitude(driverLatLng.**latitude**);  
 loc2.setLongitude(driverLatLng.**longitude**);  
  
 **float** distance = loc1.distanceTo(loc2);  
  
 **if** (distance<100){  
 **nRequest**.setText(**"Provider is Here : "**+ String.*valueOf*(distance) + **" meter/s."**);  
 }**else** {  
 **nRequest**.setText(**"Provider Found: "**+ String.*valueOf*(distance) + **" meter/s."**);  
 }  
  
  
  
 **nDriverMarker** = **nMap**.addMarker(**new** MarkerOptions().position(driverLatLng).title(**"Your Service Provider"**).icon(BitmapDescriptorFactory.*fromResource*(R.mipmap.***ic\_person***))); *// #13 Change Marker ++++++++++++  
 // nDriverMarker = nMap.addMarker(new MarkerOptions().position(driverLatLng).title("Your Service Provider"));* }  
  
  
 }  
  
 @Override  
 **public void** onCancelled(DatabaseError databaseError) {  
  
 }  
 });  
  
 }  
  
  
  
  
  
 @Override  
 **public void** onMapReady(GoogleMap googleMap) {  
 **nMap** = googleMap;  
  
*/\*  
 // Add a marker in Sydney and move the camera  
 LatLng manila = new LatLng(14.5872236, 120.9939596);  
 nMap.addMarker(new MarkerOptions().position(manila).title("Marker in Philippines"));  
 nMap.moveCamera(CameraUpdateFactory.newLatLng(manila));  
\*/* **if** (ActivityCompat.*checkSelfPermission*(**this**, android.Manifest.permission.***ACCESS\_FINE\_LOCATION***) != PackageManager.***PERMISSION\_GRANTED*** && ActivityCompat.*checkSelfPermission*(**this**, android.Manifest.permission.***ACCESS\_COARSE\_LOCATION***) != PackageManager.***PERMISSION\_GRANTED***) {  
 ActivityCompat.*requestPermissions*(ConsumerMapActivity.**this**, **new** String[]{android.Manifest.permission.***ACCESS\_FINE\_LOCATION***}, **LOCATION\_REQUEST\_CODE**);  
  
 *// return;* }  
 buildGoogleApiClient();  
 **nMap**.setMyLocationEnabled(**true**);  
  
  
 }  
  
 **protected synchronized void** buildGoogleApiClient() {  
 **nGoogleApiClient** = **new** GoogleApiClient.Builder(**this**)  
 .addConnectionCallbacks(**this**)  
 .addOnConnectionFailedListener(**this**)  
 .addApi(LocationServices.***API***)  
 .build();  
 **nGoogleApiClient**.connect();  
  
  
 }  
  
  
 @Override  
 **public void** onLocationChanged(Location location) {  
 **nLastLocation** = location;  
 LatLng latlng = **new** LatLng(location.getLatitude(), location.getLongitude());  
 **nMap**.moveCamera(CameraUpdateFactory.*newLatLng*(latlng));  
 **nMap**.moveCamera(CameraUpdateFactory.*zoomTo*(16));  
  
 *// String userId = FirebaseAuth.getInstance().getCurrentUser().getUid();  
 // DatabaseReference ref = FirebaseDatabase.getInstance().getReference("providerAvailable");  
  
 // GeoFire geoFire = new GeoFire(ref);  
 // geoFire.setLocation(userId, new GeoLocation(location.getLatitude(), location.getLongitude()));* }  
  
 */\*  
 @Override  
 public void onStatusChanged(String s, int i, Bundle bundle) {  
  
 }  
  
 @Override  
 public void onProviderEnabled(String s) {  
  
 }  
  
 @Override  
 public void onProviderDisabled(String s) {  
  
 }  
 \*/* @Override  
 **public void** onConnected(@Nullable Bundle bundle) {  
 **nLocationRequest** = **new** LocationRequest();  
 **nLocationRequest**.setInterval(1000);  
 **nLocationRequest**.setFastestInterval(1000);  
  
 **if** (ActivityCompat.*checkSelfPermission*(**this**, android.Manifest.permission.***ACCESS\_FINE\_LOCATION***) != PackageManager.***PERMISSION\_GRANTED*** && ActivityCompat.*checkSelfPermission*(**this**, android.Manifest.permission.***ACCESS\_COARSE\_LOCATION***) != PackageManager.***PERMISSION\_GRANTED***) {  
 ActivityCompat.*requestPermissions*(ConsumerMapActivity.**this**, **new** String[]{android.Manifest.permission.***ACCESS\_FINE\_LOCATION***}, **LOCATION\_REQUEST\_CODE**);  
 *// return;* }  
 LocationServices.***FusedLocationApi***.requestLocationUpdates(**nGoogleApiClient**, **nLocationRequest**, **this**);  
  
 }  
  
 @Override  
 **public void** onConnectionSuspended(**int** i) {  
  
 }  
  
 @Override  
 **public void** onConnectionFailed(@NonNull ConnectionResult connectionResult) {  
  
 }  
  
 **final int LOCATION\_REQUEST\_CODE** = 1;  
 @Override  
 **public void** onRequestPermissionsResult(**int** requestCode, @NonNull String[] permissions, @NonNull **int**[] grantResults) {  
 **super**.onRequestPermissionsResult(requestCode, permissions, grantResults);  
 **switch**(requestCode){  
 **case LOCATION\_REQUEST\_CODE**:{  
 **if** (grantResults.**length** >0 && grantResults[0] == PackageManager.***PERMISSION\_GRANTED***){  
 **mapFragment**.getMapAsync(**this**);  
 }**else**{  
 Toast.*makeText*(getApplicationContext(), **"Please provide permission"**, Toast.***LENGTH\_SHORT***).show();  
 }  
 **break**;  
 }  
 }  
  
  
 }  
  
 @Override  
 **protected void** onStop() {  
 **super**.onStop();  
 *// LocationServices.FusedLocationApi.removeLocationUpdates(nGoogleApiClient, this);  
  
 // String userId = FirebaseAuth.getInstance().getCurrentUser().getUid();  
 // DatabaseReference ref = FirebaseDatabase.getInstance().getReference("providerAvailable");  
  
 // GeoFire geoFire = new GeoFire(ref);  
 // geoFire.removeLocation(userId);* }  
  
  
}

ProviderMap========================================

**package** com.example.davidhaniel.uber;  
  
**import** android.content.Intent;  
**import** android.content.pm.PackageManager;  
**import** android.location.Location;  
**import** android.location.LocationListener;  
**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.view.View;  
**import** android.widget.Button;  
**import** android.widget.Toast;  
  
**import** com.firebase.geofire.GeoFire;  
**import** com.firebase.geofire.GeoLocation;  
**import** com.google.android.gms.common.ConnectionResult;  
**import** com.google.android.gms.common.api.GoogleApiClient;  
**import** com.google.android.gms.location.FusedLocationProviderClient;  
**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** com.google.android.gms.tasks.OnSuccessListener;  
**import** com.google.firebase.auth.FirebaseAuth;  
**import** com.google.firebase.database.DataSnapshot;  
**import** com.google.firebase.database.DatabaseError;  
**import** com.google.firebase.database.DatabaseReference;  
**import** com.google.firebase.database.FirebaseDatabase;  
**import** com.google.firebase.database.ValueEventListener;  
  
**import** java.util.List;  
**import** java.util.Map;  
  
**public class** ProviderMapsActivity **extends** FragmentActivity **implements** OnMapReadyCallback, GoogleApiClient.ConnectionCallbacks, GoogleApiClient.OnConnectionFailedListener, com.google.android.gms.location.LocationListener {  
  
 **private** GoogleMap **nMap**;  
 GoogleApiClient **nGoogleApiClient**;  
 Location **nLastLocation**;  
 LocationRequest **nLocationRequest**;  
 *// private SupportMapFragment mapFragment;  
 // FusedLocationProviderClient mFusedLocationClient;* **private** Button **nLogout**;  
 SupportMapFragment **mapFragment** = (SupportMapFragment) getSupportFragmentManager().findFragmentById(R.id.***map***);  
  
 **private** String **customerId** = **""**;  
 **private** Boolean **isLoggingOut** = **false**; *// #14 Fixing-Driver-Logout-Bug +++++++++++++* @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_provider\_maps***);  
 *// Obtain the SupportMapFragment and get notified when the map is ready to be used.* SupportMapFragment mapFragment = (SupportMapFragment) getSupportFragmentManager().findFragmentById(R.id.***map***);  
  
 **if** (ActivityCompat.*checkSelfPermission*(**this**, android.Manifest.permission.***ACCESS\_FINE\_LOCATION***) != PackageManager.***PERMISSION\_GRANTED*** && ActivityCompat.*checkSelfPermission*(**this**, android.Manifest.permission.***ACCESS\_COARSE\_LOCATION***) != PackageManager.***PERMISSION\_GRANTED***) {  
 ActivityCompat.*requestPermissions*(ProviderMapsActivity.**this**, **new** String[]{android.Manifest.permission.***ACCESS\_FINE\_LOCATION***}, **LOCATION\_REQUEST\_CODE**);  
 *// return;* }**else** {  
 mapFragment.getMapAsync(**this**);  
 }  
  
 **nLogout** = (Button)findViewById(R.id.***logout***);  
 **nLogout**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View view) {  
  
 **isLoggingOut** = **true**; *// #14 Fixing-Driver-Logout-Bug +++++++++++++* disconnectProvider(); *// #14 Fixing-Driver-Logout-Bug +++++++++++++  
 //automatic logout* FirebaseAuth.*getInstance*().signOut();  
 Intent intent = **new** Intent(ProviderMapsActivity.**this**, MainActivity.**class**);  
 startActivity(intent);  
 finish();  
 **return**;  
 }  
 });  
  
  
 *// mFusedLocationClient = LocationServices.getFusedLocationProviderClient(this);  
  
 //call a function* getAssignedCustomer();  
 }  
  
 *//creating the function* **public void** getAssignedCustomer(){  
 String driverId = FirebaseAuth.*getInstance*().getCurrentUser().getUid();  
 **final** DatabaseReference assignedCustomerRef = FirebaseDatabase.*getInstance*().getReference().child(**"Users"**).child(**"Drivers"**).child(driverId).child(**"customerRideId"**);  
  
 assignedCustomerRef.addValueEventListener(**new** ValueEventListener() {  
 @Override  
 **public void** onDataChange(DataSnapshot dataSnapshot) {  
 **if** (dataSnapshot.exists()) { *// #12 Canceling-an-Uber-Request @@@@@@@ if the child(customerRideId) is removed this..dataSnapshot.exists()..will be false (else)* **customerId** = dataSnapshot.getValue().toString();  
  
 *//call a function* getAssignedCustomerPickupLocation();  
 }**else** { *// #12 Canceling-an-Uber-Request @@@@@@@ checking if the child(customerRideId) is removed.* **customerId** = **""**; *// #12 Canceling-an-Uber-Request @@@@@@@ initiating variable back again* **if** (**pickupMarker** != **null**){ *// #12 Canceling-an-Uber-Request @@@@@@@ removing Marker* **pickupMarker**.remove();  
 }  
 **if** (**assignedCustomerPickupLocationRefListener** != **null**){ *// #12 Canceling-an-Uber-Request @@@@@@@ bugs fix...adding if statement to remove the EventListener* **assignedCustomerPickupLocationRef**.removeEventListener(**assignedCustomerPickupLocationRefListener**); *// #12 Canceling-an-Uber-Request @@@@@@@ removing EventListener* }  
  
  
  
 }  
 }  
  
  
 @Override  
 **public void** onCancelled(DatabaseError databaseError) {  
  
 }  
 });  
  
 }  
  
 *// creating a function* Marker **pickupMarker**; *// #12 Canceling-an-Uber-Request @@@@@@@ creating String variable* **private** DatabaseReference **assignedCustomerPickupLocationRef**; *// #12 Canceling-an-Uber-Request @@@@@@@ creating String variable DatabaseReference* **private** ValueEventListener **assignedCustomerPickupLocationRefListener**; *// #12 Canceling-an-Uber-Request @@@@@@@ creating String variable ValueEventListener* **public void** getAssignedCustomerPickupLocation(){  
 **assignedCustomerPickupLocationRef** = FirebaseDatabase.*getInstance*().getReference().child(**"customerRequest"**).child(**customerId**).child(**"l"**);  
  
 **assignedCustomerPickupLocationRefListener** = **assignedCustomerPickupLocationRef**.addValueEventListener(**new** ValueEventListener() {  
 @Override  
 **public void** onDataChange(DataSnapshot dataSnapshot) {  
 **if** (dataSnapshot.exists() && !**customerId**.equals(**""**)){  
 List<Object> map = (List<Object>) dataSnapshot.getValue();  
 **double** locationLat = 0;  
 **double** locationLng = 0;  
  
 **if** (map.get(0)!=**null**){  
 locationLat = Double.*parseDouble*(map.get(0).toString());  
 }  
  
 **if** (map.get(1)!=**null**){  
 locationLng = Double.*parseDouble*(map.get(1).toString());  
 }  
  
 LatLng driverLatLng = **new** LatLng(locationLat, locationLng);  
 **pickupMarker** = **nMap**.addMarker(**new** MarkerOptions().position(driverLatLng).title(**"Pickup Location"**).icon(BitmapDescriptorFactory.*fromResource*(R.mipmap.***ic\_house***))); *// #13 Change Marker  
 // pickupMarker = nMap.addMarker(new MarkerOptions().position(driverLatLng).title("Pickup Location"));* }  
  
 }  
  
 @Override  
 **public void** onCancelled(DatabaseError databaseError) {  
  
 }  
 });  
  
 }  
  
  
 @Override  
 **public void** onMapReady(GoogleMap googleMap) {  
 **nMap** = googleMap;  
  
*/\*  
 // Add a marker in Sydney and move the camera  
 LatLng manila = new LatLng(14.5872236, 120.9939596);  
 nMap.addMarker(new MarkerOptions().position(manila).title("Marker in Philippines"));  
 nMap.moveCamera(CameraUpdateFactory.newLatLng(manila));  
\*/* **if** (ActivityCompat.*checkSelfPermission*(**this**, android.Manifest.permission.***ACCESS\_FINE\_LOCATION***) != PackageManager.***PERMISSION\_GRANTED*** && ActivityCompat.*checkSelfPermission*(**this**, android.Manifest.permission.***ACCESS\_COARSE\_LOCATION***) != PackageManager.***PERMISSION\_GRANTED***) {  
 ActivityCompat.*requestPermissions*(ProviderMapsActivity.**this**, **new** String[]{android.Manifest.permission.***ACCESS\_FINE\_LOCATION***}, **LOCATION\_REQUEST\_CODE**);  
  
 *// return;* }  
 buildGoogleApiClient();  
 **nMap**.setMyLocationEnabled(**true**);  
  
  
 }  
  
 **protected synchronized void** buildGoogleApiClient() {  
 **nGoogleApiClient** = **new** GoogleApiClient.Builder(**this**)  
 .addConnectionCallbacks(**this**)  
 .addOnConnectionFailedListener(**this**)  
 .addApi(LocationServices.***API***)  
 .build();  
 **nGoogleApiClient**.connect();  
  
  
 }  
  
  
 @Override  
 **public void** onLocationChanged(Location location) {  
 **if** (getApplicationContext()!=**null**){ *//additional line...if statement* **nLastLocation** = location;  
 LatLng latlng = **new** LatLng(location.getLatitude(), location.getLongitude());  
 **nMap**.moveCamera(CameraUpdateFactory.*newLatLng*(latlng));  
 **nMap**.moveCamera(CameraUpdateFactory.*zoomTo*(16));  
  
 String userId = FirebaseAuth.*getInstance*().getCurrentUser().getUid();  
  
 *//updated DatabaseReference ref = FirebaseDatabase.getInstance().getReference("providerAvailable");  
  
 //updated GeoFire geoFire = new GeoFire(ref);  
 //updated geoFire.setLocation(userId, new GeoLocation(location.getLatitude(), location.getLongitude()));  
  
 // replaced updated* DatabaseReference refAvailable = FirebaseDatabase.*getInstance*().getReference(**"providerAvailable"**);  
 DatabaseReference refWorking = FirebaseDatabase.*getInstance*().getReference(**"providerWorking"**);  
  
 GeoFire geoFireAvailable = **new** GeoFire(refAvailable);  
 GeoFire geoFireWorking = **new** GeoFire(refWorking);  
  
  
  
 **switch** (**customerId**){  
 **case ""**:  
 geoFireWorking.removeLocation(userId);  
 geoFireAvailable.setLocation(userId, **new** GeoLocation(location.getLatitude(), location.getLongitude()));  
 **break**;  
 **default**:  
 geoFireAvailable.removeLocation(userId);  
 geoFireWorking.setLocation(userId, **new** GeoLocation(location.getLatitude(), location.getLongitude()));  
 **break**;  
  
 }  
 }  
  
  
  
 }  
  
 */\*  
 @Override  
 public void onStatusChanged(String s, int i, Bundle bundle) {  
  
 }  
  
 @Override  
 public void onProviderEnabled(String s) {  
  
 }  
  
 @Override  
 public void onProviderDisabled(String s) {  
  
 }  
 \*/* @Override  
 **public void** onConnected(@Nullable Bundle bundle) {  
 **nLocationRequest** = **new** LocationRequest();  
 **nLocationRequest**.setInterval(1000);  
 **nLocationRequest**.setFastestInterval(1000);  
  
 **if** (ActivityCompat.*checkSelfPermission*(**this**, android.Manifest.permission.***ACCESS\_FINE\_LOCATION***) != PackageManager.***PERMISSION\_GRANTED*** && ActivityCompat.*checkSelfPermission*(**this**, android.Manifest.permission.***ACCESS\_COARSE\_LOCATION***) != PackageManager.***PERMISSION\_GRANTED***) {  
 ActivityCompat.*requestPermissions*(ProviderMapsActivity.**this**, **new** String[]{android.Manifest.permission.***ACCESS\_FINE\_LOCATION***}, **LOCATION\_REQUEST\_CODE**);  
 *// return;* }  
 LocationServices.***FusedLocationApi***.requestLocationUpdates(**nGoogleApiClient**, **nLocationRequest**, **this**);  
  
 }  
  
 @Override  
 **public void** onConnectionSuspended(**int** i) {  
  
 }  
  
 @Override  
 **public void** onConnectionFailed(@NonNull ConnectionResult connectionResult) {  
  
 }  
  
 **final int LOCATION\_REQUEST\_CODE** = 1;  
 @Override  
 **public void** onRequestPermissionsResult(**int** requestCode, @NonNull String[] permissions, @NonNull **int**[] grantResults) {  
 **super**.onRequestPermissionsResult(requestCode, permissions, grantResults);  
 **switch**(requestCode){  
 **case LOCATION\_REQUEST\_CODE**:{  
 **if** (grantResults.**length** >0 && grantResults[0] == PackageManager.***PERMISSION\_GRANTED***){  
 **mapFragment**.getMapAsync(**this**);  
 }**else**{  
 Toast.*makeText*(getApplicationContext(), **"Please provide permission"**, Toast.***LENGTH\_SHORT***).show();  
 }  
 **break**;  
 }  
 }  
  
  
 }  
 **private void** disconnectProvider(){ *// #14 Fixing-Driver-Logout-Bug +++++++++++++* LocationServices.***FusedLocationApi***.removeLocationUpdates(**nGoogleApiClient**, **this**);  
  
 String userId = FirebaseAuth.*getInstance*().getCurrentUser().getUid();  
 DatabaseReference ref = FirebaseDatabase.*getInstance*().getReference(**"providerAvailable"**);  
  
 GeoFire geoFire = **new** GeoFire(ref);  
 geoFire.removeLocation(userId);  
 }  
  
 @Override  
 **protected void** onStop() {  
 **super**.onStop();  
 **if** (!**isLoggingOut**){ *// #14 Fixing-Driver-Logout-Bug +++++++++++++* disconnectProvider(); *// #14 Fixing-Driver-Logout-Bug +++++++++++++* }  
  
  
 }  
  
  
}