TOURIST GUIDE

Android Programming- CIS 600(Summer)

Date

OVERVIEW

Tourist Guide is the best app that provides the users all the nearby places near him that he might need in a new place. Tough there are many apps that provide similar features, but none provided it in a better organized fashion. We provide a list of places in the following categories:

- Eating
- Shopping
- Entertainment
- Historic Places
- Lodging
- Utilities
- ATMs

FEATURE OVERVIEW

- Custom Views
- View Pager
- Database Helpers
- Text to Speech Service
- GPS Tracking
- Broadcast Receiver
- Retrofit Library Use
- Recycler View
- In- app Purchase
- Maps

CODE SNIPPETS

Custom Views- AvatarView, PlaceView and Split Toolbar

Avatar View extends AppCompatImageButton-

```
@Override
public void onDraw(Canvas canvas) {
    saveBasicValues(canvas);

    if (viewSize == 0) {
        return;
    }

    Bitmap bitmap = cutIntoCircle(drawableToBitmap(drawable));

    if (bitmap == null) {
        return;
    }

    canvas.translate(circleCenterXValue, circleCenterYValue);

    //Draw Border
    canvas.drawCircle(circleRadius + borderWidth, circleRadius + borderWidth, circleRadius +
borderWidth, borderPaint);

    canvas.drawBitmap(bitmap, 0, 0, null);
}

private void init(Context context, AttributeSet attrs) {
    setDefaultBorderValues();

    if (attrs != null) {
        TypedArray typedArray = context.getTheme().obtainStyledAttributes(
                attrs,
                R.styleable.AvatarView,
                0, 0);
        try {
            configureBorderValues(typedArray);
        } finally {
            typedArray.recycle();
        }
    }

    borderPaint.setAntiAlias(true);
    borderPaint.setStyle(Paint.Style.FILL);
    borderPaint.setColor(borderColor);

    mainPaint.setAntiAlias(true);
    mainPaint.setColor(getResources().getColor(R.color.av_bitmap_background_color));
    mainPaint.setXfermode(new PorterDuffXfermode(PorterDuff.Mode.SRC_IN));
}

private Bitmap drawableToBitmap(Drawable drawable) {
    if (drawable == null) {
        return null;
    }
```

```
    try {
        Bitmap bitmap = Bitmap.createBitmap(viewSize, viewSize, Bitmap.Config.ARGB_8888);
        Canvas canvas = new Canvas(bitmap);
        drawable.setBounds(0, 0, viewSize, viewSize);
        drawable.draw(canvas);

        return bitmap;
    } catch (OutOfMemoryError error) {

        return null;
    }
}

private Bitmap cutIntoCircle(Bitmap bitmap) {
    if (bitmap == null) {
        return null;
    }

    try {
        Bitmap output = Bitmap.createBitmap(viewSize, viewSize, Bitmap.Config.ARGB_8888);
        Canvas canvas = new Canvas(output);

        canvas.drawARGB(0, 0, 0, 0);
        canvas.drawCircle(circleRadius + borderWidth, circleRadius + borderWidth, circleRadius,
borderPaint);

        canvas.drawBitmap(bitmap, circleRect, circleRect, mainPaint);
        return output;
    } catch (OutOfMemoryError error) {

        return null;
    }
}
```

ViewPagerFragment- creates the tabs for the application main screen.

```
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {
    // Inflate the layout for this fragment
    View v=inflater.inflate(R.layout.fragment_view_pager, container, false);;

    viewPager = (ViewPager) v.findViewById(R.id.viewpager);

    setupViewPager(viewPager);

    tabLayout = (TabLayout) v.findViewById(R.id.tabs);
    tabLayout.setupWithViewPager(viewPager);

    pageno = viewPager.getCurrentItem();
```

```
    final ViewPager.OnPageChangeListener pageChangeListener =new
ViewPager.OnPageChangeListener() {
        @Override
        public void onPageScrolled(int position, float positionOffset, int positionOffsetPixels) {
            Log.d("pageno",String.valueOf(position));
            pageno=position;

        }

        @Override
        public void onPageSelected(int position) {
            pageno=position;


        }

        @Override
        public void onPageScrollStateChanged(int state) {

        }
    };

    viewPager.addOnPageChangeListener( pageChangeListener);
    viewPager.post(new Runnable()
    {
        @Override
        public void run()
        {
            pageChangeListener .onPageSelected(viewPager.getCurrentItem()+1);
        }
    });

    setupTabIcons();

    return v;
}


private void setupTabIcons() {
    tabLayout.getTabAt(0).setIcon(tabIcons[0]);
    tabLayout.getTabAt(1).setIcon(tabIcons[1]);
    tabLayout.getTabAt(2).setIcon(tabIcons[2]);
    tabLayout.getTabAt(3).setIcon(tabIcons[3]);
    tabLayout.getTabAt(4).setIcon(tabIcons[4]);
    tabLayout.getTabAt(5).setIcon(tabIcons[5]);
    tabLayout.getTabAt(6).setIcon(tabIcons[6]);
}

private void setupViewPager(ViewPager viewPager) {
    adapter = new ViewPagerAdapter(getChildFragmentManager());
    masterFragment=new RecyclerviewFragment();
    masterFragment2=new RecyclerviewFragment2();
    masterFragment3=new Recyclerviewfragment3();
    masterFragment4=new RecyclerviewFragment4();
    masterFragment5=new RecyclerviewFragment5();
    masterFragment6=new RecyclerviewFragment6();
    masterFragment7=new RecyclerviewFragment7();
```

```
    adapter.addFragment(masterFragment,"Eating");
    adapter.addFragment(masterFragment2,"Shopping");
    adapter.addFragment(masterFragment3,"Entertainmnet");
    adapter.addFragment(masterFragment4,"Historic Places");
    adapter.addFragment(masterFragment5,"Lodging");
    adapter.addFragment(masterFragment6,"Utilities");
    adapter.addFragment(masterFragment7,"ATM");
    viewPager.setAdapter(adapter);
    viewPager.setPageTransformer(true, new CubeOutTransformer());
}
```

DataBaseHelper- Provdes SQLLite support for the application.

```
@Override
public void onCreate(SQLiteDatabase db) {
    db.execSQL("Create table touristguidesignin_table(ID INTEGER PRIMARY KEY
AUTOINCREMENT,username TEXT,password TEXT,logout Text)");
}

@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    db.execSQL("DROP TABLE IF EXISTS " + Table_Name);
    onCreate(db);
}

public boolean insertdata(String username,String password,String logout)
{
    SQLiteDatabase db=this.getWritableDatabase();
    ContentValues b1=new ContentValues();
    b1.put(COL_2,username);
    b1.put(COL_3,password);
    b1.put(COL_4,logout);
    long b2=db.insert(Table_Name, null, b1);
    if(b2==-1)
    {
        return false;
    }
    else
    {
        return true;
    }
}

public Cursor getalldata()
{
    SQLiteDatabase db=this.getWritableDatabase();
    Cursor b1=db.rawQuery("Select * from "+Table_Name,null);
    return b1;
}
```

```
public boolean deletedata(int id)
{
   SQLiteDatabase db=this.getWritableDatabase();
   long b2=db.delete(Table_Name, null,null);
   if(b2>0)
   {
      return true;
   }
   else
   {
      return false;
   }
}
```

TTSService- text to speech service is implemented in this class

```
@Override
public void onInit(int status) {
   Log.v(TAG, "oninit");
   if (status == TextToSpeech.SUCCESS) {
      int result = mTts.setLanguage(Locale.US);
      if (result == TextToSpeech.LANG_MISSING_DATA ||
            result == TextToSpeech.LANG_NOT_SUPPORTED) {
         Log.v(TAG, "Language is not available.");
      } else {

         sayHello(str);

      }
   } else {
      Log.v(TAG, "Could not initialize TextToSpeech.");
   }
}
private void sayHello(String str) {
   mTts.speak(str,
         TextToSpeech.QUEUE_FLUSH,
         null);

}
```

GPSTracker- users the device gps to gets users location

```
private class LocationListener implements android.location.LocationListener {

    public LocationListener(String provider) {
        Log.e(TAG, "LocationListener " + provider);
        mLastLocation = new Location(provider);

        sendMessage();
    }

    @Override
    public void onLocationChanged(Location location) {
        Log.e(TAG, "onLocationChanged: " + location);
        mLastLocation.set(location);
        sendMessage();
    }

    @Override
    public void onProviderDisabled(String provider) {
        Log.e(TAG, "onProviderDisabled: " + provider);
    }

    @Override
    public void onProviderEnabled(String provider) {
        Log.e(TAG, "onProviderEnabled: " + provider);
    }

    @Override
    public void onStatusChanged(String provider, int status, Bundle extras) {
        Log.e(TAG, "onStatusChanged: " + provider);
    }
}

LocationListener[] mLocationListeners = new LocationListener[]{
        new LocationListener(LocationManager.GPS_PROVIDER),
        new LocationListener(LocationManager.NETWORK_PROVIDER)
};
```

SMS Reciever- acts as a broadcast receiver, detects message for generated OTP and accepts it automatically.

```
public class SmsReceiver extends BroadcastReceiver {
    String number;
    String message ;
    @Override
    public void onReceive(Context context, Intent intent) {
        SmsMessage msg;

        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.KITKAT) {
            SmsMessage[] msgs = Telephony.Sms.Intents.getMessagesFromIntent(intent);
            msg = msgs[0];
        } else {
            Object pdus[] = (Object[]) intent.getExtras().get("pdus");
            msg = SmsMessage.createFromPdu((byte[]) pdus[0]);
```

```
        }

        number = msg.getOriginatingAddress();
        message = msg.getMessageBody();

        if(Signup.number.equals(number))
        {
            if(Signup.message.equals(message))
            {
              Signup.getInstace().setText(message.substring(36));
            }
        }
    }
}
```

POJO & POJO2- Two retrofit classes for the two maps we have in the application.

```
@SerializedName("geometry")
@Expose
private Geometry geometry;
@SerializedName("icon")
@Expose
private String icon;
@SerializedName("id")
@Expose
private String id;
@SerializedName("name")
@Expose
private String name;
@SerializedName("opening_hours")
@Expose
private OpeningHours openingHours;
@SerializedName("photos")
@Expose
private List<Photo> photos = new ArrayList<Photo>();
@SerializedName("place_id")
@Expose
private String placeId;
@SerializedName("rating")
@Expose
private Double rating;
@SerializedName("reference")
@Expose
private String reference;
@SerializedName("scope")
@Expose
private String scope;
@SerializedName("types")
@Expose
private List<String> types = new ArrayList<String>();
@SerializedName("vicinity")
@Expose
```

```
private String vicinity;
@SerializedName("price_level")
@Expose
private Integer priceLevel;
```

RecyclerViewFragment- Main Screen that displays various places in well-organized manner-

```
setHasOptionsMenu(true);
View v = inflater.inflate(R.layout.fragment_recyclerview, container, false);

recyclerView = (RecyclerView) v.findViewById(R.id.recycler_view);

mAdapter = new MovieAdapter(movieList,getContext());
final RecyclerView.LayoutManager mLayoutManager = new LinearLayoutManager(getContext());
recyclerView.setLayoutManager(mLayoutManager);
recyclerView.setItemAnimator(new DefaultItemAnimator());
recyclerView.addItemDecoration(new android.support.v7.widget.DividerItemDecoration(getContext(),
LinearLayoutManager.VERTICAL));
recyclerView.setHasFixedSize(true);
AlphaInAnimationAdapter alphaAdapter = new AlphaInAnimationAdapter(mAdapter);
ScaleInAnimationAdapter scaleadapter = new ScaleInAnimationAdapter(alphaAdapter);
scaleadapter.setDuration(700);
recyclerView.setAdapter(scaleadapter);
// recyclerView.setAdapter(mAdapter);

mAdapter.setOnItemClickListener(new MovieAdapter.ClickListener() {

  @Override
  public void onItemClick(int position, View v,List<Movie> movielist) {
      buttonClicked(v, position,movielist);
    }


  @Override
  public void onItemLongClick(int position, View v) {
    getActivity().startActionMode(new ActionBarCallBack(position));

  }

  @Override
  public void onOverflowMenuClick(final int position, View v) {
    PopupMenu popup = new PopupMenu(getActivity(), v);
    popup.setOnMenuItemClickListener(new PopupMenu.OnMenuItemClickListener() {
      @Override
      public boolean onMenuItemClick(MenuItem item) {
        int id = item.getItemId();
        switch (id) {
          case R.id.action_delete:
            movieList.remove(position);
            FadeInAnimator animator = new FadeInAnimator();
            animator.setInterpolator(new OvershootInterpolator());
            animator.setRemoveDuration(500);
```

```
                recyclerView.setItemAnimator(animator);
                mAdapter.notifyItemRemoved(position);

                mAdapter.notifyItemRangeChanged(0, movieList.size());

                return true;
            case R.id.action_duplicate:
                movieList.add(position + 1, movieList.get(position));
                FlipInBottomXAnimator animator2 = new FlipInBottomXAnimator();
                animator2.setInterpolator(new OvershootInterpolator());
                animator2.setRemoveDuration(1000);
                recyclerView.setItemAnimator(animator2);
                mAdapter.notifyItemInserted(position + 1);

                return true;
            }
            return false;
        }
    });
    MenuInflater menuInflater = popup.getMenuInflater();
    menuInflater.inflate(R.menu.cab, popup.getMenu());
    popup.show();

  }

});


@Override
public boolean onOptionsItemSelected(MenuItem item) {
   switch (item.getItemId()) {

      case R.id.action_unhide:
         MainActivity.b1.purchaseRemoveAds();
         break;

      case R.id.action_sort:
            Movie.flag = true;
            Collections.sort(movieList);
            mAdapter.notifyDataSetChanged();
            break;


      case R.id.action_logoff:
         AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());
         LayoutInflater inflater = getLayoutInflater();
         View dialogView = inflater.inflate(R.layout.exit, null);
         builder.setView(dialogView);
         final AlertDialog alert = builder.create();

         Button b = (Button) dialogView.findViewById(R.id.button51);
         Button b2 = (Button) dialogView.findViewById(R.id.button7);

         b2.setOnClickListener(new View.OnClickListener() {

            @Override
            public void onClick(View v) {
```

```
                alert.dismiss();
            }
        });

        b.setOnClickListener(new View.OnClickListener() {
                        @Override
                        public void onClick(View v) {

                            DatabaseHelper dh = new DatabaseHelper(getActivity());
                            FbDatabaseHelper dh1 = new FbDatabaseHelper(getActivity());
                            TwDatabaseHelper dh2 = new TwDatabaseHelper(getActivity());
                            GlDatabaseHelper dh3 = new GlDatabaseHelper(getActivity());

                            AccessToken accessToken = AccessToken.getCurrentAccessToken();
                            TwitterSession twitterSession =
TwitterCore.getInstance().getSessionManager().getActiveSession();

                            if (accessToken != null) {
                                if (SplashScreen.fblogout == true) {
                                    SplashScreen.fblogout = false;
                                    dh1.deletedata(1);
                                    LoginManager.getInstance().logOut();
                                }
                            }

                            if (twitterSession != null) {
                                if (SplashScreen.twlogout == true) {
                                    SplashScreen.twlogout = false;
                                    dh2.deletedata(1);
                                    CookieSyncManager.createInstance(getActivity());
                                    CookieManager cookieManager = CookieManager.getInstance();
                                    cookieManager.removeSessionCookie();
                                    TwitterCore.getInstance().getSessionManager().clearActiveSession();
                                }
                            }

                            if (SplashScreen.logout == true) {
                                SplashScreen.logout = false;
                                dh.deletedata(1);
                            }

                            if (SplashScreen.gllogout == true) {
                                SplashScreen.gllogout = false;
                                dh3.deletedata(1);

                            }
                            Intent i = new Intent(getActivity(), Signin.class);
                            startActivity(i);
                            getActivity().finish();
                        }
                    }
        );
        alert.show();

        break;

    case R.id.action_privacy:
        AlertDialog.Builder builder2 = new AlertDialog.Builder(getActivity());
```

```
            LayoutInflater inflater2 = getLayoutInflater();
            View dialogView2 = inflater2.inflate(R.layout.termsandconditions, null);
            builder2.setView(dialogView2);
            final AlertDialog alert2 = builder2.create();


            Button b3 = (Button) dialogView2.findViewById(R.id.button7);

            b3.setOnClickListener(new View.OnClickListener() {
                                @Override
                                public void onClick(View v) {
                                    alert2.dismiss();
                                }
                            }
            );

            alert2.show();
            break;

        case R.id.action_map:

            Intent i = new Intent(getActivity(), MapsActivity.class);
            i.putExtra("no", ViewPagerFragment.pageno);
            //startActivity(i);
            flipCard();
            break;

        case R.id.action_search:

            break;

    }
    return super.onOptionsItemSelected(item);
}
```

MyBilin- Class to provide the feature of in-app purchase to remove adds-

```
IabHelper.QueryInventoryFinishedListener mGotInventoryListener = new
IabHelper.QueryInventoryFinishedListener() {
    public void onQueryInventoryFinished(IabResult result,
                                         Inventory inventory) {
        Log.d(TAG, "Query inventory finished.");

        if (mHelper == null)
            return;

        // Is it a failure?
        if (result.isFailure()) {
            // complain("Failed to query inventory: " + result);
            return;
        }

        Log.d(TAG, "Query inventory was successful.");
```

```
        // Do we have the premium upgrade?
        Purchase removeAdsPurchase = inventory.getPurchase(SKU_REMOVE_ADS);
        Constants.isAdsDisabled = (removeAdsPurchase != null &&
verifyDeveloperPayload(removeAdsPurchase));
        removeAds();

        // setWaitScreen(false);
        Log.d(TAG, "Initial inventory query finished; enabling main UI.");
    }
};


IabHelper.OnIabPurchaseFinishedListener mPurchaseFinishedListener = new
IabHelper.OnIabPurchaseFinishedListener() {
    public void onIabPurchaseFinished(IabResult result, Purchase purchase) {
        Log.d(TAG, "Purchase finished: " + result + ", purchase: "
                + purchase);


        if (mHelper == null)
            return;

        if (result.isFailure()) {
            complain("Error purchasing: " + result);
            return;
        }
        if (!verifyDeveloperPayload(purchase)) {
            complain("Error purchasing. Authenticity verification failed.");
            return;
        }

        Log.d(TAG, "Purchase successful.");

        if (purchase.getSku().equals(SKU_REMOVE_ADS)) {
            MainActivity.bottomT.setVisibility(View.VISIBLE);
            removeAds();

        }
    }
};
```

MovieFragment- Fragment to display the details of the selected place

```
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState)
{
    setHasOptionsMenu(true);
    Fresco.initialize(getContext());
    View v=inflater.inflate(R.layout.fragment_movie, container, false);

    listView = (ListView) v.findViewById(R.id.mobile_list);

    iv= (SimpleDraweeView)v. findViewById(R.id.imageView);

    MainActivity.mTitle.setText(title);
```

```
    name=(TextView)v.findViewById(R.id.placeName);
    rating=(RatingBar) v.findViewById(R.id.ratingBar);
    placeAddress=(TextView)v.findViewById(R.id.description);
    ratingText = (TextView) v.findViewById(R.id.ratingText);
    pricing = (TextView) v.findViewById(R.id.type);
    count=(TextView) v.findViewById(R.id.ratingcount);

    floatingActionButton = (FloatingActionButton) v.findViewById(R.id.floatingActionButton);
    callButton = (ImageButton) v.findViewById(R.id.imageButton2);
    websiteButton = (ImageButton) v.findViewById(R.id.imageButton4);

    if(savedInstanceState!=null)
    {
        ratings=savedInstanceState.getFloat("ratings");
        pricinglevel=savedInstanceState.getString("pricing");
        title=savedInstanceState.getString("title");
        homepage=savedInstanceState.getString("homepage");
        call=savedInstanceState.getString("call");
        address=savedInstanceState.getString("address");
        id=savedInstanceState.getString("id");
        image=savedInstanceState.getString("image");
        photo=savedInstanceState.getString("photo");
        reviewcount=savedInstanceState.getString("reviewcount");

        rating.setEnabled(true);
        if(photo!=null) {
            Uri uri = Uri.parse("https://maps.googleapis.com/maps/api/place/photo?photoreference=" +
photo +
"&sensor=false&maxheight=1600&maxwidth=1600&key=AIzaSyC1bDJ9vVGjOwemAvVQyvLz0HPyv-
kLDoI");
            iv.setImageURI(uri);
        }
        else
        {
            Uri uri = Uri.parse("https://maps.googleapis.com/maps/api/place/photo?photoreference=" +
image +
"&sensor=false&maxheight=1600&maxwidth=1600&key=AIzaSyC1bDJ9vVGjOwemAvVQyvLz0HPyv-
kLDoI");
            iv.setImageURI(uri);
        }
        rating.setRating(ratings);
        name.setText(title);
        description.setText(overview);
        ArrayAdapter adapter = new ArrayAdapter<String>(getContext(),
                R.layout.activity_listview, text);
        listView.setAdapter(adapter);
    }
else
    {
        if(((MainActivity)getActivity()).id !=null)
        {
            id=((MainActivity)getActivity()).id;
            ((MainActivity)getActivity()).id=null;
        }
        else {
            no = ((MainActivity) getActivity()).movieno;
            id = ((MainActivity) getActivity()).movieList.get(no).getId();
```

```
        }
        new JSONP3().execute(id);
    }

    toolbar.setNavigationOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            if(getActivity()!=null) {
                ((MainActivity) getActivity()).onBackPressed();
            }
            else {
                drawer.openDrawer(GravityCompat.START);
            }
        }
    });
    return v;
}
```

MapsActivity and MapsFragment- shows various places on the maps and navigation feature is provided too.

```
private void build_retrofit_and_get_response(String type) {

    String url = "https://maps.googleapis.com/maps/";
    OkHttpClient client = new OkHttpClient();
    client.setConnectTimeout(10, TimeUnit.SECONDS);
    client.setReadTimeout(30, TimeUnit.SECONDS);
    Retrofit retrofit = new Retrofit.Builder()
            .baseUrl(url)
            .client(client)
            .addConverterFactory(GsonConverterFactory.create())
            .build();

    RetrofitMaps2 service = retrofit.create(RetrofitMaps2.class);

    Call<Example> call = service.getDistanceDuration("imperial", origin.latitude + "," +
origin.longitude,dest.latitude + "," + dest.longitude, type);

    call.enqueue(new Callback<Example>() {
        @Override
        public void onResponse(Response<Example> response, Retrofit retrofit) {

            try {
                //Remove previous line from map
                if (line != null) {
                    line.remove();
                    ShowDistanceDuration.setText("Duration: Not Available");
                    ShowDuration.setText("Distance: Not Available");
                }
                // This loop will go through all the results and add marker on each location.
                for (int i = 0; i < response.body().getRoutes().size(); i++) {
                    String distance =
response.body().getRoutes().get(i).getLegs().get(i).getDistance().getText();
                    String time = response.body().getRoutes().get(i).getLegs().get(i).getDuration().getText();
                    String via = response.body().getRoutes().get(i).getLegs().get(i).getDuration().getText();
```

```
                ShowDistanceDuration.setText("Duration: " + time);
                ShowDuration.setText("Distance: "+distance);
                String encodedString =
response.body().getRoutes().get(0).getOverviewPolyline().getPoints();
                List<LatLng> list = decodePoly(encodedString);
                line = mMap.addPolyline(new PolylineOptions()
                        .addAll(list)
                        .width(9)
                        .color(Color.RED)
                        .geodesic(true)
                );
            }
        } catch (Exception e) {
            Log.d("onResponse", "There is an error");
            e.printStackTrace();
        }
    }

    @Override
    public void onFailure(Throwable t) {
        Log.d("onFailure", t.toString());
    }
});

}


@Override
public void onLocationChanged(Location location) {

    Log.d("onLocationChanged", "entered");

    mLastLocation = location;
    if (mCurrLocationMarker != null) {
        mCurrLocationMarker.remove();
    }
    //Place current location marker
    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");

    // Adding colour to the marker

markerOptions.icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_GREEN));

    // Adding Marker to the Map
    mCurrLocationMarker = mMap.addMarker(markerOptions);

    //move map camera
   // mMap.moveCamera(CameraUpdateFactory.newLatLng(latLng));
    //mMap.animateCamera(CameraUpdateFactory.zoomTo(11));

    Log.d("onLocationChanged", String.format("latitude:%.3f longitude:%.3f", latitude, longitude));
```

```
    Log.d("onLocationChanged", "Exit");
    if(flag) {

        int no=getArguments().getInt("no");
        String s = null;
        if(no==0) {

            s = "restaurant";
        }
        else if(no==1)
        {
            s="store";
        }
        else if(no==2)
        {
            s="movie_theater";
        }
        else if(no==3)
        {
            s="museum";
        }
        else if(no==4)
        {
            s="lodging";
        }
        else if(no==5)
        {
            s="gas_station";
        }
        else if(no==6)
        {
            s="atm";
        }
        Toast.makeText(getActivity(),s,Toast.LENGTH_LONG).show();

        build_retrofit_and_get_response(s);
    }

}
```