http://210.16.103.166:5000/documentation#/
https://marvelapp.com/8g806he/screen/42581219
P764UTvzJKds1J&&^&x8glbTXcEEJSUilGqpxCcmnx

https://github.com/Darkpingouin/To-Do-List-Android-App

Way of calling permission:

```
String[] permissions = new String[]{
        Manifest.permission.ACCESS_FINE_LOCATION,
        Manifest.permission.ACCESS_COARSE_LOCATION,
        Manifest.permission.READ_EXTERNAL_STORAGE,
        Manifest.permission.WRITE_EXTERNAL_STORAGE,
};



private boolean checkPermissions() {

    int result;
    List<String> listPermissionsNeeded = new ArrayList<>();
    for (String p : permissions) {
        result = ContextCompat.checkSelfPermission(this, p);
        if (result != PackageManager.PERMISSION_GRANTED) {
            listPermissionsNeeded.add(p);
        }
    }

    if (!listPermissionsNeeded.isEmpty()) {
        ActivityCompat.requestPermissions(this, listPermissionsNeeded.toArray(new
String[listPermissionsNeeded.size()]), 100);
        return false;
    }
    return true;
}



package app.toobusy.provider.utils;

import android.annotation.SuppressLint;
import android.app.Activity;
```

```
import android.app.Dialog;
import android.content.Context;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.graphics.Point;
import android.graphics.PorterDuff;
import android.graphics.PorterDuffXfermode;
import android.graphics.Rect;
import android.graphics.RectF;
import android.graphics.drawable.BitmapDrawable;
import android.graphics.drawable.ColorDrawable;
import android.graphics.drawable.Drawable;
import android.media.ExifInterface;
import android.os.Build;
import android.provider.Settings;
import android.support.v4.content.ContextCompat;
import android.support.v4.widget.SwipeRefreshLayout;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.text.Html;
import android.text.TextUtils;
import android.util.Base64;
import android.util.DisplayMetrics;
import android.util.Log;
import android.view.Display;
import android.view.View;
import android.view.ViewGroup;
import android.view.Window;
import android.view.WindowManager;
import android.view.inputmethod.InputMethodManager;
import android.widget.Button;
import android.widget.EditText;
import android.widget.FrameLayout;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.TextView;

import com.google.android.gms.maps.model.LatLng;
import com.squareup.picasso.Picasso;
```

```
import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.io.UnsupportedEncodingException;
import java.net.HttpURLConnection;
import java.net.URL;
import java.net.URLDecoder;
import java.net.URLEncoder;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import java.util.List;
import java.util.TimeZone;

import app.toobusy.provider.R;
import app.toobusy.provider.models.NavDrawerItem;

import static com.bumptech.glide.load.resource.bitmap.TransformationUtils.rotateImage;


public class Func {

    private static final int INDEX_NOT_FOUND = -1;

    public static void hideKeyboard(View v, Context ctx) {
        if (v != null) {
            v.clearFocus();
            InputMethodManager keyboard = (InputMethodManager)
ctx.getSystemService(Context.INPUT_METHOD_SERVICE);
            keyboard.hideSoftInputFromWindow(v.getWindowToken(), 0);
        }
    }

    public static boolean isValidEmail(String email) {
        return !TextUtils.isEmpty(email) &&
android.util.Patterns.EMAIL_ADDRESS.matcher(email).matches();
    }
```

```
    public static boolean ValidateEditText(String text) {
        return !TextUtils.isEmpty(text) && text.trim().length() > 0;
    }

    public static boolean ValidatePhone (String phone) {
        return phone.trim().length() >= 10;
    }

    public static int editSize(EditText edit) {
        return edit.getText().toString().trim().length();
    }

    /**
     * Set title to title bar
     *
     * @param Title    title to be set
     * @param ctx      current context
     * @param mToolbar instance of toolbar
     * @param value    value
     */
    public static void set_title_to_actionbar(String Title, Context ctx, Toolbar mToolbar, Boolean
value) {
        mToolbar.setNavigationIcon(R.drawable.ic_arrow_back_black_24dp);
        ((AppCompatActivity) ctx).setSupportActionBar(mToolbar);
        ((AppCompatActivity) ctx).getSupportActionBar().setDisplayShowHomeEnabled(value);
        ((AppCompatActivity) ctx).getSupportActionBar().setDisplayUseLogoEnabled(value);
        ((AppCompatActivity) ctx).getSupportActionBar().setDisplayHomeAsUpEnabled(value);
        ((AppCompatActivity) ctx).getSupportActionBar().setTitle("");

        TextView mTitle = mToolbar.findViewById(R.id.title_tv);

        mTitle.setText(Title);
    }

    public static String DeviceId(Context ctx) {
        return Settings.Secure.getString(ctx.getContentResolver(), Settings.Secure.ANDROID_ID);
    }

    public static double distance(double lat1, double lon1, double lat2, double lon2) {
        double theta = lon1 - lon2;
        double dist = Math.sin(deg2rad(lat1))
                * Math.sin(deg2rad(lat2))
                + Math.cos(deg2rad(lat1))
```

```
            * Math.cos(deg2rad(lat2))
            * Math.cos(deg2rad(theta));
    dist = Math.acos(dist);
    dist = rad2deg(dist);
    dist = dist * 60 * 1.1515 * 1000;
    return (dist);
}

public static  List<LatLng> decodePoly(String encoded) {

    List<LatLng> poly = new ArrayList<LatLng>();
    int index = 0, len = encoded.length();
    int lat = 0, lng = 0;

    while (index < len) {
        int b, shift = 0, result = 0;
        do {
            b = encoded.charAt(index++) - 63;
            result |= (b & 0x1f) << shift;
            shift += 5;
        } while (b >= 0x20);
        int dlat = ((result & 1) != 0 ? ~(result >> 1) : (result >> 1));
        lat += dlat;

        shift = 0;
        result = 0;
        do {
            b = encoded.charAt(index++) - 63;
            result |= (b & 0x1f) << shift;
            shift += 5;
        } while (b >= 0x20);
        int dlng = ((result & 1) != 0 ? ~(result >> 1) : (result >> 1));
        lng += dlng;

        LatLng p = new LatLng((((double) lat / 1E5)), (((double) lng / 1E5)));
        poly.add(p);
        Log.e("LatLng p", "is " + p);

    }
    return poly;
}
```

```
    public static double deg2rad(double deg) {
        return (deg * Math.PI / 180.0);
    }

    public static double rad2deg(double rad) {
        return (rad * 180.0 / Math.PI);
    }


    public static Dialog TwoButtonDialog(final Context ctx, String message, View.OnClickListener
onClickListener) {
        final Dialog MessagePopup = new Dialog(ctx);
        MessagePopup.requestWindowFeature(Window.FEATURE_NO_TITLE);
        MessagePopup.getWindow().setBackgroundDrawable(new
ColorDrawable(Color.TRANSPARENT));
        MessagePopup.setContentView(R.layout.dialog_two_button);
        MessagePopup.setCancelable(true);
        MessagePopup.setCanceledOnTouchOutside(true);
        MessagePopup.show();
        ((TextView) MessagePopup.findViewById(R.id.message)).setText(message);
        ((Button) MessagePopup.findViewById(R.id.btn_yes)).setOnClickListener(onClickListener);
        ((Button) MessagePopup.findViewById(R.id.btn_no)).setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View v) {
                MessagePopup.dismiss();
            }
        });
        return MessagePopup;
    }

    public static Dialog OneButtonDialog(final Context ctx, String message, View.OnClickListener
onClickListener, String type) {
        final Dialog MessagePopup = new Dialog(ctx);
        MessagePopup.requestWindowFeature(Window.FEATURE_NO_TITLE);
        MessagePopup.getWindow().setBackgroundDrawable(new
ColorDrawable(Color.TRANSPARENT));
        MessagePopup.setContentView(R.layout.dialog_one_button);
        MessagePopup.setCancelable(true);
        MessagePopup.setCanceledOnTouchOutside(true);
        MessagePopup.show();

        if (type.equalsIgnoreCase("Dashboard")) {
```

```
            ((ImageView)
MessagePopup.findViewById(R.id.img_close)).setVisibility(View.INVISIBLE);
        }

        ((TextView)
MessagePopup.findViewById(R.id.txt_message)).setText(Html.fromHtml(message));
        ((TextView) MessagePopup.findViewById(R.id.btn_submit)).setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View v) {
                MessagePopup.dismiss();
            }
        });

        ((ImageView) MessagePopup.findViewById(R.id.img_close)).setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View v) {
                MessagePopup.dismiss();
            }
        });

        return MessagePopup;
    }


    public static ArrayList<NavDrawerItem> drawer_items(Context ctx) {
        ArrayList<NavDrawerItem> items = new ArrayList<>();
        items.add(new NavDrawerItem(ctx.getString(R.string.home), false));
        items.add(new NavDrawerItem(ctx.getString(R.string.profile), false));
        items.add(new NavDrawerItem(ctx.getString(R.string.notification), false));
        items.add(new NavDrawerItem(ctx.getString(R.string.order), false));
        items.add(new NavDrawerItem(ctx.getString(R.string.help), false));
        items.add(new NavDrawerItem(ctx.getString(R.string.logout), false));
        return items;
    }

    public static Bitmap checkImageRotaion(String filePath) {

        ExifInterface ei = null;
        try {
            ei = new ExifInterface(filePath);
        } catch (IOException e) {
```

```
            e.printStackTrace();
        }
        int orientation = ei.getAttributeInt(ExifInterface.TAG_ORIENTATION,
                ExifInterface.ORIENTATION_UNDEFINED);

        BitmapFactory.Options options = new BitmapFactory.Options();
        options.inPreferredConfig = Bitmap.Config.ARGB_8888;
        Bitmap bitmap = BitmapFactory.decodeFile(filePath, options);

        switch (orientation) {

            case ExifInterface.ORIENTATION_ROTATE_90:
                bitmap = rotateImage(bitmap, 90);
                break;

            case ExifInterface.ORIENTATION_ROTATE_180:
                bitmap = rotateImage(bitmap, 180);
                break;

            case ExifInterface.ORIENTATION_ROTATE_270:
                bitmap = rotateImage(bitmap, 270);
                break;

            case ExifInterface.ORIENTATION_NORMAL:

            default:
                break;

        }
        Log.e("rotation", "is " + orientation + "\n" + bitmap);

        return bitmap;

    }


    //=====get width of the screen =====//
    @SuppressLint("NewApi")
    public static int getWidth(Context mContext) {
        int width = 0;
        WindowManager wm = (WindowManager)
mContext.getSystemService(Context.WINDOW_SERVICE);
        Display display = wm.getDefaultDisplay();
```

```
        if (Build.VERSION.SDK_INT > 12) {
            Point size = new Point();
            display.getSize(size);
            width = size.x;
        } else {
            width = display.getWidth(); // deprecated
        }
        return width;
    }

    // imageLayoutHeightandWidth
    public static int[] getImageHeightAndWidthForProFileImageHomsecreen(Activity activity) {
        int imageHeightAndWidth[] = new int[2];
        int screenHeight = getHeight(activity);
        int screenWidth = getWidth(activity);
        int imagehiegth;
        int imagewidth;
        if ((screenHeight <= 500 && screenHeight >= 480) && (screenWidth <= 340 &&
screenWidth >= 300)) {
            imagehiegth = 200;
            imagewidth = 200;
            imageHeightAndWidth[0] = imagehiegth;
            imageHeightAndWidth[1] = imagewidth;
        } else if ((screenHeight <= 400 && screenHeight >= 300) && (screenWidth <= 240 &&
screenWidth >= 220)) {
            imagehiegth = 150;
            imagewidth = 150;
            imageHeightAndWidth[0] = imagehiegth;
            imageHeightAndWidth[1] = imagewidth;
        } else if ((screenHeight <= 840 && screenHeight >= 780) && (screenWidth <= 500 &&
screenWidth >= 440)) {
            imagehiegth = 220;
            imagewidth = 220;
            imageHeightAndWidth[0] = imagehiegth;
            imageHeightAndWidth[1] = imagewidth;
        } else if ((screenHeight <= 1280 && screenHeight >= 840) && (screenWidth <= 720 &&
screenWidth >= 500)) {
            imagehiegth = 250;
            imagewidth = 250;
            imageHeightAndWidth[0] = imagehiegth;
            imageHeightAndWidth[1] = imagewidth;
        } else {
            imagehiegth = 250;
```

```
            imagewidth = 250;
            imageHeightAndWidth[0] = imagehiegth;
            imageHeightAndWidth[1] = imagewidth;
        }
        return imageHeightAndWidth;
    }

    public static LinearLayout.LayoutParams setViewRelativeHeight(Activity activity) {
        int screenHeight = getHeight(activity);
        int screenWidth = getWidth(activity);
        LinearLayout.LayoutParams layoutParams = new
LinearLayout.LayoutParams(ViewGroup.LayoutParams.MATCH_PARENT,
                ViewGroup.LayoutParams.MATCH_PARENT);
        layoutParams.width = screenWidth;
        layoutParams.height = screenHeight / 2;

        return layoutParams;
    }

    public static String getBase64String(Bitmap bitmap) {
        ByteArrayOutputStream baos = new ByteArrayOutputStream();
        bitmap.compress(Bitmap.CompressFormat.JPEG, 100, baos);
        byte[] b = baos.toByteArray();
        String temp = null;
        try {
            System.gc();
            temp = Base64.encodeToString(b, Base64.DEFAULT);
        } catch (Exception e) {
            e.printStackTrace();
        } catch (OutOfMemoryError e) {
            baos = new ByteArrayOutputStream();
            bitmap.compress(Bitmap.CompressFormat.JPEG, 50, baos);
            b = baos.toByteArray();
            temp = Base64.encodeToString(b, Base64.DEFAULT);
            Log.e("EWN", "Out of memory error catched");
        }
        return temp;
    }

    public static int dpToPx(Context context, int dp) {
        int px = Math.round(dp * getPixelScaleFactor(context));
        return px;
    }
```

```
    ///// create file from bitmap /////
    public static File bitmapToFile(Bitmap bitmap, String name, Context ctx) {
        File filesDir = ctx.getFilesDir();
        File imageFile = new File(filesDir, name + ".jpg");

        OutputStream os;
        try {
            os = new FileOutputStream(imageFile);
            bitmap.compress(Bitmap.CompressFormat.JPEG, 100, os);
            os.flush();
            os.close();
        } catch (Exception e) {
            Log.e("BitmapToFile Error", "Error writing bitmap", e);
        }
        return imageFile;
    }

    public static Bitmap getBitmapFromURL(String strURL) {
        try {
            URL url = new URL(strURL);
            HttpURLConnection connection = (HttpURLConnection) url.openConnection();
            connection.setDoInput(true);
            connection.connect();
            InputStream input = connection.getInputStream();
            Bitmap myBitmap = BitmapFactory.decodeStream(input);
            return myBitmap;
        } catch (IOException e) {
            e.printStackTrace();
            return null;
        }
    }

    public static Bitmap drawableToBitmap(Drawable drawable) {
        if (drawable instanceof BitmapDrawable) {
            return ((BitmapDrawable) drawable).getBitmap();
        }

        final int width = !drawable.getBounds().isEmpty() ? drawable.getBounds().width() :
drawable.getIntrinsicWidth();
        final int height = !drawable.getBounds().isEmpty() ? drawable.getBounds().height() :
drawable.getIntrinsicHeight();
```

```
        final Bitmap bitmap = Bitmap.createBitmap(width <= 0 ? 1 : width, height <= 0 ? 1 : height,
Bitmap.Config.ARGB_8888);
        Canvas canvas = new Canvas(bitmap);
        drawable.setBounds(0, 0, canvas.getWidth(), canvas.getHeight());
        drawable.draw(canvas);

        return bitmap;
    }

    private static float getPixelScaleFactor(Context context) {
        DisplayMetrics displayMetrics = context.getResources().getDisplayMetrics();
        return (displayMetrics.xdpi / DisplayMetrics.DENSITY_DEFAULT);
    }

    public static String convertDateFormat(String date, int type) {
        String converted = "";
        SimpleDateFormat spf = new SimpleDateFormat(type != 6 ? "yyyy-MM-dd" :
"MM-dd-yyyy");
        Date newDate = null;
        try {
            newDate = spf.parse(date);
        } catch (ParseException e) {
            e.printStackTrace();
        }
        SimpleDateFormat spf2 = new SimpleDateFormat("dd MMM");
        SimpleDateFormat spf3 = new SimpleDateFormat("MMMM dd, yyyy");
        SimpleDateFormat spf4 = new SimpleDateFormat("MMM yyyy");
        SimpleDateFormat spf5 = new SimpleDateFormat("MM-dd-yyyy");
        SimpleDateFormat spf6 = new SimpleDateFormat("yyyy-MM-dd");
        SimpleDateFormat spf7 = new SimpleDateFormat("MMM dd");

        if (type == 0)
            converted = spf2.format(newDate);
        else if (type == 2)
            converted = spf4.format(newDate);
        else if (type == 5)
            converted = spf5.format(newDate);
        else if (type == 6)
            converted = spf6.format(newDate);
        else if (type == 7)
            converted = spf7.format(newDate);
        else
            converted = spf3.format(newDate);
```

```
    return converted;
}

public static boolean Calculate_beforedate(String DateTime, String StartDate) {
    boolean flag = false;
    try {
        SimpleDateFormat formatter = new SimpleDateFormat("MM-dd-yyyy HH:mm:00");
        Date current_date = formatter.parse(StartDate);
        Date selected_date = formatter.parse(DateTime);

        System.out.println(current_date + "  " + selected_date);

        if (current_date.after(selected_date))
            flag = true;
        else
            flag = false;

    } catch (Exception e) {
        e.printStackTrace();
    }
    return flag;
}


public static String convertDateFormatWithTime(String date) {
    String converted = "";
    SimpleDateFormat spf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
    spf.setTimeZone(TimeZone.getTimeZone("UTC"));
    Date newDate = null;
    try {
        newDate = spf.parse(date);
    } catch (ParseException e) {
        e.printStackTrace();
    }
    SimpleDateFormat spf2 = new SimpleDateFormat("dd MMM hh:mmaa");
    spf2.setTimeZone(TimeZone.getDefault());
    converted = spf2.format(newDate);
    return converted;
}

public static String getCurrentDateTime(int format) {
    String time = "";
```

```
        SimpleDateFormat sdf = new SimpleDateFormat(format == 1 ? "yyyy-MM-dd HH:mm:ss" :
"yyyy-MM-dd");
        sdf.setTimeZone(TimeZone.getTimeZone("UTC"));
        try {
            Date current_date = sdf.parse(sdf.format(Calendar.getInstance().getTime()));
            time = sdf.format(current_date);
        } catch (ParseException e) {
            e.printStackTrace();
            time = "N/A";
        }

        return time;
    }

    public static String getDateDiffString(String dateOne, int type) {

        String dateString = dateOne;
        Date current_date = null, selected_date = null;
        try {
            SimpleDateFormat sdf1 = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
            sdf1.setTimeZone(TimeZone.getTimeZone("UTC"));

            Date date = sdf1.parse(dateOne);

            sdf1.setTimeZone(TimeZone.getDefault());
            Date crdate = new SimpleDateFormat("yyyy-MM-dd
HH:mm:ss").parse(sdf1.format(date));
            String dd = sdf1.format(crdate);

            String[] datetime = dd.split(" ");
            current_date = sdf1.parse(sdf1.format(Calendar.getInstance().getTime()));
            selected_date = sdf1.parse(dd);

            long timeOne = selected_date.getTime();
            long timeTwo = current_date.getTime();
            long diff = timeTwo - timeOne;
            //long diffMinutes = diff / (60 * 1000) % 60;
            //long diffHours = diff / (60 * 60 * 1000) % 24;
            long diffDays = diff / (24 * 60 * 60 * 1000);

            SimpleDateFormat formatter2 = new SimpleDateFormat("yyyy-MM-dd");
            Date current_date2 =
formatter2.parse(formatter2.format(Calendar.getInstance().getTime()));
```

```
            Date selected_date2 = formatter2.parse(datetime[0]);

            SimpleDateFormat sdf = new SimpleDateFormat("HH:mm:ss");
            SimpleDateFormat sdfs = new SimpleDateFormat("hh:mmaa");
            dateString = sdfs.format(sdf.parse(datetime[1]));

            if (selected_date2.equals(current_date2)) {
                if (type == 1)
                    dateString = dateString.replace("am", "AM").replace("pm", "PM");
                else
                    dateString = "Today at " + dateString.replace("am", "AM").replace("pm", "PM");

                return dateString;

            } else {
                if (diffDays <= 1) {
                    if (type == 1)
                        dateString = "Yesterday";
                    else
                        dateString = "Yesterday at " + dateString.replace("am", "AM").replace("pm",
"PM");
                } else {
                    SimpleDateFormat formatter3 = new SimpleDateFormat("MMM dd");
                    if (type == 1)
                        dateString = formatter3.format(selected_date2);
                    else
                        dateString = formatter3.format(selected_date2) + " at " +
dateString.replace("am", "AM").replace("pm", "PM");
                }

            }
        } catch (Exception e) {
            e.printStackTrace();
        }

        return dateString;
    }

    public static String dateDifferenceInDays(String startDate, String endDate) {
        String dayDiff = "";
        try {
            SimpleDateFormat formatter = new SimpleDateFormat("MM-dd-yyyy");
            startDate = formatter.format(formatter.parse(startDate));
```

```
        Date d1 = formatter.parse(startDate);
        Date d2 = formatter.parse(endDate);
        if (d2.equals(d1)) {
            dayDiff = "0";
        } else {
            long timeOne = d1.getTime();
            long timeTwo = d2.getTime();
            long diff = timeTwo - timeOne;
            long diffDays = diff / (24 * 60 * 60 * 1000);
            dayDiff = diffDays + "";
            Log.e("dayDiff", dayDiff);
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
    return dayDiff;
}

public static String getTimeLeftString(String dateOne) {
    String dateString = "";
    Date current_date = null, selected_date = null;
    try {
        SimpleDateFormat tzFormat = new SimpleDateFormat("yyyy-MM-dd");
        tzFormat.setTimeZone(TimeZone.getTimeZone("UTC"));
        String UTCDate = tzFormat.format(tzFormat.parse(dateOne));
        Date date = tzFormat.parse(UTCDate);
        tzFormat.setTimeZone(TimeZone.getDefault());
        Date crdate = new SimpleDateFormat("yyyy-MM-dd").parse(tzFormat.format(date));
        String dd = tzFormat.format(crdate);

        String[] datetime = dd.split(" ");
        current_date = tzFormat.parse(tzFormat.format(Calendar.getInstance().getTime()));
        selected_date = tzFormat.parse(dd);

        long timeOne = current_date.getTime();
        long timeTwo = selected_date.getTime();
        long diff = timeTwo - timeOne;
        long diffDays = diff / (24 * 60 * 60 * 1000);

        int year = (int) (diffDays / 365);
        int rest = (int) (diffDays % 365);
        int month = rest / 30;
        rest = rest % 30;
```

```
            int weeks = rest / 7;

            SimpleDateFormat formatter2 = new SimpleDateFormat("yyyy-MM-dd");
            Date current_date2 =
formatter2.parse(formatter2.format(Calendar.getInstance().getTime()));
            Date selected_date2 = formatter2.parse(datetime[0]);

            if (current_date2.equals(selected_date2)) {
               dateString = "In Progress";
            } else if (current_date2.after(selected_date2)) {
               dateString = "In Progress";
            } else {
               if (diffDays >= 365)
                  dateString = year > 1 ? year + " Years" : year + " Year";
               else if (month > 0)
                  dateString = month > 1 ? month + " Months" : month + " Month";
               else if (diffDays % 7 == 0 && diffDays != 0)
                  dateString = weeks > 1 ? weeks + " Weeks" : weeks + " Week";
               else if (diffDays > 0 && diffDays <= 30)
                  dateString = diffDays > 1 ? diffDays + " days" : diffDays + " day";
               else if (diffDays == 0)
                  dateString = "1 day";
               else
                  return "N/A";
            }
         } catch (Exception e) {
            e.printStackTrace();
         }
         return dateString;
      }

      public static String dateDifferenceInDays(String DateTime) {
         try {
            SimpleDateFormat formatter = new SimpleDateFormat("yyyy-MM-dd");
            DateTime = formatter.format(formatter.parse(DateTime));
            Date current_date = formatter.parse(formatter.format(Calendar.getInstance().getTime()));
            Date selected_date = formatter.parse(DateTime);
            if (selected_date.equals(current_date)) {
               DateTime = "0";
            } else {
               long timeOne = current_date.getTime();
               long timeTwo = selected_date.getTime();
               long diff = timeTwo - timeOne;
```

```
            long diffDays = diff / (24 * 60 * 60 * 1000);

            DateTime = diffDays + "";

          }
       } catch (Exception e) {
          e.printStackTrace();
       }
       return DateTime;
    }

    public static String deviceTime(String dateOne, int type) {
       String dd = "";
       try {
          SimpleDateFormat sdf1 = new SimpleDateFormat(type == 0 ? "HH:mm" :
"yyyy-MM-dd");
          sdf1.setTimeZone(TimeZone.getTimeZone("UTC"));
          String dateOne1 = sdf1.format(sdf1.parse(dateOne));
          Date date = sdf1.parse(dateOne1);
          sdf1.setTimeZone(TimeZone.getDefault());

          Date crdate = new SimpleDateFormat(type == 0 ? "HH:mm" :
"yyyy-MM-dd").parse(sdf1.format(date));
          dd = sdf1.format(crdate);

       } catch (Exception e) {
          e.printStackTrace();
       }
       return dd;
    }

    public static String deviceTimetoUTC(String time, int type) {
       SimpleDateFormat sdf1;
       if (type == 0)
          sdf1 = new SimpleDateFormat("HH:mm");
       else
          sdf1 = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");

       String date = null;
       try {
          Date date1 = sdf1.parse(time);
```

```
            sdf1.setTimeZone(TimeZone.getTimeZone("UTC"));
            date = sdf1.format(date1);
        } catch (Exception e) {
            e.printStackTrace();
        }
        return date;
    }

    public static String returnTime(String DateTime) {
        Calendar calendar = Calendar.getInstance();
        try {
            Date date = new SimpleDateFormat("HH:mm:ss").parse(DateTime);
            calendar.setTime(date);
        } catch (Exception e) {
            e.printStackTrace();
        }
        String hour = calendar.get(Calendar.HOUR_OF_DAY) + "";
        String minute = calendar.get(Calendar.MINUTE) + "";
        String second = calendar.get(Calendar.SECOND) + "";
        String am_pm = "";
        int h = Integer.parseInt(hour);
        if (h >= 12) {
            am_pm = "PM";
            if (h > 12) {
                h = h - 12;
                hour = h + "";
            }
        } else {
            am_pm = "AM";
        }
        if (hour.length() == 1) {
            hour = "0" + hour;
        }
        if (minute.length() == 1) {
            minute = "0" + minute;
        }
        return hour + ":" + minute + " " + am_pm;
    }

    ///// create bitmap  circular in shape /////
    public static Bitmap getCircleBitmap(Bitmap bitmap) {
        Bitmap output = null;
        try {
```

```
            output = Bitmap.createBitmap(100, 100, Bitmap.Config.ARGB_8888);

            final Canvas canvas = new Canvas(output);

            final int color = Color.RED;
            final Paint paint = new Paint();
            final Rect rect = new Rect(0, 0, bitmap.getWidth(), bitmap.getHeight());
            final RectF rectF = new RectF(rect);

            paint.setAntiAlias(true);
            canvas.drawARGB(0, 0, 0, 0);
            paint.setColor(color);
            canvas.drawOval(rectF, paint);
            paint.setXfermode(new PorterDuffXfermode(PorterDuff.Mode.SRC_IN));
            canvas.drawBitmap(bitmap, rect, rect, paint);
        } catch (Exception e) {
            e.printStackTrace();
        }
        return output;
    }

    @SuppressLint("NewApi")
    public static int getHeight(Context mContext) {
        int height = 0;
        WindowManager wm = (WindowManager) mContext
                .getSystemService(Context.WINDOW_SERVICE);
        Display display = wm.getDefaultDisplay();
        if (Build.VERSION.SDK_INT > 12) {
            Point size = new Point();
            display.getSize(size);
            height = size.y;
        } else {
            height = display.getHeight(); // deprecated
        }
        return height;
    }

    public static FrameLayout.LayoutParams setViewRelativeHeight(Activity activity, int half) {
        int screenHeight = getHeight(activity);
        int screenWidth = getWidth(activity);
        FrameLayout.LayoutParams layoutParams = new
FrameLayout.LayoutParams(ViewGroup.LayoutParams.MATCH_PARENT,
                ViewGroup.LayoutParams.MATCH_PARENT);
```

```
        layoutParams.width = screenWidth;
        if (half == 1)
            layoutParams.height = screenHeight / 2;
        else
            layoutParams.height = screenHeight;

        return layoutParams;
    }

    public static LinearLayout.LayoutParams setViewLinearHeight(Activity activity, int half) {
        int screenHeight = getHeight(activity);
        int screenWidth = getWidth(activity);
        LinearLayout.LayoutParams layoutParams = new
LinearLayout.LayoutParams(ViewGroup.LayoutParams.MATCH_PARENT,
                ViewGroup.LayoutParams.MATCH_PARENT);
        layoutParams.width = screenWidth;
        if (half == 1)
            layoutParams.height = screenHeight / 2;
        else
            layoutParams.height = screenHeight;

        return layoutParams;
    }

    public static void setCircularImageView(ImageView imageView, String imagepath, Context
context, int errorImage) {
        if (imagepath == null || imagepath.trim().length() <= 0 || imagepath.isEmpty())
            imagepath = "NoImage";
        Picasso.with(context).load(imagepath).placeholder(errorImage).error(errorImage)
                .transform(new CircleTransform()).fit().into(imageView);
    }

    public static void setSquareImageView(ImageView imageView, String imagepath, Context
context, int errorImage) {
        if (imagepath == null || imagepath.trim().length() <= 0 || imagepath.isEmpty())
            imagepath = "NoImage";

Picasso.with(context).load(imagepath).placeholder(errorImage).error(errorImage).into(imageVie
w);
    }

    public static void setSquareImageViewResize(ImageView imageView, String imagepath,
Context context, int errorImage) {
```

```
        if (imagepath == null || imagepath.trim().length() <= 0 || imagepath.isEmpty())
            imagepath = "NoImage";

Picasso.with(context).load(imagepath).placeholder(errorImage).error(errorImage).into(imageVie
w);
    }

    public static int getStatusBarHeight(Activity activity) {
        int result = 0;
        int resourceId = activity.getResources().getIdentifier("status_bar_height", "dimen",
"android");
        if (resourceId > 0) {
            result = activity.getResources().getDimensionPixelSize(resourceId);
        }
        return result;
    }

    public static void setStatusBarTranslucent(boolean makeTranslucent, Activity activity) {
        if (makeTranslucent) {

activity.getWindow().addFlags(WindowManager.LayoutParams.FLAG_TRANSLUCENT_STATU
S);
        } else {

activity.getWindow().clearFlags(WindowManager.LayoutParams.FLAG_TRANSLUCENT_STAT
US);
        }
    }

    public static void setSwipeRefreshLayout(SwipeRefreshLayout swipeRefreshLayout, Context
context) {
        swipeRefreshLayout.setColorSchemeColors(ContextCompat.getColor(context,
R.color.white));

swipeRefreshLayout.setProgressBackgroundColorSchemeColor(ContextCompat.getColor(cont
ext, R.color.greenColor));
    }

    public static boolean checkDate(String startDate) {
        boolean flag = false;
        try {
            SimpleDateFormat formatter = new SimpleDateFormat("yyyy-MM-dd");
            Date current_date = formatter.parse(formatter.format(Calendar.getInstance().getTime()));
```

```
        Date selected_date = formatter.parse(startDate);

        //equals() returns true if both the dates are equal
        if (current_date.equals(selected_date)) {
            flag = true;
        }
        // after() will return true if and only if date1 is after date 2
        if (current_date.after(selected_date)) {
            flag = true;
        }

    } catch (Exception e) {
        e.printStackTrace();
    }
    return flag;
}

public static boolean Calculate_afterdate(String DateTime, String StartDate) {
    boolean flag = false;
    try {
        SimpleDateFormat formatter = new SimpleDateFormat("yyyy-MM-dd");
        Date current_date = formatter.parse(StartDate);
        Date selected_date = formatter.parse(DateTime);

        System.out.println(current_date + "  " + selected_date);

        if (selected_date.equals(current_date)) {
            flag = true;
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
    return flag;
}

public static String convertDate(String strDob) {
    String date = null;
    SimpleDateFormat df1 = new SimpleDateFormat("dd MMM, yyyy");
    SimpleDateFormat df = new SimpleDateFormat("yyyy-MM-dd");
    try {
        date = df.format(df1.parse(strDob));
    } catch (Exception e) {
        e.printStackTrace();
```

```
    }
    return date;
}

public static String convertDatefromAPI(String strDob) {
    String date = null;
    SimpleDateFormat df1 = new SimpleDateFormat("yyyy-MM-dd");
    SimpleDateFormat df = new SimpleDateFormat("dd MMM, yyyy");
    try {
        date = df.format(df1.parse(strDob));
    } catch (Exception e) {
        e.printStackTrace();
    }
    return date;
}

public static String getBase64(String text) {
    byte[] data = null;
    try {
        data = text.getBytes("UTF-8");
    } catch (UnsupportedEncodingException e1) {
        e1.printStackTrace();
    }
    String base64 = Base64.encodeToString(data, Base64.DEFAULT);
    return base64;
}

public static String difference(String str1, String str2) {
    if (str1 == null) {
        return str2;
    }
    if (str2 == null) {
        return str1;
    }
    int at = indexOfDifference(str1, str2);
    if (at == INDEX_NOT_FOUND) {
        return "";
    }
    return str2.substring(at);
}

public static int indexOfDifference(CharSequence cs1, CharSequence cs2) {
    if (cs1 == cs2) {
```

```
            return INDEX_NOT_FOUND;
        }
        if (cs1 == null || cs2 == null) {
            return 0;
        }
        int i;
        for (i = 0; i < cs1.length() && i < cs2.length(); ++i) {
            if (cs1.charAt(i) != cs2.charAt(i)) {
                break;
            }
        }
        if (i < cs2.length() || i < cs1.length()) {
            return i;
        }
        return INDEX_NOT_FOUND;
    }
}
```

****

SPANNABLE STRING BUILDER

```
 void changeTextColor() {
        SpannableStringBuilder builder = new SpannableStringBuilder();

        SpannableString first = new SpannableString("Already have an account? Sign in ");
        first.setSpan(new ForegroundColorSpan(Color.parseColor("#596379")), 0, first.length(),
Spannable.SPAN_INCLUSIVE_INCLUSIVE);

        SpannableString second = new SpannableString("here ");
        second.setSpan(new UnderlineSpan(), 0, second.length(), 0);
        second.setSpan(new ForegroundColorSpan(Color.parseColor("#47cba3")), 0,
second.length(), Spannable.SPAN_INCLUSIVE_INCLUSIVE);

        builder.append(first);
        builder.append(second);

        already_account.setText(builder);

    }
```

****

HANDLE THE CUSTOM ERROR RESPONSE IN RETROFIT

```
try {
    JSONObject jsonObject = new JSONObject(response.errorBody().string());
    mView.showMessage(jsonObject.getString("message"));
} catch (IOException | JSONException e) {
    e.printStackTrace();
}
```

OPEN EXTERNAL GOOGLE MAP WITH MARKER

```
Double myLatitude = 44.433106;
    Double myLongitude = 26.103687;
    String labelLocation = "Jorgesys @ Bucharest";
    Intent intent = new Intent(Intent.ACTION_VIEW, Uri.parse("geo:<" + myLatitude  + ">,<" +
myLongitude + ">?q=<" + myLatitude  + ">,<" + myLongitude + ">(" + labelLocation + ")"));
    startActivity(intent);
```

DYNAMICALLY PARSE JSON DATA

```
private void parseResult(TypedInput data) {
    try {
        Gson gson = new GsonBuilder().create();
        TypedByteArray typedByteArray = (TypedByteArray) data;
        if (typedByteArray != null) {
            String json = new String(typedByteArray.getBytes());
            JSONObject jsonObj = new JSONObject(json);

            JSONArray dataArr = jsonObj.getJSONArray("data");
            JSONObject mainDataObj = dataArr.getJSONObject(0);

            Iterator<?> keys = mainDataObj.keys();
            while (keys.hasNext()) {
                String key = (String) keys.next();
                if (mainDataObj.get(key) instanceof JSONArray) {
                    Log.d("Key---", key); // do whatever you want with it
                    JSONArray jsonArray = (JSONArray) mainDataObj.get(key);
                    if (jsonArray != null) {
                        Type listType = new TypeToken<List<ServiceslistitemsPojo>>() {
                        }.getType();
                        subCatKeysArraylist = gson.fromJson(String.valueOf(jsonArray), listType);
                        Log.d("id---", subCatKeysArraylist.get(0).getCategory_name() + "       " +
subCatKeysArraylist.size());
                        serviceslistitemsPojos.add(subCatKeysArraylist);
                    }
```

```
                }
//                  subCatKeysArraylist.clear();
              }
            }
        } catch (JSONException e) {
            e.printStackTrace();
        }
    }
```

POJO CLASS OF DYNAMIC PARSING OF JSON DATA

```
public class ServiceslistitemsPojo implements Parcelable {
    private String category_id;
    private String category_name;
    private String name;
    private String id;
    private String price;
    private String img;
    private String status;
    int count = 0;

    public ServiceslistitemsPojo(Parcel in) {
        category_id = in.readString();
        category_name = in.readString();
        name = in.readString();
        id = in.readString();
        price = in.readString();
        img = in.readString();
        status = in.readString();
        count = in.readInt();
    }

    public static final Creator<ServiceslistitemsPojo> CREATOR = new
Creator<ServiceslistitemsPojo>() {
        @Override
        public ServiceslistitemsPojo createFromParcel(Parcel in) {
            return new ServiceslistitemsPojo(in);
        }

        @Override
        public ServiceslistitemsPojo[] newArray(int size) {
            return new ServiceslistitemsPojo[size];
```

```
    }
};

public ServiceslistitemsPojo(String category_id, String category_name, String name, String
id, String price, String img, String status) {
    this.category_id=category_id;
    this.category_name=category_name;
    this.name=name;
    this.id=id;
    this.price=price;
    this.img=img;
    this.status=status;

}

public int getCount() {
    return count;
}

public void setCount(int count) {
    this.count = count;
}

public String getCategory_id() {
    return category_id;
}

public void setCategory_id(String category_id) {
    this.category_id = category_id;
}

public String getCategory_name() {
    return category_name;
}

public void setCategory_name(String category_name) {
    this.category_name = category_name;
}

public String getName() {
    return name;
}
```

```
public void setName(String name) {
    this.name = name;
}

public String getId() {
    return id;
}

public void setId(String id) {
    this.id = id;
}

public String getPrice() {
    return price;
}

public void setPrice(String price) {
    this.price = price;
}

public String getImg() {
    return img;
}

public void setImg(String img) {
    this.img = img;
}

public String getStatus() {
    return status;
}

public void setStatus(String status) {
    this.status = status;
}

@Override
public int describeContents() {
    return 0;
}

@Override
public void writeToParcel(Parcel parcel, int i) {
```

```
        parcel.writeString(category_id);
        parcel.writeString(category_name);
        parcel.writeString(name);
        parcel.writeString(id);
        parcel.writeString(price);
        parcel.writeString(img);
        parcel.writeString(status);
        parcel.writeInt(count);

    }
}
```

DYNAMIC PARSING OF REST API'S

```
public static void checkObjectOrArray(TypedInput data, OnApiResponseListener
onApiResponseListener, String tag) {
        Gson gson = new GsonBuilder().create();
        ArrayList<DataPojo> DetaPojosList = new ArrayList<DataPojo>();
        try {

            TypedByteArray typedByteArray = (TypedByteArray) data;
            if (typedByteArray != null) {
                String json = new String(typedByteArray.getBytes());

                JSONObject jsonObj = new JSONObject(json);
                Object o = jsonObj.get("data");

                if (o instanceof JSONObject) {
                    //you have an object
                    DataPojo dataPojo = gson.fromJson(jsonObj.getJSONObject("data").toString(),
DataPojo.class);
                    DetaPojosList.add(dataPojo);
                    onApiResponseListener.onSuccess(DetaPojosList, tag);
                } else if (o instanceof JSONArray) {
                    //you have an array

                    JSONArray jsonArray = (JSONArray) jsonObj.get("data");
                    if (jsonArray != null) {


                        Type listType = new TypeToken<List<DataPojo>>() {
                        }.getType();
                        DetaPojosList = gson.fromJson(String.valueOf(jsonArray), listType);
                        onApiResponseListener.onSuccess(DetaPojosList, tag);
```

```
            }
        }

    }
} catch (Exception e) {
    e.printStackTrace();
}
}
```

MULTI LANGUAGE APP

1. Locale class

```
public class LocaleUtils {
    private static final String SELECTED_LANGUAGE = "language";

    public static Context onAttach(Context context) {
        String lang = getPersistedData(context,
Locale.getDefault().getLanguage());
        return setLocale(context, lang);
    }

    public static Context onAttach(Context context, String defaultLanguage) {
        String lang = getPersistedData(context, defaultLanguage);
        return setLocale(context, lang);
    }

    public static String getLanguage(Context context) {
        return getPersistedData(context, Locale.getDefault().getLanguage());
    }

    public static Context setLocale(Context context, String language) {
        persist(context, language);

        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.N) {
            return updateResources(context, language);
        }

        return updateResourcesLegacy(context, language);
    }

    private static String getPersistedData(Context context, String
defaultLanguage) {
        SharedPreferences preferences =
PreferenceManager.getDefaultSharedPreferences(context);
```

```
        return preferences.getString(SELECTED_LANGUAGE, defaultLanguage);
    }

    private static void persist(Context context, String language) {
        SharedPreferences preferences =
PreferenceManager.getDefaultSharedPreferences(context);
        SharedPreferences.Editor editor = preferences.edit();
        editor.putString(SELECTED_LANGUAGE, language);
        editor.apply();
    }

    @TargetApi(Build.VERSION_CODES.N)
    private static Context updateResources(Context context, String language) {
        Locale locale = new Locale(language);
        Locale.setDefault(locale);

        Configuration configuration = context.getResources().getConfiguration();
        configuration.setLocale(locale);
        configuration.setLayoutDirection(locale);

        return context.createConfigurationContext(configuration);
    }

    @SuppressWarnings("deprecation")
    private static Context updateResourcesLegacy(Context context, String
language) {
        Locale locale = new Locale(language);
        Locale.setDefault(locale);

        Resources resources = context.getResources();

        Configuration configuration = resources.getConfiguration();
        configuration.locale = locale;
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.JELLY_BEAN_MR1) {
            configuration.setLayoutDirection(locale);
        }
        resources.updateConfiguration(configuration,
resources.getDisplayMetrics());
        return context;
    }
}
```

2.Share preference

```
public class SharedHelper {
    public static SharedPreferences sharedPreferences;
```

```
    public static SharedPreferences.Editor editor;

    public static void putKey(Context context, String Key, String
Value) {
        sharedPreferences = context.getSharedPreferences("Cache",
Context.MODE_PRIVATE);
        editor = sharedPreferences.edit();
        editor.putString(Key, Value);
        editor.commit();

    }

    public static String getKey(Context contextGetKey, String
Key) {
        sharedPreferences =
contextGetKey.getSharedPreferences("Cache",
Context.MODE_PRIVATE);
        String Value = sharedPreferences.getString(Key, "");
        return Value;

    }

    public static void clearSharedPreferences(Context context)
    {
        sharedPreferences = context.getSharedPreferences("Cache",
Context.MODE_PRIVATE);
        sharedPreferences.edit().clear().commit();
    }
}
```

3. Main Activity

```
public class MainActivity extends AppCompatActivity {
    Spinner spinner;
    Locale myLocale;
    String currentLanguage = "en", currentLang;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```

```
        setContentView(R.layout.activity_main);
        currentLanguage = getIntent().getStringExtra(currentLang);
        spinner = findViewById(R.id.spinner);

        List<String> list = new ArrayList<>();
        list.add("Select language");
        list.add("English");
        list.add("Español");
        list.add("Français");
        list.add("Hindi");

        ArrayAdapter<String> adapter = new ArrayAdapter<>(this,
android.R.layout.simple_spinner_item, list);

adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
        spinner.setAdapter(adapter);

        spinner.setOnItemSelectedListener(new
AdapterView.OnItemSelectedListener() {
            @Override
            public void onItemSelected(AdapterView<?> adapterView, View view,
int position, long l) {
                switch (position) {
                    case 0:
                        break;
                    case 1:
                        setLanguage("en");
                        break;
                    case 2:
                        setLanguage("es");
                        break;
                    case 3:
                        setLanguage("fr");
                        break;
                    case 4:
                        setLanguage("hi");
                        break;
                }
            }

            @Override
            public void onNothingSelected(AdapterView<?> adapterView) {

            }
        });
    }

    private void setLanguage(String lng) {
```

```
        SharedHelper.putKey(MainActivity.this, "language", lng);
        String languageCode = SharedHelper.getKey(MainActivity.this,
"language");
        LocaleUtils.setLocale(this, languageCode);

        Intent mainIntent = new Intent(this, MainActivity.class);
        mainIntent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK |
Intent.FLAG_ACTIVITY_NEW_TASK);
        startActivity(mainIntent);
    }

    @Override
    protected void attachBaseContext(Context newBase) {
        super.attachBaseContext(LocaleUtils.onAttach(newBase));

    }
}
```

UPDATE LOCATION IN BACKGROUND SERVICE AND UPDATE THE UI WITH BROADCAST RECEIVER

```
public class LocationService extends Service {
    private GoogleApiClient mGoogleApiClient;
    private Location mLastLocation;
    private LocationRequest mLocationRequest;
    private FusedLocationProviderClient mFusedLocationClient;

    public LocationService() {
        super();
    }

    @Nullable
    @Override
    public IBinder onBind(Intent intent) {
        return null;
    }

    @Override
    public void onCreate() {
        buildGoogleApiClient();
        createLocationRequest();
        if (mGoogleApiClient != null) {
            mGoogleApiClient.connect();
        }
        super.onCreate();
    }
```

```
    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {
        // Resuming the periodic location updates
        if (mGoogleApiClient != null) {
            if (mGoogleApiClient.isConnected()) {
                startLocationUpdates();
            }
        }
        return super.onStartCommand(intent, START_FLAG_REDELIVERY, startId);
    }

    private void createLocationRequest() {
        mLocationRequest = new LocationRequest();
        mLocationRequest.setInterval(1000);
        mLocationRequest.setFastestInterval(1000);
        mLocationRequest.setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY);
        //mLocationRequest.setSmallestDisplacement(100);


    }

    private void onLocationChanged(Location location) {
        SharedPreferences settings =
PreferenceManager.getDefaultSharedPreferences(getApplicationContext());
        SharedPreferences.Editor editor = settings.edit();
        editor.putString(Constants.LAT, "" + location.getLatitude());
        editor.putString(Constants.LNG, "" + location.getLongitude());
        editor.apply();

        Intent broadCastIntent = new Intent();
        broadCastIntent.setAction("location");
        sendBroadcast(broadCastIntent);
    }

    private synchronized void buildGoogleApiClient() {
        try {
            mGoogleApiClient = new
GoogleApiClient.Builder(getApplicationContext())
                    .addConnectionCallbacks(new
GoogleApiClient.ConnectionCallbacks() {
                        @Override
                        public void onConnected(Bundle bundle) {
                            startLocationUpdates();
                        }

                        @Override
                        public void onConnectionSuspended(int i) {
```

```
                            mGoogleApiClient.connect();
                        }
                    })
                    .addOnConnectionFailedListener(new
GoogleApiClient.OnConnectionFailedListener() {
                        @Override
                        public void onConnectionFailed(ConnectionResult
connectionResult) {
                            //Toast.makeText(getApplicationContext(), "Failed",
Toast.LENGTH_LONG).show();
                        }
                    })
                    .addApi(LocationServices.API).build();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }


    private void startLocationUpdates() {
        mFusedLocationClient =
LocationServices.getFusedLocationProviderClient(this);
        if (ContextCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED
&& checkSelfPermission(Manifest.permission.ACCESS_COARSE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {
            // TODO: Consider calling
            //    Activity#requestPermissions
            // here to request the missing permissions, and then overriding
            //   public void onRequestPermissionsResult(int requestCode,
String[] permissions,
            //                                          int[] grantResults)
            // to handle the case where the user grants the permission. See the
documentation
            // for Activity#requestPermissions for more details.
            return;
        }
        mFusedLocationClient.getLastLocation()
                .addOnSuccessListener(new OnSuccessListener<Location>() {
                    @Override
                    public void onSuccess(Location location) {
                        if (location != null) {
//                          BaseActivity.
                            SharedPreferences settings =
PreferenceManager.getDefaultSharedPreferences(getApplicationContext());
                            SharedPreferences.Editor editor = settings.edit();
                            editor.putString(Constants.LAT, "" +
location.getLatitude());
```

```
                            editor.putString(Constants.LNG, "" +
location.getLongitude());
                            editor.apply();

//
PreferenceClass.setStringPreference(LocationService.this, Constants.LAT, "" +
location.getLatitude());
//
PreferenceClass.setStringPreference(LocationService.this, Constants.LNG, "" +
location.getLongitude());

                            Intent broadCastIntent = new Intent();
                            broadCastIntent.setAction("location");
                            sendBroadcast(broadCastIntent);
                        }
                        // stopSelf();
                    }
                });


        // Create LocationSettingsRequest object using location request
        LocationSettingsRequest.Builder builder = new
LocationSettingsRequest.Builder();
        builder.addLocationRequest(mLocationRequest);
        LocationSettingsRequest locationSettingsRequest = builder.build();

        // Check whether location settings are satisfied
        //
https://developers.google.com/android/reference/com/google/android/gms/location
/SettingsClient
        SettingsClient settingsClient =
LocationServices.getSettingsClient(this);
        settingsClient.checkLocationSettings(locationSettingsRequest);


        // new Google API SDK v11 uses getFusedLocationProviderClient(this)
        if (ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED
&& ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_COARSE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {
            // TODO: Consider calling
            //    ActivityCompat#requestPermissions
            // here to request the missing permissions, and then overriding
            //   public void onRequestPermissionsResult(int requestCode,
String[] permissions,
            //                                          int[] grantResults)
```

```
            // to handle the case where the user grants the permission. See the
documentation
            // for ActivityCompat#requestPermissions for more details.
            return;
        }

        mFusedLocationClient.requestLocationUpdates(mLocationRequest, new
LocationCallback() {

                    @Override
                    public void onLocationResult(LocationResult locationResult)
{
                        // do work here
                        onLocationChanged(locationResult.getLastLocation());
                    }
                },
                Looper.myLooper());


    }

    private void stopLocationUpdates() {

        mFusedLocationClient.removeLocationUpdates(new LocationCallback());
      /*  LocationServices.FusedLocationApi.removeLocationUpdates(
                mGoogleApiClient, new LocationListener() {
                    @Override
                    public void onLocationChanged(Location location) {
                        mLastLocation = location;
                        // Displaying the new location on UI

                    }
                });*/
    }


    @Override
    public boolean stopService(Intent name) {
        stopLocationUpdates();
        return super.stopService(name);
    }

    @Override
    public void onDestroy() {
        super.onDestroy();
        if (mGoogleApiClient.isConnected()) {
            mGoogleApiClient.disconnect();
        }
```

```
        }


}
```