uddhavgautam / **com_liveEarthquakesAlerts_controller_adapters_ListViewA...**

Created 2 minutes ago

Code Revisions 1

Edit Delete Star 0 Embed `<script src="https://gis` Download ZIP

com_liveEarthquakesAlerts_controller_adapters_ListViewAdapter.java Raw

```java
package com.liveEarthquakesAlerts.controller.adapters;

import android.app.Activity;
import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.TextView;

import com.liveEarthquakesAlerts.R;
import com.liveEarthquakesAlerts.model.database.EarthQuakes;

import java.util.Date;
import java.util.List;

/**
 * Created by  Uddhav Gautam  on 7.3.2016. upgautam@ualr.edu
 */
public class ListViewAdapter extends ArrayAdapter<EarthQuakes> { // just to GUI present for earthquake records in ListView

    private static LayoutInflater inflater = null;
    private List<EarthQuakes> list;
    private Activity act;

    public ListViewAdapter(Activity activity, List<EarthQuakes> datas) {
        super(activity, R.layout.list_row, datas);
```

```java
            list = datas;
            act = activity;
            inflater = (LayoutInflater) activity.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
        }

        public View getView(int position, View convertView, ViewGroup parent) {

            View vi = convertView;
            if (convertView == null)
                vi = inflater.inflate(R.layout.list_row, null);

            TextView tvLocation = (TextView) vi.findViewById(R.id.tv_location);
            TextView tvDate = (TextView) vi.findViewById(R.id.tv_date);
            TextView tvDepth = (TextView) vi.findViewById(R.id.tv_depth);
            TextView tvMag = (TextView) vi.findViewById(R.id.tv_mag);

            EarthQuakes earthQuake = list.get(position);

            tvLocation.setText(earthQuake.getLocationName());
            tvDate.setText(": " + new Date(earthQuake.getDateMilis()).toLocaleString());
            tvDepth.setText(": " + Float.toString(earthQuake.getDepth()) + " KM");
            tvMag.setText(Float.toString(earthQuake.getMagnitude()));

            float magnitude = earthQuake.getMagnitude();

            if (magnitude < 3) {
                tvMag.setBackgroundColor(act.getResources().getColor(R.color.COLOR_GREEN));
            } else if (magnitude >= 3 && magnitude < 5) {
                tvMag.setBackgroundColor(act.getResources().getColor(R.color.COLOR_YELLOW));
            } else if (magnitude >= 5) {
                tvMag.setBackgroundColor(act.getResources().getColor(R.color.COLOR_RED));
            }

            return vi;
        }

        @Override
        public EarthQuakes getItem(int position) {
            return super.getItem(position);
        }

        @Override
        public int getCount() {
            return super.getCount();
        }

        @Override
        public long getItemId(int position) {
            return super.getItemId(position);
        }

        @Override
        public int getPosition(EarthQuakes item) {
            return super.getPosition(item);
        }

        @Override
        public void notifyDataSetChanged() {
            super.notifyDataSetChanged();
        }

    }
```

com_liveEarthquakesAlerts_controller_adapters_MarkerInfoAdapter.java Raw

```
    package com.liveEarthquakesAlerts.controller.adapters;

    import android.view.LayoutInflater;
    import android.view.View;
    import android.widget.TextView;

    import com.google.android.gms.maps.GoogleMap.InfoWindowAdapter;
    import com.google.android.gms.maps.model.Marker;
    import com.liveEarthquakesAlerts.R;
    import com.liveEarthquakesAlerts.model.database.EarthQuakes;
```

```
import java.util.Date;

/**
 * Created by  Uddhav Gautam  on 7.3.2016. upgautam@ualr.edu
 */
public class MarkerInfoAdapter implements InfoWindowAdapter { // this is to mark location via google maps when user clicks eartl

    LayoutInflater inflater = null;

    TextView tvLoc, tvMag, tvDepth, tvDate, tvLat, tvLng;

    public MarkerInfoAdapter(LayoutInflater inflater) {
        this.inflater = inflater;
    }

    @Override
    public View getInfoContents(Marker marker) {

        View info = inflater.inflate(R.layout.marker_info, null);

        tvLoc = (TextView) info.findViewById(R.id.tv1);
        tvMag = (TextView) info.findViewById(R.id.tv2);
        tvDate = (TextView) info.findViewById(R.id.tv3);
        tvDepth = (TextView) info.findViewById(R.id.tv4);
        tvLat = (TextView) info.findViewById(R.id.tv5);
        tvLng = (TextView) info.findViewById(R.id.tv6);

        String snippet = marker.getSnippet();

        EarthQuakes earthQuakes = new EarthQuakes().getEarthquakesById(Long.parseLong(snippet));

        tvLoc.setText(" : " + earthQuakes.getLocationName());
        tvMag.setText(" : " + earthQuakes.getMagnitude());
        tvDepth.setText(" : " + earthQuakes.getDepth() + " KM");
        tvDate.setText(" : " + new Date(earthQuakes.getDateMilis()).toLocaleString());
        tvLat.setText(" : " + earthQuakes.getLatitude());
        tvLng.setText(" : " + earthQuakes.getLongitude());
```

```
            return info;
        }

        @Override
        public View getInfoWindow(Marker marker) {
            // TODO Auto-generated method stub
            return null;
        }

    }
```

com_liveEarthquakesAlerts_controller_services_earthquakes_EarthquakesDataSyncService.java Raw

```
package com.liveEarthquakesAlerts.controller.services.earthquakes;


import android.app.Service;
import android.content.Context;
import android.content.Intent;
import android.os.IBinder;
import android.util.Log;

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 com.liveEarthquakesAlerts.controller.utils.App;
import com.liveEarthquakesAlerts.controller.utils.BusStatus;
import com.liveEarthquakesAlerts.controller.utils.OnLineTracker;
import com.liveEarthquakesAlerts.controller.utils.SaveResponseToDB;

/**
 * Created by  Uddhav Gautam  on 7.3.2016. upgautam@ualr.edu
 */
public class EarthquakesDataSyncService extends Service {
    public static Context AppContextService;
    private int mStartMode;
    private IBinder mBinder;
    private boolean mAllowRebind;
    private DatabaseReference databaseReference;
//    private Handler handler;

    @Override
    public void onCreate() {
        AppContextService = getApplicationContext();
        App.bus.register(this);
        databaseReference = FirebaseDatabase.getInstance().getReference().getRoot().child("realTimeEarthquakes");

//        handler = new Handler(Looper.getMainLooper());
    }

    @Override
    public int onStartCommand(Intent intent, int flags, int startId) { //this will call onStart()
        ValueEventListener valueEventListener = new ValueEventListener() {
            @Override
            public void onDataChange(DataSnapshot dataSnapshot) {
                if (OnLineTracker.isOnline(AppContextService)) { //check every time online
                    SaveResponseToDB clientHelper = new SaveResponseToDB(); //clears the database in constructor
                    Log.i("Inside", "on start command!");
                    clientHelper.getDataFromFirebase(dataSnapshot);
                    App.bus.post(new BusStatus(123)); //post event into the Otto bus

                } else {
                    App.bus.post(new BusStatus(999));
                }
            }

            @Override
            public void onCancelled(DatabaseError databaseError) {

            }
        };
```

```java
            databaseReference.addValueEventListener(valueEventListener);

            return Service.START_STICKY;
        }

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

        @Override
        public boolean onUnbind(Intent intent) {
            return mAllowRebind;
        }

        @Override
        public void onRebind(Intent intent) {

        }

        @Override
        public void onDestroy() {
            App.bus.unregister(this);
        }

    }
```

com_liveEarthquakesAlerts_controller_services_locations_LocationTracker.java Raw

```java
    package com.liveEarthquakesAlerts.controller.services.locations;

    import android.app.NotificationManager;
    import android.app.PendingIntent;
    import android.app.Service;
    import android.content.Context;
    import android.content.Intent;
    import android.graphics.Color;
    import android.location.Location;
```

```
import android.media.RingtoneManager;
import android.net.Uri;
import android.os.Bundle;
import android.os.Handler;
import android.os.IBinder;
import android.os.Looper;
import android.support.annotation.NonNull;
import android.support.annotation.Nullable;
import android.support.design.widget.FloatingActionButton;
import android.support.v4.app.NotificationCompat;
import android.support.v4.app.TaskStackBuilder;
import android.telephony.SmsManager;
import android.util.Log;
import android.view.View;

import com.google.android.gms.common.ConnectionResult;
import com.google.android.gms.common.api.GoogleApiClient;
import com.google.android.gms.location.LocationListener;
import com.google.android.gms.location.LocationRequest;
import com.google.android.gms.location.LocationServices;
import com.google.android.gms.location.LocationSettingsRequest;
import com.liveEarthquakesAlerts.R;
import com.liveEarthquakesAlerts.controller.utils.AppSettings;
import com.liveEarthquakesAlerts.controller.utils.CheckRiskEarthquakes;
import com.liveEarthquakesAlerts.model.LocationPOJO;
import com.liveEarthquakesAlerts.model.database.EarthQuakes;
import com.liveEarthquakesAlerts.model.database.LastEarthquakeDate;
import com.liveEarthquakesAlerts.model.database.RiskyEarthquakes;
import com.liveEarthquakesAlerts.view.MainActivity;
import com.odoo.FavoriteNumberBean;

import java.util.ArrayList;
import java.util.List;

/* The IntentService class provides a straightforward structure for running an operation on a single background thread. */
public class LocationTracker extends Service
        implements LocationListener,
        GoogleApiClient.ConnectionCallbacks,
        GoogleApiClient.OnConnectionFailedListener {


    private static final String TAG = "LocationTracker";
    public static boolean isServiceRunning = false;
    private Handler handler;
    private List<RiskyEarthquakes> riskyEarthquakes;
    private int count = 0;
    private LocationRequest mLocationRequest;
    private GoogleApiClient mGoogleApiClient;
    private LocationSettingsRequest.Builder builderLocationSettings;

    private Location location; // location
    private LocationSettingsRequest mLocationSettingsRequest;
    private LocationPOJO locationPOJO;
    private String messageEarthquake;

    public LocationTracker() {
    }

    @Override
    public void onCreate() {
        Log.i(TAG, "On Create");
        buildGoogleApiClient();
        mGoogleApiClient.connect();
        handler = new Handler(Looper.getMainLooper());
        super.onCreate();


    }

    protected void createLocationRequest() {
        //remove location updates so that it resets
        LocationServices.FusedLocationApi.removeLocationUpdates(mGoogleApiClient, this); //Import should not be android.Locatior
        //import should be import com.google.android.gms.location.LocationListener;
        mLocationRequest = new LocationRequest();
        mLocationRequest.setInterval(10000);
        mLocationRequest.setSmallestDisplacement(500); //500 meters changed
```

```
            mLocationRequest.setFastestInterval(5000);
            mLocationRequest.setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY);
            //restart location updates with the new interval
            LocationServices.FusedLocationApi.requestLocationUpdates(mGoogleApiClient, mLocationRequest, this);

        }

        protected synchronized void buildLocationSettingsRequest() {
            builderLocationSettings = new LocationSettingsRequest.Builder(); //null builder
            builderLocationSettings.addLocationRequest(mLocationRequest);
            mLocationSettingsRequest = builderLocationSettings.build();

        }

        protected synchronized void buildGoogleApiClient() {
            Log.i(TAG, "Building GoogleApiClient");

            mGoogleApiClient = new GoogleApiClient.Builder(this)
                    .addApi(LocationServices.API)
                    .addConnectionCallbacks(this)
                    .addOnConnectionFailedListener(this)
                    .build();

        }


        @Override
        public int onStartCommand(Intent intent, int flags, int startId) { //this calls onStart()

            return super.onStartCommand(intent, flags, startId);
        }

        @Override
        public void onDestroy() {
            Log.i(TAG, "Service destroyed!");
            isServiceRunning = false;
            mGoogleApiClient.disconnect();
            super.onDestroy();
```

```java
            }

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


            @Override
            public void onConnected(@Nullable Bundle bundle) {
                count++;
                Log.i(TAG, "GoogleApiClient connected!");
        //        buildLocationSettingsRequest();
                createLocationRequest();
                location = LocationServices.FusedLocationApi.getLastLocation(mGoogleApiClient);
                LocationPOJO.location = location;
                Log.i(TAG, " LocationWhat: " + count + " " + LocationPOJO.location); //may return null because, I can't guarantee locati
            }

            @Override
            public void onConnectionSuspended(int i) {

            }

            @Override
            public void onConnectionFailed(@NonNull ConnectionResult connectionResult) {
                Log.i(TAG, "GoogleApiClient failed!");

            }

            @Override
            public void onLocationChanged(Location location) {
                Log.i(TAG, "Location Changed!");
                this.location = location;

                //update bean on every location changed
                LocationPOJO.location = location;
                //keep tracking of every risky earthquakes until they are no more dangerous to victim
                RiskyEarthquakes riskyEarthquakes = new RiskyEarthquakes();
                List<RiskyEarthquakes> allRiskyEarthquakes = riskyEarthquakes.GetAllData();

                //update the RiskyEarthquakes
                for (RiskyEarthquakes r : allRiskyEarthquakes) {
                    if (!CheckRiskEarthquakes.checkRisky(r)) {
                        r.DeleteRow(r.getDateMilis());
                    }
                }

                for (RiskyEarthquakes r : allRiskyEarthquakes) {
                    while (CheckRiskEarthquakes.checkRisky(r)) {
                        //Notify user
                        notificationHandler();
                        //Notify emergency only one time, then pop the "I am Ok" button to click and push messages "I am ok"
                        sendMsgToEmergencyContacts();
                    }
                }
            }

            private void notificationHandler() {

                handler.post(new Runnable() {
                    @Override
                    public void run() {
                        showNotification();
                    }
                });
            }

            private void showNotification() {
                List<RiskyEarthquakes> newEarthquakes = new RiskyEarthquakes().newEarthquakes();

                if (newEarthquakes.size() > 0) { //if there are earthquakes

                    if (AppSettings.getInstance().isNotifications()) {
                        createNotification(getString(R.string.EarthquakesDetect), "" + newEarthquakes.get(0).getMagnitude() + "  |  " +
                        messageEarthquake = "Earthquake Hit !!" + newEarthquakes.get(0).getMagnitude() + "  |  " + newEarthquakes.get(0)

                    }



                    LastEarthquakeDate led = new LastEarthquakeDate();
                    led.setDateMilis(new EarthQuakes().GetLastEarthQuakeDate());
                    led.Insert();
                }
            }

            private void sendMsgToEmergencyContacts() {
                FavoriteNumberBean favoriteNumberBean = new FavoriteNumberBean(false);
                ArrayList<String> mobileList = favoriteNumberBean.getMobileNumber();
                SmsManager smsManager = SmsManager.getDefault();
                //send every emergency contacts the messages
                for (String phone : mobileList) {
                    smsManager.sendTextMessage(phone, null, messageEarthquake, null, null);
                }
                //now create "I am Ok button", tap it to send "I am Ok" messages to every emergency contacts

                handler.post(new Runnable() {
                    @Override
                    public void run() {
                        //create a floating button
                        FloatingActionButton floatingActionButton = new FloatingActionButton(getApplicationContext());
                        floatingActionButton.setOnClickListener(new View.OnClickListener() {
                            @Override
                            public void onClick(View v) {
                                FavoriteNumberBean favoriteNumberBean = new FavoriteNumberBean(false);
                                ArrayList<String> mobileList = favoriteNumberBean.getMobileNumber();
                                SmsManager smsManager = SmsManager.getDefault();
                                //send every emergency contacts the messages
                                for (String phone : mobileList) {
                                    smsManager.sendTextMessage(phone, null, "I am Ok!", null, null);
                                }
                            }
                        });
                    }
                });
            }

            private void createNotification(String strContentTitle, String strContentText) {

                NotificationCompat.Builder builder = new NotificationCompat.Builder(getApplicationContext()) //
                        .setSmallIcon(R.drawable.icon1) //
                        .setContentTitle(strContentTitle) //
                        .setContentText(strContentText);

                Intent resultIntent = new Intent(this, MainActivity.class);
                TaskStackBuilder stackBuilder = TaskStackBuilder.create(this);
                stackBuilder.addParentStack(MainActivity.class);
                stackBuilder.addNextIntent(resultIntent);
                PendingIntent resultPendingIntent = stackBuilder.getPendingIntent(0, PendingIntent.FLAG_UPDATE_CURRENT);

                builder.setContentIntent(resultPendingIntent);
                builder.setAutoCancel(true);
                builder.setLights(Color.BLUE, 500, 500);

                if (AppSettings.getInstance().isVibration()) {
                    long[] pattern = {500, 500};
                    builder.setVibrate(pattern);
                }
                if (AppSettings.getInstance().isSound()) {
                    Uri alarmSound = RingtoneManager.getDefaultUri(RingtoneManager.TYPE_NOTIFICATION);
                    builder.setSound(alarmSound);
                }

                NotificationManager manager = (NotificationManager) getSystemService(Context.NOTIFICATION_SERVICE);
                manager.notify(0, builder.build());
            }
        }
```

com_liveEarthquakesAlerts_controller_utils_Animator.java Raw

```java
package com.liveEarthquakesAlerts.controller.utils;

import android.view.View;
import android.view.animation.AlphaAnimation;
import android.view.animation.Animation;
import android.widget.TextView;

import com.liveEarthquakesAlerts.R;


/**
 * Created by upg on 11/1/16.
 */

public class Animator extends Animation {
    private static Animation anim;
    private static Animator animator = null;
    public boolean isSetAnimation = false;
    private int animationTime;
    private View view;


    private Animator(View view) {

        this.view = view;
        anim = new AlphaAnimation(0.0f, 1.0f);

    }

    public static Animator getAnimator(View view) {
        if (animator == null) {
            animator = new Animator(view);
        }
        return animator;
    }

    public void setAnimation(int animationTime1) {
        animationTime = animationTime1;
```

```
            if (anim != null) {
                anim.setDuration(animationTime); //You can manage the time of the blink with this parameter
                anim.setStartOffset(20);
                anim.setRepeatMode(Animation.REVERSE);
                anim.setRepeatCount(Animation.INFINITE);
                if (view instanceof TextView) {
                    TextView textView = (TextView) view;
                    textView.setCompoundDrawablesWithIntrinsicBounds(R.drawable.danger, 0, 0, 0);
                    textView.startAnimation(anim);
                }
                this.isSetAnimation = true;
            }
        }

        public void stopAnimation(TextView tvBanner1) {
            anim.cancel();
            anim.reset();
            view.clearAnimation();
            this.isSetAnimation = false;
        }
    }
```

com_liveEarthquakesAlerts_controller_utils_App.java Raw

```
package com.liveEarthquakesAlerts.controller.utils;

import android.app.Application;
import android.content.Context;

/**
 * Created by  Uddhav Gautam  on 7.3.2016. upgautam@ualr.edu
 */
public class App extends Application { //App class for: 1) providing application context 2)Starting Otto Bus

    public static Context AppContext;
    public static OttoBus bus;
    //bus is just a Otto bus. It uses handler for the communication between activity and fragments or activity and services.
    @Override
    public void onCreate() {
        super.onCreate();

        AppContext = getApplicationContext(); //Now, AppContext is the context of the Application class (App class here)

//              Mint.initAndStartSession(App.this, getString(R.string.Mint_apiKey));

//        Mint.initAndStartSession(App.this, "29463cb0"); //I am using Splunk Mint SDK to initialize and do start session
        // of this App providing the Splunk Mint API key. 29463cb0 is the key. Splunk Mint, here, I am using for the Data collec

        bus = OttoBus.getOttoBus(); //Class.method. This technique doesn't initialize the constructor
    }
}
```

com_liveEarthquakesAlerts_controller_utils_AppSettings.java Raw

```
package com.liveEarthquakesAlerts.controller.utils;

import android.content.Context;
import android.content.SharedPreferences;
import android.preference.PreferenceManager;
import android.util.Log;

import com.liveEarthquakesAlerts.R;

import java.util.Map;

/**
 * Created by  Uddhav Gautam  on 7.3.2016. upgautam@ualr.edu
 */
public class AppSettings {

    public static AppSettings pojoPref = null;
    private static Context ctx = App.AppContext; //Get the same context of App class. Context is like security permission manage

    private int TimeInterval, ProximityMiles, Magnitude, Sorting;
```

```
        private boolean Notifications, isFavourite, Vibration, Sound; // for checking notification, vibration and sound whether to c

        private String Key_TimeInterval, Key_Magnitude, Key_Proxmity, ProximityMilesDesc;
        private String Key_Sorting;
        private String Key_Notifications;
        private String Key_Vibration;
        private String Key_Sound;
        private String Key_Emergency;


        AppSettings() { // constructor but no public. Because this class is based on Singleton design pattern

            Key_Proxmity = ctx.getResources().getString(R.string.listPref_Key_Proximity);

            Key_TimeInterval = ctx.getResources().getString(R.string.listPref_Key_TimeInterval);
            Key_Magnitude = ctx.getResources().getString(R.string.listPref_Key_Magnitude);
            Key_Sorting = ctx.getResources().getString(R.string.listPref_Key_Sorting);
            Key_Notifications = ctx.getResources().getString(R.string.CheckBoxPref_Key_Notifications);
            Key_Vibration = ctx.getResources().getString(R.string.CheckBoxPref_Key_Vibration);
            Key_Sound = ctx.getResources().getString(R.string.CheckBoxPref_Key_Sound);

            isFavourite = false;

            Key_Emergency = ctx.getResources().getString(R.string.CheckBoxPref_Key_Phone);

            SharedPreferences pref = PreferenceManager.getDefaultSharedPreferences(ctx);
            Map<String, ?> allEntries = pref.getAll(); //generic hashmap<string as key, anything as value>

            String asd = (String) allEntries.get(Key_TimeInterval); //return all values of  time interval (eg, last 24 hours)

            TimeInterval = Integer.parseInt(asd);
            Magnitude = Integer.parseInt((String) allEntries.get(Key_Magnitude));
            Sorting = Integer.parseInt((String) allEntries.get(Key_Sorting));

            ProximityMilesDesc = (String) allEntries.get(Key_Proxmity);
            Log.d("ProximityMilesDesc", ProximityMilesDesc);

            Notifications = (Boolean) allEntries.get(Key_Notifications);
            Vibration = (Boolean) allEntries.get(Key_Vibration);
            Sound = (Boolean) allEntries.get(Key_Sound);
            isFavourite = (Boolean) allEntries.get(Key_Emergency);

            if (isFavourite()) { //check through getter
                setFavourite(isFavourite);
            }
            if (ProximityMilesDesc.equalsIgnoreCase("200 miles")) {
                setProximityMiles(200);
            } else if (ProximityMilesDesc.equalsIgnoreCase("World-wide")) {
                setProximityMiles(0);
            }

        }

        public static void setDefaultSettings() {
            PreferenceManager.setDefaultValues(ctx, R.xml.pref, false); //false tells not to read again.

        }

        public static AppSettings getInstance() {
            return pojoPref == null ? new AppSettings() : pojoPref;
        }

        public int getProximityMiles() {
            return ProximityMiles;
        }

        public void setProximityMiles(int proximityMiles) {
            ProximityMiles = proximityMiles;
        }

        public int getTimeInterval() {
            return TimeInterval;
        }

        public void setTimeInterval(int timeInterval) {
            TimeInterval = timeInterval;
        }

        public int getMagnitude() {
            return Magnitude;
        }

        public void setMagnitude(int magnitude) {
            Magnitude = magnitude;
        }

        public int getSorting() {
            return Sorting;
        }

        public void setSorting(int sorting) {
            Sorting = sorting;
        }

        public boolean isSound() {
            return Sound;
        } //for boolean getter, it starts with "is" not "get"

        public void setSound(boolean sound) {
            Sound = sound;
        }

        public boolean isVibration() {
            return Vibration;
        }

        public void setVibration(boolean vibration) {
            Vibration = vibration;
        }

        public boolean isNotifications() {
            return Notifications;
        }
```

```
        public void setNotifications(boolean notifications) {
            Notifications = notifications;
        }

        public boolean isFavourite() {
            return isFavourite;
        }

        public void setFavourite(boolean favourite) {
            isFavourite = favourite;
        }
    }
```

com_liveEarthquakesAlerts_controller_utils_BusStatus.java Raw

```
    package com.liveEarthquakesAlerts.controller.utils;

    /**
     * Created by  Uddhav Gautam  on 7.3.2016. upgautam@ualr.edu
     */
    public class BusStatus { //to main the OttoBus status

        private int status;

        public BusStatus(int s) { //constructor working as a setter
            status = s;
        }

        public int getStatus() {
            return status;
        }

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

com_liveEarthquakesAlerts_controller_utils_CheckRiskEarthquakes.java

Raw

```java
package com.liveEarthquakesAlerts.controller.utils;

import android.location.Location;

import com.liveEarthquakesAlerts.model.LocationPOJO;
import com.liveEarthquakesAlerts.model.database.EarthQuakes;
import com.liveEarthquakesAlerts.model.database.RiskyEarthquakes;

/**
 * Created by  Uddhav Gautam  on 3/24/17.
 */

public class CheckRiskEarthquakes {


    public static boolean checkRisky(EarthQuakes item) {
        Location userLocation = LocationPOJO.location;

        Location finalLoc = new Location("Risky Earthquake");
        finalLoc.setLatitude(item.getLatitude());
        finalLoc.setLongitude(item.getLongitude());

        boolean status = false;

        if (userLocation != null) {
            double distanceInMeters = finalLoc.distanceTo(userLocation);
            double distanceValInMiles = distanceInMeters * 0.000621371;


            if (distanceValInMiles < 200 && item.getSig() > 500) { //we assume risky, need collaboration with GeoScientist
                status = true;
            }
        }
        return status;
    }
```

```
            public static boolean checkRisky(Float latitude, Float longitude, Integer sig) {
                Location userLocation = LocationPOJO.location;

                Location finalLoc = new Location("Risky Earthquake");
                finalLoc.setLatitude(latitude);
                finalLoc.setLongitude(longitude);

                boolean status = false;

                if (userLocation != null) {
                    double distanceInMeters = finalLoc.distanceTo(userLocation);
                    double distanceValInMiles = distanceInMeters * 0.000621371;


                    if (distanceValInMiles < 200 && sig > 500) { //we assume risky, need collaboration with GeoScientist
                        status = true;
                    }
                }
                return status;
            }

            public static boolean checkRisky(RiskyEarthquakes item) {
                Location userLocation = LocationPOJO.location;

                Location finalLoc = new Location("Risky Earthquake");
                finalLoc.setLatitude(item.getLatitude());
                finalLoc.setLongitude(item.getLongitude());

                boolean status = false;

                if (userLocation != null) {
                    double distanceInMeters = finalLoc.distanceTo(userLocation);
                    double distanceValInMiles = distanceInMeters * 0.000621371;


                    if (distanceValInMiles < 200 && item.getSig() > 500) { //we assume risky, need collaboration with GeoScientist
                        status = true;
                    }
                }
                return status;
            }
        }
```

com_liveEarthquakesAlerts_controller_utils_CreateRequestUrl.java Raw

```
package com.liveEarthquakesAlerts.controller.utils;

import android.location.Location;

/**
 * Created by  Uddhav Gautam  on 7.3.2016. upgautam@ualr.edu
 */
public class CreateRequestUrl {

    public static String URL_USGS(int day) {

        String str = "all_hour";

        if (day == 0) {
            str = "all_hour";
        } else if (day == 1) {
            str = "all_day";
        } else if (day == 2) {
            str = "all_week";
        }

        return "http://earthquake.usgs.gov/earthquakes/feed/v1.0/summary/" + str + ".geojson";
        //https://earthquake.usgs.gov/earthquakes/feed/v1.0/summary/all_day.geojson
    }

    public static String URL_USGS(int day, int i, Location currentLoc) {
        int bboxRadius = i;

        String str = "all_hour";
        if (day == 0) {
            str = "all_hour";
        } else if (day == 1) {
            str = "all_day";
        } else if (day == 2) {
            str = "all_week";
        }

        return "http://earthquake.usgs.gov/earthquakes/feed/v1.0/summary/" + str + ".geojson";
    }
}
```

com_liveEarthquakesAlerts_controller_utils_OnLineTracker.java Raw

```
package com.liveEarthquakesAlerts.controller.utils;

import android.content.Context;
import android.net.ConnectivityManager;
import android.net.NetworkInfo;

//import com.splunk.mint.Mint;

/**
 * Created by  Uddhav Gautam  on 7.3.2016. upgautam@ualr.edu
 */
public class OnLineTracker {

    public static int syncPeriod = 1000 * 15; //15 seconds
    public static String DATEFORMAT = "yyyy-MM-dd HH:mm:ss";
    public static String DATEFORMAT_SEISMICPORTAL = "yyyy-MM-dd'T'HH:mm:ss.S'Z'";

    public static boolean isOnline(Context ctx) {
        ConnectivityManager cm = (ConnectivityManager) ctx.getSystemService(Context.CONNECTIVITY_SERVICE);
        NetworkInfo netInfo = cm.getActiveNetworkInfo();
        return netInfo != null && netInfo.isConnectedOrConnecting();
    }

    public static void catchException(Exception ex) {
        ex.printStackTrace();
//        Mint.logException(ex);
    }
}
```

com_liveEarthquakesAlerts_controller_utils_OttoBus.java Raw

```
package com.liveEarthquakesAlerts.controller.utils;

import android.os.Handler;
import android.os.Looper;

import com.squareup.otto.Bus;

/**
 * Created by  Uddhav Gautam  on 7.3.2016. upgautam@ualr.edu
 */
public class OttoBus extends Bus { //this is Otto Bus

    private static OttoBus ottoBus = new OttoBus();
    private final Handler mHandler = new Handler(Looper.getMainLooper());

    private OttoBus() {
    }

    public static OttoBus getOttoBus() {
        return ottoBus;
    }

    @Override
    public void post(final Object event) {
        if (Looper.myLooper() == Looper.getMainLooper()) {

            super.post(event);
        } else {
            mHandler.post(new Runnable() {
                @Override
                public void run() {
                    OttoBus.super.post(event);
                    }
                });
            }
        }
    }
```

com_liveEarthquakesAlerts_controller_utils_RiskyEarthquakeTracker.java Raw

```
package com.liveEarthquakesAlerts.controller.utils;

/**
 * Created by  Uddhav Gautam  on 3/24/17.
 */

public class RiskyEarthquakeTracker {

}
```

com_liveEarthquakesAlerts_controller_utils_SaveResponseToDB.java Raw

```
package com.liveEarthquakesAlerts.controller.utils;

import android.util.Log;

import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.gson.Gson;
import com.google.gson.GsonBuilder;
import com.google.gson.reflect.TypeToken;
import com.liveEarthquakesAlerts.model.database.DatabaseHelper;
import com.liveEarthquakesAlerts.model.database.EarthQuakes;
import com.liveEarthquakesAlerts.model.database.RiskyEarthquakes;
import com.liveEarthquakesAlerts.model.sources.pOJOFolderUSGS.POJOUSGS;
import com.liveEarthquakesAlerts.model.sources.pOJOFolderUSGS.insidePOJOFolderUSGS.featuresFolderUSGS.FeaturesUSGS;
import com.liveEarthquakesAlerts.model.sources.pOJOFolderUSGS.insidePOJOFolderUSGS.featuresFolderUSGS.insideFeaturesUSGS.Geometr
import com.liveEarthquakesAlerts.model.sources.pOJOFolderUSGS.insidePOJOFolderUSGS.featuresFolderUSGS.insideFeaturesUSGS.Propert
import com.liveEarthquakesAlerts.model.sources.pOJOFolderUSGS.insidePOJOFolderUSGS.metadataFolderUSGS.MetadataUSGS;
```

```
        import java.lang.reflect.Type;
        import java.math.BigDecimal;
        import java.util.Date;
        import java.util.HashMap;
        import java.util.Map;

        import okhttp3.OkHttpClient;
        import okhttp3.Request;
        import okhttp3.Response;

        //We need database helper, so that, before we do insert new records, we clear database first

        /**
         * Created by  Uddhav Gautam  on 7.3.2016. upgautam@ualr.edu
         */
        public class SaveResponseToDB { //this class updates EarthQuakes Bean


            private String locationName, jsonOriginal = null, str1;
            private Integer sig, decimalPlace = 1;
            private Long time;
            private Float longitude, latitude, depth, magnitude;

            public SaveResponseToDB() {
                DatabaseHelper.getDbHelper().clearDatabase();
            }

            public static String getJson(String reqUrl) throws Exception {
                Request request = new Request.Builder().url(reqUrl).build(); //Request builder is used to get JSON url
                Response response = new OkHttpClient().newCall(request).execute(); //OkHttpClient is HTTP client to request
                return response.isSuccessful() ? response.body().string() : "";
            }

            public void getDataFromFirebase(final DataSnapshot dataSnapshot) { //save every earthquake fields like magnitude, latitude e

                new Thread(new Runnable() { //should do network operation using separate thread; can't do from main thread
                    @Override
                    public void run() {
                        try {

                            for (DataSnapshot feature : dataSnapshot.child("features").getChildren()) {

                                //get all required data

                                time = Long.parseLong(feature.child("properties").child("time").getValue().toString());
                                if (new Date().getTime() - time > 11000) { //old data
                                    //upload the JSON again
                                    updateFirebase(CreateRequestUrl.URL_USGS(0), FirebaseDatabase.getInstance().getReference().getRoot()
                                    return;
                                }

                                str1 = feature.child("properties").child("place").getValue().toString().trim().toUpperCase();
                                locationName = str1.substring(str1.indexOf("of") + 3);
                                sig = Integer.parseInt(feature.child("properties").child("sig").getValue().toString());
                                magnitude = Float.parseFloat(feature.child("properties").child("mag").getValue().toString());
                                longitude = Float.parseFloat(feature.child("geometry").child("coordinates").child("0").getValue().toStri
                                latitude = Float.parseFloat(feature.child("geometry").child("coordinates").child("1").getValue().toStrin
                                depth = Float.parseFloat(feature.child("geometry").child("coordinates").child("2").getValue().toString()


                                //update Earthquake object
                                EarthQuakes eq = new EarthQuakes(); //EarthQuakes is a bean
                                eq.setSig(sig);
                                eq.setDateMilis(time);
                                eq.setDepth(round(depth, decimalPlace)); //depth means altitude. In this way, database is getting update
                                eq.setLatitude(latitude);
                                eq.setLongitude(longitude);
                                eq.setLocationName(locationName);
                                eq.setMagnitude(round(magnitude, decimalPlace));

                                eq.Insert();

                                if (CheckRiskEarthquakes.checkRisky(latitude, longitude, sig)) {
                                    RiskyEarthquakes riskyEarthquakes = new RiskyEarthquakes();
                                    riskyEarthquakes.setSig(sig);
                                    riskyEarthquakes.setDateMilis(time);
                                    riskyEarthquakes.setDepth(round(depth, decimalPlace)); //depth means altitude. In this way, database
                                    riskyEarthquakes.setLatitude(latitude);
                                    riskyEarthquakes.setLongitude(longitude);
                                    riskyEarthquakes.setLocationName(locationName);
                                    riskyEarthquakes.setMagnitude(round(magnitude, decimalPlace));
                                    riskyEarthquakes.Insert();
                                }
                            }

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

            }

            public void updateFirebase(final String url, final DatabaseReference databaseReference) { //save every earthquake fields lik

                try {

                    final Gson gson = new GsonBuilder().setPrettyPrinting().setDateFormat(OnLineTracker.DATEFORMAT).create();
                    final Type listType = new TypeToken<POJOUSGS<String, MetadataUSGS, FeaturesUSGS<PropertiesUSGS, GeometryUSGS>, Float
                    }.getType();


                    new Thread(new Runnable() { //should do network operation using separate thread; can't do from main thread
                        @Override
                        public void run() {
                            try {
                                jsonOriginal = getJson(url);

                                if (jsonOriginal == null || jsonOriginal.length() < 1) { // JSON is null or empty , jsonOriginal.length(
                                    return;
                                }

                                Log.i("Jsonoriginal", jsonOriginal);
                                POJOUSGS<String, MetadataUSGS, FeaturesUSGS<PropertiesUSGS, GeometryUSGS>, Float> items = gson.fromJson(


                                if (items == null || items.getFeatures() == null || items.getFeatures().size() == 0) { //check if item r
                                    return;
                                }

                                Log.i("items", items.toString());

                                databaseReference.child("realTimeEarthquakes").setValue(items); //upload jsonOriginal on new "realTimeEa
                                String jsonString = "{  \n" +
                                        "      \"metaInfo\": {\n" +
                                        "        \"count\": \"serversCount\",\n" +
                                        "        \"needToBeServer\": \"false\"\n" +
                                        "      },\n" +
                                        "      \"servers\": [\n" +
                                        "        {\n" +
                                        "          \"id\": \"myid\",\n" +
                                        "          \"lastOnline\": \"timeMilis\"\n" +
                                        "        }\n" +
                                        "      ]\n" +
                                        "   }\n";
                                Map<String, Object> jsonMap = new Gson().fromJson(jsonString, new TypeToken<HashMap<String, Object>>() {
                                }.getType());


                                databaseReference.child("serverTrack").setValue(jsonMap); //upload jsonOriginal on new "realTimeEarthqua


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


                } catch (Exception e) {
                    OnLineTracker.catchException(e);
```

```
            }

        }

        public float round(float d, int decimalPlace) {
            BigDecimal bd = new BigDecimal(Float.toString(d));
            bd = bd.setScale(decimalPlace, BigDecimal.ROUND_HALF_UP);
            return bd.floatValue();
        }

    }
```

com_liveEarthquakesAlerts_model_database_DatabaseHelper.java Raw

```
    package com.liveEarthquakesAlerts.model.database;

    import android.content.Context;
    import android.database.sqlite.SQLiteDatabase;

    import com.j256.ormlite.android.apptools.OrmLiteSqliteOpenHelper;
    import com.j256.ormlite.dao.Dao;
    import com.j256.ormlite.support.ConnectionSource;
    import com.j256.ormlite.table.TableUtils;
    import com.liveEarthquakesAlerts.controller.utils.App;
    import com.liveEarthquakesAlerts.controller.utils.OnLineTracker;

    import java.sql.SQLException;

    /**
     * Created by  Uddhav Gautam  on 7.3.2016. upgautam@ualr.edu
     */
    public class DatabaseHelper extends OrmLiteSqliteOpenHelper {

        private static final String DATABASE_NAME = "lastearthquakes.db";
        private static final int DATABASE_VERSION = 1;
        private static DatabaseHelper dbHelper;
        private static Object syncObject = new Object();
        private final Context myContext;
        private Dao<EarthQuakes, Long> EarthQuakesDataHelper = null;
        private Dao<LastEarthquakeDate, Integer> LastEarthquakeDateDataHelper = null;

        public DatabaseHelper(Context context) {
            super(context, DATABASE_NAME, null, DATABASE_VERSION);
            this.myContext = context;
        }

        public static DatabaseHelper getDbHelper() {
            synchronized (syncObject) {
                if (dbHelper == null) {
                    dbHelper = new DatabaseHelper(App.AppContext);
                }
            }
            return dbHelper;
        }

        @Override
        public void onCreate(SQLiteDatabase db, ConnectionSource connectionSource) {
            try {
                TableUtils.createTable(connectionSource, EarthQuakes.class);
                TableUtils.createTable(connectionSource, LastEarthquakeDate.class);
            } catch (java.sql.SQLException e) {
                OnLineTracker.catchException(e);
            }
        }

        @Override
        public void onUpgrade(SQLiteDatabase db, ConnectionSource connectionSource, int oldVersion, int newVersion) {
            try {
                TableUtils.dropTable(connectionSource, EarthQuakes.class, true);
                TableUtils.dropTable(connectionSource, LastEarthquakeDate.class, true);
                onCreate(db, connectionSource);
            } catch (java.sql.SQLException e) {
                OnLineTracker.catchException(e);
            }
        }
```

```
        public void clearDatabase() {
            ConnectionSource connectionSource = getConnectionSource();
            try {
                TableUtils.clearTable(connectionSource, EarthQuakes.class);
            } catch (SQLException e) {
                OnLineTracker.catchException(e);
            }
        }

        public Dao<EarthQuakes, Long> getEarthQuakesDataHelper() throws SQLException {
            if (EarthQuakesDataHelper == null) {
                EarthQuakesDataHelper = getDao(EarthQuakes.class);
            }
            return EarthQuakesDataHelper;
        }

        public Dao<LastEarthquakeDate, Integer> getLastEarthquakeDateDataHelper() throws SQLException {
            if (LastEarthquakeDateDataHelper == null) {
                LastEarthquakeDateDataHelper = getDao(LastEarthquakeDate.class);
            }
            return LastEarthquakeDateDataHelper;
        }

    }
```

com_liveEarthquakesAlerts_model_database_DatabaseHelperRisky.java Raw

```
    package com.liveEarthquakesAlerts.model.database;

    import android.content.Context;
    import android.database.sqlite.SQLiteDatabase;

    import com.j256.ormlite.android.apptools.OrmLiteSqliteOpenHelper;
    import com.j256.ormlite.dao.Dao;
    import com.j256.ormlite.support.ConnectionSource;
    import com.j256.ormlite.table.TableUtils;
    import com.liveEarthquakesAlerts.controller.utils.App;
```

```
import com.liveEarthquakesAlerts.controller.utils.OnLineTracker;

import java.sql.SQLException;

/**
 * Created by  Uddhav Gautam  on 3/23/17.
 */

public class DatabaseHelperRisky extends OrmLiteSqliteOpenHelper {

    private static final String DATABASE_NAME = "lastearthquakesRisky.db";
    private static final int DATABASE_VERSION = 1;
    private static DatabaseHelperRisky dbHelper;
    private static Object syncObject = new Object();
    private final Context myContext;
    private Dao<RiskyEarthquakes, Long> EarthQuakesDataHelperRisky = null;
    private Dao<LastEarthquakeDateRisky, Integer> LastEarthquakeDateDataHelperRisky = null;

    public DatabaseHelperRisky(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
        this.myContext = context;
    }

    public static DatabaseHelperRisky getDbHelper() {
        synchronized (syncObject) {
            if (dbHelper == null) {
                dbHelper = new DatabaseHelperRisky(App.AppContext);
            }
        }
        return dbHelper;
    }

    @Override
    public void onCreate(SQLiteDatabase db, ConnectionSource connectionSource) {
        try {
            TableUtils.createTable(connectionSource, RiskyEarthquakes.class);
            TableUtils.createTable(connectionSource, LastEarthquakeDateRisky.class);
        } catch (java.sql.SQLException e) {
            OnLineTracker.catchException(e);
        }
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, ConnectionSource connectionSource, int oldVersion, int newVersion) {
        try {
            TableUtils.dropTable(connectionSource, RiskyEarthquakes.class, true);
            TableUtils.dropTable(connectionSource, LastEarthquakeDateRisky.class, true);
            onCreate(db, connectionSource);
        } catch (java.sql.SQLException e) {
            OnLineTracker.catchException(e);
        }
    }

    public void clearDatabase() {
        ConnectionSource connectionSource = getConnectionSource();
        try {
            TableUtils.clearTable(connectionSource, RiskyEarthquakes.class);
        } catch (SQLException e) {
            OnLineTracker.catchException(e);
        }
    }

    public Dao<RiskyEarthquakes, Long> getEarthQuakesDataHelperRisky() throws SQLException {
        if (EarthQuakesDataHelperRisky == null) {
            EarthQuakesDataHelperRisky = getDao(RiskyEarthquakes.class);
        }
        return EarthQuakesDataHelperRisky;
    }

    public Dao<LastEarthquakeDateRisky, Integer> getLastEarthquakeDateDataHelperRisky() throws SQLException {
        if (LastEarthquakeDateDataHelperRisky == null) {
            LastEarthquakeDateDataHelperRisky = getDao(LastEarthquakeDateRisky.class);
        }
        return LastEarthquakeDateDataHelperRisky;
    }
```

```
    }
```

com_liveEarthquakesAlerts_model_database_EarthQuakes.java

Raw

```
package com.liveEarthquakesAlerts.model.database;

import android.os.Parcel;
import android.os.Parcelable;
import android.util.Log;

import com.j256.ormlite.dao.Dao;
import com.j256.ormlite.field.DatabaseField;
import com.j256.ormlite.stmt.DeleteBuilder;
import com.j256.ormlite.stmt.PreparedQuery;
import com.j256.ormlite.stmt.QueryBuilder;
import com.j256.ormlite.table.DatabaseTable;
import com.liveEarthquakesAlerts.controller.utils.AppSettings;
import com.liveEarthquakesAlerts.controller.utils.OnLineTracker;
import com.liveEarthquakesAlerts.model.LocationPOJO;

import java.sql.SQLException;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Comparator;
import java.util.List;

/**
 * Created by  Uddhav Gautam  on 7.3.2016. upgautam@ualr.edu
 */
@DatabaseTable(tableName = "EarthQuakes")
public class EarthQuakes implements Parcelable, Comparator<EarthQuakes> {

    public static final Creator<EarthQuakes> CREATOR = new Creator<EarthQuakes>() { // earthquakes ko chunk space is CREATOR
        @Override
        public EarthQuakes createFromParcel(Parcel in) {
            return new EarthQuakes(in);
        }
```

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

            @DatabaseField(id = true)
            private Long DateMilis;
            @DatabaseField
            private String LocationName;
            @DatabaseField
            private double Latitude;
            @DatabaseField
            private double Longitude;
            @DatabaseField
            private float Magnitude;
            @DatabaseField
            private float Depth;
            @DatabaseField
            private int sig;
            @DatabaseField
            private int Day;
            @DatabaseField
            private int Month;

            public EarthQuakes() {
            }

            protected EarthQuakes(Parcel in) {
                LocationName = in.readString();
                Latitude = in.readDouble();
                Longitude = in.readDouble();
                Magnitude = in.readFloat();
                Depth = in.readFloat();
                sig = in.readInt();
                Day = in.readInt();
                Month = in.readInt();
            }
```

```java
    public static Long backDate() {

        int value = AppSettings.getInstance().getTimeInterval();

        int goBack = 7;

        if (value == 0) {
            goBack = 0; //last hour
        } else if (value == 1) {
            goBack = 1; //last 1 day
        } else if (value == 2) {
            goBack = 7; //last 7 days
        }

        Calendar cal = Calendar.getInstance();
        cal.add(Calendar.DAY_OF_MONTH, -goBack);
        return cal.getTimeInMillis();
    }

    public int getSig() {
        return sig;
    }

    public void setSig(int sig) {
        this.sig = sig;
    }

    public void Insert() {

        Calendar cal = Calendar.getInstance();
        cal.setTimeInMillis(DateMilis);
        Day = cal.get(Calendar.DAY_OF_MONTH);
        Month = cal.get(Calendar.MONTH) + 1;

        try {
            Dao<EarthQuakes, Long> MissionsInsert = (DatabaseHelper.getDbHelper()).getEarthQuakesDataHelper();
            EarthQuakes existenceCheck = MissionsInsert.queryForId(this.DateMilis);
```

```
                if (existenceCheck != null) {
                    MissionsInsert.update(this); //delete and then create
                } else {
                    MissionsInsert.create(this);
                }

            } catch (SQLException e) {
                OnLineTracker.catchException(e);
            }
        }

        public List<EarthQuakes> GetAllData() {

            List<EarthQuakes> data = new ArrayList<>();

            try {

                Dao<EarthQuakes, Long> dao = DatabaseHelper.getDbHelper().getEarthQuakesDataHelper();
                QueryBuilder<EarthQuakes, Long> qBuilder = dao.queryBuilder();

                int sortingType = AppSettings.getInstance().getSorting();
                Long backdate = backDate();

                qBuilder.where()//
                        .gt("Magnitude", AppSettings.getInstance().getMagnitude()) //
                        .and()//
                        .gt("DateMilis", backdate);


                if (sortingType == 0) {
                    qBuilder.orderBy("DateMilis", true);
                } else if (sortingType == 1) {
                    qBuilder.orderBy("DateMilis", false);
                } else if (sortingType == 2) {
                    qBuilder.orderBy("Magnitude", true);
                } else if (sortingType == 3) {
                    qBuilder.orderBy("Magnitude", false);
                }

                PreparedQuery<EarthQuakes> pQuery = qBuilder.prepare();
                data = dao.query(pQuery);

            } catch (SQLException e) {
                OnLineTracker.catchException(e);
            }

//          Log.i("Row1", data.get(0).getLocationName());
            return data;
        }

        public List<EarthQuakes> GetAllDataUserProximity() {
            List<EarthQuakes> data = new ArrayList<>();
            try {
                Dao<EarthQuakes, Long> dao = DatabaseHelper.getDbHelper().getEarthQuakesDataHelper();
                QueryBuilder<EarthQuakes, Long> qBuilder = dao.queryBuilder();
                int sortingType = AppSettings.getInstance().getSorting();
                Long backdate = backDate();
                qBuilder.where()//
                        .gt("Magnitude", AppSettings.getInstance().getMagnitude()) //
                        .and()//
                        .gt("DateMilis", backdate)
                        .and()//
                        .gt("Latitude", LocationPOJO.location.getLatitude() - 2.91)
                        .and()//
                        .gt("Longitude", LocationPOJO.location.getLongitude() - 2.89)
                        .and()//
                        .lt("Latitude", LocationPOJO.location.getLatitude() + 2.91)
                        .and()//
                        .lt("Longitude", LocationPOJO.location.getLongitude() + 2.89);

                if (sortingType == 0) {
                    qBuilder.orderBy("DateMilis", true);
                } else if (sortingType == 1) {
                    qBuilder.orderBy("DateMilis", false);
                } else if (sortingType == 2) {
```

```
                qBuilder.orderBy("Magnitude", true);
            } else if (sortingType == 3) {
                qBuilder.orderBy("Magnitude", false);
            }
            PreparedQuery<EarthQuakes> pQuery = qBuilder.prepare();
            data = dao.query(pQuery);

        } catch (SQLException e) {
            OnLineTracker.catchException(e);
        }
        return data;
    }

    public Long GetLastEarthQuakeDate() {

        List<EarthQuakes> data = new ArrayList<>();

        try {

            Dao<EarthQuakes, Long> dao = DatabaseHelper.getDbHelper().getEarthQuakesDataHelper();
            QueryBuilder<EarthQuakes, Long> qBuilder = dao.queryBuilder();

            qBuilder.orderBy("DateMilis", false);

            PreparedQuery<EarthQuakes> pQuery = qBuilder.prepare();
            data = dao.query(pQuery);

        } catch (SQLException e) {
            OnLineTracker.catchException(e);
        }

        Log.i("NewDateMilis", String.valueOf(data.get(0).getDateMilis()));

        return data.get(0).getDateMilis();
    }

    public List<EarthQuakes> newEarthquakes() {
        List<EarthQuakes> data = new ArrayList<>();


        try {

            Dao<EarthQuakes, Long> dao = DatabaseHelper.getDbHelper().getEarthQuakesDataHelper();
            QueryBuilder<EarthQuakes, Long> qBuilder = dao.queryBuilder();


            qBuilder.distinct().where()
                    .gt("Magnitude", AppSettings.getInstance().getMagnitude()) //
                    .and()//
                    .gt("DateMilis", new LastEarthquakeDate().GetLastEarthquakeMilisDate());


            qBuilder.orderBy("DateMilis", false);

            PreparedQuery<EarthQuakes> pQuery = qBuilder.prepare();
            data = dao.query(pQuery);

        } catch (SQLException e) {
            OnLineTracker.catchException(e);
        }

        return data;
    }

    public int GetRowCount() {
        int count = 0;

        try {
            Dao<EarthQuakes, Long> dao = DatabaseHelper.getDbHelper().getEarthQuakesDataHelper();
            count = (int) dao.countOf();
        } catch (Exception e) {
            OnLineTracker.catchException(e);
        }

        return count;
    }

    public void DeleteRow(long deleteId) {
        try {

            Dao<EarthQuakes, Long> dao = DatabaseHelper.getDbHelper().getEarthQuakesDataHelper();
            DeleteBuilder<EarthQuakes, Long> deleteBuilder = dao.deleteBuilder();
            deleteBuilder.where().eq("DateMilis", deleteId);
            deleteBuilder.delete();
        } catch (Exception e) {
            OnLineTracker.catchException(e);
        }
    }

    public EarthQuakes getEarthquakesById(Long DateMilis) {

        List<EarthQuakes> eqList = new ArrayList<>();

        try {
            Dao<EarthQuakes, Long> dao = DatabaseHelper.getDbHelper().getEarthQuakesDataHelper();
            QueryBuilder<EarthQuakes, Long> qBuilder = dao.queryBuilder();
            qBuilder.distinct().where().eq("DateMilis", DateMilis);
            PreparedQuery<EarthQuakes> pQuery = qBuilder.prepare();
            eqList = dao.query(pQuery);

        } catch (Exception e) {
            OnLineTracker.catchException(e);
        }

        return eqList.get(0);
    }

    public List<EarthQuakes> getEarthquakesByDay(int day, int month) {

        List<EarthQuakes> data = new ArrayList<>();

        try {
            Dao<EarthQuakes, Long> dao = DatabaseHelper.getDbHelper().getEarthQuakesDataHelper();
            QueryBuilder<EarthQuakes, Long> qBuilder = dao.queryBuilder();

            int sortingType = AppSettings.getInstance().getSorting();

            qBuilder.where()//
                    .eq("Source", 0) //
                    .and()//
                    .gt("Magnitude", AppSettings.getInstance().getMagnitude());


            if (sortingType == 0) {
                qBuilder.orderBy("DateMilis", true);
            } else if (sortingType == 1) {
                qBuilder.orderBy("DateMilis", false);
            } else if (sortingType == 2) {
                qBuilder.orderBy("Magnitude", true);
            } else if (sortingType == 3) {
                qBuilder.orderBy("Magnitude", false);
            }

            PreparedQuery<EarthQuakes> pQuery = qBuilder.prepare();
            data = dao.query(pQuery);

        } catch (SQLException e) {
            OnLineTracker.catchException(e);
        }

        return data;
    }

    public List<EarthQuakes> getColumn(String ColumnName) throws SQLException {
        Dao<EarthQuakes, Long> dao = DatabaseHelper.getDbHelper().getEarthQuakesDataHelper();
        List<EarthQuakes> results = dao.queryBuilder().distinct().selectColumns(ColumnName).query();
        return results;
    }

    public Long getDateMilis() {
        return DateMilis;
    }

    public void setDateMilis(Long dateMilis) {
        DateMilis = dateMilis;
    }

    public String getLocationName() {
        return LocationName;
    }

    public void setLocationName(String locationName) {
        LocationName = locationName;
    }

    public double getLatitude() {
        return Latitude;
    }

    public void setLatitude(double latitude) {
        Latitude = latitude;
    }

    public double getLongitude() {
        return Longitude;
    }

    public void setLongitude(double longitude) {
        Longitude = longitude;
    }

    public float getMagnitude() {
        return Magnitude;
    }

    public void setMagnitude(float magnitude) {
        Magnitude = magnitude;
    }
```

```
            public float getDepth() {
                return Depth;
            }

            public void setDepth(float depth) {
                Depth = depth;
            }

            public int getDay() {
                return Day;
            }

            public void setDay(int day) {
                Day = day;
            }

            public int getMonth() {
                return Month;
            }

            public void setMonth(int month) {
                Month = month;
            }

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

            @Override
            public void writeToParcel(Parcel dest, int flags) {

                dest.writeString(LocationName);
                dest.writeDouble(Latitude);
                dest.writeDouble(Longitude);
                dest.writeFloat(Magnitude);
                dest.writeFloat(Depth);
```

```
        dest.writeInt(sig);
        dest.writeInt(Day);
        dest.writeInt(Month);
    }

    @Override
    public int compare(EarthQuakes lhs, EarthQuakes rhs) {
        return (int) (lhs.getDateMilis() - rhs.getDateMilis());
    }


}
```

com_liveEarthquakesAlerts_model_database_LastEarthquakeDate.java Raw

```
package com.liveEarthquakesAlerts.model.database;

import android.os.Parcel;
import android.os.Parcelable;

import com.j256.ormlite.dao.Dao;
import com.j256.ormlite.field.DatabaseField;
import com.j256.ormlite.table.DatabaseTable;
import com.liveEarthquakesAlerts.controller.utils.OnLineTracker;

import java.sql.SQLException;
import java.util.Comparator;

/**
 * Created by  Uddhav Gautam  on 7.3.2016. upgautam@ualr.edu
 */

@DatabaseTable(tableName = "LastEarthquakeDate")
public class LastEarthquakeDate implements Parcelable, Comparator<LastEarthquakeDate> {

    public static final Creator<LastEarthquakeDate> CREATOR = new Creator<LastEarthquakeDate>() {
        @Override
        public LastEarthquakeDate createFromParcel(Parcel in) {
```

```
                    return new LastEarthquakeDate(in);
                }

                @Override
                public LastEarthquakeDate[] newArray(int size) {
                    return new LastEarthquakeDate[size];
                }
            };
            @DatabaseField(id = true)
            private int id;
            @DatabaseField
            private Long DateMilis;

            public LastEarthquakeDate() {
                this.id = 1;
            }

            protected LastEarthquakeDate(Parcel in) {
                id = in.readInt();
            }

            public void Insert() {

                try {
                    Dao<LastEarthquakeDate, Integer> Missionsinsert = (DatabaseHelper.getDbHelper()).getLastEarthquakeDateDataHelper();
                    LastEarthquakeDate existenceCheck = Missionsinsert.queryForId(this.id);

                    if (existenceCheck != null) {
                        Missionsinsert.update(this);
                    } else {
                        Missionsinsert.create(this);
                    }

                } catch (SQLException e) {
                    OnLineTracker.catchException(e);
                }
            }
```

```java
            public Long GetLastEarthquakeMilisDate() {
                LastEarthquakeDate lastDate = null;
                try {
                    Dao<LastEarthquakeDate, Integer> dao = DatabaseHelper.getDbHelper().getLastEarthquakeDateDataHelper();
                    lastDate = dao.queryForId(1);
                } catch (Exception e) {
                    OnLineTracker.catchException(e);
                }
                return lastDate.getDateMilis();
            }

            public int GetRowCount() {
                int count = 0;

                try {
                    Dao<LastEarthquakeDate, Integer> dao = DatabaseHelper.getDbHelper().getLastEarthquakeDateDataHelper();
                    count = (int) dao.countOf();
                } catch (Exception e) {
                    OnLineTracker.catchException(e);
                }

                return count;
            }

            public int getId() {
                return id;
            }

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

            public Long getDateMilis() {
                return DateMilis;
            }

            public void setDateMilis(Long dateMilis) {
                DateMilis = dateMilis;
```

```
        }

        @Override
        public int describeContents() {
            // TODO Auto-generated method stub
            return 0;
        }

        @Override
        public void writeToParcel(Parcel dest, int flags) {
            // TODO Auto-generated method stub

            dest.writeInt(id);
        }

        @Override
        public int compare(LastEarthquakeDate lhs, LastEarthquakeDate rhs) {
            return (int) (lhs.DateMilis - rhs.DateMilis);
        }

    }
```

com_liveEarthquakesAlerts_model_database_LastEarthquakeDateRisky.java

Raw

```
    package com.liveEarthquakesAlerts.model.database;

    import android.os.Parcel;
    import android.os.Parcelable;

    import com.j256.ormlite.dao.Dao;
    import com.j256.ormlite.field.DatabaseField;
    import com.j256.ormlite.table.DatabaseTable;
    import com.liveEarthquakesAlerts.controller.utils.OnLineTracker;

    import java.sql.SQLException;
    import java.util.Comparator;
```

```
/**
 * Created by  Uddhav Gautam  on 7.3.2016. upgautam@ualr.edu
 */

@DatabaseTable(tableName = "LastEarthquakeDateRisky")
public class LastEarthquakeDateRisky implements Parcelable, Comparator<LastEarthquakeDateRisky> {

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

        @Override
        public LastEarthquakeDateRisky[] newArray(int size) {
            return new LastEarthquakeDateRisky[size];
        }
    };
    @DatabaseField(id = true)
    private int id;
    @DatabaseField
    private Long DateMilis;

    public LastEarthquakeDateRisky() {
        this.id = 1;
    }

    protected LastEarthquakeDateRisky(Parcel in) {
        id = in.readInt();
    }

    public void Insert() {

        try {
            Dao<LastEarthquakeDateRisky, Integer> Missionsinsert = (DatabaseHelperRisky.getDbHelper()).getLastEarthquakeDateData
            LastEarthquakeDateRisky existenceCheck = Missionsinsert.queryForId(this.id);

            if (existenceCheck != null) {
                Missionsinsert.update(this);
            } else {
                Missionsinsert.create(this);
            }

        } catch (SQLException e) {
            OnLineTracker.catchException(e);
        }
    }

    public Long GetLastEarthquakeMilisDate() {
        LastEarthquakeDateRisky lastDate = null;
        try {
            Dao<LastEarthquakeDateRisky, Integer> dao = DatabaseHelperRisky.getDbHelper().getLastEarthquakeDateDataHelperRisky()
            lastDate = dao.queryForId(1);
        } catch (Exception e) {
            OnLineTracker.catchException(e);
        }
        return lastDate.getDateMilis();
    }

    public int GetRowCount() {
        int count = 0;

        try {
            Dao<LastEarthquakeDateRisky, Integer> dao = DatabaseHelperRisky.getDbHelper().getLastEarthquakeDateDataHelperRisky()
            count = (int) dao.countOf();
        } catch (Exception e) {
            OnLineTracker.catchException(e);
        }

        return count;
    }

    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }

    public Long getDateMilis() {
        return DateMilis;
    }

    public void setDateMilis(Long dateMilis) {
        DateMilis = dateMilis;
    }

    @Override
    public int describeContents() {
        // TODO Auto-generated method stub
        return 0;
    }

    @Override
    public void writeToParcel(Parcel dest, int flags) {
        // TODO Auto-generated method stub

        dest.writeInt(id);
    }

    @Override
    public int compare(LastEarthquakeDateRisky lhs, LastEarthquakeDateRisky rhs) {
        return (int) (lhs.DateMilis - rhs.DateMilis);
    }

}
```

com_liveEarthquakesAlerts_model_database_RiskyEarthquakes.java Raw

```
    package com.liveEarthquakesAlerts.model.database;

    import android.os.Parcel;
```

```
import android.os.Parcelable;
import android.util.Log;

import com.j256.ormlite.dao.Dao;
import com.j256.ormlite.field.DatabaseField;
import com.j256.ormlite.stmt.DeleteBuilder;
import com.j256.ormlite.stmt.PreparedQuery;
import com.j256.ormlite.stmt.QueryBuilder;
import com.j256.ormlite.table.DatabaseTable;
import com.liveEarthquakesAlerts.controller.utils.AppSettings;
import com.liveEarthquakesAlerts.controller.utils.OnLineTracker;

import java.sql.SQLException;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Comparator;
import java.util.List;

/**
 * Created by  Uddhav Gautam  on 7.3.2016. upgautam@ualr.edu
 */
@DatabaseTable(tableName = "RiskyEarthquakes")
public class RiskyEarthquakes implements Parcelable, Comparator<RiskyEarthquakes> {

    public static final Creator<RiskyEarthquakes> CREATOR = new Creator<RiskyEarthquakes>() { // earthquakes ko chunk space is (
        @Override
        public RiskyEarthquakes createFromParcel(Parcel in) {
            return new RiskyEarthquakes(in);
        }

        @Override
        public RiskyEarthquakes[] newArray(int size) {
            return new RiskyEarthquakes[size];
        }
    };

    @DatabaseField(id = true)
    private Long DateMilis;
```

```java
    @DatabaseField
    private String LocationName;
    @DatabaseField
    private double Latitude;
    @DatabaseField
    private double Longitude;
    @DatabaseField
    private float Magnitude;
    @DatabaseField
    private float Depth;
    @DatabaseField
    private int sig;
    @DatabaseField
    private int Day;
    @DatabaseField
    private int Month;

    public RiskyEarthquakes() {
    }

    protected RiskyEarthquakes(Parcel in) {
        LocationName = in.readString();
        Latitude = in.readDouble();
        Longitude = in.readDouble();
        Magnitude = in.readFloat();
        Depth = in.readFloat();
        sig = in.readInt();
        Day = in.readInt();
        Month = in.readInt();
    }

    public static Long backDate() {

        int value = AppSettings.getInstance().getTimeInterval();

        int goBack = 7;

        if (value == 0) {
            goBack = 0; //last hour
        } else if (value == 1) {
            goBack = 1; //last 1 day
        } else if (value == 2) {
            goBack = 7; //last 7 days
        }

        Calendar cal = Calendar.getInstance();
        cal.add(Calendar.DAY_OF_MONTH, -goBack);
        return cal.getTimeInMillis();
    }

    public int getSig() {
        return sig;
    }

    public void setSig(int sig) {
        this.sig = sig;
    }

    public void Insert() {

        Calendar cal = Calendar.getInstance();
        cal.setTimeInMillis(DateMilis);
        Day = cal.get(Calendar.DAY_OF_MONTH);
        Month = cal.get(Calendar.MONTH) + 1;

        try {
            Dao<RiskyEarthquakes, Long> MissionsInsert = (DatabaseHelperRisky.getDbHelper()).getEarthQuakesDataHelperRisky();
            RiskyEarthquakes existenceCheck = MissionsInsert.queryForId(this.DateMilis);

            if (existenceCheck != null) {
                MissionsInsert.update(this); //delete and then create
            } else {
                MissionsInsert.create(this);
            }

        } catch (SQLException e) {
            OnLineTracker.catchException(e);
        }
    }

    public List<RiskyEarthquakes> GetAllData() {

        List<RiskyEarthquakes> data = new ArrayList<>();

        try {

            Dao<RiskyEarthquakes, Long> dao = DatabaseHelperRisky.getDbHelper().getEarthQuakesDataHelperRisky();
            QueryBuilder<RiskyEarthquakes, Long> qBuilder = dao.queryBuilder();

            int sortingType = AppSettings.getInstance().getSorting();
            Long backdate = backDate();

            qBuilder.where()//
                    .gt("Magnitude", AppSettings.getInstance().getMagnitude()) //
                    .and()//
                    .gt("DateMilis", backdate);


            if (sortingType == 0) {
                qBuilder.orderBy("DateMilis", true);
            } else if (sortingType == 1) {
                qBuilder.orderBy("DateMilis", false);
            } else if (sortingType == 2) {
                qBuilder.orderBy("Magnitude", true);
            } else if (sortingType == 3) {
                qBuilder.orderBy("Magnitude", false);
            }

            PreparedQuery<RiskyEarthquakes> pQuery = qBuilder.prepare();
            data = dao.query(pQuery);

        } catch (SQLException e) {
            OnLineTracker.catchException(e);
        }
```

```
//        Log.i("Row1", data.get(0).getLocationName());
        return data;
    }

    public Long GetLastEarthQuakeDate() {

        List<RiskyEarthquakes> data = new ArrayList<>();

        try {

            Dao<RiskyEarthquakes, Long> dao = DatabaseHelperRisky.getDbHelper().getEarthQuakesDataHelperRisky();
            QueryBuilder<RiskyEarthquakes, Long> qBuilder = dao.queryBuilder();

            qBuilder.orderBy("DateMilis", false);

            PreparedQuery<RiskyEarthquakes> pQuery = qBuilder.prepare();
            data = dao.query(pQuery);

        } catch (SQLException e) {
            OnLineTracker.catchException(e);
        }

        Log.i("NewDateMilis", String.valueOf(data.get(0).getDateMilis()));

        return data.get(0).getDateMilis();
    }

    public List<RiskyEarthquakes> newEarthquakes() {

        List<RiskyEarthquakes> data = new ArrayList<>();


        try {

            Dao<RiskyEarthquakes, Long> dao = DatabaseHelperRisky.getDbHelper().getEarthQuakesDataHelperRisky();
            QueryBuilder<RiskyEarthquakes, Long> qBuilder = dao.queryBuilder();
```

```
                qBuilder.distinct().where()
                        .gt("Magnitude", AppSettings.getInstance().getMagnitude()) //
                        .and()//
                        .gt("DateMilis", new LastEarthquakeDate().GetLastEarthquakeMilisDate());


                qBuilder.orderBy("DateMilis", false);

                PreparedQuery<RiskyEarthquakes> pQuery = qBuilder.prepare();
                data = dao.query(pQuery);

            } catch (SQLException e) {
                OnLineTracker.catchException(e);
            }

            return data;
        }

        public int GetRowCount() {
            int count = 0;

            try {
                Dao<RiskyEarthquakes, Long> dao = DatabaseHelperRisky.getDbHelper().getEarthQuakesDataHelperRisky();
                count = (int) dao.countOf();
            } catch (Exception e) {
                OnLineTracker.catchException(e);
            }

            return count;
        }

        public void DeleteRow(long deleteId) {
            try {

                Dao<RiskyEarthquakes, Long> dao = DatabaseHelperRisky.getDbHelper().getEarthQuakesDataHelperRisky();
                DeleteBuilder<RiskyEarthquakes, Long> deleteBuilder = dao.deleteBuilder();
                deleteBuilder.where().eq("DateMilis", deleteId);
                deleteBuilder.delete();
            } catch (Exception e) {
                OnLineTracker.catchException(e);
            }
        }

        public RiskyEarthquakes getEarthquakesById(Long DateMilis) {

            List<RiskyEarthquakes> eqList = new ArrayList<>();

            try {
                Dao<RiskyEarthquakes, Long> dao = DatabaseHelperRisky.getDbHelper().getEarthQuakesDataHelperRisky();
                QueryBuilder<RiskyEarthquakes, Long> qBuilder = dao.queryBuilder();
                qBuilder.distinct().where().eq("DateMilis", DateMilis);
                PreparedQuery<RiskyEarthquakes> pQuery = qBuilder.prepare();
                eqList = dao.query(pQuery);

            } catch (Exception e) {
                OnLineTracker.catchException(e);
            }

            return eqList.get(0);
        }

        public List<RiskyEarthquakes> getEarthquakesByDay(int day, int month) {

            List<RiskyEarthquakes> data = new ArrayList<>();

            try {

                Dao<RiskyEarthquakes, Long> dao = DatabaseHelperRisky.getDbHelper().getEarthQuakesDataHelperRisky();
                QueryBuilder<RiskyEarthquakes, Long> qBuilder = dao.queryBuilder();

                int sortingType = AppSettings.getInstance().getSorting();

                qBuilder.where()//
                        .eq("Source", 0) //
                        .and()//
```

```
                    .gt("Magnitude", AppSettings.getInstance().getMagnitude());


            if (sortingType == 0) {
                qBuilder.orderBy("DateMilis", true);
            } else if (sortingType == 1) {
                qBuilder.orderBy("DateMilis", false);
            } else if (sortingType == 2) {
                qBuilder.orderBy("Magnitude", true);
            } else if (sortingType == 3) {
                qBuilder.orderBy("Magnitude", false);
            }

            PreparedQuery<RiskyEarthquakes> pQuery = qBuilder.prepare();
            data = dao.query(pQuery);

        } catch (SQLException e) {
            OnLineTracker.catchException(e);
        }

        return data;
    }

    public List<RiskyEarthquakes> getColumn(String ColumnName) throws SQLException {
        Dao<RiskyEarthquakes, Long> dao = DatabaseHelperRisky.getDbHelper().getEarthQuakesDataHelperRisky();
        List<RiskyEarthquakes> results = dao.queryBuilder().distinct().selectColumns(ColumnName).query();
        return results;
    }

    public Long getDateMilis() {
        return DateMilis;
    }

    public void setDateMilis(Long dateMilis) {
        DateMilis = dateMilis;
    }

    public String getLocationName() {
        return LocationName;
    }

    public void setLocationName(String locationName) {
        LocationName = locationName;
    }

    public double getLatitude() {
        return Latitude;
    }

    public void setLatitude(double latitude) {
        Latitude = latitude;
    }

    public double getLongitude() {
        return Longitude;
    }

    public void setLongitude(double longitude) {
        Longitude = longitude;
    }

    public float getMagnitude() {
        return Magnitude;
    }

    public void setMagnitude(float magnitude) {
        Magnitude = magnitude;
    }

    public float getDepth() {
        return Depth;
    }

    public void setDepth(float depth) {
        Depth = depth;
    }
```

```
            public int getDay() {
                return Day;
            }

            public void setDay(int day) {
                Day = day;
            }

            public int getMonth() {
                return Month;
            }

            public void setMonth(int month) {
                Month = month;
            }

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

            @Override
            public void writeToParcel(Parcel dest, int flags) {

                dest.writeString(LocationName);
                dest.writeDouble(Latitude);
                dest.writeDouble(Longitude);
                dest.writeFloat(Magnitude);
                dest.writeFloat(Depth);
                dest.writeInt(sig);
                dest.writeInt(Day);
                dest.writeInt(Month);
            }

            @Override
            public int compare(RiskyEarthquakes lhs, RiskyEarthquakes rhs) {
                return (int) (lhs.getDateMilis() - rhs.getDateMilis());
```

```
        }


    }
```

com_liveEarthquakesAlerts_model_LocationPOJO.java

Raw

```
package com.liveEarthquakesAlerts.model;

import android.location.Location;

/**
 * Created by  Uddhav Gautam  on 3/22/17.
 */

public class LocationPOJO {
    public static Location location = null;
    private final String TAG = "LocationPOJO";

    public LocationPOJO() {
    }

}
```

com_liveEarthquakesAlerts_model_sources_pOJOFolderUSGS_insidePOJOFolderUSGS_featuresFolderUSGS_…

Raw

```
package com.liveEarthquakesAlerts.model.sources.pOJOFolderUSGS.insidePOJOFolderUSGS.featuresFolderUSGS;

/**
 * Created by  Uddhav Gautam  on 7.3.2016. upgautam@ualr.edu
 */
public class FeaturesUSGS<P, G> { //generic representation. Two different generic types.
// These types can be any type, PropertiesUSGS, GeometryUSGS, id are 4 attributes of feature.
    // Each feature means each earthquake.
    // Generic is only for object, not for primitive types

    private String type;
```

```
        private P properties; //added PropertiesUSGS as first generic type
        private G geometry; //added GeometryUSGS as second generic type
        private String id;

        public String getType() {
            return type;
        }

        public void setType(String type) {
            this.type = type;
        }

        public P getProperties() {
            return properties;
        }

        public void setProperties(P properties) {
            this.properties = properties;
        }

        public G getGeometry() {
            return geometry;
        }

        public void setGeometry(G geometry) {
            this.geometry = geometry;
        }

        public String getId() {
            return id;
        }

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

    }
```

com_liveEarthquakesAlerts_model_sources_pOJOFolderUSGS_insidePOJOFolderUSGS_featuresFolderUSGS_… Raw

```
package com.liveEarthquakesAlerts.model.sources.pOJOFolderUSGS.insidePOJOFolderUSGS.featuresFolderUSGS.insideFeaturesUSGS;

/**
 * Created by  Uddhav Gautam  on 7.3.2016. upgautam@ualr.edu
 */

import java.util.List;

public class GeometryUSGS {

    private String type;
    private List<Float> coordinates;

    public String getType() { //return type of GeometryUSGS. eg, point, line, Polygon etc. Here it is always point
        return type;
    }

    public void setType(String type) {
        this.type = type;
    }

    public List<Float> getCoordinates() {
        return coordinates; //coordinate must be array. Here it is List. Since all arrays are list.
    } //lat, long, alt of point

    public void setCoordinate(List<Float> coordinates) {
        this.coordinates = coordinates;
    }

}
```

com_liveEarthquakesAlerts_model_sources_pOJOFolderUSGS_insidePOJOFolderUSGS_featuresFolderUSGS_… Raw

```
package com.liveEarthquakesAlerts.model.sources.pOJOFolderUSGS.insidePOJOFolderUSGS.featuresFolderUSGS.insideFeaturesUSGS;

/**
```

```
 * Created by  Uddhav Gautam  on 7.3.2016. upgautam@ualr.edu
 */
public class PropertiesUSGS {

    private float mag;
    private String place;
    private String time;
    private String updated;
    private int tz;
    private String url;
    private String status;
    private int sig; //significance of earthquake. This tells how dangerous earthquake iS. Value is in between 0 to 1000
    private String net;
    private String code;
    private String ids;
    private String sources;
    private String types;
    private String gap;
    private String magnitudeType;

    private Object nst;
    private Object dmin;
    private Object rms;
    private Object felt;
    private Object cdi;
    private Object mmi;
    private Object alert;
    private Object tsunami;

    public float getMag() {
        return mag;
    }

    public void setMag(float mag) {
        this.mag = mag;
    }

    public String getPlace() {
        return place;
    }

    public void setPlace(String place) {
        this.place = place;
    }

    public String getTime() {
        return time;
    }

    public void setTime(String time) {
        this.time = time;
    }

    public String getUpdated() {
        return updated;
    }

    public void setUpdated(String updated) {
        this.updated = updated;
    }

    public int getTz() {
        return tz;
    }

    public void setTz(int tz) {
        this.tz = tz;
    }

    public String getUrl() {
        return url;
    }

    public void setUrl(String url) {
        this.url = url;
    }
```

```
            public String getStatus() {
                return status;
            }

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

            public int getSig() {
                return sig;
            }

            public void setSig(int sig) {
                this.sig = sig;
            }

            public String getNet() {
                return net;
            }

            public void setNet(String net) {
                this.net = net;
            }

            public String getCode() {
                return code;
            }

            public void setCode(String code) {
                this.code = code;
            }

            public String getIds() {
                return ids;
            }

            public void setIds(String ids) {
```

```
            this.ids = ids;
        }

        public String getSources() {
            return sources;
        }

        public void setSources(String sources) {
            this.sources = sources;
        }

        public String getTypes() {
            return types;
        }

        public void setTypes(String types) {
            this.types = types;
        }

        public String getGap() {
            return gap;
        }

        public void setGap(String gap) {
            this.gap = gap;
        }

        public String getMagnitudeType() {
            return magnitudeType;
        }

        public void setMagnitudeType(String magnitudeType) {
            this.magnitudeType = magnitudeType;
        }

        public Object getNst() {
            return nst;
        }
```

```
            public void setNst(Object nst) {
                this.nst = nst;
            }

            public Object getDmin() {
                return dmin;
            }

            public void setDmin(Object dmin) {
                this.dmin = dmin;
            }

            public Object getRms() {
                return rms;
            }

            public void setRms(Object rms) {
                this.rms = rms;
            }

            public Object getFelt() {
                return felt;
            }

            public void setFelt(Object felt) {
                this.felt = felt;
            }

            public Object getCdi() {
                return cdi;
            }

            public void setCdi(Object cdi) {
                this.cdi = cdi;
            }

            public Object getMmi() {
```

```
            return mmi;
        }

        public void setMmi(Object mmi) {
            this.mmi = mmi;
        }

        public Object getAlert() {
            return alert;
        }

        public void setAlert(Object alert) {
            this.alert = alert;
        }

        public Object getTsunami() {
            return tsunami;
        }

        public void setTsunami(Object tsunami) {
            this.tsunami = tsunami;
        }

    }
```

com_liveEarthquakesAlerts_model_sources_pOJOFolderUSGS_insidePOJOFolderUSGS_metadataFolderUSGS_… Raw

```
    package com.liveEarthquakesAlerts.model.sources.pOJOFolderUSGS.insidePOJOFolderUSGS.metadataFolderUSGS;

    /**
     * Created by  Uddhav Gautam  on 7.3.2016. upgautam@ualr.edu
     */
    public class MetadataUSGS {

        private long generated;
        private String url;
        private String title;
        private int status;
        private String api;
        private int count;

        public long getGenerated() {
            return generated;
        }

        public void setGenerated(long generated) {
            this.generated = generated;
        }

        public String getUrl() {
            return url;
        }

        public void setUrl(String url) {
            this.url = url;
        }

        public String getTitle() {
            return title;
        }

        public void setTitle(String title) {
            this.title = title;
        }

        public int getStatus() {
            return status;
        }

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

        public String getApi() {
            return api;
        }
```

```
    public void setApi(String api) {
        this.api = api;
    }

    public int getCount() {
        return count;
    }

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

com_liveEarthquakesAlerts_model_sources_pOJOFolderUSGS_POJOUSGS.java

Raw

```
package com.liveEarthquakesAlerts.model.sources.pOJOFolderUSGS;

import java.util.List;

/**
 * Created by  Uddhav Gautam  on 7.3.2016. upgautam@ualr.edu
 */
public class POJOUSGS<String, M, F, Float> {

    private String type;
    private M metadata; //added MetadataUSGS as M type generic. M becomes MetadataUSGS when
    //I call                        POJOUSGS<String, MetadataUSGS, FeaturesUSGS<PropertiesUSGS, GeometryUSGS>, Float> items = g
//  where listType is as:           final Type listType = new TypeToken<POJOUSGS<String, MetadataUSGS, FeaturesUSGS<PropertiesU

    private List<F> features; //added FeaturesUSGS as List of F type.
    private List<Float> bbox;


    public List<Float> getBbox() {
        return bbox;
    }
```

```
        public void setBbox(List<Float> bbox) {
            this.bbox = bbox;
        }

        public String getType() {
            return type;
        }

        public void setType(String type) {
            this.type = type;
        }

        public M getMetadata() {
            return metadata;
        }

        public void setMetadata(M metadata) {
            this.metadata = metadata;
        }

        public List<F> getFeatures() {
            return features;
        }

        public void setFeatures(List<F> features) {
            this.features = features;
        }

    }
```

com_liveEarthquakesAlerts_view_MainActivity.java Raw

```
    package com.liveEarthquakesAlerts.view;

    import android.app.ProgressDialog;
    import android.content.Intent;
    import android.location.Location;
```

```
        import android.os.Bundle;
        import android.support.v7.app.AppCompatActivity;
        import android.support.v7.widget.Toolbar;
        import android.util.Log;
        import android.view.Menu;
        import android.view.MenuItem;
        import android.view.View;
        import android.widget.AbsListView;
        import android.widget.AbsListView.OnScrollListener;
        import android.widget.AdapterView;
        import android.widget.ListView;
        import android.widget.TextView;

        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 com.liveEarthquakesAlerts.R;
        import com.liveEarthquakesAlerts.controller.adapters.ListViewAdapter;
        import com.liveEarthquakesAlerts.controller.services.earthquakes.EarthquakesDataSyncService;
        import com.liveEarthquakesAlerts.controller.services.locations.LocationTracker;
        import com.liveEarthquakesAlerts.controller.utils.Animator;
        import com.liveEarthquakesAlerts.controller.utils.App;
        import com.liveEarthquakesAlerts.controller.utils.AppSettings;
        import com.liveEarthquakesAlerts.controller.utils.BusStatus;
        import com.liveEarthquakesAlerts.controller.utils.CheckRiskEarthquakes;
        import com.liveEarthquakesAlerts.controller.utils.CreateRequestUrl;
        import com.liveEarthquakesAlerts.controller.utils.OnLineTracker;
        import com.liveEarthquakesAlerts.controller.utils.SaveResponseToDB;
        import com.liveEarthquakesAlerts.model.LocationPOJO;
        import com.liveEarthquakesAlerts.model.database.EarthQuakes;
        import com.liveEarthquakesAlerts.model.database.LastEarthquakeDate;
        import com.liveEarthquakesAlerts.model.database.LastEarthquakeDateRisky;
        import com.squareup.otto.Subscribe;

        import java.util.List;
```

```
/**
 * Created by  Uddhav Gautam  on 7.3.2016. upgautam@ualr.edu
 */
public class MainActivity extends AppCompatActivity implements AdapterView.OnItemLongClickListener, AdapterView.OnItemClickListe

    public static String bannerText;
    private final String TAG = "MainActivity";
    private ProgressDialog pd;
    private ListView list;
    private int currentScrollState, currentFirstVisibleItem, currentVisibleItemCount, currentTotalItemCount;
    private ListViewAdapter adapter;
    private TextView tvEmptyMessage;
    private TextView tvBanner;
    private boolean isConnectToInternet = true;

    @Override
    public void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Toolbar mToolbar = (Toolbar) findViewById(R.id.toolbar1);


////        remove the left margin from the logo
        mToolbar.setPadding(2, 0, 0, 0);//for tab otherwise give space in tab
        mToolbar.setContentInsetsAbsolute(0, 0);

        setSupportActionBar(mToolbar);
//
////        set the logo icon
        mToolbar.setLogo(R.drawable.icon1);

        AppSettings.setDefaultSettings(); //SingleFragmentActivity -- AppSettings -- other classes

        if (new LastEarthquakeDate().GetRowCount() == 0) { //if no earthquakes already in our database, then find total no of ne
            //checking just one earthquake record exists if enough to tell, this apk has previously synced the earthquakes
            LastEarthquakeDate led = new LastEarthquakeDate();
```

```
                //sets datemilis for any arbitrary record
                led.setDateMilis(606175200000l); //some date of 1989. This is starting datemilis
                Log.i("Datemilis", String.valueOf(led.getDateMilis()));
                led.Insert(); //this ultimately creates a earthquake row
                //this ensures the LastEarthquakeDate table is not null

            }

            if (new LastEarthquakeDateRisky().GetRowCount() == 0) {
                LastEarthquakeDateRisky led = new LastEarthquakeDateRisky();
                led.setDateMilis(606175200000l);
                Log.i("Datemilis", String.valueOf(led.getDateMilis()));
                led.Insert();

            }


            tvEmptyMessage = (TextView) findViewById(R.id.tv_empty_message);
            tvBanner = (TextView) findViewById(R.id.mile_banner);
            tvBanner.setText(" ");
            list = (ListView) findViewById(R.id.list2); //adds ListView in this, MainActivity. This list is for storing
            // earthquakes record in GUI

            list.setOnItemLongClickListener(this); // adds listeners on that ListView
            list.setOnScrollListener(this); //
            list.setOnItemClickListener(this);

//          start service to track user location
            if (!LocationTracker.isServiceRunning) { //if service not running
                Intent intent = new Intent(getApplicationContext(), LocationTracker.class); //start service
                startService(intent);
            }

            initializeFirebaseRealtimeDB();


//          start service to fetch earthquakes
```

```
            Intent intent = new Intent(getBaseContext(), EarthquakesDataSyncService.class); //start service
            startService(intent);

            pd = new ProgressDialog(MainActivity.this); //show progressbar
            pd.setProgressStyle(ProgressDialog.STYLE_SPINNER);
            pd.setTitle(getString(R.string.PleaseWait));
            pd.setMessage(getString(R.string.DatasLoading));
            pd.setCancelable(true);
            pd.setIndeterminate(true);
            pd.show();

        }

        private void initializeFirebaseRealtimeDB() {
            final DatabaseReference databaseReference = FirebaseDatabase.getInstance().getReference().getRoot();
            Log.i("reference1", databaseReference.toString());

            final ValueEventListener valueEventListener = new ValueEventListener() {
                @Override
                public void onDataChange(DataSnapshot dataSnapshot) {
                    if (!dataSnapshot.getChildren().iterator().hasNext()) {//empty

                        //create Firebase Realtime DB jsonOriginal structure and upload earthquake JSON

                        SaveResponseToDB clientHelper = new SaveResponseToDB(); //clears the database in constructor
                        clientHelper.updateFirebase(CreateRequestUrl.URL_USGS(0), databaseReference);

    //                    unregister valueEventListener
                        databaseReference.removeEventListener(this);
                    }
                }

                @Override
                public void onCancelled(DatabaseError databaseError) {

                }
            };
            databaseReference.addValueEventListener(valueEventListener);
        }

        @Override
        protected void onStart() {
            super.onStart();
            App.bus.register(this); //registration of Otto Bus
        }

        @Override
        protected void onStop() {
            super.onStop();
            App.bus.unregister(this); //Unregister of Otto Bus
        }

        @Override
        protected void onResume() {
            super.onResume();

            if (isConnectToInternet) {
                List<EarthQuakes> EarthQuakeList = new EarthQuakes().GetAllData();

                if (EarthQuakeList.size() > 0) {
                    adapter = new ListViewAdapter(MainActivity.this, EarthQuakeList);
                    adapter.notifyDataSetChanged();
                    list.setAdapter(adapter);
                    list.setSelectionFromTop(currentFirstVisibleItem, 0);
                }
            }

        }

        @Subscribe
        public void messageReceived(BusStatus event) {
            Log.i("MainActivity", event.getStatus() + " ");

            if (event.getStatus() == 999) {
                isConnectToInternet = false;
                list.setEmptyView(tvEmptyMessage);
                list.setAdapter(null);
            } else {
                isConnectToInternet = true;
                List<EarthQuakes> EarthQuakeList;
                Log.i("ConnectInternet", "true");
                if (AppSettings.getInstance().getProximityMiles() == 0) {
                    EarthQuakeList = new EarthQuakes().GetAllDataUserProximity();
                } else {
                    EarthQuakeList = new EarthQuakes().GetAllData();
                }
                if (EarthQuakeList.size() > 0) {
                    Log.i("EarthquakeData", "Yes");
                    adapter = new ListViewAdapter(MainActivity.this, EarthQuakeList);
                    adapter.notifyDataSetChanged();
                    list.setAdapter(adapter);
                    list.setSelectionFromTop(currentFirstVisibleItem, 0); // (x,y)
                }

            }

            if (pd != null && pd.isShowing()) {
                Log.i("Inside pd", "pd is running");
                pd.dismiss();
                pd = null;
            }
        }

        @Override
        public boolean onCreateOptionsMenu(Menu menu) {

            getMenuInflater().inflate(R.menu.activity_main, menu);
            return true;
        }

        @Override
        public boolean onPrepareOptionsMenu(Menu menu) {
//            Show the Emergency contact icon
            Log.i("Visible", String.valueOf(View.VISIBLE));
            if (AppSettings.getInstance().isFavourite()) {
                menu.getItem(0).setVisible(true);
//                On every emergency contacts enabled, animate the Emergency contact icon. Will do later

            } else {
                menu.getItem(0).setVisible(false);
            }
            return super.onPrepareOptionsMenu(menu);
        }

        @Override
        public boolean onOptionsItemSelected(MenuItem item) {

            if (item.getItemId() == R.id.action_main) {

                Intent i1 = new Intent(MainActivity.this, SettingsActivity.class); //It is actually communicating with SettingsActiv
                //explicit intent to start SingleFragmentActivity
                startActivityForResult(i1, 7777); //Update Menu from this activity. //(Intent, requestCode)

                return true;
            }

            if (item.getItemId() == R.id.e_contact) {

                Intent i2 = new Intent(MainActivity.this, com.odoo.HomeActivity.class); //explicit intent to start SingleFragmentAct
                startActivity(i2);
                return true;
            }


            return super.onOptionsItemSelected(item); //This is default method calling which simply return false. This makes sure it
        }

        @Override
        protected void onActivityResult(int requestCode, int resultCode, Intent data) {
            super.onActivityResult(requestCode, resultCode, data);

            // Check which request we're responding to
            if (data != null && resultCode == RESULT_OK && requestCode == 7777) {

                Boolean boolVal = data.getBooleanExtra("str", false); //default value is false
                if (boolVal) {
                    invalidateOptionsMenu();
                }

            }

        }

        @Override
        public void onScrollStateChanged(AbsListView view, int scrollState) {
            this.currentScrollState = scrollState;
            this.isScrollCompleted();
        }

        @Override
        public void onScroll(AbsListView view, int firstVisibleItem, int visibleItemCount, int totalItemCount) {
            this.currentFirstVisibleItem = firstVisibleItem;
            this.currentVisibleItemCount = visibleItemCount;
            this.currentTotalItemCount = totalItemCount;
        }

        private void isScrollCompleted() {

            if (currentFirstVisibleItem + currentVisibleItemCount >= currentTotalItemCount) {
                if (this.currentVisibleItemCount > 0 && this.currentScrollState == OnScrollListener.SCROLL_STATE_IDLE) {

                }
            }
        }

        @Override
        public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
            EarthQuakes eq = (EarthQuakes) parent.getAdapter().getItem(position);

            if (CheckRiskEarthquakes.checkRisky(eq)) {
                Animator animator = Animator.getAnimator(tvBanner); //get the Animator to do the animation for tvBanner TextView
                if (animator.isSetAnimation) {
                    tvBanner.setCompoundDrawablesWithIntrinsicBounds(0, 0, 0, 0);
                    animator.stopAnimation(tvBanner);
                }

            }
            Location userLocation = LocationPOJO.location;

            Location finalLoc = new Location("Risky Earthquake");
            finalLoc.setLatitude(eq.getLatitude());
            finalLoc.setLongitude(eq.getLongitude());


            if (userLocation != null) {
                double distanceInMeters = finalLoc.distanceTo(userLocation);
                double distanceValInMiles = distanceInMeters * 0.000621371;
                bannerText = String.format("%.2f", distanceValInMiles) + " miles far!";
                tvBanner.setText(bannerText);

            }
        }

        @Override
        public boolean onItemLongClick(AdapterView<?> parent1, View view, int position, long id) {
            EarthQuakes eq = (EarthQuakes) parent1.getAdapter().getItem(position);

            if (CheckRiskEarthquakes.checkRisky(eq)) {
                Animator animator = Animator.getAnimator(tvBanner); //get the Animator to do the animation for tvBanner TextView
                if (animator.isSetAnimation) {
                    tvBanner.setCompoundDrawablesWithIntrinsicBounds(0, 0, 0, 0);
                    animator.stopAnimation(tvBanner);
                }

            }
            Location userLocation = LocationPOJO.location;

            Location finalLoc = new Location("Risky Earthquake");
```

```
                finalLoc.setLatitude(eq.getLatitude());
                finalLoc.setLongitude(eq.getLongitude());

                if (userLocation != null) {
                    double distanceInMeters = finalLoc.distanceTo(userLocation);
                    double distanceValInMiles = distanceInMeters * 0.000621371;
                    bannerText = String.format("%.2f", distanceValInMiles) + " miles far!";
                    tvBanner.setText(bannerText);

                }

        //Display the Google Maps
                try {
                    Intent i = new Intent(MainActivity.this, MapsActivity.class); //explicit intent
                    i.putExtra("selectedItem", eq.getDateMilis());
                    startActivity(i);
                } catch (Exception e) {
                    OnLineTracker.catchException(e);
                }
                return false;

            }


        }
```

com_liveEarthquakesAlerts_view_MapsActivity.java

Raw

```
        package com.liveEarthquakesAlerts.view;

        import android.app.Dialog;
        import android.graphics.Color;
        import android.location.Location;
        import android.os.Bundle;
        import android.support.v4.app.FragmentActivity;
        import android.util.Log;
        import android.view.Gravity;
```

```
import android.widget.Toast;

import com.google.android.gms.common.ConnectionResult;
import com.google.android.gms.common.GooglePlayServicesUtil;
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.maps.model.Polyline;
import com.google.android.gms.maps.model.PolylineOptions;
import com.liveEarthquakesAlerts.R;
import com.liveEarthquakesAlerts.controller.adapters.MarkerInfoAdapter;
import com.liveEarthquakesAlerts.controller.utils.OnLineTracker;
import com.liveEarthquakesAlerts.model.LocationPOJO;
import com.liveEarthquakesAlerts.model.database.EarthQuakes;

import java.util.ArrayList;

/**
 * Created by  Uddhav Gautam  on 7.3.2016. upgautam@ualr.edu
 */
public class MapsActivity extends FragmentActivity implements OnMapReadyCallback, GoogleMap.OnMapClickListener, GoogleMap.OnMapL

    private final String TAG = "MapsActivity";
    MarkerOptions mOptions = new MarkerOptions();
    private GoogleMap myMap;
    private MarkerInfoAdapter infoAdapter;
    private long itemId;
    private SupportMapFragment mapFragment;
    private long DateMilis;

    private LatLng currentLatLng;
    private LatLng destLatLng1;
    private Location location;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_maps);

        itemId = getIntent().getLongExtra("selectedItem", 0);

        if (itemId == 0) {
            finish();
        } else {
            DateMilis = itemId;
        }
        location = LocationPOJO.location;
        Log.i(TAG, " Location: " + location);
        setUpMapIfNeeded();
    }

    @Override
    protected void onResume() {
        super.onResume();
        setUpMapIfNeeded();
    }

    private void setUpMapIfNeeded() {
        if (myMap == null) {
            mapFragment = (SupportMapFragment) getSupportFragmentManager().findFragmentById(R.id.map);
            mapFragment.getMapAsync(this);
        } else {
            Toast.makeText(getApplicationContext(), "myMap null!", Toast.LENGTH_LONG).show();
        }
    }

    private void setUpMap() {

        try {
            if (location != null) {
                myMap.setMapType(GoogleMap.MAP_TYPE_NORMAL);

                infoAdapter = new MarkerInfoAdapter(MapsActivity.this.getLayoutInflater());
                myMap.clear();
                myMap.setInfoWindowAdapter(infoAdapter);

                myMap.setMyLocationEnabled(true);
                myMap.setOnMapClickListener(this);
                myMap.setOnMapLongClickListener(this);
                myMap.setOnMarkerClickListener(this);
                myMap.setOnMarkerDragListener(this);

                EarthQuakes currentEarthquakes = new EarthQuakes().getEarthquakesById(itemId);

//             gets user LatLng
                this.currentLatLng = new LatLng(location.getLatitude(), location.getLongitude());
                mOptions.position(currentLatLng); //reach the position of the LatLng for that mark
                mOptions.title("Current Location");
                mOptions.visible(true); //makes visible
                mOptions.icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_BLUE));
                myMap.addMarker(mOptions);

                for (EarthQuakes earthQuakes : new EarthQuakes().GetAllData()) {
//                 gets marker LatLng

                    LatLng destLatLng = new LatLng(earthQuakes.getLatitude(), earthQuakes.getLongitude());


                    mOptions.position(destLatLng); //reach the position of the LatLng for that mark
                    mOptions.snippet(Long.toString(earthQuakes.getDateMilis())); //takes whatever descriptions
                    mOptions.visible(true); //makes visible


                    float magnitude = earthQuakes.getMagnitude();

                    if (magnitude < 3) {
                        mOptions.icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_GREEN));
                    } else if (magnitude >= 3 && magnitude < 5) {
```

```
                    mOptions.icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_YELLOW));
                } else if (magnitude >= 5) {
                    mOptions.icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_RED));
                }

                if (earthQuakes.getDateMilis().equals(currentEarthquakes.getDateMilis())) {
                    myMap.moveCamera(CameraUpdateFactory.newLatLngZoom(destLatLng, 6));
                    Marker marker = myMap.addMarker(mOptions);
                    marker.showInfoWindow();
                } else {
                    myMap.addMarker(mOptions);
                }


//                add the line only between selected earthQuake and current location
                this.destLatLng1 = new LatLng(earthQuakes.getEarthquakesById(DateMilis).getLatitude(), earthQuakes.getEarthq

                ArrayList<LatLng> locList = new ArrayList<LatLng>();
                locList.add(currentLatLng);
                locList.add(destLatLng1);

                int setColor = Color.BLUE;
                if (magnitude < 3) {
                    setColor = Color.GREEN;
                } else if (magnitude >= 3 && magnitude < 5) {
                    setColor = Color.YELLOW;
                } else if (magnitude >= 5) {
                    setColor = Color.RED;
                }

                Polyline pl = myMap.addPolyline((new PolylineOptions()).addAll(locList)
                        .width(15)
                        .color(setColor)
                        .geodesic(false));
                pl.setClickable(true);


                myMap.setOnPolylineClickListener(new GoogleMap.OnPolylineClickListener() {
```

```java
                                @Override
                                public void onPolylineClick(Polyline polyline) {

                                    String val = " Hi ";
                                    if (MainActivity.bannerText != null) {
                                        val = MainActivity.bannerText;
                                    }

                                    try {
                                        Toast toast = Toast.makeText(getBaseContext(), val, Toast.LENGTH_LONG);
                                        toast.setGravity(Gravity.CENTER, 0, 0);
                                        toast.show();
                                    } catch (Exception e) {
                                        Toast.makeText(getApplicationContext(), "Oh Gautam!", Toast.LENGTH_LONG).show();
                                    }

                                }
                            });
                        }

                    }

                } catch (Exception e) {
                    OnLineTracker.catchException(e);
                    Toast.makeText(this, getString(R.string.MapCanNotBeDisplayed), Toast.LENGTH_LONG).show();
                    finish();
                }


            }


            @Override
            public void onMapReady(GoogleMap gmap) {
                //DO WHATEVER YOU WANT WITH GOOGLEMAP
                myMap = gmap;
                if (myMap != null) {
```

```
            int resultCode = GooglePlayServicesUtil.isGooglePlayServicesAvailable(getApplicationContext());

            if (resultCode != ConnectionResult.SUCCESS) {
                Dialog dialog = GooglePlayServicesUtil.getErrorDialog(resultCode, this, 1);
                dialog.show();
                return;
            } else {
                setUpMap();
            }
        }

    }

    @Override
    public void onMapClick(LatLng latLng) {

    }

    @Override
    public void onMapLongClick(LatLng latLng) {

    }

    @Override
    public boolean onMarkerClick(Marker marker) {
        return false;
    }

    @Override
    public void onMarkerDragStart(Marker marker) {

    }

    @Override
    public void onMarkerDrag(Marker marker) {

    }
```

```
        @Override
        public void onMarkerDragEnd(Marker marker) {

        }

        @Override
        public void onBackPressed() {
            finish();
        }


    }
```

com_liveEarthquakesAlerts_view_SettingsActivity.java

Raw

```
    package com.liveEarthquakesAlerts.view;

    import android.os.Bundle;
    import android.support.v7.app.AppCompatActivity;
    import android.support.v7.widget.Toolbar;
    import android.view.MenuItem;

    import com.liveEarthquakesAlerts.R;

    public class SettingsActivity extends AppCompatActivity {

        @Override
        protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_settings);

            Toolbar mToolbar = (Toolbar) findViewById(R.id.toolbar2);
            mToolbar.setTitle("Settings");

            setSupportActionBar(mToolbar);

            getSupportActionBar().setDisplayHomeAsUpEnabled(true);
            getSupportActionBar().setHomeButtonEnabled(true);


    ////        remove the left margin from the logo
            mToolbar.setPadding(2, 0, 0, 0);//for tab otherwise give space in tab
            mToolbar.setContentInsetsAbsolute(0, 0);

            setSupportActionBar(mToolbar);



            // Display the fragment as the main content
            getFragmentManager().beginTransaction()
                    .replace(R.id.blankFragment, new SettingsFragment())
                    .commit();



        }

        @Override
        public boolean onOptionsItemSelected(MenuItem menuItem) {
            return super.onOptionsItemSelected(menuItem);
        }

    }
```

com_liveEarthquakesAlerts_view_SettingsFragment.java Raw

```
    package com.liveEarthquakesAlerts.view;

    import android.content.Intent;
    import android.content.SharedPreferences;
    import android.os.Bundle;
    import android.preference.CheckBoxPreference;
    import android.preference.EditTextPreference;
    import android.preference.ListPreference;
    import android.preference.MultiSelectListPreference;
    import android.preference.Preference;
    import android.preference.PreferenceCategory;
```

```
import android.preference.PreferenceFragment;
import android.preference.PreferenceGroup;
import android.util.Log;
import android.view.MenuItem;

import com.liveEarthquakesAlerts.R;
import com.liveEarthquakesAlerts.controller.utils.AppSettings;


/**
 * Created by  Uddhav Gautam  on 1/5/17.
 */
public class SettingsFragment extends PreferenceFragment implements Preference.OnPreferenceChangeListener, SharedPreferences.OnS
    private String Key_TimeInterval, Key_Magnitude, Key_Proximity;
    private String Key_Sorting, Key_EmergencyPhoneContactEnabled, Key_Notifications, Key_Vibration, Key_Sound;
    private ListPreference lpTimeInterval, lpMagnitude, lpSorting, lpProximity;
    private CheckBoxPreference cbNotifications, cbVibration, cbSound, cbEmergencyContacts;

    @SuppressWarnings("deprecation")
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        addPreferencesFromResource(R.xml.pref);

        Key_Proximity = getResources().getString(R.string.listPref_Key_Proximity);

        Key_TimeInterval = getResources().getString(R.string.listPref_Key_TimeInterval);
        Key_Magnitude = getResources().getString(R.string.listPref_Key_Magnitude);
        Key_Sorting = getResources().getString(R.string.listPref_Key_Sorting);
        Key_Notifications = getResources().getString(R.string.CheckBoxPref_Key_Notifications);
        Key_Vibration = getResources().getString(R.string.CheckBoxPref_Key_Vibration);
        Key_Sound = getResources().getString(R.string.CheckBoxPref_Key_Sound);
        Key_EmergencyPhoneContactEnabled = getResources().getString(R.string.CheckBoxPref_Key_Phone);

        lpProximity = (ListPreference) findPreference(Key_Proximity);

        lpTimeInterval = (ListPreference) findPreference(Key_TimeInterval);
        lpMagnitude = (ListPreference) findPreference(Key_Magnitude);
```

```
            lpSorting = (ListPreference) findPreference(Key_Sorting);
            cbNotifications = (CheckBoxPreference) findPreference(Key_Notifications);
            cbVibration = (CheckBoxPreference) findPreference(Key_Vibration);
            cbSound = (CheckBoxPreference) findPreference(Key_Sound);
            cbEmergencyContacts = (CheckBoxPreference) findPreference(Key_EmergencyPhoneContactEnabled);


            lpProximity.setOnPreferenceChangeListener(this);

            lpTimeInterval.setOnPreferenceChangeListener(this);
            lpMagnitude.setOnPreferenceChangeListener(this);
            cbNotifications.setOnPreferenceChangeListener(this);
            cbVibration.setOnPreferenceChangeListener(this);
            cbSound.setOnPreferenceChangeListener(this);
            cbEmergencyContacts.setOnPreferenceChangeListener(this);

            initSummary(getPreferenceScreen());


            if (cbNotifications.isChecked()) {
                cbVibration.setEnabled(true);
                cbSound.setEnabled(true);
                cbEmergencyContacts.setEnabled(true);
            } else {
                cbVibration.setEnabled(false);
                cbSound.setEnabled(false);
                cbEmergencyContacts.setEnabled(false);
            }
        }

        @Override
        public void onResume() {
            super.onResume();
            getPreferenceScreen().getSharedPreferences().registerOnSharedPreferenceChangeListener(this);
        }

        @Override
        public void onPause() {
            super.onPause();
            getPreferenceScreen().getSharedPreferences().unregisterOnSharedPreferenceChangeListener(this);
        }

        @Override
        public void onSharedPreferenceChanged(SharedPreferences sharedPreferences, String key) {
            updatePrefSummary(findPreference(key));
        }

        private void initSummary(Preference p) {
            if (p instanceof PreferenceGroup) {
                PreferenceGroup pGrp = (PreferenceGroup) p;
                for (int i = 0; i < pGrp.getPreferenceCount(); i++) {
                    initSummary(pGrp.getPreference(i));
                }
            } else {
                updatePrefSummary(p);
            }
        }

        private void updatePrefSummary(Preference p) {
            if (p instanceof ListPreference) { //if p is instance of top level
                ListPreference listPref = (ListPreference) p;
                p.setSummary(listPref.getEntry());
            }

            if (p instanceof MultiSelectListPreference) {
                EditTextPreference editTextPref = (EditTextPreference) p;
                p.setSummary(editTextPref.getText());
            }

            if (p instanceof CheckBoxPreference) { //preference p can be instantiated from CheckBoxPreference
                if (p instanceof PreferenceCategory) {
                    PreferenceCategory pCat = (PreferenceCategory) p;
                    if (pCat.getTitle().equals("Notifications")) {
                        CheckBoxPreference checkBoxPref = (CheckBoxPreference) p;
                        p.setSummary(checkBoxPref.isChecked() ? getResources().getString(R.string.statu_on) : getResources().getStri
                    }
                }
            }
        }

        @Override
        public boolean onOptionsItemSelected(MenuItem menuItem) {
            if (AppSettings.getInstance().isFavourite()) {
                Intent i = new Intent(getActivity(), MainActivity.class);
                i.putExtra("str", true); //(key,value)
                getActivity().setResult(getActivity().RESULT_OK, i); //100 is request code
                getActivity().finish();
            }
            return super.onOptionsItemSelected(menuItem);
        }

        @Override
        public boolean onPreferenceChange(Preference preference, Object newValue) {
            String key = preference.getKey();
            String value = newValue.toString();
            if (key.equalsIgnoreCase(Key_TimeInterval)) { //time interval
                lpTimeInterval.setSummary(lpTimeInterval.getEntries()[Integer.parseInt(value)]);
            } else if (key.equalsIgnoreCase(Key_Proximity)) { //proximity
                lpProximity.setSummary(lpProximity.getEntries()[Integer.parseInt(value)]);
                Log.i("Summary", lpProximity.getEntries()[Integer.parseInt(value)].toString());
            } else if (key.equalsIgnoreCase(Key_Magnitude)) { //magnitude
                lpMagnitude.setSummary(lpMagnitude.getEntries()[Integer.parseInt(value)]);
            } else if (key.equalsIgnoreCase(Key_Sorting)) { //sorting
                lpSorting.setSummary(lpSorting.getEntries()[Integer.parseInt(value)]);
            } else if (key.equalsIgnoreCase(Key_Notifications)) {
                if (value.equals("true")) {
                    cbNotifications.setSummary(getResources().getString(R.string.statu_on));
                    cbVibration.setEnabled(true);
                    cbSound.setEnabled(true);
                    cbEmergencyContacts.setEnabled(true);
                } else {
                    cbNotifications.setSummary(getResources().getString(R.string.statu_off));
                    cbVibration.setEnabled(false);
                    cbVibration.setChecked(false);
```

```
                        cbVibration.setSummary(getResources().getString(R.string.statu_off));
                        cbSound.setEnabled(false);
                        cbSound.setChecked(false);
                        cbSound.setSummary(getResources().getString(R.string.statu_off));
                        cbEmergencyContacts.setEnabled(false);
                        cbEmergencyContacts.setChecked(false);
                        cbEmergencyContacts.setSummary(getResources().getString(R.string.statu_off));
                    }
                } else if (key.equalsIgnoreCase(Key_Vibration)) {
                    if (value.equals("true")) {
                        cbVibration.setSummary(getResources().getString(R.string.statu_on));
                    } else {
                        cbVibration.setSummary(getResources().getString(R.string.statu_off));
                    }
                } else if (key.equalsIgnoreCase(Key_Sound)) {
                    if (value.equals("true")) {
                        cbSound.setSummary(getResources().getString(R.string.statu_on));
                    } else {
                        cbSound.setSummary(getResources().getString(R.string.statu_off));
                    }
                } else if (key.equalsIgnoreCase(Key_EmergencyPhoneContactEnabled)) {
                    if (value.equals("true")) {
                        cbEmergencyContacts.setSummary(getResources().getString(R.string.sync_on));
                    } else {
                        cbEmergencyContacts.setSummary(getResources().getString(R.string.sync_off));
                    }
                }
                return true;
            }
        }
```

com_odoo_AddContact.java Raw

```
        package com.odoo;

        import android.content.ContentValues;
        import android.content.Intent;
```

```java
import android.graphics.Bitmap;
import android.os.Bundle;
import android.support.v4.app.NavUtils;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.text.TextUtils;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.CheckBox;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.Toast;

import com.liveEarthquakesAlerts.R;
import com.odoo.table.ResPartner;
import com.odoo.utils.BitmapUtils;

public class AddContact extends AppCompatActivity implements View.OnClickListener {

    private Toolbar toolbar;
    private EditText editName, editMobileNumber, editPhoneNumber, editCity, editEmail, editState, editCountry,
            editPincode, editWebsite, editFax, editStreet, editStreet2;
    private String imageString = "null";
    private ResPartner resPartner;
    private ImageView profileImage;
    private CheckBox checkBoxIsCompany;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_add_contact);

        toolbar = (Toolbar) findViewById(R.id.profile_toolbar);
        setSupportActionBar(toolbar);
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        getSupportActionBar().setDisplayShowHomeEnabled(true);
        init();
    }

    private void init() {

        resPartner = new ResPartner(this);

        profileImage = (ImageView) findViewById(R.id.avatar);
        profileImage.setOnClickListener(this);

        editName = (EditText) findViewById(R.id.editName);
        editMobileNumber = (EditText) findViewById(R.id.editMobileNumber);
        editPhoneNumber = (EditText) findViewById(R.id.editPhoneNumber);
        editEmail = (EditText) findViewById(R.id.editEmail);
        editStreet = (EditText) findViewById(R.id.editStreet);
        editStreet2 = (EditText) findViewById(R.id.editStreet2);
        editCity = (EditText) findViewById(R.id.editCity);
        editState = (EditText) findViewById(R.id.editState);
        editCountry = (EditText) findViewById(R.id.editCountry);
        editPincode = (EditText) findViewById(R.id.editPincode);
        editWebsite = (EditText) findViewById(R.id.editWebsite);
        editFax = (EditText) findViewById(R.id.editFax);
        checkBoxIsCompany = (CheckBox) findViewById(R.id.checkboxIsCompany);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.add_contact_menu, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {

        if (item.getItemId() == R.id.menuSave) {

            editName.setError(null);
            if (TextUtils.isEmpty(editName.getText())) {
                editName.setError("Name Required");
                editName.requestFocus();
                return true;
            }

            ContentValues values = new ContentValues();

            values.put("name", editName.getText().toString());

            if (imageString.equals("") || imageString.equals("null")) {
                values.put("image_medium", "false");
            } else {
                values.put("image_medium", imageString);
            }

            if (checkBoxIsCompany.isChecked()) {
                values.put("company_type", "company");
            } else {
                values.put("company_type", "person");
            }

            if (editMobileNumber.getText().toString().equals("")) {
                values.put("mobile", "false");
            } else {
                values.put("mobile", editMobileNumber.getText().toString());
            }

            if (editPhoneNumber.getText().toString().equals("")) {
                values.put("phone", "false");
            } else {
                values.put("phone", editPhoneNumber.getText().toString());
            }

            if (editCity.getText().toString().equals("")) {
                values.put("city", "false");
            } else {
                values.put("city", editCity.getText().toString());
            }
```

```java
                    if (editStreet.getText().toString().equals("")) {
                        values.put("street", "false");
                    } else {
                        values.put("street", editStreet.getText().toString());
                    }

                    if (editStreet2.getText().toString().equals("")) {
                        values.put("street2", "false");
                    } else {
                        values.put("street2", editStreet2.getText().toString());
                    }

                    if (editEmail.getText().toString().equals("")) {
                        values.put("email", "false");
                    } else {
                        values.put("email", editEmail.getText().toString());
                    }

                    if (editWebsite.getText().toString().equals("")) {
                        values.put("website", "false");
                    } else {
                        values.put("website", editWebsite.getText().toString());
                    }

                    if (editState.getText().toString().equals("")) {
                        values.put("state_id", "0");
                    } else {
                        //TODO: state name

                        values.put("state_id", "1");
                    }

                    if (editCountry.getText().toString().equals("")) {
                        values.put("country_id", "0");
                    } else {
                        //TODO: Country name
                        values.put("country_id", "1");
                    }

                    if (editFax.getText().toString().equals("")) {
                        values.put("fax", "false");
                    } else {
                        values.put("fax", editFax.getText().toString());
                    }

                    if (editPincode.getText().toString().equals("")) {
                        values.put("zip", "false");
                    } else {
                        values.put("zip", editPincode.getText().toString());
                    }

                    resPartner.create(values);
                    Toast.makeText(this, R.string.new_contact_create, Toast.LENGTH_SHORT).show();
                    finish();
                }

                if (item.getItemId() == android.R.id.home) {
                    finish();
                }
                return super.onOptionsItemSelected(item);
            }

            @Override
            public void onClick(View v) {
                selectImage();
            }

            private void selectImage() {
                Intent intent = new Intent();

                intent.setType("image/*");
                intent.setAction(Intent.ACTION_GET_CONTENT);

                intent.putExtra("crop", "true");
                intent.putExtra("aspectX", 0);
                intent.putExtra("aspectY", 0);
                intent.putExtra("outputX", 200);
                intent.putExtra("outputY", 200);

                try {
                    intent.putExtra("return-data", true);
                    startActivityForResult(Intent.createChooser(intent,
                            "Complete action using"), 1);

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

            @Override
            protected void onActivityResult(int requestCode, int resultCode, Intent data) {
                super.onActivityResult(requestCode, resultCode, data);

                if (requestCode == 1) {

                    if (data != null) {
                        Bundle extras = data.getExtras();
                        if (extras != null) {
                            Bitmap bitmap = extras.getParcelable("data");
                            profileImage.setImageBitmap(bitmap);
                            imageString = BitmapUtils.bitmapToBase64(bitmap);
                        }
                    } else {
                        NavUtils.getParentActivityIntent(this);
                    }
                }
            }
        }
```

com_odoo_auth_OdooAuthenticator.java Raw

```
        package com.odoo.auth;
```

```
import android.accounts.AbstractAccountAuthenticator;
import android.accounts.Account;
import android.accounts.AccountAuthenticatorResponse;
import android.accounts.AccountManager;
import android.accounts.NetworkErrorException;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;

import com.odoo.LoginActivity;

/**
 * Created by  Uddhav Gautam  on 19/4/16.
 */
public class OdooAuthenticator extends AbstractAccountAuthenticator {

    public static final String AUTH_TYPE = "com.odoo.contacts.auth";
    private Context mContext;

    public OdooAuthenticator(Context context) {
        super(context);
        mContext = context;
    }

    @Override
    public Bundle editProperties(AccountAuthenticatorResponse response, String accountType) {
        return null;
    }

    @Override
    public Bundle addAccount(AccountAuthenticatorResponse response, String accountType, String authTokenType, String[] requiredF
        Bundle data = new Bundle();
        data.putParcelable(AccountManager.KEY_INTENT, new Intent(mContext, LoginActivity.class));

        return data;
    }

    @Override
```

```
        public Bundle confirmCredentials(AccountAuthenticatorResponse response, Account account, Bundle options) throws NetworkError
            return null;
        }

        @Override
        public Bundle getAuthToken(AccountAuthenticatorResponse response, Account account, String authTokenType, Bundle options) thr
            return null;
        }

        @Override
        public String getAuthTokenLabel(String authTokenType) {
            return null;
        }

        @Override
        public Bundle updateCredentials(AccountAuthenticatorResponse response, Account account, String authTokenType, Bundle options
            return null;
        }

        @Override
        public Bundle hasFeatures(AccountAuthenticatorResponse response, Account account, String[] features) throws NetworkErrorExce
            return null;
        }
    }
```

com_odoo_auth_OdooAuthenticatorServices.java Raw

```
    package com.odoo.auth;

    import android.app.Service;
    import android.content.Intent;
    import android.os.IBinder;
    import android.support.annotation.Nullable;

    /**
     * Created by  Uddhav Gautam  on 19/4/16.
     */
```

```
public class OdooAuthenticatorServices extends Service {

    private static final Object mAuthenticatorLock = new Object();
    private OdooAuthenticator authenticator;

    @Override
    public void onCreate() {
        super.onCreate();
        synchronized (mAuthenticatorLock) {
            if (authenticator == null) {
                authenticator = new OdooAuthenticator(getApplicationContext());
            }
        }
    }

    @Nullable
    @Override
    public IBinder onBind(Intent intent) {
        return authenticator.getIBinder();
    }
}
```

com_odoo_ContactDetailActivity.java Raw

```
package com.odoo;

import android.Manifest;
import android.accounts.Account;
import android.accounts.AccountManager;
import android.content.ContentProviderOperation;
import android.content.ContentProviderResult;
import android.content.ContentValues;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.OperationApplicationException;
import android.content.pm.PackageManager;
import android.graphics.Bitmap;
import android.net.Uri;
import android.os.Build;
import android.os.Bundle;
import android.os.RemoteException;
import android.provider.ContactsContract;
import android.provider.ContactsContract.CommonDataKinds.Phone;
import android.provider.ContactsContract.Contacts.Data;
import android.support.annotation.NonNull;
import android.support.design.widget.CollapsingToolbarLayout;
import android.support.design.widget.CoordinatorLayout;
import android.support.design.widget.FloatingActionButton;
import android.support.design.widget.Snackbar;
import android.support.v4.app.ActivityCompat;
import android.support.v4.app.NavUtils;
import android.support.v7.app.AlertDialog;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.RelativeLayout;
import android.widget.TextView;
import android.widget.Toast;

import com.liveEarthquakesAlerts.R;
import com.odoo.auth.OdooAuthenticator;
import com.odoo.orm.ListRow;
import com.odoo.table.ResCountry;
import com.odoo.table.ResPartner;
import com.odoo.table.ResState;
import com.odoo.utils.BitmapUtils;

import java.io.ByteArrayOutputStream;
import java.util.ArrayList;
import java.util.List;
```

```
public class ContactDetailActivity extends AppCompatActivity implements View.OnClickListener {

    private static final int REQUEST_CODE_ASK_PERMISSIONS_WRITE_CONTACT = 11;
    private static final int REQUEST_CODE_ASK_PERMISSIONS_CALL_CONTACT = 22;
    private static final int REQUEST_CODE_ASK_PERMISSIONS_SEND_SMS = 33;
    private ResPartner resPartner;
    private ResState resState;
    private ResCountry resCountry;
    private Toolbar toolbar;
    private CollapsingToolbarLayout collapsingToolbarLayout;
    private FloatingActionButton fabEdit;
    private TextView textMobileNumber, textPhoneNumber, textEmail, textStreet, textStreet2,
            textCity, textState, textCountry, textPincode, textWebsite, textFax;
    private ImageView profileImage, callImage;
    private LinearLayout contactNumberLayout, emailLayout, addressLayout, websiteLayout, faxLayout;
    private RelativeLayout mobileLayout, phoneLayout;
    private EditText editMobileNumber, editPhoneNumber, editCity, editEmail, editState, editCountry,
            editPincode, editWebsite, editFax, editStreet, editStreet2;
    private String stringName, stringMobileNumber, stringPhoneNumber, stringEmail, stringStreet, stringStreet2,
            stringCity, stringPincode, stringStateId, stringStateName, stringCountryId,
            stringCountryName, stringWebsite, stringFax, stringImage;
    private int _id;
    private String address;
    private CoordinatorLayout coordinatorLayout;
    private LinearLayout viewLayout, editLayout;
    private Intent dial;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.contact_detail_activity);

        toolbar = (Toolbar) findViewById(R.id.profile_toolbar);
        setSupportActionBar(toolbar);
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        getSupportActionBar().setDisplayShowHomeEnabled(true);

        coordinatorLayout = (CoordinatorLayout) findViewById(R.id.rootLayout);
        collapsingToolbarLayout = (CollapsingToolbarLayout) findViewById(R.id.profile_collapsing);

        fabEdit = (FloatingActionButton) findViewById(R.id.fabEdit);
        fabEdit.setOnClickListener(this);

        resState = new ResState(this);
        resCountry = new ResCountry(this);

        init();

        _id = getIntent().getIntExtra("id", 0);
        resPartner = new ResPartner(this);
        List<ListRow> rows = resPartner.select("_id = ?", String.valueOf(_id));
        for (ListRow row : rows) {

            stringName = row.getString("name");
            stringMobileNumber = row.getString("mobile");
            stringPhoneNumber = row.getString("phone");
            stringEmail = row.getString("email");
            stringStreet = row.getString("street");
            stringStreet2 = row.getString("street2");
            stringCity = row.getString("city");
            stringPincode = row.getString("zip");
            stringStateId = row.getString("state_id");
            stringCountryId = row.getString("country_id");
            stringFax = row.getString("fax");
            stringWebsite = row.getString("website");
            stringImage = row.getString("image_medium");

            //TODO: state_name and Country_name from id
            stringStateName = "false";
            stringCountryName = "false";

            collapsingToolbarLayout.setTitle(row.getString("name"));
            //contact number
            textMobileNumber.setText(stringMobileNumber);
            if (stringMobileNumber.equals("false") && !stringPhoneNumber.equals("false")) {
                mobileLayout.setVisibility(View.GONE);
                callImage.setVisibility(View.VISIBLE);
            }

            if (stringMobileNumber.equals("false")) {
                mobileLayout.setVisibility(View.GONE);
            }

            textPhoneNumber.setText(stringPhoneNumber);
            if (stringPhoneNumber.equals("false")) {
                phoneLayout.setVisibility(View.GONE);
            }

            if (stringMobileNumber.equals("false") && stringPhoneNumber.equals("false")) {
                contactNumberLayout.setVisibility(View.GONE);
            }

            //email
            textEmail.setText(stringEmail);
            if (stringEmail.equals("false")) {
                emailLayout.setVisibility(View.GONE);
            }

            //address
            textStreet.setText(stringStreet);
            textStreet.setVisibility(stringStreet.equals("false") ? View.GONE : View.VISIBLE);

            textStreet2.setText(stringStreet2);
            textStreet2.setVisibility(stringStreet2.equals("false") ? View.GONE : View.VISIBLE);

            textCity.setText(stringCity);
            textCity.setVisibility(stringCity.equals("false") ? View.GONE : View.VISIBLE);

            textPincode.setText(stringPincode);
            textPincode.setVisibility(stringPincode.equals("false") ? View.GONE : View.VISIBLE);

            textState.setText(stringStateId);
            textState.setVisibility(stringStateId.equals("0") ? View.GONE : View.VISIBLE);
            textCountry.setText(stringCountryId);
            textCountry.setVisibility(stringCountryId.equals("0") ? View.GONE : View.VISIBLE);

            if (stringStreet.equals("false") && stringStreet2.equals("false") &&
                    stringCity.equals("false") && stringPincode.equals("false") &&
                    stringStateId.equals("0") && stringCountryId.equals("0")) {
                addressLayout.setVisibility(View.GONE);
            }

            //website
            textWebsite.setText(stringWebsite);
            if (stringWebsite.equals("false")) {
                websiteLayout.setVisibility(View.GONE);
            }

            //fax
            textFax.setText(stringFax);
            if (stringFax.equals("false")) {
                faxLayout.setVisibility(View.GONE);
            }

            //profile image
            if (!stringImage.equals("false")) {
                profileImage.setImageBitmap(BitmapUtils.getBitmapImage(this, stringImage));
            } else {
                profileImage.setImageBitmap(BitmapUtils.getAlphabetImage(this,
                        row.getString("name")));
            }

        }
    }

    private void init() {

        textMobileNumber = (TextView) findViewById(R.id.textMobileNumber);
        textPhoneNumber = (TextView) findViewById(R.id.textPhoneNumber);
        textEmail = (TextView) findViewById(R.id.textEmail);
        textCity = (TextView) findViewById(R.id.textCity);
        textStreet = (TextView) findViewById(R.id.textStreet);
        textStreet2 = (TextView) findViewById(R.id.textStreet2);
        textState = (TextView) findViewById(R.id.textState);
        textCountry = (TextView) findViewById(R.id.textCountry);
        textWebsite = (TextView) findViewById(R.id.textWebsite);
        textFax = (TextView) findViewById(R.id.textFax);
        textPincode = (TextView) findViewById(R.id.textPincode);

        profileImage = (ImageView) findViewById(R.id.avatar);
        callImage = (ImageView) findViewById(R.id.imageCall2);

        contactNumberLayout = (LinearLayout) findViewById(R.id.contactNumberLayout);
        emailLayout = (LinearLayout) findViewById(R.id.emailLayout);
        addressLayout = (LinearLayout) findViewById(R.id.addressLayout);
        websiteLayout = (LinearLayout) findViewById(R.id.websiteLayout);
        faxLayout = (LinearLayout) findViewById(R.id.faxLayout);

        viewLayout = (LinearLayout) findViewById(R.id.viewLayout);
        editLayout = (LinearLayout) findViewById(R.id.editLayout);

        mobileLayout = (RelativeLayout) findViewById(R.id.mobileLayout);
        phoneLayout = (RelativeLayout) findViewById(R.id.phoneLayout);

        editMobileNumber = (EditText) findViewById(R.id.editMobileNumber);
        editPhoneNumber = (EditText) findViewById(R.id.editPhoneNumber);
        editEmail = (EditText) findViewById(R.id.editEmail);
        editStreet = (EditText) findViewById(R.id.editStreet);
        editStreet2 = (EditText) findViewById(R.id.editStreet2);
        editCity = (EditText) findViewById(R.id.editCity);
        editState = (EditText) findViewById(R.id.editState);
        editCountry = (EditText) findViewById(R.id.editCountry);
        editPincode = (EditText) findViewById(R.id.editPincode);
        editWebsite = (EditText) findViewById(R.id.editWebsite);
        editFax = (EditText) findViewById(R.id.editFax);

    }
```

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

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

                    fabEdit.setImageResource(R.drawable.ic_done_24dp);

                    viewLayout.setVisibility(View.GONE);
                    editLayout.setVisibility(View.VISIBLE);

                    editMobileNumber.setText(stringMobileNumber.equals("false") ? "" : stringMobileNumber);
                    editPhoneNumber.setText(stringPhoneNumber.equals("false") ? "" : stringPhoneNumber);
                    editEmail.setText(stringEmail.equals("false") ? "" : stringEmail);
                    editCity.setText(stringCity.equals("false") ? "" : stringCity);
                    editStreet.setText(stringStreet.equals("false") ? "" : stringStreet);
                    editStreet2.setText(stringStreet2.equals("false") ? "" : stringStreet2);
                    editState.setText(stringStateId.equals("0") ? "" : stringStateId);
                    editCountry.setText(stringCountryId.equals("0") ? "" : stringCountryId);
                    editWebsite.setText(stringWebsite.equals("false") ? "" : stringWebsite);
                    editFax.setText(stringFax.equals("false") ? "" : stringFax);
                    editPincode.setText(stringPincode.equals("false") ? "" : stringPincode);

                    profileImage.setClickable(true);
                    profileImage.setOnClickListener(new View.OnClickListener() {
                        @Override
                        public void onClick(View v) {
                            selectImage();
                        }
                    });

                    fabEdit.setOnClickListener(new View.OnClickListener() {
                        @Override
                        public void onClick(View v) {
                            updateRecords();
                            finish();
                        }
                    });
                }
            }

            private void updateRecords() {

                ContentValues values = new ContentValues();

                if (editMobileNumber.getText().toString().equals("")) {
                    values.put("mobile", "false");
                } else {
                    values.put("mobile", editMobileNumber.getText().toString());
                }

                if (editPhoneNumber.getText().toString().equals("")) {
                    values.put("phone", "false");
                } else {
                    values.put("phone", editPhoneNumber.getText().toString());
                }

                if (editCity.getText().toString().equals("")) {
                    values.put("city", "false");
                } else {
                    values.put("city", editCity.getText().toString());
                }

                if (editStreet.getText().toString().equals("")) {
                    values.put("street", "false");
                } else {
                    values.put("street", editStreet.getText().toString());
                }

                if (editStreet2.getText().toString().equals("")) {
                    values.put("street2", "false");
                } else {
                    values.put("street2", editStreet2.getText().toString());
                }

                if (editEmail.getText().toString().equals("")) {
                    values.put("email", "false");
                } else {
                    values.put("email", editEmail.getText().toString());
                }

                if (editWebsite.getText().toString().equals("")) {
                    values.put("website", "false");
                } else {
                    values.put("website", editWebsite.getText().toString());
                }

                if (editState.getText().toString().equals("")) {
                    values.put("state_id", "0");
                } else {
                    //TODO: state name
                    values.put("state_id", "1");
                }

                if (editCountry.getText().toString().equals("")) {
                    values.put("country_id", "0");
                } else {
                    //TODO: Country name
                    values.put("country_id", "1");
                }

                if (editFax.getText().toString().equals("")) {
                    values.put("fax", "false");
                } else {
                    values.put("fax", editFax.getText().toString());
                }

                if (editPincode.getText().toString().equals("")) {
                    values.put("zip", "false");
                } else {
                    values.put("zip", editPincode.getText().toString());
                }

                resPartner.update(values, "_id = ? ", String.valueOf(_id));
                Toast.makeText(ContactDetailActivity.this, "Contact Updated", Toast.LENGTH_SHORT).show();
```

```
            }

            private void selectImage() {
                Intent intent = new Intent();

                intent.setType("image/*");
                intent.setAction(Intent.ACTION_GET_CONTENT);

                intent.putExtra("crop", "true");
                intent.putExtra("aspectX", 0);
                intent.putExtra("aspectY", 0);
                intent.putExtra("outputX", 200);
                intent.putExtra("outputY", 200);

                try {
                    intent.putExtra("return-data", true);
                    startActivityForResult(Intent.createChooser(intent,
                            "Complete action using"), 1);

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

            @Override
            protected void onActivityResult(int requestCode, int resultCode, Intent data) {
                super.onActivityResult(requestCode, resultCode, data);

                if (requestCode == 1) {

                    if (data != null) {
                        Bundle extras = data.getExtras();
                        if (extras != null) {
                            Bitmap bitmap = extras.getParcelable("data");
                            profileImage.setImageBitmap(bitmap);
                            stringImage = BitmapUtils.bitmapToBase64(bitmap);
                            ContentValues values = new ContentValues();
                            values.put("image_medium", stringImage);
                            resPartner.update(values, "_id = ? ", String.valueOf(_id));
                        }
                    } else {
                        NavUtils.getParentActivityIntent(this);
                    }
                }
            }

            @Override
            public boolean onCreateOptionsMenu(Menu menu) {
                getMenuInflater().inflate(R.menu.menu_contact_profile, menu);
                return true;
            }

            @Override
            public boolean onOptionsItemSelected(MenuItem item) {

                int id = item.getItemId();

                switch (id) {
                    case android.R.id.home:
                        finish();
                        break;

                    case R.id.menu_call:
                        if (stringMobileNumber.equals("false")) {
                            if (stringPhoneNumber.equals("false")) {
                                Toast.makeText(this, "Number not found", Toast.LENGTH_LONG).show();
                            } else {
                                callToContact(stringPhoneNumber);
                            }
                        } else {
                            callToContact(stringMobileNumber);
                        }
                        break;

                    case R.id.menu_add_contact_to_device:
                        stringImage = stringImage.equals("false") ? "" : stringImage;
                        stringMobileNumber = stringMobileNumber.equals("false") ? "" : stringMobileNumber;
                        stringPhoneNumber = stringPhoneNumber.equals("false") ? "" : stringPhoneNumber;
                        stringEmail = stringEmail.equals("false") ? "" : stringEmail;
                        stringStreet = stringStreet.equals("false") ? "" : stringStreet;
                        stringStreet2 = stringStreet2.equals("false") ? "" : stringStreet2;
                        stringCity = stringCity.equals("false") ? "" : stringCity;
                        stringCountryName = stringCountryName.equals("false") ? "" : stringCountryName;
                        stringWebsite = stringWebsite.equals("false") ? "" : stringWebsite;
                        stringFax = stringFax.equals("false") ? "" : stringFax;
                        stringPincode = stringPincode.equals("false") ? "" : stringPincode;

                        addContactToDevice(stringName, stringImage, stringMobileNumber, stringPhoneNumber, stringEmail,
                                stringStreet, stringStreet2, stringCity, stringCountryName, stringFax,
                                stringWebsite, stringPincode);

                        break;

                    case R.id.menu_send_message:
                        if (stringMobileNumber.equals("false")) {
                            if (stringPhoneNumber.equals("false")) {
                                Toast.makeText(this, "Number not found", Toast.LENGTH_LONG).show();
                            } else {
                                sendMessage(stringPhoneNumber);
                            }
                        } else {
                            sendMessage(stringMobileNumber);
                        }
                        break;

                    case R.id.menu_send_mail:
                        if (stringEmail.equals("false")) {
                            Toast.makeText(this, "Email not found", Toast.LENGTH_LONG).show();
                        } else {
                            Intent mailIntent = new Intent(Intent.ACTION_SEND);
                            mailIntent.setData(Uri.parse("mailto:"));
                            mailIntent.setType("text/plain");
                            mailIntent.putExtra(Intent.EXTRA_EMAIL, new String[]{stringEmail});
                            startActivity(mailIntent);
                        }
                        break;

                    case R.id.menu_delete:

                        AlertDialog.Builder builder = new AlertDialog.Builder(this);
                        builder.setTitle("You want to delete contact ?");
                        builder.setPositiveButton("Ok", new DialogInterface.OnClickListener() {
                            @Override
                            public void onClick(DialogInterface dialog, int which) {
                                resPartner.delete("_id = ? ", String.valueOf(_id));
                                Toast.makeText(ContactDetailActivity.this, "Contact Deleted", Toast.LENGTH_LONG).show();
                                finish();
                            }
                        });
                        builder.setNegativeButton("Cancle", new DialogInterface.OnClickListener() {
                            @Override
                            public void onClick(DialogInterface dialog, int which) {
                                dialog.dismiss();
                            }
                        });
                        builder.create().show();

                        break;
                }

                return super.onOptionsItemSelected(item);
            }

            private void addContactToDevice(String stringName, String stringImage, String stringMobileNumber,
                                            String stringPhoneNumber, String stringEmail, String stringStreet,
                                            String stringStreet2, String stringCity, String stringCountryName,
                                            String stringFax, String stringWebsite, String stringPincode) {

                ArrayList<ContentProviderOperation> ops = new ArrayList<>();
                int rawContactInsertIndex = 0;

                ops.add(ContentProviderOperation.newInsert(ContactsContract.RawContacts.CONTENT_URI)
                        .withValue(ContactsContract.RawContacts.ACCOUNT_TYPE, OdooAuthenticator.AUTH_TYPE)
                        .withValue(ContactsContract.RawContacts.ACCOUNT_NAME, getAccount().name).build());

                // Display name
                ops.add(ContentProviderOperation
                        .newInsert(ContactsContract.Data.CONTENT_URI)
                        .withValueBackReference(Data.RAW_CONTACT_ID, rawContactInsertIndex)
                        .withValue(Data.MIMETYPE,
                                ContactsContract.CommonDataKinds.StructuredName.CONTENT_ITEM_TYPE)
                        .withValue(ContactsContract.CommonDataKinds.StructuredName.DISPLAY_NAME,
                                stringName)
                        .build());

                // avatar
                if (!stringImage.equals("false") && !stringImage.isEmpty()) {
                    Bitmap bitmap = BitmapUtils.getBitmapImage(this, stringImage);
                    ByteArrayOutputStream baos = new ByteArrayOutputStream();
                    bitmap.compress(Bitmap.CompressFormat.JPEG, 80, baos);

                    ops.add(ContentProviderOperation
                            .newInsert(ContactsContract.Data.CONTENT_URI)
                            .withValueBackReference(Data.RAW_CONTACT_ID, rawContactInsertIndex)
                            .withValue(Data.MIMETYPE,
                                    ContactsContract.CommonDataKinds.Photo.CONTENT_ITEM_TYPE)
                            .withValue(ContactsContract.CommonDataKinds.Photo.PHOTO,
                                    baos.toByteArray()).build());
                }

                // Mobile number
                ops.add(ContentProviderOperation
                        .newInsert(ContactsContract.Data.CONTENT_URI)
                        .withValueBackReference(ContactsContract.Data.RAW_CONTACT_ID,
                                rawContactInsertIndex)
                        .withValue(Data.MIMETYPE, Phone.CONTENT_ITEM_TYPE)
                        .withValue(Phone.NUMBER, stringMobileNumber)
                        .withValue(Phone.TYPE, Phone.TYPE_MOBILE).build());

                // Phone number
```

```
                ops.add(ContentProviderOperation.newInsert(ContactsContract.Data.CONTENT_URI)
                        .withValueBackReference(ContactsContract.Data.RAW_CONTACT_ID, rawContactInsertIndex)
                        .withValue(ContactsContract.Data.MIMETYPE, Phone.CONTENT_ITEM_TYPE)
                        .withValue(Phone.NUMBER, stringPhoneNumber)
                        .withValue(Phone.TYPE, Phone.TYPE_HOME)
                        .build());

                // Fax number
                ops.add(ContentProviderOperation.newInsert(ContactsContract.Data.CONTENT_URI)
                        .withValueBackReference(ContactsContract.Data.RAW_CONTACT_ID, rawContactInsertIndex)
                        .withValue(ContactsContract.Data.MIMETYPE, Phone.CONTENT_ITEM_TYPE)
                        .withValue(Phone.NUMBER, stringFax)
                        .withValue(Phone.TYPE, Phone.TYPE_OTHER_FAX)
                        .build());

                // Email
                ops.add(ContentProviderOperation
                        .newInsert(ContactsContract.Data.CONTENT_URI)
                        .withValueBackReference(ContactsContract.Data.RAW_CONTACT_ID,
                                rawContactInsertIndex)
                        .withValue(Data.MIMETYPE, ContactsContract.CommonDataKinds.Email.CONTENT_ITEM_TYPE)
                        .withValue(ContactsContract.CommonDataKinds.Email.DATA, stringEmail)
                        .withValue(ContactsContract.CommonDataKinds.Email.TYPE,
                                ContactsContract.CommonDataKinds.Email.TYPE_WORK).build());

                // Website
                ops.add(ContentProviderOperation
                        .newInsert(ContactsContract.Data.CONTENT_URI)
                        .withValueBackReference(ContactsContract.Data.RAW_CONTACT_ID,
                                rawContactInsertIndex)
                        .withValue(Data.MIMETYPE, ContactsContract.CommonDataKinds.Website.CONTENT_ITEM_TYPE)
                        .withValue(ContactsContract.CommonDataKinds.Website.URL, stringWebsite)
                        .withValue(ContactsContract.CommonDataKinds.Website.TYPE,
                                ContactsContract.CommonDataKinds.Website.TYPE_HOME).build());

                // address, city, zip, county

                address = String.valueOf(new StringBuilder(stringStreet).append(", ").append(stringStreet2));
                if (stringStreet.equals("")) {
                    if (stringStreet2.equals("")) {
                        address = "";
                    } else {
                        address = stringStreet2;
                    }
                } else {
                    address = stringStreet;
                }

                ops.add(ContentProviderOperation
                        .newInsert(ContactsContract.Data.CONTENT_URI)
                        .withValueBackReference(ContactsContract.Data.RAW_CONTACT_ID,
                                rawContactInsertIndex)
                        .withValue(Data.MIMETYPE,
                                ContactsContract.CommonDataKinds.StructuredPostal.CONTENT_ITEM_TYPE)
                        .withValue(ContactsContract.CommonDataKinds.StructuredPostal.STREET, address)
                        .withValue(Data.MIMETYPE,
                                ContactsContract.CommonDataKinds.StructuredPostal.CONTENT_ITEM_TYPE)
                        .withValue(ContactsContract.CommonDataKinds.StructuredPostal.CITY,
                                stringCity)
                        .withValue(Data.MIMETYPE,
                                ContactsContract.CommonDataKinds.StructuredPostal.CONTENT_ITEM_TYPE)
                        .withValue(ContactsContract.CommonDataKinds.StructuredPostal.POSTCODE,
                                stringPincode)
                        //TODO : country name
                        /*.withValue(ContactsContract.CommonDataKinds.StructuredPostal.COUNTRY,
                                stringCountryName)*/
                        .build());

                try {
                    if (ActivityCompat.checkSelfPermission(this, Manifest.permission.WRITE_CONTACTS) != PackageManager.PERMISSION_GRANTE
                        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
                            requestPermissions(new String[]{Manifest.permission.WRITE_CONTACTS}, REQUEST_CODE_ASK_PERMISSIONS_WRITE_CONT
                        }
                    } else {
                        ContentProviderResult[] res = this.getContentResolver().applyBatch(
                                ContactsContract.AUTHORITY, ops);
                        if (res.length > 0) {
                            final ContentProviderResult result = res[0];
                            Snackbar.make(coordinatorLayout, R.string.contact_created, Snackbar.LENGTH_LONG)
                                    .setAction(R.string.label_view, new View.OnClickListener() {
                                        @Override
                                        public void onClick(View v) {
                                            Intent intent = new Intent(Intent.ACTION_VIEW, result.uri);
                                            startActivity(intent);
                                        }
                                    }).show();

                        }
                    }
                } catch (RemoteException | OperationApplicationException e) {
                    e.printStackTrace();
                }
            }

            public void sendMessage(String number) {
                Intent smsIntent = new Intent(Intent.ACTION_VIEW);
                smsIntent.setType("vnd.android-dir/mms-sms");
                smsIntent.putExtra("address", number);
                smsIntent.putExtra("sms_body", "");
                if (ActivityCompat.checkSelfPermission(this, Manifest.permission.SEND_SMS) != PackageManager.PERMISSION_GRANTED) {
                    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
                        requestPermissions(new String[]{Manifest.permission.SEND_SMS}, REQUEST_CODE_ASK_PERMISSIONS_SEND_SMS);
                    }
                } else {
                    startActivity(smsIntent);
                }
            }

            public void callToContact(String number) {
                Uri phoneCall;
                phoneCall = Uri.parse("tel:" + number);
                dial = new Intent(Intent.ACTION_CALL, phoneCall);
                if (ActivityCompat.checkSelfPermission(this, Manifest.permission.CALL_PHONE) != PackageManager.PERMISSION_GRANTED) {
                    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
                        requestPermissions(new String[]{Manifest.permission.CALL_PHONE}, REQUEST_CODE_ASK_PERMISSIONS_CALL_CONTACT);
                    }
                } else {
                    startActivity(dial);
                }
            }

            private Account getAccount() {
                AccountManager accountManager = (AccountManager) getSystemService(ACCOUNT_SERVICE);
                Account[] accounts = accountManager.getAccountsByType(OdooAuthenticator.AUTH_TYPE);
                if (accounts.length == 1) {
                    return accounts[0];
                }
                return null;
            }

            @Override
            public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {
                switch (requestCode) {
                    case REQUEST_CODE_ASK_PERMISSIONS_CALL_CONTACT:
                        startActivity(dial);
                        break;
                }
                super.onRequestPermissionsResult(requestCode, permissions, grantResults);
            }
        }
```

com_odoo_ContactFragment.java Raw

```
    package com.odoo;

    import android.Manifest;
    import android.annotation.TargetApi;
    import android.content.ContentValues;
    import android.content.DialogInterface;
    import android.content.Intent;
    import android.content.pm.PackageManager;
```

```java
import android.database.Cursor;
import android.net.Uri;
import android.os.Build;
import android.os.Bundle;
import android.support.annotation.Nullable;
import android.support.v4.app.ActivityCompat;
import android.support.v4.app.Fragment;
import android.support.v4.app.LoaderManager;
import android.support.v4.content.CursorLoader;
import android.support.v4.content.Loader;
import android.support.v7.app.AlertDialog;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.CompoundButton;
import android.widget.ImageView;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;
import android.widget.ToggleButton;

import com.liveEarthquakesAlerts.R;
import com.odoo.orm.ListRow;
import com.odoo.orm.OListAdapter;
import com.odoo.table.RecentContact;
import com.odoo.table.ResPartner;
import com.odoo.utils.BitmapUtils;

import java.util.HashMap;

/**
 * A simple {@link Fragment} subclass.
 */
public class ContactFragment extends Fragment implements
        LoaderManager.LoaderCallbacks<Cursor>, OListAdapter.OnViewBindListener,
        AdapterView.OnItemClickListener, AdapterView.OnItemLongClickListener {
```

```
    private static final int REQUEST_CODE_ASK_PERMISSIONS_CALL_CONTACT = 11;
    private ResPartner resPartner;
    private OListAdapter oListAdapter;
    private ListView contactList;
    private RecentContact recentContact;

    private HashMap<Integer, Boolean> favToogleCache = new HashMap<>();

    public ContactFragment() {
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        return inflater.inflate(R.layout.fragment_contact, container, false);

    }

    @Override
    public void onViewCreated(View view, @Nullable Bundle savedInstanceState) {
        super.onViewCreated(view, savedInstanceState);
        resPartner = new ResPartner(getContext());
        recentContact = new RecentContact(getContext());
        contactList = (ListView) view.findViewById(R.id.contactList);
        oListAdapter = new OListAdapter(getContext(), null, R.layout.contact_list_item);
        oListAdapter.setOnViewBindListener(this);
        contactList.setAdapter(oListAdapter);
        contactList.setOnItemClickListener(this);
        contactList.setOnItemLongClickListener(this);

        getLoaderManager().initLoader(0, null, this);
    }

    @Override
    public void onResume() {
        super.onResume();
        getLoaderManager().initLoader(0, null, this);
    }

    @Override
    public void onViewBind(View view, Cursor cursor, final ListRow row) {

        TextView textContactName, textContactEmail, textContactCity, textContactNumber;
        ImageView profileImage, isCompany;

        final ToggleButton toggleFavourite = (ToggleButton) view.findViewById(R.id.toggleIsFavourite);

        textContactName = (TextView) view.findViewById(R.id.textViewName);
        textContactEmail = (TextView) view.findViewById(R.id.textViewEmail);
        textContactCity = (TextView) view.findViewById(R.id.textViewCity);
        textContactNumber = (TextView) view.findViewById(R.id.textViewContact);
        profileImage = (ImageView) view.findViewById(R.id.profile_image);
//              isCompany = (ImageView) view.findViewById(R.id.isCompany);

        String stringName, stringEmail, stringCity, stringMobile, stringImage, stringCompanyType,
                stringToggle;
        stringName = row.getString("name");
        stringEmail = row.getString("email");
        stringCity = row.getString("city");
        stringMobile = row.getString("mobile");
        stringImage = row.getString("image_medium");
//              stringCompanyType = row.getString("company_type");
        stringToggle = row.getString("isFavourite");


        textContactName.setText(stringName);
        textContactEmail.setText(stringEmail);
        textContactEmail.setVisibility(stringEmail.equals("false") ? View.GONE : View.VISIBLE);

        textContactCity.setText(stringCity);
        textContactCity.setVisibility(stringCity.equals("false") ? View.GONE : View.VISIBLE);

        textContactNumber.setText(stringMobile);
        textContactNumber.setVisibility(stringMobile.equals("false") ? View.GONE : View.VISIBLE);
```

```
//        isCompany.setVisibility(stringCompanyType.equals("person") ? View.GONE : View.VISIBLE);

        if (stringImage.equals("false")) {
            profileImage.setImageBitmap(BitmapUtils.getAlphabetImage(getContext(), stringName));

        } else {
            profileImage.setImageBitmap(BitmapUtils.getBitmapImage(getContext(),
                    stringImage));
        }

        boolean isFavourite = !stringToggle.equals("false");
        if (favToogleCache.containsKey(row.getInt("_id"))) {
            isFavourite = favToogleCache.get(row.getInt("_id"));
        } else {
            favToogleCache.put(row.getInt("_id"), isFavourite);
        }
        toggleFavourite.setChecked(isFavourite);
        toggleFavourite.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
            @Override
            public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
                ContentValues values = new ContentValues();
                favToogleCache.put(row.getInt("_id"), toggleFavourite.isChecked());
                String favString = " marked emergency contacts";
                if (toggleFavourite.isChecked()) {
                    values.put("isFavourite", "true"); //(key, value)
                } else {
                    favString = " unmarked from emergency contacts";
                    values.put("isFavourite", "false");
                }
                Toast.makeText(getContext(), "Contact " + favString, Toast.LENGTH_SHORT).show();


                resPartner.update(values, "_id = ? ", String.valueOf(row.getInt("_id"))); //update(ContentValues values, String

                Log.i("Columns", resPartner.select().toString());

                getContext().getContentResolver().notifyChange(resPartner.uri(), null); //(uri, ContentObserver)
                /* Notify registered observers that a row was updated and attempt to sync changes to the network.
```

```
                 To register, call registerContentObserver().
                  By default, CursorAdapter objects will get this notification.
                   */

               }
           });
       }

       @Override
       public Loader<Cursor> onCreateLoader(int id, Bundle args) {
           return new CursorLoader(getContext(), resPartner.uri(), null, null, null, null);
       }

       @Override
       public void onLoadFinished(Loader<Cursor> loader, Cursor data) {
           oListAdapter.changeCursor(data);
           if (data.getCount() <= 0) {
               ((HomeActivity) getActivity()).syncData();
           }
       }

       @Override
       public void onLoaderReset(Loader<Cursor> loader) {
           oListAdapter.changeCursor(null);
       }

       @Override
       public void onItemClick(AdapterView<?> parent, View view, int position, long id) {

           Cursor cr = (Cursor) oListAdapter.getItem(position);

           Intent intent = new Intent(getActivity(), ContactDetailActivity.class);
           intent.putExtra("id", cr.getInt(cr.getColumnIndex("_id")));

           ContentValues values = new ContentValues();
           values.put("contact_id", cr.getInt(cr.getColumnIndex("_id")));
           recentContact.update_or_create(values, "contact_id = ? ", cr.getInt(cr.getColumnIndex("_id")) + "");
           getContext().getContentResolver().notifyChange(resPartner.uri(), null);
```

```
                startActivity(intent);
            }

            @Override
            public boolean onItemLongClick(AdapterView<?> parent, View view, int position, long id) {
                final Cursor cr = (Cursor) oListAdapter.getItem(position);
                final String stringMobileNumber = cr.getString(cr.getColumnIndex("mobile"));
                final String stringPhoneNumber = cr.getString(cr.getColumnIndex("phone"));
                final String stringEmail = cr.getString(cr.getColumnIndex("email"));

                final CharSequence[] options = {"Delete", "Call", "Send Mail"};

                AlertDialog.Builder builder = new AlertDialog.Builder(getContext()); //
                builder.setTitle("SELECT ANY ONE!");
                builder.setItems(options, new DialogInterface.OnClickListener() {

                    @Override
                    public void onClick(DialogInterface arg0, int item) {

                        arg0.dismiss();
                        if (options[item].equals("Delete")) {

                            AlertDialog.Builder alert = new AlertDialog.Builder(getContext());
                            alert.setTitle("You want to delete contact ?");
                            alert.setPositiveButton("Ok", new DialogInterface.OnClickListener() {
                                @Override
                                public void onClick(DialogInterface dialog, int which) {
                                    resPartner.delete("_id = ? ", String.valueOf(cr.getInt(cr.getColumnIndex("_id"))));
                                    Toast.makeText(getContext(), "Contact Deleted", Toast.LENGTH_LONG).show();
                                }
                            });
                            alert.setNegativeButton("Cancle", new DialogInterface.OnClickListener() {
                                @Override
                                public void onClick(DialogInterface dialog, int which) {
                                    dialog.dismiss();
                                }
                            });
                            alert.create().show();
                        }
                        if (options[item].equals("Call")) {
                            if (stringMobileNumber.equals("false")) {
                                if (stringPhoneNumber.equals("false")) {
                                    Toast.makeText(getContext(), "Number not found", Toast.LENGTH_LONG).show();
                                } else {
                                    callToContact(stringPhoneNumber);
                                }
                            } else {
                                callToContact(stringMobileNumber);
                            }
                        }

                        if (options[item].equals("Send Mail")) {
                            if (stringEmail.equals("false")) {
                                Toast.makeText(getContext(), "Email not found", Toast.LENGTH_LONG).show();
                            } else {
                                Intent mailIntent = new Intent(Intent.ACTION_SEND);
                                mailIntent.setData(Uri.parse("mailto:"));
                                mailIntent.setType("text/plain");
                                mailIntent.putExtra(Intent.EXTRA_EMAIL, new String[]{stringEmail});
                                startActivity(mailIntent);
                            }
                        }
                    }
                });
                builder.show();
                return true;
            }

            @TargetApi(Build.VERSION_CODES.M)
            public void callToContact(String number) {
                Uri phoneCall;
                Intent dial;
                phoneCall = Uri.parse("tel:" + number);
                dial = new Intent(Intent.ACTION_CALL, phoneCall);
                if (ActivityCompat.checkSelfPermission(getContext(), Manifest.permission.CALL_PHONE) != PackageManager.PERMISSION_GRANTE
                requestPermissions(new String[]{Manifest.permission.CALL_PHONE}, REQUEST_CODE_ASK_PERMISSIONS_CALL_CONTACT);
            } else {
                startActivity(dial);
            }
        }

    }
```

com_odoo_FavoriteFragment.java Raw

```
package com.odoo;


import android.content.ContentValues;
import android.content.Intent;
import android.database.Cursor;
import android.net.Uri;
import android.os.Bundle;
import android.support.annotation.Nullable;
import android.support.v4.app.Fragment;
import android.support.v4.app.LoaderManager;
import android.support.v4.content.CursorLoader;
import android.support.v4.content.Loader;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.ImageView;
import android.widget.ListView;
import android.widget.TextView;

import com.liveEarthquakesAlerts.R;
import com.odoo.orm.ListRow;
import com.odoo.orm.OListAdapter;
import com.odoo.table.RecentContact;
import com.odoo.table.ResPartner;
```

```java
import com.odoo.utils.BitmapUtils;


/**
 * A simple {@link Fragment} subclass.
 */
public class FavoriteFragment extends Fragment implements OListAdapter.OnViewBindListener,
        LoaderManager.LoaderCallbacks<Cursor>, AdapterView.OnItemClickListener {

    private ResPartner resPartner;
    private OListAdapter oListAdapter;
    private ListView favContactList;
    private RecentContact recentContact;

    public FavoriteFragment() {
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        return inflater.inflate(R.layout.fragment_favourite, container, false);
    }

    @Override
    public void onViewCreated(View view, @Nullable Bundle savedInstanceState) {
        super.onViewCreated(view, savedInstanceState);
        resPartner = new ResPartner(getContext());
        recentContact = new RecentContact(getContext());
        favContactList = (ListView) view.findViewById(R.id.favContactList);
        oListAdapter = new OListAdapter(getContext(), null, R.layout.favourite_list_item);
        oListAdapter.setOnViewBindListener(this);
        favContactList.setAdapter(oListAdapter);
        favContactList.setOnItemClickListener(this);
        getLoaderManager().initLoader(0, null, this);

    }

    @Override
```

```
        public void onViewBind(View view, Cursor cursor, ListRow row) {
            TextView textContactName, textContactEmail, textContactCity, textContactNumber;
            ImageView profileImage, isCompany;

            textContactName = (TextView) view.findViewById(R.id.textViewName);
            textContactEmail = (TextView) view.findViewById(R.id.textViewEmail);
            textContactCity = (TextView) view.findViewById(R.id.textViewCity);
            textContactNumber = (TextView) view.findViewById(R.id.textViewContact);
            profileImage = (ImageView) view.findViewById(R.id.profile_image);
//            isCompany = (ImageView) view.findViewById(R.id.isCompany);

            String stringName, stringEmail, stringCity, stringMobile, stringImage, stringCompanyType;

            stringName = row.getString("name");
            stringEmail = row.getString("email");
            stringCity = row.getString("city");
            stringMobile = row.getString("mobile");
            Log.i("Mobile", stringMobile);
            //write mobile number to bean and get it from there
            FavoriteNumberBean favoriteNumberBean = new FavoriteNumberBean(); //clears the list first
            favoriteNumberBean.addToArrayList(stringMobile);
            stringImage = row.getString("image_medium");
            stringCompanyType = row.getString("company_type");

            textContactName.setText(stringName);
            textContactEmail.setText(stringEmail);
            textContactEmail.setVisibility(stringEmail.equals("false") ? View.GONE : View.VISIBLE);

            textContactCity.setText(stringCity);
            textContactCity.setVisibility(stringCity.equals("false") ? View.GONE : View.VISIBLE);

            textContactNumber.setText(stringMobile);
            textContactNumber.setVisibility(stringMobile.equals("false") ? View.GONE : View.VISIBLE);

            if (stringImage.equals("false")) {
                profileImage.setImageBitmap(BitmapUtils.getAlphabetImage(getContext(), stringName));
            } else {
                profileImage.setImageBitmap(BitmapUtils.getBitmapImage(getContext(),
                        stringImage));
            }
        }

        @Override
        public Loader<Cursor> onCreateLoader(int id, Bundle args) { //loader are threads, therefore, we see various callbacks
            Uri uri = Uri.parse("content://com.odoo.contacts.res_partner/res_partner");
            return new CursorLoader(getContext(), uri, null, "isFavourite = ? ", new String[]{"true"}, null);
        }

        @Override
        public void onLoadFinished(Loader<Cursor> loader, Cursor data) {
            oListAdapter.changeCursor(data);
        }

        @Override
        public void onLoaderReset(Loader<Cursor> loader) {
            oListAdapter.changeCursor(null);
        }

        @Override
        public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
            Cursor cr = (Cursor) oListAdapter.getItem(position);

            Intent intent = new Intent(getActivity(), ContactDetailActivity.class);
            intent.putExtra("id", cr.getInt(cr.getColumnIndex("_id")));
            ContentValues values = new ContentValues();
            values.put("contact_id", cr.getInt(cr.getColumnIndex("_id")));
            recentContact.update_or_create(values, "contact_id = ? ", cr.getInt(cr.getColumnIndex("_id")) + "");
            getContext().getContentResolver().notifyChange(resPartner.uri(), null);
            startActivity(intent);
        }
    }
```

com_odoo_FavoriteNumberBean.java Raw

```
    package com.odoo;
```

```
import java.util.ArrayList;

/**
 * Created by  Uddhav Gautam  on 3/26/17.
 */

public class FavoriteNumberBean {
    private ArrayList<String> mobileNumber = new ArrayList<>();
    private boolean isToClear = false;

    public FavoriteNumberBean(boolean isToClear) {
        this.isToClear = isToClear;
    }

    public FavoriteNumberBean() {
        mobileNumber.clear();
    }

    public ArrayList<String> getMobileNumber() {
        return mobileNumber;
    }

    public void setMobileNumber(ArrayList<String> mobileNumber) {
        this.mobileNumber = mobileNumber;
    }

    public void addToArrayList(String stringMobile) {
        mobileNumber.add(stringMobile);
    }
}
```

com_odoo_HomeActivity.java Raw

```
package com.odoo;

import android.Manifest;
import android.accounts.Account;
import android.accounts.AccountManager;
```

```java
import android.content.ContentResolver;
import android.content.ContentValues;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.database.Cursor;
import android.os.Build;
import android.os.Bundle;
import android.provider.BaseColumns;
import android.provider.ContactsContract;
import android.support.design.widget.FloatingActionButton;
import android.support.design.widget.TabLayout;
import android.support.v4.app.ActivityCompat;
import android.support.v4.app.Fragment;
import android.support.v4.app.FragmentManager;
import android.support.v4.app.FragmentPagerAdapter;
import android.support.v4.view.ViewPager;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.SearchView;
import android.support.v7.widget.Toolbar;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Toast;

import com.liveEarthquakesAlerts.R;
import com.odoo.auth.OdooAuthenticator;
import com.odoo.orm.sync.ContactSyncAdapter;
import com.odoo.table.ResPartner;


public class HomeActivity extends AppCompatActivity implements TabLayout.OnTabSelectedListener {

    private static final int REQUEST_CODE_ASK_PERMISSIONS_READ_CONTACTS = 11;
    private SectionsPagerAdapter mSectionsPagerAdapter;
    private ViewPager mViewPager;
    private TabLayout tabLayout;
    private SearchView searchview;
    private ResPartner resPartner;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_home);


        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        getSupportActionBar().setHomeButtonEnabled(true);


        resPartner = new ResPartner(this); // ResPartner(Context)

        searchview = (SearchView) findViewById(R.id.contactSearchView);
        searchview.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // Redirecting to global contact search activity
                startActivity(new Intent(HomeActivity.this, SearchContactActivity.class));
            }
        });


        tabLayout = (TabLayout) findViewById(R.id.tab_layout);
        //code for tab
        tabLayout.addTab(tabLayout.newTab().setText("All-Contacts"));
        tabLayout.addTab(tabLayout.newTab().setText("Emergency-Contacts"));

//          tabLayout.setOnTabSelectedListener(this); //deprecated method
        tabLayout.addOnTabSelectedListener(this); //addOnTabSelectedListener(OnTabSelectedListener)
```

```
            mSectionsPagerAdapter = new SectionsPagerAdapter(getSupportFragmentManager()); //SectionPagerAdapter is a FragmentPagerA

            //code for swipe
            mViewPager = (ViewPager) findViewById(R.id.container);
            mViewPager.setAdapter(mSectionsPagerAdapter);
            tabLayout.setupWithViewPager(mViewPager);

            FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
            fab.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View view) {
                    startActivity(new Intent(HomeActivity.this, AddContact.class));
                }
            });

        }

        @Override
        public boolean onCreateOptionsMenu(Menu menu) {
            getMenuInflater().inflate(R.menu.menu_home, menu);
            return true;
        }

        public void syncData() {
            AccountManager accountManager = (AccountManager) getSystemService(ACCOUNT_SERVICE);
            Account[] accounts = accountManager.getAccountsByType(OdooAuthenticator.AUTH_TYPE);
            if (accounts.length == 1) {
                ContentResolver.requestSync(accounts[0], ContactSyncAdapter.AUTHORITY,
                        Bundle.EMPTY);
                Toast.makeText(HomeActivity.this, "Sync started", Toast.LENGTH_SHORT).show();
            }
        }

        @Override
        public boolean onOptionsItemSelected(MenuItem item) {
            int id = item.getItemId();
            switch (id) {
    //            case R.id.menu_sync:
```

```
//                syncData();
//                break;
            case R.id.menu_remove_emergency_contact:
                ContentValues values = new ContentValues(); // a HashMap //This class is used to store a set of values that the

                values.put("isFavourite", "false");
                resPartner.update(values, "isFavourite = ? ", "true"); //'where' in the query is 'key' or 'index' to search row
//(String table, String whereClause, String[] whereArgs)
                /*
                public int update(ContentValues values, String where, String... args) {
        if (!uri().equals(Uri.EMPTY)) {
            return mContext.getContentResolver().update(uri(), values, where, args);
        } else {
            SQLiteDatabase db = getWritableDatabase();
            int count = db.update(getTableName(), values, where, args);
            db.close();
            return count;
        }
    }
                 */



//                this.getContentResolver().notifyChange(resPartner.uri(), null); //null as a ContentObserver. It means, By defa


                //ContentObservers receives the callbacks from the Listeners. Means, it observes the Listeners. Adapter is a typ
                //So, events get registered with ContentObserver.
//                We can register like this ----> getContentResolver().registerContentObserver(SOME_URI, true, yourObserver);

                /*
                Notify registered observers that a row was updated and attempt to sync changes to the network.
                who can register ContentObserver?
                Any android component. Activity can by calling getContentResolver().registerContentObserver()
                //Fragment can register by calling getActivity().getContentResolver().registerContentObserver()
                 */
                break;
            case R.id.menu_import_contact:
                if (ActivityCompat.checkSelfPermission(this, Manifest.permission.READ_CONTACTS) != PackageManager.PERMISSION_GRA
                    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) { //if Build version is greater or equal to 23 then, we
                        requestPermissions(new String[]{Manifest.permission.READ_CONTACTS}, REQUEST_CODE_ASK_PERMISSIONS_READ_CO
                    }
                } else importContacts();

                break;

        }
        return super.onOptionsItemSelected(item);
    }

    private void importContacts() {
        ContentResolver cr = this.getContentResolver();

        Cursor cursor = cr.query(ContactsContract.Contacts.CONTENT_URI, null, null, null, null);
        if (cursor != null) {
            if (cursor.moveToFirst()) {
                do {
                    int contact_id = cursor.getInt(cursor
                            .getColumnIndex(BaseColumns._ID));
                    String contact_name = cursor.getString(cursor
                            .getColumnIndex("display_name"));
                    String contact_image = cursor.getString(cursor
                            .getColumnIndex("photo_uri"));

                    Cursor phoneCR = cr.query(
                            ContactsContract.CommonDataKinds.Phone.CONTENT_URI,
                            null, ContactsContract.CommonDataKinds.Phone.CONTACT_ID
                                    + " = ?", new String[]{contact_id + ""},
                            null);

                    if (phoneCR != null && phoneCR.moveToFirst()) {
                        String contact_number = phoneCR
                                .getString(phoneCR
                                        .getColumnIndex(ContactsContract.CommonDataKinds.Phone.NUMBER));
                        ContentValues values = new ContentValues();
                        values.put("name", contact_name);
```

```
                                values.put("image_medium", contact_image);
                                values.put("mobile", contact_number);
                                resPartner.update_or_create(values, "name = ? ", contact_name);
                            }

                        } while (cursor.moveToNext());
                        Log.d("TAG", cursor.getCount() + " contacts import");
                        cursor.close();
                    }
                }
            }

            @Override
            public void onTabSelected(TabLayout.Tab tab) {
                mViewPager.setCurrentItem(tab.getPosition());
            }

            @Override
            public void onTabUnselected(TabLayout.Tab tab) {

            }

            @Override
            public void onTabReselected(TabLayout.Tab tab) {

            }


            public static class PlaceholderFragment extends Fragment {
                private static final String ARG_SECTION_NUMBER = "section_number";

                public PlaceholderFragment() {
                }

                public static PlaceholderFragment newInstance(int sectionNumber) {
                    PlaceholderFragment fragment = new PlaceholderFragment();
                    Bundle args = new Bundle();
                    args.putInt(ARG_SECTION_NUMBER, sectionNumber);
                    fragment.setArguments(args);
                    return fragment;
                }

                @Override
                public View onCreateView(LayoutInflater inflater, ViewGroup container,
                                         Bundle savedInstanceState) {
                    View rootView = inflater.inflate(R.layout.fragment_contact, container, false);
                    return rootView;
                }
            }

            public class SectionsPagerAdapter extends FragmentPagerAdapter {

                public SectionsPagerAdapter(FragmentManager fm) {
                    super(fm);
                }

                @Override
                public Fragment getItem(int position) {
                    switch (position) {
                        case 0:
                            ContactFragment contactFragment = new ContactFragment();
                            return contactFragment;

                        case 1:
                            FavoriteFragment favoriteFragment = new FavoriteFragment();
                            return favoriteFragment;
                    }
                    return PlaceholderFragment.newInstance(position + 1);
                }

                @Override
                public int getCount() {
                    return 2;
                }

                @Override
                public CharSequence getPageTitle(int position) {
                    switch (position) {

                        case 0:
                            return "All-Contacts";

                        case 1:
                            return "Emergency-Contacts";
                    }
                    return null;
                }
            }

        }
```

com_odoo_LoginActivity.java Raw

```
    package com.odoo;

    import android.accounts.Account;
    import android.accounts.AccountManager;
    import android.app.ProgressDialog;
    import android.content.ContentResolver;
    import android.content.DialogInterface;
    import android.content.Intent;
    import android.os.Bundle;
    import android.support.v7.app.AlertDialog;
    import android.support.v7.app.AppCompatActivity;
    import android.util.Log;
    import android.view.View;
    import android.widget.EditText;
    import android.widget.Toast;

    import com.liveEarthquakesAlerts.R;
    import com.odoo.auth.OdooAuthenticator;
    import com.odoo.orm.sync.ContactSyncAdapter;
```

```
import java.util.List;

import odoo.Odoo;
import odoo.handler.OdooVersionException;
import odoo.helper.OUser;
import odoo.listeners.IDatabaseListListener;
import odoo.listeners.IOdooConnectionListener;
import odoo.listeners.IOdooLoginCallback;
import odoo.listeners.OdooError;

public class LoginActivity extends AppCompatActivity implements IOdooLoginCallback, View.OnClickListener,
        IOdooConnectionListener {

    private EditText edtHost, edtUsername, edtPassword;
    private Odoo odoo;
    private ProgressDialog progressDialog;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);

        AccountManager manager = (AccountManager) getSystemService(ACCOUNT_SERVICE);
        Account[] accounts = manager.getAccountsByType(OdooAuthenticator.AUTH_TYPE);
        if (accounts.length > 0) {
            // account found. redirecting to home screen.
            redirectToHome();
        }

        //code for Login object Initialization
        edtHost = (EditText) findViewById(R.id.edtHost);
        edtUsername = (EditText) findViewById(R.id.edtUsername);
        edtPassword = (EditText) findViewById(R.id.edtPassword);

        findViewById(R.id.btnLogin).setOnClickListener(this);
    }

    @Override
```

```
            public void onClick(View v) {

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

                    edtHost.setError(null);
                    if (edtHost.getText().toString().trim().isEmpty()) {
                        edtHost.setError(getString(R.string.error_host_name_required));
                        edtHost.requestFocus();
                        return;
                    }
                    edtUsername.setError(null);
                    if (edtUsername.getText().toString().trim().isEmpty()) {
                        edtUsername.setError(getString(R.string.error_username_required));
                        edtUsername.requestFocus();
                        return;
                    }
                    edtPassword.setError(null);
                    if (edtPassword.getText().toString().trim().isEmpty()) {
                        edtPassword.setError(getString(R.string.error_password_required));
                        edtPassword.requestFocus();
                        return;
                    }

                    login();
                }
            }

            private void login() {
                String host_url = stripURL(edtHost.getText().toString().trim());
                try {
                    odoo = Odoo.createInstance(this, host_url);
                    odoo.setOnConnect(this);
                } catch (OdooVersionException e) {
                    e.printStackTrace();
                }
            }

            private String stripURL(String host) {
```

```
        if (host.contains("http://") || host.contains("https://")) {
            return host;
        } else {
            return "http://" + host;
        }
    }


    @Override
    public void onConnect(final Odoo odoo) {
        odoo.getDatabaseList(new IDatabaseListListener() {
            @Override
            public void onDatabasesLoad(List<String> list) {
                if (list.size() > 1) {
                    showDatabaseSelection(list);
                } else {
                    // auto select first database and login.
                    loginTo(list.get(0));
                }
            }
        });
    }

    private void loginTo(String database) {
        String username = edtUsername.getText().toString().trim();
        String password = edtPassword.getText().toString().trim();
        odoo.authenticate(username, password, database, this);
        progressDialog = new ProgressDialog(this);
        progressDialog.setCancelable(false);
        progressDialog.setTitle(R.string.please_wait);
        progressDialog.setMessage(getString(R.string.login_in_progress));
        progressDialog.show();
    }

    @Override
    public void onError(OdooError odooError) {
        if (progressDialog != null && progressDialog.isShowing())
            progressDialog.dismiss();
        Log.e("odoo connection", odooError.getMessage(), odooError.getThrowable());
        Toast.makeText(this, R.string.unable_to_connect_odoo, Toast.LENGTH_SHORT).show();
    }

    @Override
    public void onLoginSuccess(Odoo odoo, OUser oUser) {
        progressDialog.dismiss();
        AccountManager manager = (AccountManager) getSystemService(ACCOUNT_SERVICE);
        Account account = new Account(oUser.getAndroidName(), OdooAuthenticator.AUTH_TYPE);
        if (manager.addAccountExplicitly(account, oUser.getPassword(), oUser.getAsBundle())) {
            ContentResolver.setSyncAutomatically(account, ContactSyncAdapter.AUTHORITY, true);
            redirectToHome();
        }
    }

    @Override
    public void onLoginFail(OdooError odooError) {
        progressDialog.dismiss();
        Toast.makeText(LoginActivity.this, R.string.invalid_username_or_password, Toast.LENGTH_SHORT).show();
    }

    private void redirectToHome() {
        startActivity(new Intent(this, HomeActivity.class));
        finish();
    }


    private void showDatabaseSelection(final List<String> databases) {
        AlertDialog.Builder builder = new AlertDialog.Builder(this);
        builder.setTitle(R.string.select_database);
        builder.setSingleChoiceItems(databases.toArray(new String[databases.size()]), 0,
                new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        dialog.dismiss();
                        String database = databases.get(which);
                        loginTo(database);
                    }
                });
        builder.create().show();
    }


}
```

com_odoo_orm_ListRow.java

Raw

```
        package com.odoo.orm;

        import android.database.Cursor;

        import java.util.HashMap;

        /**
         * Created by  Uddhav Gautam  on 25/4/16.
         */
        public class ListRow extends HashMap<String, Object> {

            public ListRow() {

            }

            public ListRow(Cursor cursor) {
                for (String col : cursor.getColumnNames()) {
                    int index = cursor.getColumnIndex(col);
                    switch (cursor.getType(index)) {
                        case Cursor.FIELD_TYPE_STRING:
                            put(col, cursor.getString(index));
                            break;
                        case Cursor.FIELD_TYPE_INTEGER:
                            put(col, cursor.getInt(index));
                            break;
                        case Cursor.FIELD_TYPE_BLOB:
                            put(col, cursor.getBlob(index));
                            break;
                        case Cursor.FIELD_TYPE_FLOAT:
```

```java
                        put(col, cursor.getFloat(index));
                    }

                }
            }

            public int getInt(String key) {
                return containsKey(key) ? Integer.parseInt(get(key) + "") : -1;

            }

            public String getString(String key) {
                return containsKey(key) ? get(key) + "" : "false";
            }



        }
```

com_odoo_orm_OColumn.java Raw

```java
        package com.odoo.orm;

        import com.odoo.orm.types.ColumnType;

        public class OColumn {
            public String name, label, relModel;
            public ColumnType columnType;
            public Boolean primaryKey = false, autoIncrement = false, isLocal = false;
            public Object defValue = null;

            public OColumn(String label, ColumnType columnType) {
                this(label, columnType, null);
            }

            public OColumn(String label, ColumnType columnType, String relModel) {
                this.label = label;
                this.columnType = columnType;
                this.relModel = relModel;
            }

            public OColumn makePrimaryKey() {
                primaryKey = true;
                return this;
            }

            public OColumn makeAutoIncrement() {
                autoIncrement = true;
                return this;
            }

            public OColumn setDefault(Object defValue) {
                this.defValue = defValue;
                return this;
            }

            public OColumn makeLocal() {
                isLocal = true;
                return this;
            }
        }
```

com_odoo_orm_OListAdapter.java Raw

```
    package com.odoo.orm;

    import android.content.Context;
    import android.database.Cursor;
    import android.view.LayoutInflater;
    import android.view.View;
    import android.view.ViewGroup;
    import android.widget.CursorAdapter;

    public class OListAdapter extends CursorAdapter {
        public static final String TAG = OListAdapter.class.getSimpleName();
        private int res_id = 0;
        private OnViewBindListener mOnViewBindListener;
```

```java
        private OnNewViewInflateListener mOnNewViewInflateListener;
        private Context mContext;

        public OListAdapter(Context context, Cursor c, int layout) {
            super(context, c, false);
            mContext = context;
            res_id = layout;
        }

        public int getResource() {
            return res_id;
        }

        @Override
        public View newView(Context context, Cursor cursor, ViewGroup parent) {
            if (mOnNewViewInflateListener != null) {
                return mOnNewViewInflateListener.onNewView(context, cursor, parent);
            } else {
                return LayoutInflater.from(mContext).inflate(getResource(), parent,
                        false);
            }
        }

        @Override
        public void bindView(View view, Context context, Cursor cursor) {
            if (mOnViewBindListener != null) {
                ListRow row = new ListRow(cursor);
                mOnViewBindListener.onViewBind(view, cursor, row);
            }
        }

        public void setOnViewBindListener(OnViewBindListener bindListener) {
            mOnViewBindListener = bindListener;
        }

        public void setOnNewViewInflateListener(OnNewViewInflateListener listener) {
            mOnNewViewInflateListener = listener;
        }
```

```
    public interface OnNewViewInflateListener {
        View onNewView(Context context, Cursor cursor, ViewGroup parent);
    }

    public interface OnViewBindListener {
        void onViewBind(View view, Cursor cursor, ListRow row);
    }
}
```

com_odoo_orm_OModel.java

Raw

```
package com.odoo.orm;

import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.net.Uri;
import android.provider.BaseColumns;
import android.util.Log;

import com.odoo.orm.types.ColumnType;
import com.odoo.table.ModelRegistry;

import java.lang.reflect.Field;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.HashMap;
import java.util.List;

/**
 * Created by  Uddhav Gautam  on 25/4/16.
 */
public abstract class OModel extends SQLiteOpenHelper implements BaseColumns {
    public static final String DB_NAME = "OdooContacts.db";
    public static final int DB_VERSION = 1;
    OColumn _id = new OColumn("Local ID", ColumnType.INTEGER)
            .makeAutoIncrement()
            .makePrimaryKey().makeLocal();
    OColumn id = new OColumn("Server ID", ColumnType.INTEGER)
            .setDefault("0");
    OColumn _write_date = new OColumn("Local Write date", ColumnType.DATETIME)
            .setDefault("false").makeLocal();
    OColumn is_dirty = new OColumn("Dirty record", ColumnType.BOOLEAN).setDefault("false")
            .makeLocal();
    private Context mContext;
    private String mModelName;

    public OModel(Context context, String model) {
        super(context, DB_NAME, null, DB_VERSION);
        mContext = context;
        mModelName = model;
    }

    public static OModel createInstance(String modelName, Context context) {
        ModelRegistry registry = new ModelRegistry();
        HashMap<String, OModel> models = registry.models(context);
        for (String modelKey : models.keySet()) {
            OModel m = models.get(modelKey);
            if (m.getModelName().equals(modelName)) {
                return m;
            }
        }
        return null;
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        HashMap<String, OModel> map = new ModelRegistry().models(mContext);
        for (OModel model : map.values()) {
            StatementBuilder sqlBuilder = new StatementBuilder(model);
            String sql = sqlBuilder.createStatement();
            if (sql != null) {
                db.execSQL(sql);
                Log.v("Database", "Model registered : " + model.getModelName());
            }
        }
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {

    }

    public String getTableName() {
        return mModelName.replace(".", "_");
    }

    public String getModelName() {
        return mModelName;
    }

    public List<OColumn> getColumns() {
        List<OColumn> columns = new ArrayList<>();
        List<Field> fields = new ArrayList<>();
        fields.addAll(Arrays.asList(getClass().getSuperclass().getDeclaredFields()));
        fields.addAll(Arrays.asList(getClass().getDeclaredFields()));

        for (Field field : fields) {
            field.setAccessible(true);
            if (field.getType().isAssignableFrom(OColumn.class)) {
                try {
                    OColumn column = (OColumn) field.get(this);
                    column.name = field.getName();
                    columns.add(column);
                } catch (IllegalAccessException e) {
                    e.printStackTrace();
                }
            }
        }
        return columns;
    }
```

```
        public List<ListRow> select() {

            return select(null);
        }

        public int create(ContentValues contentValues) {
            if (!uri().equals(Uri.EMPTY)) {
                Uri uri = mContext.getContentResolver().insert(uri(), contentValues);
                return Integer.parseInt(uri.getLastPathSegment());
            } else {
                SQLiteDatabase database = getWritableDatabase();
                Long id = database.insert(getTableName(), null, contentValues);
                database.close();
                return id.intValue();
            }
        }

        public List<ListRow> select(String where, String... args) {
            List<ListRow> rows = new ArrayList<>();
            SQLiteDatabase db = getReadableDatabase();
            args = args.length > 0 ? args : null;
            Cursor cursor = db.query(getTableName(), null, where, args, null, null, "_id DESC");
            if (cursor.moveToFirst()) {

                do {
                    rows.add(new ListRow(cursor));
                } while (cursor.moveToNext());
            }
            cursor.close();
            db.close();
            return rows;
        }

        public int update(ContentValues values, String where, String... args) { //(String table, String whereClause, String[] whereA
            if (!uri().equals(Uri.EMPTY)) { //if not empty
                return mContext.getContentResolver().update(uri(), values, where, args);
            } else {
                SQLiteDatabase db = getWritableDatabase();
                int count = db.update(getTableName(), values, where, args);
                Log.i("Table name", getTableName());
                db.close();
                return count;
            }
        }

        public int delete(String where, String... args) {
            if (!uri().equals(Uri.EMPTY)) {
                return mContext.getContentResolver().delete(uri(), where, args);
            } else {
                SQLiteDatabase db = getWritableDatabase();
                int count = db.delete(getTableName(), where, args);
                db.close();
                return count;
            }
        }

        public int count() {
            int count = 0;
            Cursor cr = null;
            SQLiteDatabase db = getReadableDatabase();

            cr = db.rawQuery("select count(*) as total from " + getTableName(), null);

            if (cr.moveToFirst()) {
                count = cr.getInt(0);
            }

            cr.close();
            db.close();

            return count;
        }

        public String[] getServerColumn() {
            List<String> columns = new ArrayList<>();
            for (OColumn column : getColumns()) {
                if (!column.isLocal) {
                    columns.add(column.name);
                }
            }
            return columns.toArray(new String[columns.size()]);
        }

        public int update_or_create(ContentValues values, String where, String... args) {
            List<ListRow> records = select(where, args);
            if (records.size() > 0) {
                // Update record
                ListRow row = records.get(0);
                update(values, where, args);
                return row.getInt(_ID);
            } else {
                // create new record
                return create(values);
            }
        }

        public Uri uri() {
            return Uri.EMPTY;
        }
    }
```

com_odoo_orm_StatementBuilder.java

Raw

```
    package com.odoo.orm;

    /**
     * Created by  Uddhav Gautam  on 25/4/16.
     */
    public class StatementBuilder {
        private OModel mTable;
        public StatementBuilder(OModel table) {
            this.mTable = table;
        }

        public String createStatement() {

            StringBuffer sql = new StringBuffer();
            sql.append("CREATE TABLE IF NOT EXISTS ")
                    .append(mTable.getTableName())
                    .append(" (");

            StringBuffer columns = new StringBuffer();
            for (OColumn column : mTable.getColumns()) {
                columns.append(column.name)
                        .append(" ")
                        .append(column.columnType.toString());

                if (column.primaryKey) {
                    columns.append(" PRIMARY KEY ");
                }
                if (column.autoIncrement) {
                    columns.append(" AUTOINCREMENT ");
                }
                if (column.defValue != null) {
                    columns.append(" DEFAULT '").append(column.defValue.toString()).append("'");
                }
                columns.append(" , ");
            }
            String columnString = columns.toString();
            sql.append(columnString.substring(0, columnString.length() - 2)).append(" )");
            return sql.toString();
        }
    }
```

com_odoo_orm_sync_ContactSyncAdapter.java

Raw

```
    package com.odoo.orm.sync;
```

```
import android.accounts.Account;
import android.accounts.AccountManager;
import android.content.AbstractThreadedSyncAdapter;
import android.content.ContentProviderClient;
import android.content.ContentValues;
import android.content.Context;
import android.content.SharedPreferences;
import android.content.SyncResult;
import android.os.Bundle;
import android.util.Log;

import com.odoo.orm.ListRow;
import com.odoo.orm.OColumn;
import com.odoo.orm.OModel;
import com.odoo.orm.types.ColumnType;
import com.odoo.table.ResPartner;
import com.odoo.utils.ODateUtils;

import java.util.ArrayList;
import java.util.List;

import odoo.Odoo;
import odoo.handler.OdooVersionException;
import odoo.helper.ODomain;
import odoo.helper.ORecordValues;
import odoo.helper.OUser;
import odoo.helper.OdooFields;
import odoo.helper.utils.gson.OdooRecord;
import odoo.helper.utils.gson.OdooResult;

public class ContactSyncAdapter extends AbstractThreadedSyncAdapter {

    public static final String KEY_LAST_SYNC_DATETIME = "last_sync_datetime";
    public static final String AUTHORITY = "com.odoo.contacts.res_partner";
    private Context mContext;
    private Odoo odoo;
    private OUser mUser;
    private SharedPreferences pref;
    private AccountManager accountManager;

    public ContactSyncAdapter(Context context, boolean autoInitialize) {
        super(context, autoInitialize);
        mContext = context;
        pref = context.getSharedPreferences("sync_meta", Context.MODE_PRIVATE);
        accountManager = (AccountManager) context.getSystemService(Context.ACCOUNT_SERVICE);
    }

    @Override
    public void onPerformSync(Account account, Bundle extras, String authority,
                              ContentProviderClient provider, SyncResult syncResult) {

        Log.v("Perform Sync", "Sync started.");
        // Finding details of user by account and account manager
        mUser = findUser(account);
        try {

            // Because onPerformSync works in background. we have to use synchronized method in this scope.
            // Quick connecting with odoo in synchronized mode
            odoo = Odoo.createQuickInstance(mContext, mUser.getHost());
            // Quick authenticating with user in synchronized mode
            mUser = odoo.authenticate(mUser.getUsername(), mUser.getPassword(), mUser.getDatabase());

            // Creating Respartner database table object
            ResPartner partner = new ResPartner(mContext);

            /**
             * Creating record got from server also updating if record is newer than local
             */
            List<Integer> recordIds = createOrUpdateRecords(partner);
            Log.v("Create Or Update", recordIds + " records affected locally");


            /**
             * Creating record on server if any local record with zero (0) id
             */
            List<Integer> createdIds = createRecordsOnServer(partner);
```

```
                Log.v("Creating on server", createdIds + " records created on server");

                /**
                 * Storing last sync date and time to preferences.
                 */
                String lastSyncOn = ODateUtils.getCurrentDateTime();
                Log.v("Sync Finished", "Sync finished on " + lastSyncOn);
//                Log.v("Sync Finished:", recordIds.toString() + " records created/updated =>>" + lastSyncOn);

                pref.edit().putString(KEY_LAST_SYNC_DATETIME, lastSyncOn).apply();
            } catch (OdooVersionException e) {
                e.printStackTrace();
            }
        }


        private List<Integer> createOrUpdateRecords(ResPartner partner) {
            OdooFields fields = new OdooFields();
            fields.addAll(partner.getServerColumn());
            ODomain domain = new ODomain();

            /**
             * Adding create_date compare with last sync date.
             */
            String lastSyncDatetime = pref.getString(KEY_LAST_SYNC_DATETIME, null);
            if (lastSyncDatetime != null) {
                String utcLastSyncDate = ODateUtils.convertToUTC(lastSyncDatetime,
                        ODateUtils.DEFAULT_FORMAT);

                domain.add("|");
                domain.add("create_date", ">=", utcLastSyncDate);
                domain.add("write_date", ">=", utcLastSyncDate);
            }

            // getting records from odoo server.
            OdooResult result = odoo.searchRead(partner.getModelName(), fields, domain, 0, 0, null);

            List<Integer> recordIds = new ArrayList<>();
```

```
                // filtering each of the record with the column values.
                for (OdooRecord record : result.getRecords()) {
                    ContentValues values = new ContentValues();

                    // looping with each column and setting the content value as per its type.
                    for (OColumn column : partner.getColumns()) {
                        if (!column.isLocal) {
                            switch (column.columnType) {
                                case VARCHAR:
                                    values.put(column.name, record.getString(column.name));
                                    break;
                                case BOOLEAN:
                                    values.put(column.name, record.getBoolean(column.name));
                                    break;
                                case BLOB:
                                    values.put(column.name, record.getString(column.name));
                                    break;
                                case INTEGER:
                                    values.put(column.name, record.getInt(column.name));
                                    break;
                                case DATETIME:
                                    values.put(column.name, record.getString(column.name));
                                    break;
                                case MANY2ONE:

                                    // Here many to one record refers another table.
                                    // so creating primary record for that table and adding
                                    // primary key field unique id to reference value

                                    OdooRecord m2oRecord = record.getM2O(column.name);
                                    int m2oId = 0;
                                    if (m2oRecord != null) {
                                        String modelName = column.relModel;
                                        OModel model = OModel.createInstance(modelName, mContext);
                                        if (model != null) {
                                            ContentValues m2oValues = new ContentValues();
                                            m2oValues.put("id", m2oRecord.getInt("id"));
                                            m2oValues.put("name", m2oRecord.getString("name"));
                                            m2oId = model.update_or_create(m2oValues, "id = ?",
                                                    m2oRecord.getInt("id") + "");
                                        }
                                    }
                                    values.put(column.name, m2oId);
                                    break;
                            }
                        }
                    }
                    // Creating or updating record in the database..
                    int record_id = partner.update_or_create(values, "id = ?", record.getInt("id") + "");
                    recordIds.add(record_id);
                }
                return recordIds;
            }


            private List<Integer> createRecordsOnServer(ResPartner partner) {
                List<Integer> ids = new ArrayList<>();
                for (ListRow row : partner.select("id = ?", "0")) {
                    ORecordValues values = new ORecordValues();

                    for (OColumn column : partner.getColumns()) {
                        if (!column.isLocal && !column.name.equals("id")) {
                            Object value = row.get(column.name);
                            if (!value.toString().equals("false") ||
                                    column.columnType == ColumnType.BOOLEAN) {
                                switch (column.columnType) {
                                    case MANY2ONE:
                                        //TODO: To be implemented.
                                        continue;
                                }
                                values.put(column.name, value);
                            }
                        }
                    }
                    // Creating record on server
                    OdooResult result = odoo.createRecord(partner.getModelName(), values);
                    int newServerId = result.getInt("result");
                    ids.add(newServerId);
                    // Updating local record with new created server id
                    ContentValues newValues = new ContentValues();
                    newValues.put("id", newServerId);
                    partner.update(newValues, "_id = ?", row.getString("_id"));
                }
                return ids;
            }

            /**
             * creates the detail for user of account. for connecting with odoo
             *
             * @param account object of device account
             * @return object of odoo user
             */
            private OUser findUser(Account account) {
                OUser user = new OUser();
                user.setHost(accountManager.getUserData(account, "host"));
                user.setUsername(accountManager.getUserData(account, "username"));
                user.setPassword(accountManager.getPassword(account));
                user.setDatabase(accountManager.getUserData(account, "database"));
                return user;
            }

        }
```

com_odoo_orm_sync_providers_ContactProvider.java

Raw

```
    package com.odoo.orm.sync.providers;

    import android.content.ContentProvider;
    import android.content.ContentValues;
    import android.content.Context;
    import android.database.Cursor;
    import android.database.sqlite.SQLiteDatabase;
```

```
        import android.database.sqlite.SQLiteQueryBuilder;
        import android.net.Uri;

        import com.odoo.table.ResPartner;

        public class ContactProvider extends ContentProvider {
            public static final String TAG = ContactProvider.class.getSimpleName();
            public ResPartner resPartner;

            @Override
            public boolean onCreate() {
                resPartner = new ResPartner(getContext());
                return true;
            }


            @Override
            public Cursor query(Uri uri, String[] projection, String selection,
                                String[] selectionArgs, String sortOrder) {
                SQLiteQueryBuilder queryBuilder = new SQLiteQueryBuilder();
                Cursor cr;
                queryBuilder.setTables("res_partner");

                cr = queryBuilder.query(resPartner.getReadableDatabase(), projection,
                        selection, selectionArgs, null, null, sortOrder);
                Context ctx = getContext();
                assert ctx != null;
                assert cr != null;
                cr.setNotificationUri(ctx.getContentResolver(), uri);
                return cr;
            }

            @Override
            public String getType(Uri uri) {
                return uri.toString();
            }

            @Override
            public Uri insert(Uri uri, ContentValues values) {
                SQLiteDatabase database = resPartner.getWritableDatabase();
                Long id = database.insert(resPartner.getTableName(), null, values);
                Context ctx = getContext();
                assert ctx != null;
                ctx.getContentResolver().notifyChange(uri, null);
                return Uri.withAppendedPath(uri, id + "");
            }

            @Override
            public int delete(Uri uri, String selection, String[] selectionArgs) {
                SQLiteDatabase database = resPartner.getWritableDatabase();
                int count = database.delete(resPartner.getTableName(), selection, selectionArgs);
                Context ctx = getContext();
                assert ctx != null;
                ctx.getContentResolver().notifyChange(uri, null);
                return count;
            }

            @Override
            public int update(Uri uri, ContentValues values, String selection,
                              String[] selectionArgs) {
                SQLiteDatabase database = resPartner.getWritableDatabase();
                int count = database.update(resPartner.getTableName(), values, selection, selectionArgs);
                Context ctx = getContext();
                assert ctx != null;
                ctx.getContentResolver().notifyChange(uri, null);
                return count;
            }

        }
```

com_odoo_orm_sync_SyncService.java Raw

```
    package com.odoo.orm.sync;

    import android.app.Service;
    import android.content.Intent;
```

```java
import android.os.IBinder;
import android.support.annotation.Nullable;

public class SyncService extends Service {

    private static final Object sSyncAdapterLock = new Object();
    private ContactSyncAdapter syncAdapter;

    @Override
    public void onCreate() {
        synchronized (sSyncAdapterLock) {
            if (syncAdapter == null) {
                syncAdapter = new ContactSyncAdapter(getApplicationContext(), true);
            }
        }
    }

    @Nullable
    @Override
    public IBinder onBind(Intent intent) {
        return syncAdapter.getSyncAdapterBinder();
    }
}
```

com_odoo_orm_types_ColumnType.java Raw

```java
package com.odoo.orm.types;

/**
 * Created by sha on 25/4/16.
 */
public enum ColumnType {
    VARCHAR("VARCHAR"),
    INTEGER("INTEGER"),
    BLOB("BLOB"),
    MANY2ONE("INTEGER"),
    DATETIME("varchar"),
    BOOLEAN("boolean");
```

```
        String type;

        ColumnType(String type) {
            this.type = type;
        }
    }
```

com_odoo_SearchContactActivity.java

Raw

```
    package com.odoo;

    import android.content.Intent;
    import android.database.Cursor;
    import android.net.Uri;
    import android.os.Bundle;
    import android.support.annotation.Nullable;
    import android.support.v4.app.LoaderManager;
    import android.support.v4.content.CursorLoader;
    import android.support.v4.content.Loader;
    import android.support.v7.app.AppCompatActivity;
    import android.text.Editable;
    import android.text.TextWatcher;
    import android.view.View;
    import android.widget.AdapterView;
    import android.widget.EditText;
    import android.widget.ImageView;
    import android.widget.ListView;
    import android.widget.TextView;
    import android.widget.ToggleButton;

    import com.liveEarthquakesAlerts.R;
    import com.odoo.orm.ListRow;
    import com.odoo.orm.OListAdapter;
    import com.odoo.utils.BitmapUtils;


    public class SearchContactActivity extends AppCompatActivity implements
```

```
        LoaderManager.LoaderCallbacks<Cursor>, TextWatcher, OListAdapter.OnViewBindListener, AdapterView.OnItemClickListener {

    private OListAdapter adapter;
    private ListView listView;
    private EditText edtSearchBox;
    private String searchFor = null;

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.search_contact_activity);
        edtSearchBox = (EditText) findViewById(R.id.edtSearchBox);
        edtSearchBox.addTextChangedListener(this);
        init();
    }

    private void init() {
        listView = (ListView) findViewById(R.id.contactList);
        adapter = new OListAdapter(this, null, R.layout.contact_list_item);
        adapter.setOnViewBindListener(this);
        listView.setAdapter(adapter);
        listView.setOnItemClickListener(this);
        getSupportLoaderManager().initLoader(0, null, this);
    }

    @Override
    public Loader<Cursor> onCreateLoader(int id, Bundle data) {
        Uri uri = Uri.parse("content://com.odoo.contacts.res_partner/res_partner");

        String where = null;
        String[] args = null;
        if (searchFor != null) {
            where = " name like ?";
            args = new String[]{"%" + searchFor + "%"};
        }
        return new CursorLoader(this, uri, null, where, args, null);
    }
```

```
        @Override
        public void onLoadFinished(Loader<Cursor> loader, Cursor data) {
            adapter.changeCursor(data);
        }

        @Override
        public void onLoaderReset(Loader<Cursor> loader) {
            adapter.changeCursor(null);
        }

        @Override
        public void onViewBind(View view, Cursor cursor, ListRow row) {

            TextView textContactName, textContactEmail, textContactCity, textContactNumber;
            ImageView profileImage, isCompany;

            final ToggleButton toggleFavourite = (ToggleButton) view.findViewById(R.id.toggleIsFavourite);

            textContactName = (TextView) view.findViewById(R.id.textViewName);
            textContactEmail = (TextView) view.findViewById(R.id.textViewEmail);
            textContactCity = (TextView) view.findViewById(R.id.textViewCity);
            textContactNumber = (TextView) view.findViewById(R.id.textViewContact);
            profileImage = (ImageView) view.findViewById(R.id.profile_image);
//            isCompany = (ImageView) view.findViewById(R.id.isCompany);

            String stringName, stringEmail, stringCity, stringMobile, stringImage, stringCompanyType;
            stringName = row.getString("name");
            stringEmail = row.getString("email");
            stringCity = row.getString("city");
            stringMobile = row.getString("mobile");
            stringImage = row.getString("image_medium");
            stringCompanyType = row.getString("company_type");


            textContactName.setText(stringName);
            textContactEmail.setText(stringEmail);
            textContactEmail.setVisibility(stringEmail.equals("false") ? View.GONE : View.VISIBLE);
            textContactCity.setText(stringCity);
            textContactCity.setVisibility(stringCity.equals("false") ? View.GONE : View.VISIBLE);

            textContactNumber.setText(stringMobile);
            textContactNumber.setVisibility(stringMobile.equals("false") ? View.GONE : View.VISIBLE);

//          isCompany.setVisibility(stringCompanyType.equals("person") ? View.GONE : View.VISIBLE);

            if (stringImage.equals("false")) {
                profileImage.setImageBitmap(BitmapUtils.getAlphabetImage(this, stringName));
            } else {
                profileImage.setImageBitmap(BitmapUtils.getBitmapImage(this,
                        stringImage));
            }
            toggleFavourite.setVisibility(View.GONE);
        }

        @Override
        public void beforeTextChanged(CharSequence s, int start, int count, int after) {

        }

        @Override
        public void onTextChanged(CharSequence s, int start, int before, int count) {
            searchFor = s.toString();
            if (s.toString().trim().isEmpty()) {
                searchFor = null;
            }
            getSupportLoaderManager().restartLoader(0, null, this);
        }

        @Override
        public void afterTextChanged(Editable s) {

        }

        @Override
        public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
```

```
        Cursor cr = (Cursor) adapter.getItem(position);
        Intent intent = new Intent(this, ContactDetailActivity.class);
        intent.putExtra("id", cr.getInt(cr.getColumnIndex("_id")));
        startActivity(intent);
        finish();
    }
}
```

com_odoo_table_ModelRegistry.java

Raw

```
package com.odoo.table;

import android.content.Context;

import com.odoo.orm.OModel;

import java.util.HashMap;

public class ModelRegistry {
    public HashMap<String, OModel> models(Context context) {
        HashMap<String, OModel> models = new HashMap<>();
        models.put("res.partner", new ResPartner(context));
        models.put("res.country", new ResCountry(context));
        models.put("res.state", new ResState(context));
        models.put("recent.contact", new RecentContact(context));
        return models;
    }
}
```

com_odoo_table_RecentContact.java

Raw

```
package com.odoo.table;

import android.content.Context;

import com.odoo.orm.OColumn;
import com.odoo.orm.OModel;
import com.odoo.orm.types.ColumnType;
```

```
public class RecentContact extends OModel {

    OColumn contact_id = new OColumn("Contact Id", ColumnType.INTEGER);
    OColumn write_date = new OColumn("Write Date", ColumnType.VARCHAR);

    public RecentContact(Context context) {
        super(context, "recent.contact");
    }
}
```

com_odoo_table_ResCountry.java Raw

```
package com.odoo.table;

import android.content.Context;

import com.odoo.orm.OColumn;
import com.odoo.orm.OModel;
import com.odoo.orm.types.ColumnType;

public class ResCountry extends OModel {

    OColumn name = new OColumn("Name", ColumnType.VARCHAR);

    public ResCountry(Context context) {
        super(context, "res.country");
    }
}
```

com_odoo_table_ResPartner.java Raw

```
package com.odoo.table;

import android.content.Context;
import android.net.Uri;

import com.odoo.orm.OColumn;
```

```java
import com.odoo.orm.OModel;
import com.odoo.orm.types.ColumnType;

public class ResPartner extends OModel {

    OColumn name = new OColumn("Name", ColumnType.VARCHAR);
    OColumn company_type = new OColumn("Company Type", ColumnType.VARCHAR);
    OColumn street = new OColumn("Street", ColumnType.VARCHAR);
    OColumn street2 = new OColumn("Street2", ColumnType.VARCHAR);
    OColumn city = new OColumn("City", ColumnType.VARCHAR);
    OColumn state_id = new OColumn("State", ColumnType.MANY2ONE, "res.state");
    OColumn country_id = new OColumn("Country", ColumnType.MANY2ONE, "res.country");
    OColumn zip = new OColumn("ZIP", ColumnType.VARCHAR);
    OColumn website = new OColumn("Website", ColumnType.VARCHAR);
    OColumn phone = new OColumn("Phone", ColumnType.VARCHAR);
    OColumn mobile = new OColumn("Mobile", ColumnType.VARCHAR);
    OColumn fax = new OColumn("Fax", ColumnType.VARCHAR);
    OColumn email = new OColumn("Email", ColumnType.VARCHAR);
    OColumn image_medium = new OColumn("Image", ColumnType.BLOB);
    OColumn isFavourite = new OColumn("isFavourite", ColumnType.VARCHAR).makeLocal(); // local column


    public ResPartner(Context context) {
        super(context, "res.partner");
    }

    @Override
    public Uri uri() {
        return Uri.parse("content://com.odoo.contacts.res_partner/res_partner");
    }
}
```

com_odoo_table_ResState.java Raw

```java
package com.odoo.table;

import android.content.Context;
import com.odoo.orm.OColumn;
import com.odoo.orm.OModel;
import com.odoo.orm.types.ColumnType;

public class ResState extends OModel {

    OColumn name = new OColumn("Name", ColumnType.VARCHAR);

    public ResState(Context context) {
        super(context, "res.state");
    }
}
```

com_odoo_utils_BitmapUtils.java Raw

```java
package com.odoo.utils;

import android.content.ContentResolver;
import android.content.Context;
import android.content.res.Resources;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.graphics.Rect;
import android.graphics.Typeface;
import android.net.Uri;
import android.text.TextPaint;
import android.util.Base64;

import com.liveEarthquakesAlerts.R;

import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.io.InputStream;

public class BitmapUtils {
    public static final int THUMBNAIL_SIZE = 500;

    /**
     * Gets the bitmap image.
     *
     * @param context the context
     * @param base64  the base64
     * @return the bitmap image
     */
    public static Bitmap getBitmapImage(Context context, String base64) {
        byte[] imageAsBytes = Base64.decode(base64.getBytes(), 5);
        return BitmapFactory.decodeByteArray(imageAsBytes, 0,
                imageAsBytes.length);

    }

    public static Bitmap getAlphabetImage(Context context, String content) {
        Resources res = context.getResources();
        Bitmap mDefaultBitmap = BitmapFactory.decodeResource(res, android.R.drawable.sym_def_app_icon);
        int width = mDefaultBitmap.getWidth();
        int height = mDefaultBitmap.getHeight();
        TextPaint mPaint = new TextPaint();
        mPaint.setTypeface(Typeface.create("sans-serif-condensed", 0));
        mPaint.setColor(Color.WHITE);
        mPaint.setTextAlign(Paint.Align.CENTER);
        mPaint.setAntiAlias(true);
        int textSize = res.getDimensionPixelSize(R.dimen.text_size_large);
        Bitmap bitmap = Bitmap.createBitmap(width, height, Bitmap.Config.ARGB_8888);
        Canvas canvas = new Canvas();
        Rect mBounds = new Rect();
        canvas.setBitmap(bitmap);
        canvas.drawColor(OStringColorUtil.getStringColor(context, content));
        if (content == null || content.trim().length() == 0) {
            content = "?";
        }
        char[] alphabet = {Character.toUpperCase(content.trim().charAt(0))};
        mPaint.setTextSize(textSize);
        mPaint.getTextBounds(alphabet, 0, 1, mBounds);
```

```
            canvas.drawText(alphabet, 0, 1, 0 + width / 2,
                    0 + height / 2 + (mBounds.bottom - mBounds.top) / 2, mPaint);
            return bitmap;
        }

        public static String bitmapToBase64(Bitmap bitmap) {
            ByteArrayOutputStream byteArrayOutputStream = new ByteArrayOutputStream();
            bitmap.compress(Bitmap.CompressFormat.PNG, 100, byteArrayOutputStream);
            byte[] byteArray = byteArrayOutputStream.toByteArray();
            return Base64.encodeToString(byteArray, 0);
        }

        private static byte[] readBytes(Uri uri, ContentResolver resolver, boolean thumbnail)
                throws IOException {
            // this dynamically extends to take the bytes you read
            InputStream inputStream = resolver.openInputStream(uri);
            ByteArrayOutputStream byteBuffer = new ByteArrayOutputStream();

            if (!thumbnail) {
                // this is storage overwritten on each iteration with bytes
                int bufferSize = 1024;
                byte[] buffer = new byte[bufferSize];

                // we need to know how may bytes were read to write them to the
                // byteBuffer
                int len = 0;
                while ((len = inputStream.read(buffer)) != -1) {
                    byteBuffer.write(buffer, 0, len);
                }
            } else {
                Bitmap imageBitmap = BitmapFactory.decodeStream(inputStream);
                int thumb_width = imageBitmap.getWidth() / 2;
                int thumb_height = imageBitmap.getHeight() / 2;
                if (thumb_width > THUMBNAIL_SIZE) {
                    thumb_width = THUMBNAIL_SIZE;
                }
                if (thumb_width == THUMBNAIL_SIZE) {
                    thumb_height = ((imageBitmap.getHeight() / 2) * THUMBNAIL_SIZE)
```

```
                            / (imageBitmap.getWidth() / 2);
                    }
                    imageBitmap = Bitmap.createScaledBitmap(imageBitmap, thumb_width, thumb_height, false);
                    imageBitmap.compress(Bitmap.CompressFormat.JPEG, 100, byteBuffer);
                }
                return byteBuffer.toByteArray();
            }

            public static String uriToBase64(Uri uri, ContentResolver resolver) {
                return uriToBase64(uri, resolver, false);
            }

            public static String uriToBase64(Uri uri, ContentResolver resolver, boolean thumbnail) {
                String encodedBase64 = "";
                try {
                    byte[] bytes = readBytes(uri, resolver, thumbnail);
                    encodedBase64 = Base64.encodeToString(bytes, 0);
                } catch (IOException e1) {
                    e1.printStackTrace();
                }
                return encodedBase64;
            }
        }
```

com_odoo_utils_ODateUtils.java Raw

```
        package com.odoo.utils;

        import android.util.Log;

        import java.text.SimpleDateFormat;
        import java.util.Date;
        import java.util.TimeZone;

        public class ODateUtils {

            public static final String TAG = ODateUtils.class.getCanonicalName();
            public static final String DEFAULT_FORMAT = "yyyy-MM-dd HH:mm:ss";
```

```java
    public static final String DEFAULT_DATE_FORMAT = "yyyy-MM-dd";
    public static final String DEFAULT_TIME_FORMAT = "HH:mm:ss";

    public static String getCurrentDateTime() {
        Date date = new Date();
        return createDate(date, DEFAULT_FORMAT, false);
    }

    public static String parseDate(String dateTime, String dateFormat, String toFormat) {
        return createDate(createDateObject(dateTime, dateFormat, false), toFormat, true);
    }

    public static Date createDateObject(String date, String dateFormat, boolean hasDefaultTimezone) {
        Date dateObj = null;
        try {
            SimpleDateFormat simpleDateFormat = new SimpleDateFormat(dateFormat);
            if (!hasDefaultTimezone) {
                simpleDateFormat.setTimeZone(TimeZone.getTimeZone("GMT"));
            }
            dateObj = simpleDateFormat.parse(date);
        } catch (Exception e) {
            Log.e(TAG, e.getMessage());
        }
        return dateObj;
    }

    private static String createDate(Date date, String defaultFormat, Boolean utc) {
        SimpleDateFormat gmtFormat = new SimpleDateFormat();
        gmtFormat.applyPattern(defaultFormat);
        TimeZone gmtTime = (utc) ? TimeZone.getTimeZone("GMT") : TimeZone.getDefault();
        gmtFormat.setTimeZone(gmtTime);
        return gmtFormat.format(date);
    }

    public static String convertToUTC(String dateTime, String dateFormat) {
        return createDate(createDateObject(dateTime, dateFormat, true), dateFormat, true);
    }
```

```
}
```

com_odoo_utils_OStringColorUtil.java

Raw

```
package com.odoo.utils;

import android.content.Context;
import android.content.res.Resources;
import android.content.res.TypedArray;
import android.graphics.Color;

import com.liveEarthquakesAlerts.R;

import java.util.Locale;

/**
 * Created by  Uddhav Gautam  on 3/5/16.
 */
public class OStringColorUtil {

    public static int getStringColor(Context context, String content) {
        Resources res = context.getResources();
        TypedArray mColors = res.obtainTypedArray(R.array.letter_tile_colors);
        int MAX_COLORS = mColors.length();
        int firstCharAsc = content.toUpperCase(Locale.getDefault()).charAt(0);
        int index = (firstCharAsc % MAX_COLORS);
        if (index > MAX_COLORS - 1) {
            index = index / 2;
        }
        int color = mColors.getColor(index, Color.WHITE);
        mColors.recycle();
        return color;
    }

}
```

com_odoo_widgets_BezelImageView.java

Raw

```
/*
 * Copyright 2014 Google Inc. All rights reserved.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

package com.odoo.widgets;

import android.content.Context;
import android.content.res.TypedArray;
import android.graphics.Bitmap;
import android.graphics.Canvas;
import android.graphics.ColorMatrix;
import android.graphics.ColorMatrixColorFilter;
import android.graphics.Paint;
import android.graphics.PorterDuff;
import android.graphics.PorterDuffXfermode;
import android.graphics.Rect;
import android.graphics.RectF;
import android.graphics.drawable.Drawable;
import android.support.v4.view.ViewCompat;
import android.util.AttributeSet;

import com.liveEarthquakesAlerts.R;


public class BezelImageView extends android.support.v7.widget.AppCompatImageView {
    private Paint mBlackPaint;
```

```
        private Paint mMaskedPaint;

        private Rect mBounds;
        private RectF mBoundsF;

        private Drawable mBorderDrawable;
        private Drawable mMaskDrawable;

        private ColorMatrixColorFilter mDesaturateColorFilter;
        private boolean mDesaturateOnPress = false;

        private boolean mCacheValid = false;
        private Bitmap mCacheBitmap;
        private int mCachedWidth;
        private int mCachedHeight;
        private Context mContext;

        public BezelImageView(Context context) {
            this(context, null);
        }

        public BezelImageView(Context context, AttributeSet attrs) {
            this(context, attrs, 0);
        }

        public BezelImageView(Context context, AttributeSet attrs, int defStyle) {
            super(context, attrs, defStyle);
            mContext = context;
            // Attribute initialization
            final TypedArray a = context.obtainStyledAttributes(attrs,
                    R.styleable.BezelImageView, defStyle, 0);

            mMaskDrawable = a.getDrawable(R.styleable.BezelImageView_maskDrawable);
            if (mMaskDrawable != null) {
                mMaskDrawable.setCallback(this);
            }

            mBorderDrawable = a
```

```java
                .getDrawable(R.styleable.BezelImageView_borderDrawable);
        if (mBorderDrawable != null) {
            mBorderDrawable.setCallback(this);
        }

        mDesaturateOnPress = a.getBoolean(
                R.styleable.BezelImageView_desaturateOnPress,
                mDesaturateOnPress);

        a.recycle();
        otherInit();
    }

    public void autoSetMaskDrawable() {
        mMaskDrawable = mContext.getResources().getDrawable(
                R.drawable.circle_bg_gray);
        otherInit();
    }

    private void otherInit() {
        // Other initialization
        mBlackPaint = new Paint();
        mBlackPaint.setColor(0xff000000);

        mMaskedPaint = new Paint();
        mMaskedPaint
                .setXfermode(new PorterDuffXfermode(PorterDuff.Mode.SRC_IN));

        // Always want a cache allocated.
        mCacheBitmap = Bitmap.createBitmap(1, 1, Bitmap.Config.ARGB_8888);

        if (mDesaturateOnPress) {
            // Create a desaturate color filter for pressed state.
            ColorMatrix cm = new ColorMatrix();
            cm.setSaturation(0);
            mDesaturateColorFilter = new ColorMatrixColorFilter(cm);
        }
    }
```

```
        @Override
        protected boolean setFrame(int l, int t, int r, int b) {
            final boolean changed = super.setFrame(l, t, r, b);
            mBounds = new Rect(0, 0, r - l, b - t);
            mBoundsF = new RectF(mBounds);

            if (mBorderDrawable != null) {
                mBorderDrawable.setBounds(mBounds);
            }
            if (mMaskDrawable != null) {
                mMaskDrawable.setBounds(mBounds);
            }

            if (changed) {
                mCacheValid = false;
            }

            return changed;
        }

        @Override
        protected void onDraw(Canvas canvas) {
            if (mBounds == null) {
                return;
            }

            int width = mBounds.width();
            int height = mBounds.height();

            if (width == 0 || height == 0) {
                return;
            }

            if (!mCacheValid || width != mCachedWidth || height != mCachedHeight) {
                // Need to redraw the cache
                if (width == mCachedWidth && height == mCachedHeight) {
                    // Have a correct-sized bitmap cache already allocated. Just
                    // erase it.
                    mCacheBitmap.eraseColor(0);
                } else {
                    // Allocate a new bitmap with the correct dimensions.
                    mCacheBitmap.recycle();
                    // noinspection AndroidLintDrawAllocation
                    mCacheBitmap = Bitmap.createBitmap(width, height,
                            Bitmap.Config.ARGB_8888);
                    mCachedWidth = width;
                    mCachedHeight = height;
                }

                Canvas cacheCanvas = new Canvas(mCacheBitmap);
                if (mMaskDrawable != null) {
                    int sc = cacheCanvas.save();
                    mMaskDrawable.draw(cacheCanvas);
                    mMaskedPaint
                            .setColorFilter((mDesaturateOnPress && isPressed()) ? mDesaturateColorFilter
                                    : null);
                    cacheCanvas.saveLayer(mBoundsF, mMaskedPaint,
                            Canvas.HAS_ALPHA_LAYER_SAVE_FLAG
                                    | Canvas.FULL_COLOR_LAYER_SAVE_FLAG);
                    super.onDraw(cacheCanvas);
                    cacheCanvas.restoreToCount(sc);
                } else if (mDesaturateOnPress && isPressed()) {
                    int sc = cacheCanvas.save();
                    cacheCanvas.drawRect(0, 0, mCachedWidth, mCachedHeight,
                            mBlackPaint);
                    mMaskedPaint.setColorFilter(mDesaturateColorFilter);
                    cacheCanvas.saveLayer(mBoundsF, mMaskedPaint,
                            Canvas.HAS_ALPHA_LAYER_SAVE_FLAG
                                    | Canvas.FULL_COLOR_LAYER_SAVE_FLAG);
                    super.onDraw(cacheCanvas);
                    cacheCanvas.restoreToCount(sc);
                } else {
                    super.onDraw(cacheCanvas);
                }
                if (mBorderDrawable != null) {
                    mBorderDrawable.draw(cacheCanvas);
                }
            }

            // Draw from cache
            canvas.drawBitmap(mCacheBitmap, mBounds.left, mBounds.top, null);
        }

        @Override
        protected void drawableStateChanged() {
            super.drawableStateChanged();
            if (mBorderDrawable != null && mBorderDrawable.isStateful()) {
                mBorderDrawable.setState(getDrawableState());
            }
            if (mMaskDrawable != null && mMaskDrawable.isStateful()) {
                mMaskDrawable.setState(getDrawableState());
            }
            if (isDuplicateParentStateEnabled()) {
                ViewCompat.postInvalidateOnAnimation(this);
            }
        }

        @Override
        public void invalidateDrawable(Drawable who) {
            if (who == mBorderDrawable || who == mMaskDrawable) {
                invalidate();
            } else {
                super.invalidateDrawable(who);
            }
        }

        @Override
        protected boolean verifyDrawable(Drawable who) {
            return who == mBorderDrawable || who == mMaskDrawable
                    || super.verifyDrawable(who);
        }
    }
```

Write Preview

Leave a comment

Attach files by dragging & dropping, selecting them, or pasting from the clipboard.

Styling with Markdown is supported

Comment