```
package com.example.covid_19alertapp.roomdatabase;

import android.content.Context;
import android.location.Location;
import android.util.Log;

import com.example.covid_19alertapp.extras.LogTags;
import com.example.covid_19alertapp.roomdatabase.VisitedLocations;
import com.example.covid_19alertapp.roomdatabase.VisitedLocationsDao;
import com.example.covid_19alertapp.roomdatabase.VisitedLocationsDatabase;

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

public abstract class LocalDBContainer {
    /*
    fit location in container
    insert to local DB
     */

    private static VisitedLocationsDatabase database;
    private static VisitedLocationsDao visitedLocationsDao;

    // container based on current position
    private static List<String> diagonalRangePoint =new ArrayList<>();



    public static void addToLocalDB(Location location, String dateTime, Context context) {
        // get the current container
        calculateContainer(location.getLatitude(), location.getLongitude(), "Bangladesh");


        // now send container and dateTime to RoomDB


        // get the database config stuff
        database = VisitedLocationsDatabase.getDatabase(context);
        visitedLocationsDao = database.visitedLocationsDao();


        final List<VisitedLocations> visitedLocationList = new ArrayList<>();


        for (String drp: diagonalRangePoint) {


            // format = "lat1,lon1,lat2,lon2_dateTime"
            String conatainerDateTimeComposite = drp+"_"+dateTime;


            visitedLocationList.add(
                    new VisitedLocations(conatainerDateTimeComposite, 1)
            );


        }


        Log.d(LogTags.LocalDBContainer_TAG, "addToLocalDB: db entry list size =
"+visitedLocationList.size()+"\n\n");


        // insert to db in a separate thread
        database.databaseWriteExecutor.execute(new Runnable() {
            @Override
            public void run() {
```

```
            for(VisitedLocations entry: visitedLocationList){
            // insert/update for each entry

                try {
                    // try to insert to db
                    visitedLocationsDao.insertLocations(entry);

                    Log.d(LogTags.LocalDBContainer_TAG, "run: room entry created");

                }catch (Exception e){
                    // entry already exists, update count

                    visitedLocationsDao.update(entry.getConatainerDateTimeComposite());

                    Log.d(LogTags.LocalDBContainer_TAG, "run: room entry updated");
                }

            }
        }
    });

}



public static List<String> calculateContainer(Double lat, Double lon, String country)
{
    Double latDevider=0.000000d, lonDevider=0.000000d, latX, lony;
```

```
        // reset the previous list
        diagonalRangePoint =new ArrayList<>();

        // this is so nice
        if(country.equals("Bangladesh")){
            latDevider=.0002000d;
            lonDevider=.0002000d;

        }

        latX=Math.floor(lat/latDevider)*latDevider;
        lony=Math.floor(lon/lonDevider)*lonDevider;
        //upper left          upper right
        Double boxA_X,boxA_Y,boxC_X,boxC_Y;                //upper box
        boxA_X=latX;                                    //#### C
        boxA_Y=lony;                    //left          // #  #   right box(x,y)
        boxC_X=latX+latDevider;                         //  #  #
        boxC_Y=lony+lonDevider;                         //(A)####
        //    #  #   lower


diagonalRangePoint.add(checkLatLongLength(Double.toString(boxA_X))+","+checkLatLongLength(Doubl
e.toString(boxA_Y))+","+checkLatLongLength(Double.toString(boxC_X))+","+checkLatLongLength(Double
.toString(boxC_Y)));



        if(lat- boxA_X<latDevider/4){
            //left box's diagonal points are to be inserted
            diagonalRangePoint.add(checkLatLongLength(Double.toString(boxA_X-
latDevider))+","+checkLatLongLength(Double.toString(boxA_Y))+","+checkLatLongLength(Double.toStrin
g(boxA_X))+","+checkLatLongLength(Double.toString(boxC_Y)));


        }
        else if(boxC_X-lat<latDevider/4){
            //right box's diagonal points are to be inserted

diagonalRangePoint.add(checkLatLongLength(Double.toString(boxC_X))+","+checkLatLongLength(Doubl
e.toString(boxA_Y))+","+checkLatLongLength(Double.toString(boxC_X+latDevider))+","+checkLatLongLe
ngth(Double.toString(boxC_Y)));


        }
        if(lon- boxA_Y<latDevider/4){
            //lower box's diagonal points are to be inserted

diagonalRangePoint.add(checkLatLongLength(Double.toString(boxA_X))+","+checkLatLongLength(Doubl
e.toString(boxA_Y-
lonDevider))+","+checkLatLongLength(Double.toString(boxC_X))+","+checkLatLongLength(Double.toStri
ng(boxA_Y)));


        }
        else if(boxC_Y-lon<lonDevider/4){
            //Upper box's diagonal points are to be inserted



diagonalRangePoint.add(checkLatLongLength(Double.toString(boxA_X))+","+checkLatLongLength(Doubl
e.toString(boxC_Y))+","+checkLatLongLength(Double.toString(boxC_X))+","+checkLatLongLength(Double
.toString(boxC_Y+lonDevider)));


        }
        if(boxC_X-lat <latDevider/4 && boxC_Y-lon<lonDevider/4){
            //Upper Right  box's diagonal points are to be inserted
```

```
diagonalRangePoint.add(checkLatLongLength(Double.toString(boxC_X))+","+checkLatLongLength(Double.toString(boxC_Y))+","+checkLatLongLength(Double.toString(boxC_X+latDevider))+","+checkLatLongLength(Double.toString(boxC_Y+lonDevider)));


    }
    else if(lat- boxA_X <latDevider/4 && lon- boxA_Y<lonDevider/4){
      //Lower left box's diagonal points are to be inserted
      diagonalRangePoint.add(checkLatLongLength(Double.toString(boxA_X-latDevider))+","+checkLatLongLength(Double.toString(boxA_Y-lonDevider))+","+checkLatLongLength(Double.toString(boxA_X))+","+checkLatLongLength(Double.toString(boxA_Y)));


    }
    else if(lat- boxA_X <latDevider/4 && boxC_Y-lon<lonDevider/4){
      //Upper Left  box's diagonal points are to be inserted
      diagonalRangePoint.add((Double.toString(boxA_X-latDevider))+","+checkLatLongLength(Double.toString(boxC_Y))+","+checkLatLongLength(Double.toString(boxA_X))+","+checkLatLongLength(Double.toString(boxC_Y+latDevider)));


    }
    else if(boxC_X-lat <latDevider/4 && lon- boxA_Y<lonDevider/4){
      //Lower Right  box's diagonal points are to be inserted

diagonalRangePoint.add(checkLatLongLength(Double.toString(boxC_X))+","+checkLatLongLength(Double.toString(boxA_Y-lonDevider))+","+checkLatLongLength(Double.toString(boxC_X+latDevider))+","+checkLatLongLength(Double.toString(boxA_Y)));
    }


    Log.d(LogTags.LocalDBContainer_TAG, "calculateContainer: diagonalPoints size = "+diagonalRangePoint.size());
```

```
        return diagonalRangePoint;


    }


    //This method keeps the lenght of the String same all the time
    private static String checkLatLongLength(String latLonDigits){
        int index;
        int len=latLonDigits.length();
        int decimalPointIndex=latLonDigits.indexOf('.');
        int checkRequiredDigits=len-decimalPointIndex-1;
        if(checkRequiredDigits<6){
            for(index=checkRequiredDigits;index<6;index++)
                latLonDigits=latLonDigits+"0";
        }
        else if(checkRequiredDigits>6){
            return latLonDigits.substring(0, len -checkRequiredDigits+6 );
        }


        return latLonDigits;
    }
}
```