SAMPLE COLLECTION MOBILE APP WITH HQ OVERSIGHT

MOBILE APP

- Download a blank Android project which has Parse Android SDK installed in it.
- Open the project in Eclipse IDE.
- Open ParseApplication.java file and add your application key and client key.
- Also enable the public read access to this object.

```
@Override
  public void onCreate()
  {
    super.onCreate();

    // Initialize Crash Reporting.
    ParseCrashReporting.enable(this);

    // Enable Local Datastore.
    Parse.enableLocalDatastore(this);

    // Add your initialization code here
    String App_Id="YoqCAzZdCM3mLjPdcfBA32ZIlMqEJJ8jkmNFHCYa";
    String Client_Key="qRkySLLp6txa0kHzHpYOxxmnDp2X1rWnd4N2Vxt5";

    Parse.initialize(this, App_Id, Client_Key);
   // Parse.initialize(this);

    ParseUser.enableAutomaticUser();
    ParseACL defaultACL = new ParseACL();

    // Optionally enable public read access.
    defaultACL.setPublicReadAccess(true);
    ParseACL.setDefaultACL(defaultACL, true);

  }
```

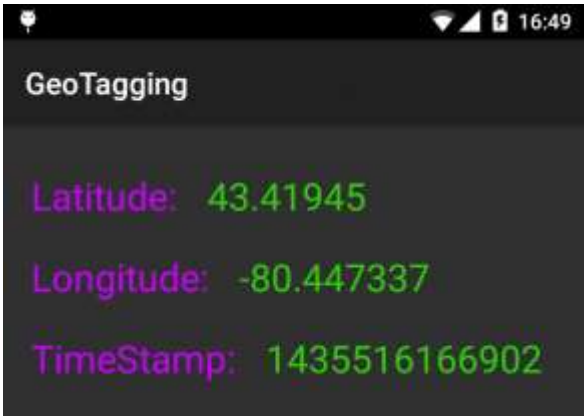
- Now open ParseStarterActivity.java file and add the following code:

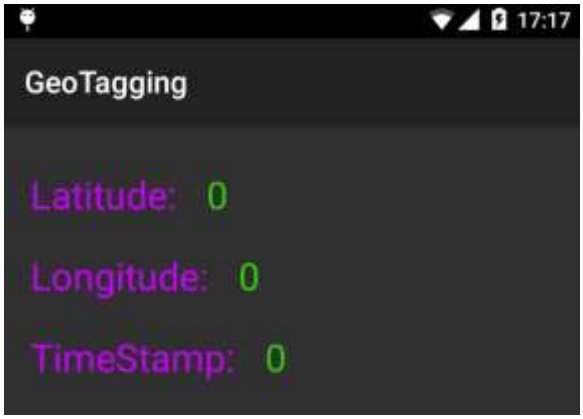
```
public class ParseStarterProjectActivity extends Activity {

        LocationManager myLocationManager;
        String PROVIDER = "";
        Location location;

       /** Called when the activity is first created. */
       public void onCreate(Bundle savedInstanceState) {
              super.onCreate(savedInstanceState);
              setContentView(R.layout.main);

              ParseAnalytics.trackAppOpenedInBackground(getIntent());

              //Retrieve location based on the cell network or Wi-fi
               PROVIDER = LocationManager.NETWORK_PROVIDER;//.GPS_PROVIDER;
               myLocationManager = (LocationManager)getSystemService(Context.LOCATION_SERVICE);

               //get last known location, if available
               location=getLocation(PROVIDER);
```

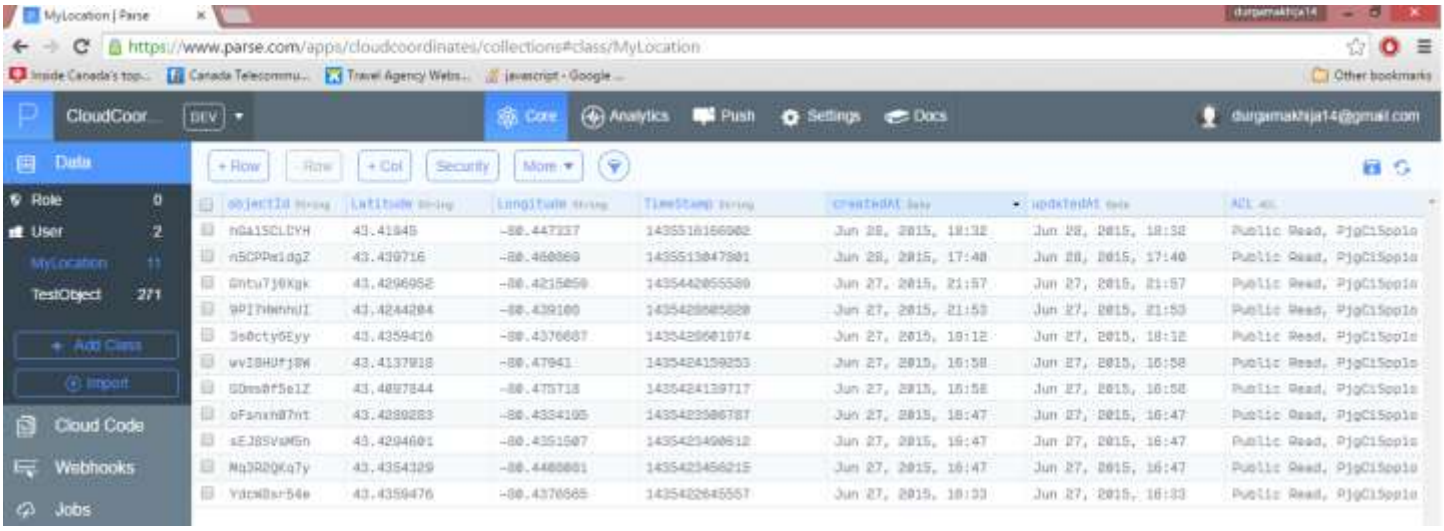
```
            saveMyLocation(location);

            //ParseObject.createWithoutData("MyLocation", "hKXGHSkSRH").deleteEventually();
      }

        public Location getLocation(String provider)
        {
                  if (myLocationManager.isProviderEnabled(provider))
                  {
                        myLocationManager.requestLocationUpdates(provider, 0, 0,
myLocationListener);
                        if (myLocationManager != null)
                        {
                              location = myLocationManager.getLastKnownLocation(provider);
                              return location;
                        }
                  }
                  return null;
            }

        private void saveMyLocation(Location l)
        {
              String lat="0";
              String lon="0";
              String time="0";

              if(l != null)
              {
                   //if location object is not null,
                   //then convert the latitude, longitude and timestamp into strings
                   //and save in the corresponding variables.
                  lat=String.valueOf(l.getLatitude());
                  lon=String.valueOf(l.getLongitude());
                  time=String.valueOf(l.getTime());

                  //Create a parseobject named MyLocation
                  //store all these location details in it
                  //and upload to the cloud.
                  ParseObject myLocation = new ParseObject("MyLocation");
                  myLocation.put("Latitude", lat);
                  myLocation.put("Longitude", lon);
                  myLocation.put("TimeStamp", time);
                  myLocation.saveInBackground();
              }

              // display latitude
              TextView txtLat= (TextView) findViewById(R.id.Latitude2);
              txtLat.setText(lat);

              // display longitude
              TextView txtLon= (TextView) findViewById(R.id.Longitude2);
              txtLon.setText(lon);

              // display timestamp
              TextView txtTime= (TextView) findViewById(R.id.Time2);
              txtTime.setText(time);
        }
```

```
        private final LocationListener myLocationListener = new LocationListener()
        {
            @Override
            public void onLocationChanged(final Location location) {
                saveMyLocation(location);
            }
            @Override
            public void onProviderDisabled(String provider) {
                // TODO Auto-generated method stub
            }
            @Override
            public void onProviderEnabled(String provider) {
                // TODO Auto-generated method stub
            }
            @Override
            public void onStatusChanged(String provider, int status, Bundle extras) {
                // TODO Auto-generated method stub
            }
        };

}
```

- In the code above, first the location is received using either the cell network or the Wi-Fi connection.
- Then the function saveMyLocation displays the location object details in the textview on the screen.
- Location is received:
  - Its latitude, longitude and timestamp are displayed on the mobile screen.
  - The parse object is created named MyLocation, all these details are stored in this object and this object is then uploaded to the cloud.
- Location is not received:
  - By default 0 is displayed on the screen for latitude, longitude and timestamp values.
  - The location details are not uploaded to the cloud.







DESKTOP MAP

- Download a blank Visual Studio project which has Parse JavaScript SDK installed in it.
- Open the project in visual studio.
- In default.html file add an iframe to display the Google map.
- Add the following code in default.html file.

```
<p><span style="color:#66CD00; font-size:22px; margin-left:30px;">
       Sample Collection Mobile App with HQ Oversight</span></p>
<iframe id="Map" src="ms-appx-web:///map.html" style="width:100%;height:100%;"></iframe>
```

- Add map.html and map.css files to the project.
- Open map.css file and add the following code.

```
html, body, #mapdisplay {
    margin: 0;
    padding: 0;
    height: 100%;
}
```

- Open map.html file and add the following code to it.

```
<!DOCTYPE html>
<html>
<head>
    <title>Map</title>

    <!-- Google Maps API reference -->
    <script src="https://maps.googleapis.com/maps/api/js?sensor=false&libraries=visualization">
    </script>

    <script src="/lib/parse-1.4.2.js"></script>

    <!-- mapframe references  -->
    <link href="/map.css" rel="stylesheet" />

</head>

<body>
```

```
        <div id="mapdisplay"></div>
</body>
</html>
```

- The map is displayed in the div element with id “mapdisplay”.
- Now add the script tag within the head tag. This script tag contains the entire code to display map, get coordinates from parse, display markers for each coordinate and refresh the markers after every 15 seconds.

```
<script>

var APPLICATION_ID = "YoqCAzZdCM3mLjPdcfBA32ZIlMqEJJ8jkmNFHCYa";
var JAVASCRIPT_KEY = "PA63YUlxqLCZl0wHjYF7hrNZQAydkzVn2Yhm3ui6";
var CLASS_NAME = "MyLocation";
var CoorinatesArray=[];
var MarkersArray=[];

var map;
var myCenter = new google.maps.LatLng(43.4359863, -80.4376289); //Fergus avenue coordinates
var infoWindow;

//this function initializes the google map
function Initialize()
{
    CenterMap(myCenter);

    // Initialize Parse with your Application ID and JavaScript key from the
    // Parse dashboard.
           //Parse.initialize("YoqCAzZdCM3mLjPdcfBA32ZIlMqEJJ8jkmNFHCYa",
   "PA63YUlxqLCZl0wHjYF7hrNZQAydkzVn2Yhm3ui6");

    Parse.initialize(APPLICATION_ID, JAVASCRIPT_KEY);
    GetCoordinates(CLASS_NAME);
}

//this function centers the map at the given position.
function CenterMap(myCenter)
{
    map = new google.maps.Map(document.getElementById('mapdisplay'),
        {
            zoom: 15,
            center: myCenter,
            mapTypeId: google.maps.MapTypeId.ROADMAP
            //.TERRAIN
        });
}

//this function gets the coordinates from the cloud
function GetCoordinates(CLASS_NAME)
{
    var LocationObject = Parse.Object.extend(CLASS_NAME);
    var query = new Parse.Query(LocationObject);
    query.descending('TimeStamp');
    query.find({
        success: function (results)
        {
            if (results.length > 0)
            {
                CoorinatesArray = new Array();
                // The object was retrieved successfully.
                for (var i = 0; i < results.length; i++)
```

```
                    {
                        var obj = results[i];
                        var lat = obj.get("Latitude");
                        var lon = obj.get("Longitude");
                        var time = obj.get("TimeStamp");
                        if (lat != null && lat != "0" && lat != 0)
                        {
                            if (lon != null && lon != "0" && lon != 0)
                            {
                                CoorinatesArray[i] = { latitude: lat, longitude: lon, timestamp: time
};
                            }
                        }
                    }
                    AddMarkers(CoorinatesArray);
                }
            },
            error: function (error)
            {
                CoorinatesArray=new Array();
            }
        });
    }

    //this function adds the marker on the map for each coordinate data collected from the cloud
    function AddMarkers(dataresults)
    {
        ResetMarkers(MarkersArray);
        for (var i = 0; i < dataresults.length; i++)
        {
            var coordinate = dataresults[i];
            var lat = coordinate["latitude"];
            var lon = coordinate["longitude"];
            var time = coordinate["timestamp"];
            var latLong = new google.maps.LatLng(lat, lon);
            var myIcon = 'images/marker.png';
            if (i == 0)
            {
                //display the latest coordinate with pink marker
                myIcon = 'images/pink.png';
            }
            else
            {
                //display all other coordinates with blue marker
                myIcon = 'images/blue.png';
            }

            var marker = new google.maps.Marker({
                position: latLong,
                map: map,
                animation: google.maps.Animation.BOUNCE,
                icon: myIcon
            });
            MarkersArray.push(marker);
            var mycontent = GetContent(lat, lon, time);
            infoWindow = new google.maps.InfoWindow;
            BindInfoWindow(marker, map, infoWindow, mycontent);
        }

        //sets the timer to get coordinates from cloud after every 15 seconds
        setTimeout(function ()
```

```
        {
            GetCoordinates(CLASS_NAME);
        }, 15000);
    }

    function BindInfoWindow(marker, map, infoWindow, html)
    {
        //binds the info window to the marker on mouseover event
        google.maps.event.addListener(marker, 'mouseover', function () {
            infoWindow.setContent(html);
            infoWindow.open(map, marker);
        });

        //unbinds the info window from the marker on mouseout event
        google.maps.event.addListener(marker, 'mouseout', function () {
            infoWindow.close();
        });
    }

    //this function resets the markers array and removes old markers from the map.
    function ResetMarkers(arr)
    {
       for (var i = 0; i < arr.length; i++)
       {
           arr[i].setMap(null);
       }
       arr = [];
    }

    //converts timestamp string into date time format as (YYYY-MM-DD HH:MM:SS)
    function FormatDate(timestamp)
    {
        var d = new Date(parseFloat(timestamp));
               return d.getFullYear() + '/' + (d.getMonth() + 1) + '/' + d.getDate()+" "+d.getHours()+"
       : "+d.getMinutes() +" : "+d.getSeconds();
    }

    //this function accepts 3 string variables from location object
    //creates a div element with each of these variables in the p tag.
    //this div tag is used as a content for infowindow for each variable
    function GetContent(lat, lon, time)
    {
        var mytime=FormatDate(time);
        var html = "<div><p><b><span style='color: #008000;'>Latitude: </span><span style='color:
#800080;'>" + lat + "</span></b></p>" +
            "<p><b><span style='color: #008000;'>Longitude: </span><span style='color: #800080;'>" +
lon + "</span></b></p>" +
            "<p><b><span style='color: #008000;'>Time: </span><span style='color: #800080;'>" + mytime
+ "</span></b></p></div>"
        return html;
    }

     google.maps.event.addDomListener(window, 'load', Initialize);

    </script>
```

- First add your parse Application key and JavaScript key to update parse initialize call.
- When map.html is loaded in the iframe, google map is displayed in by the Initialize function.
- Map is centered at the given coordinates (here Fergus avenue, Kitchener).
- GetCoordinates function then collects all the coordinates from the cloud.

- The class name on the cloud is "MyLocation". The coordinates are collected in the CoordinatesArray which is then passed to the AddMarkers function.
- ResetMarkers function removes any previous markers from the map and resets the markers array.
- Then for each coordinate in the CoordinatesArray, a marker is created and added to the map.
- The marker for the latest coordinate is displayed in pink color while rest all coordinates have blur markers.
- Then an info window is also attached with each marker. The content displayed in this marker is latitude, longitude and time when this coordinate was collected.
- This whole process of collecting coordinates and displaying them on map is refreshed every 15 seconds as the function GetCoordinates is called after 15 seconds.

The screenshot below shows the working map on the desktop.