Code for code inspection

Description:

The core functionality of our project is the Map Activity. Whenever user selects a route from the route list, we show the Map Activity to the user, i.e. TrackerMapActivity.

We have following features in this activity.

- Display the location of buses running on that route on the Map
- Checkbox option for user to see the bus stops of that route.
- Checkbox option for user to see the schedule of that route on that day.
- We have option to see the current location of the user.

In android structure, we create an Activity which is in java and the UI is written in XML which android system interprets.

Following is the code for Activity in Java:

```
package edu.unt.transportation.bustrackingsystem;

import android.Manifest;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.os.Bundle;
import android.support.annotation.NonNull;
import android.support.v4.app.ActivityCompat;
import android.support.v4.content.ContextCompat;
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.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.CheckBox;
import android.widget.LinearLayout;
import android.widget.ListView;
import android.widget.Spinner;

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.firebase.database.ChildEventListener;
```

```
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 java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import edu.unt.transportation.bustrackingsystem.model.BusStop;
import edu.unt.transportation.bustrackingsystem.model.StopSchedule;
import edu.unt.transportation.bustrackingsystem.model.Vehicle;

import static edu.unt.transportation.bustrackingsystem.R.drawable.bus;

/**
 * This is the central activity of the project which displays the bus
locations on the Google Maps and It has the option of showing bus
stops and
 * bus stop schedule on the Map.<br>
 *     <b>Data Structure :</b><br>
 *     We have used ArrayList in the class to store the list of
vehicles, bus stops.
 *     We have used HashMap to Map the bus name to the list of
schedule. <br>
 * Created by Anurag Chitnis on 10/5/2016. <br>
 *     <b>Revision History: </b><br>
 *     11/10/2016 4:50 PM Anurag Chitnis Added the Options Menu<br>
 *     11/10/2016 12:37 PM    Anurag Chitnis Changed the method of
invocation of Tracker Map Activity<br>
 *     11/9/2016 10:40 PM Gill Wasserman Added next 3 times at each
main stop to Route List adapter.  Selecting a route takes you to
TrackerMapActivity<br>
 *     11/9/2016 7:01 PM  Anurag Chitnis Changed the layout parameters
for Schedule list, Added error handling<br>
 *     11/8/2016 12:43 PM Anurag Chitnis Added the view for showing
schedule list on the Map<br>
 *     11/7/2016 12:26 PM Anurag Chitnis Added unit test class for
tracker map activity<br>
 *     11/3/2016 1:57 PM  Anurag Chitnis Added Bus stop display
feature on the google map<br>
 *     11/1/2016 7:52 PM  Anurag Chitnis Updated the tracker map to
show the runtime updates<br>
 *     10/29/2016 7:19 PM Anurag Chitnis Added the my location feature
and permissions model for map<br>
 *     10/28/2016 5:32 PM Anurag Chitnis Added the Tracker Map
Activity<br>
 */

public class TrackerMapActivity extends AppCompatActivity implements
```

```
OnMapReadyCallback,
        ActivityCompat.OnRequestPermissionsResultCallback,
ValueEventListener {

    /**
     * We are using this tag to print the logs in this class
     */
    private static final String TAG =
TrackerMapActivity.class.getName();
    private static final String FIREBASE_VEHICLES = "vehicles";

    public static String KEY_ROUTE_ID = "keyRouteId";

    /**
     * Instance of Google Map to show currently running buses and bus
stops
     */
    private GoogleMap mMap;
    private DatabaseReference mDatabase;
    /**
     * List of the vehicles running on the currently selected
     */
    private List<Vehicle> vehicleList = new ArrayList<>();
    /**
     * List of the bus stops available on the currently selected route
     */
    private List<BusStop> busStopList = new ArrayList<>();
    /**
     * Reference of the 'vehicles' node, where we have all the vehicle
objects located
     */
    private DatabaseReference vehiclesRef;
    /**
     * We are keeping all the bus stop markers in a list so that we
can remove them and add them on the UI,
     * whenever user toggles the 'show bus stops' checkbox
     */
    private List<Marker> mBusStopMarkerList = new ArrayList<>();
    /**
     * Map of markers to keep track of all the markers added for bus
tops and remove them whene ever necessary
     */
    private Map<String, Marker> markerMap;
    /**
     * Listener to receive callbacks for bus stops on the selected
route
     */
    private BusStopListener busStopListener;
    /**
     * Checkbox view to show and hide bus stop location
     */
    private CheckBox mStopCheckBox;
```

```
    /**
     * Checkbox view to show and hide schedule list view
     */
    private CheckBox mScheduleCheckBox;
    /**
     * List of schedule which will appear on the display
     */
    private List<String> scheduleList;
    /**
     * Map to keep track of the busName and the linked schedule
     * Key - busName, Value - List of schedule
     */
    private Map<String, List<String>> scheduleListMap;
    /**
     * List of all the names of bus stops which are scheduled, i.e we
have a schedule of timings for those bus stops
     * Example: For Discovery Park route, we have dp_main and gab as
schedules stops
     */
    private List<String> scheduledStopsList;
    /**
     * Adapter for the list view on which we show the schedule
     */
    private ArrayAdapter scheduleListAdapter;
    /**
     * Adapter for the spinner where we show the list of scheduled
stops
     */
    private ArrayAdapter<String> scheduledStopsAdapter;
    /**
     * Schedule list layout : layout with transparent black background
     * we show this layout when user clicks on 'show schedule'
otherwise its visibility is set to 'GONE' be default
     */
    private LinearLayout scheduleListLayout;
    /**
     * This is the route id which we get from the previous activity
     */
    private String routeID;

    /**
     * Get the list of vehicles available on the currently selected
route
     * @return List of Vehicles
     */
    public List<Vehicle> getVehicleList() {
        return vehicleList;
    }

    /**
     * Get the list of bus stops available on the currently selected
route
```

```
     * @return List of Bus stops
     */
    public List<BusStop> getBusStopList() {
        return busStopList;
    }

    /**
     * Request code for location permission request.
     *
     * @see #onRequestPermissionsResult(int, String[], int[])
     */
    private static final int LOCATION_PERMISSION_REQUEST_CODE = 1;

    /**
     * Flag indicating whether a requested permission has been denied
after returning in
     * {@link #onRequestPermissionsResult(int, String[], int[])}.
     */
    private boolean mPermissionDenied = false;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_tracker_map);

        routeID = getIntent().getStringExtra(KEY_ROUTE_ID);
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);
        if(getSupportActionBar() != null) {
            getSupportActionBar().setDisplayHomeAsUpEnabled(true);
            getSupportActionBar().setDisplayShowHomeEnabled(true);
        }

        scheduleList = new ArrayList<>();
        scheduledStopsList = new ArrayList<>();
        scheduleListMap = new HashMap<>();

        mStopCheckBox = (CheckBox) findViewById(R.id.stopCheckbox);
        mScheduleCheckBox = (CheckBox)
findViewById(R.id.scheduleCheckbox);

        scheduleListAdapter = new ArrayAdapter<>(this,
                R.layout.schedule_listview, scheduleList);

        scheduleListLayout = (LinearLayout)
findViewById(R.id.scheduleListLayout);

        ListView listView = (ListView)
findViewById(R.id.scheduleList);
        listView.setAdapter(scheduleListAdapter);

        Spinner spinner = (Spinner) findViewById(R.id.busStopSpinner);
```

```
        // Create an ArrayAdapter using the string array and a default
spinner layout
        scheduledStopsAdapter = new ArrayAdapter<>(this,
R.layout.spinner_item, scheduledStopsList);
        // Specify the layout to use when the list of choices appears

scheduledStopsAdapter.setDropDownViewResource(android.R.layout.simple_
spinner_dropdown_item);
        // Apply the scheduleListAdapter to the spinner
        spinner.setAdapter(scheduledStopsAdapter);

        spinner.setOnItemSelectedListener(new
AdapterView.OnItemSelectedListener() {
            @Override
            public void onItemSelected(AdapterView<?> parent, View
view, int position, long id) {
                String busStopName = (String)
parent.getItemAtPosition(position);
                List<String> scheduleStringList =
scheduleListMap.get(busStopName);
                scheduleList.clear();
                if(scheduleStringList != null) {
                    scheduleList.addAll(scheduleStringList);
                }
                else {
                    Log.e(TAG,"while taking the spinner found that
scheduleStringList is null");
                }
                scheduleListAdapter.notifyDataSetChanged();
            }

            @Override
            public void onNothingSelected(AdapterView<?> parent) {
                Log.w(TAG,"Spinner: Nothing is selected");
            }
        });

        // Set up Google Maps
        SupportMapFragment mapFragment = (SupportMapFragment)

getSupportFragmentManager().findFragmentById(R.id.trackerMap);
        mapFragment.getMapAsync(this);

        markerMap = new HashMap<>();

        mDatabase = FirebaseDatabase.getInstance().getReference();
        vehiclesRef = mDatabase.child(FIREBASE_VEHICLES);
        busStopListener = new BusStopListener();
        /**
         * Set the routeID, whose vehicleMap we want to listen to and
register for the listener
         */
```

```
        VehicleMapChangeListener vehicleMapChangeListener = new
VehicleMapChangeListener();
        if (null != routeID) {
            vehicleMapChangeListener.registerListener(routeID);
            DatabaseReference busStopRef = mDatabase.child("routes/" +
routeID + "/busStopMap");
            busStopRef.addListenerForSingleValueEvent(new
ValueEventListener() {
                @Override
                public void onDataChange(DataSnapshot dataSnapshot) {
                    for (DataSnapshot busStop :
dataSnapshot.getChildren()) {
                        Log.d(TAG, "busStopRef : onDataChange " +
busStop.getKey());

busStopListener.registerListener(busStop.getKey());
                    }
                }

                @Override
                public void onCancelled(DatabaseError databaseError) {
                    Log.e(TAG, "busStopRef : onCancelled() "+
databaseError.getMessage());
                }
            });
        } else {
            Log.e(TAG,"RouteID is null");
        }
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it
is present.
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case R.id.action_signIn:
                startActivity(new Intent(this, SignInActivity.class));
                return true;

            default:
                // If we got here, the user's action was not
recognized.
                // Invoke the superclass to handle it.
                return super.onOptionsItemSelected(item);

        }
```

```
    }

    @Override
    public void onMapReady(GoogleMap googleMap) {
        mMap = googleMap;
        enableMyLocation();
        LatLng mapCenter = new LatLng(33.2139981,-97.1483429);
        mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(mapCenter,
13));
    }

    @Override
    public void onDataChange(DataSnapshot dataSnapshot) {
        Log.d(TAG,"vehicleChangeListener : onDataChange
"+dataSnapshot.getKey());
        Vehicle vehicleSnapShot =
dataSnapshot.getValue(Vehicle.class);
        LatLng vehiclePosition = new
LatLng(vehicleSnapShot.getLatitude(),vehicleSnapShot.getLongitude());
        if(vehicleList.contains(vehicleSnapShot)) {
            // Replace the old vehicle object with the new one
            vehicleList.set(vehicleList.indexOf(vehicleSnapShot),
vehicleSnapShot);
            /**
             * If the vehicle is already present on the UI and in the
vehicleList,
             * most probably the position is updated, so we update the
marker position
             */

markerMap.get(vehicleSnapShot.getVehicleID()).setPosition(vehiclePosit
ion);
        }
        else {
            /**
             * Perform following operations if new vehicle is added on
the route
             */
            vehicleList.add(vehicleSnapShot);
            Marker newVehicleMarker = mMap.addMarker(new
MarkerOptions()
                    .icon(BitmapDescriptorFactory.fromResource(bus))
                    .position(vehiclePosition)
                    .flat(true));

markerMap.put(vehicleSnapShot.getVehicleID(),newVehicleMarker);
        }
    }

    @Override
    public void onCancelled(DatabaseError databaseError) {
        Log.e(TAG, "vehicleChangeListener : onCancelled() "+
```

```
databaseError.getMessage());
    }

    /**
     * This method is invoked when the user clicks on show stops
checkbox on the user interface
     * @param view View of the checkbox which is being toggled
     */
    public void onStopToggled(View view) {

        if(mStopCheckBox.isChecked()) {

            for (BusStop busStop : busStopList) {
                LatLng busStopLocation = new
LatLng(busStop.getLatitude(), busStop.getLongitude());
                mBusStopMarkerList.add(mMap.addMarker(new
MarkerOptions()

.icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HU
E_AZURE))
                        .position(busStopLocation)
                        .title(busStop.getStopName())
                        .flat(true)));
            }
        } else {
            for(Marker marker : mBusStopMarkerList) {
                marker.remove();
            }
        }

    }

    /**
     * This method is invoked when user clicks on the show schedule
checkbox on the User Interface
     * @param view View of the checkbox which is being toggled
     */
    public void onScheduleToggled(View view) {
        if(mScheduleCheckBox.isChecked()) {
            loadSchedule();
            scheduleListLayout.setVisibility(View.VISIBLE);
        }
        else
            scheduleListLayout.setVisibility(View.GONE);
    }

    /**
     * Enables the My Location layer if the fine location permission
has been granted.
     */
    private void enableMyLocation() {
        if (ContextCompat.checkSelfPermission(this,
```

```
android.Manifest.permission.ACCESS_FINE_LOCATION)
                != PackageManager.PERMISSION_GRANTED) {
            // Permission to access the location is missing.
            PermissionUtils.requestPermission(this,
LOCATION_PERMISSION_REQUEST_CODE,
                    android.Manifest.permission.ACCESS_FINE_LOCATION,
true);
        } else if (mMap != null) {
            // Access to the location has been granted to the app.
            mMap.setMyLocationEnabled(true);
        }
    }

    @Override
    public void onRequestPermissionsResult(int requestCode, @NonNull
String[] permissions,
                                           @NonNull int[]
grantResults) {
        if (requestCode != LOCATION_PERMISSION_REQUEST_CODE) {
            return;
        }

        if (PermissionUtils.isPermissionGranted(permissions,
grantResults,
                Manifest.permission.ACCESS_FINE_LOCATION)) {
            // Enable the my location layer if the permission has been
granted.
            enableMyLocation();
        } else {
            // Display the missing permission error dialog when the
fragments resume.
            mPermissionDenied = true;
        }
    }

    @Override
    protected void onResumeFragments() {
        super.onResumeFragments();
        if (mPermissionDenied) {
            // Permission was not granted, display error dialog.
            showMissingPermissionError();
            mPermissionDenied = false;
        }
    }

    /**
     * Displays a dialog with error message explaining that the
location permission is missing.
     */
    private void showMissingPermissionError() {
        PermissionUtils.PermissionDeniedDialog
                .newInstance(true).show(getSupportFragmentManager(),
```

```
"dialog");
    }

    /**
     * This method parses the list of bus stops and add the scheduled
bus stops and schedule for those bus stops in the appropriate list.
     * We later use these list to show the data in spinner and list
view respectively
     */
    private void loadSchedule() {
        scheduledStopsList.clear();
        for(BusStop busStop : busStopList) {
            Map<String, List<StopSchedule>> scheduleMap =
busStop.getRouteSchedule();
            if(scheduleMap != null) {
                scheduledStopsList.add(busStop.getStopName());
                scheduledStopsAdapter.notifyDataSetChanged();
                List<StopSchedule> stopScheduleList =
scheduleMap.get(routeID);
                if(stopScheduleList != null) {
                    for (StopSchedule stopSchedule : stopScheduleList)
{
                        scheduleListMap.put(busStop.getStopName(),
stopSchedule.getTimingsList());
                    }
                }
                else
                    Log.e(TAG,"stopScheduleList is null");
            }
        }

        scheduleListAdapter.notifyDataSetChanged();
    }

    /**
     * This class receives the callback for all the BusStops which
exist on the currently selected route.
     * Let us  keep the list of BusStop objects, so they can be
displayed on the UI whenever user requests it
     */
    private class BusStopListener implements ValueEventListener {

        DatabaseReference busStopReference;

        void registerListener(String stopID) {
            busStopReference = mDatabase.child("stops");

busStopReference.child(stopID).addListenerForSingleValueEvent(this);
        }

        @Override
        public void onDataChange(DataSnapshot dataSnapshot) {
```

```
            BusStop busStopSnapShot =
dataSnapshot.getValue(BusStop.class);
            if(!busStopList.contains(busStopSnapShot)) {
                busStopList.add(busStopSnapShot);
            }
        }

        @Override
        public void onCancelled(DatabaseError databaseError) {
            Log.e(TAG, "BusStopListener : onCancelled() "+
databaseError.getMessage());
        }
    }

    /**
     * This class tracks the changes to the vehicle map of the given
route
     */
    private class VehicleMapChangeListener implements
ChildEventListener {

        DatabaseReference vehicleMapRef;

        void registerListener(String routeID) {
            vehicleMapRef =
mDatabase.child("routes/"+routeID+"/vehicleMap");
            vehicleMapRef.addChildEventListener(this);
        }

        @Override
        public void onChildAdded(DataSnapshot dataSnapshot, String s)
{
            Log.d(TAG,"onChildAdded "+dataSnapshot.getKey());
            //Register to listen to the changes made to all the
vehicles on this route

vehiclesRef.child(dataSnapshot.getKey()).addValueEventListener(Tracker
MapActivity.this);
        }

        @Override
        public void onChildChanged(DataSnapshot dataSnapshot, String
s) {

        }

        @Override
        public void onChildRemoved(DataSnapshot dataSnapshot) {
            //Remove the listener for this bus as we don't want to
receive updates from it anymore

vehiclesRef.child(dataSnapshot.getKey()).removeEventListener(TrackerMa
```

```
pActivity.this);
        }

        @Override
        public void onChildMoved(DataSnapshot dataSnapshot, String s)
{

        }

        @Override
        public void onCancelled(DatabaseError databaseError) {
            Log.e(TAG, "VehicleMapChangeListener : onCancelled() "+
databaseError.getMessage());
        }
    }
}
```

Following Code is the UI component used in the above Activity:

```
<!--
Layout File used by TrackerMapActivity.
Created By Anurag Chitnis
-->
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:orientation="vertical">
<!--
Shows App Bar - toolbar on the UI
-->
    <android.support.design.widget.AppBarLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/app_bar_layout"
        android:theme="@style/AppTheme.AppBarOverlay">

        <android.support.v7.widget.Toolbar
            android:id="@+id/toolbar"
            android:layout_width="match_parent"
            android:layout_height="?attr/actionBarSize"
            android:background="?attr/colorPrimary"
            app:popupTheme="@style/AppTheme.PopupOverlay" />

    </android.support.design.widget.AppBarLayout>

    <RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="match_parent"
        android:layout_height="match_parent">
```

```
        <!--
        Shows Google Map Fragment on the UI
        -->
        <fragment
xmlns:android="http://schemas.android.com/apk/res/android"
            android:id="@+id/trackerMap"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            class="com.google.android.gms.maps.SupportMapFragment" />
        <!--
            This layout shows Checkboxes for bus stops and schedules
on the UI
        -->
        <LinearLayout
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentLeft="true"
            android:layout_alignParentStart="true"
            android:id="@+id/checkBoxLayout"
            android:layout_alignTop="@id/trackerMap"
            android:padding="6dp"
            android:orientation="vertical"
            android:layout_alignParentEnd="true">

            <CheckBox
                android:id="@+id/stopCheckbox"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:onClick="onStopToggled"
                android:text="@string/stops" />

            <CheckBox
                android:id="@+id/scheduleCheckbox"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:onClick="onScheduleToggled"
                android:text="Show Schedule" />

        </LinearLayout>
        <!--
            This layout shows the schedule list along with the spinner
to select the name of bus stop
        -->

        <LinearLayout
            android:layout_gravity="bottom|left"
            android:background="#A000"
            android:orientation="vertical"
            android:id="@+id/scheduleListLayout"
            android:layout_below="@id/checkBoxLayout"
            android:padding="5dp"
            android:layout_height="match_parent"
```

```
            android:layout_width="180dp"
            android:visibility="gone">

            <Spinner
                android:id="@+id/busStopSpinner"
                android:layout_width="match_parent"
                android:layout_height="wrap_content" />

            <ListView
                android:id="@+id/scheduleList"
                android:layout_width="match_parent"
                android:layout_height="wrap_content" >
            </ListView>

        </LinearLayout>
    </RelativeLayout>
</LinearLayout>
```