IJCSC 0973-7391 Volume 7 • Number 2 March 2016 - Sept 2016 pp. 8-11 available online at www.csjournalss.com

A Novel Approach of Traffic Updates Based on User Feedback for Smart City

Mihir Mavalankar, Akshata Patel, Usha Patel
Institute of Technology, Nirma University

Abstract: This paper gives idea about the implementation of Application for Traffic Updatesbased on user inputs. We have made a mobile application for the residents of one city to see the status of traffic in various parts of the city and at key traffic junctions. This is purely an academic exercise and not available to the consumer. With information from this applicationcommuters can plan their route in a better way. This application uses the Google Maps API extensively to provide the traffic data given by Google and it augments this information by information collected from the inputs of the users. Inputs given by users on the ground on their own will are very accurate and valuable. We acknowledge the possibility of a user purposefully feeding wrong information but this we believe will be rare and won't affect the bigger picture. This application gives a more balanced and accurate picture of the real situation. This application uses the Firebase service for handling the work on the server side.For the purposes of this implementation we have integrated the most basic services provided by Firebase into the application by using their API. Traffic data especially in rural areas and poor countries is not always accurate and this paper looks at one of the possible ways to address this issue.
Keywords: traffic updates, Google Maps API, crowdsourcing, Firebase, SHA Key, Firebase API, user input, traffic junctions.

Introduction

These days vehicles traffic is major issue for big cities. Currently mobile device users use many different services for navigation like Google Maps. These applications take traffic data from a variety of sources, analyse it and then give the current traffic conditions. Some of the sources of information are listed below:-

1. Analyses the GPS-determined locations transmitted to Google by a large number of mobile phone users.
2. By calculating the speed of users along a stretch of road, Google is able to generate a live traffic map.
3. Traffic sensors and cameras on roads.
4. Cellular telephone companies constantly monitor the locations of user devices. One tracking method is trilateration, whereby the distance to three or more surrounding cell phone towers is measured.
5. Government departments of transportation and private data providers.

While these sources of data are useful but many times they give inaccurate data leading to faulty predictions. In this paper we try to address this by providing another source of data that may help us to make accurate predictions.

ALGORITHM

In our application we have enabled the integration of the traffic data which the Google maps API provides. Due to this the user will see different streets labelled with a different color (green, orange or yellow depending on the level of traffic). But what we have implemented differently is that we have augmented this information with inputs from the users of the application and so the user has two different sources to look from and then make a decision. Whenever a user makes an entry the algorithm given below is followed.

Steps:-

1. The user chooses from three choices for a given junction, the choices being high, medium and low.
2. This request is then sent to the server for processing via the respective API.
3. In the server's database the level which the user entered is incremented while the other two are decremented for that traffic junction.
4. On completion of this change the user gets an acknowledgement message on his application.

The algorithm given above increments the input field that the user chose and decrements the other two. This is done so the latest entry gets the most importance and so that it can cancel out the effect of entries made by previous users as they are not up to date with the current situation. Further, one can implement a time stamp system for each entry in the server. With this one can keep track of what time a certain entry was made and remove that entry after a small but definite period of time. This may be slightly more intensive for the server but is possible due to the capabilities of contemporary systems.

Another issue that may be faced is of checking for users that are giving faulty input or malicious entities that are flooding the server with faulty or incorrect requests. To combat this one needs to keep a hash table of IP addresses and find patterns in this table. One can find this if any one particular IP is sending an unusual amount of traffic data in a short time or if the user inputs have been consistently different from the inputs of other users. This is a standard practice among many companies.

IMPLEMENTATION

To make this project we have used the Android Studio Development environment. This is a free development kit provided by Google for developers to make applications that can run on the Android system. There are also other setups that one can use to make applications for Android. Eclipse development environment paired with Android Development Toolkit (ADT) is also a powerful tool for making applications on Android. Being a part of the Linux family Android code has to be written in Java and to compile one needs a Java Virtual machine (Dalvik). Android also provides the developer with the Android Virtual device manager. With this one can create a virtual device in the system you are working on to test your application. This removes the requirement of having the need of an actual Android device to help while testing the tour application. In this virtual device manager you can simulate any Nexus device (as it runs on Android) and see and test how your application looks and works on different devices.

API are an acronym for the term Application Program Interface which as the name suggests provides protocols and tools to build software applications and they differ depending upon the software in question. In Android, different apps have different needs and so they need different API's. For example if your application in a weather reporting one it will need access to a weather data API which may be paid or free depending upon the maker of the API. If one needs to store large amounts of data online they can use the Google Cloud API.
In our application we have extensively used two main API's namely the Google Maps API and the Firebase one. We integrated the Ahmedabad city map into our app through Google Maps API which Google provides for free to the developers for use. Also some of the traffic data that is shown comes from this API which we will discuss later on. The maps API gives us access to a terrain or normal map of any part of the world. We have used the Firebase API to read and write data to the Firebase Database that we created for this application. Our experiment is based on Ahmedabad city, which is one of the most populated city of India.

Integrating Google Maps:

1. To put a map of Ahmedabad city we used a fragment on a regular activity.
2. To get this map we had to activate the Google Maps API through the Goggle developer account. To activate this API, we had to provide our SHA Key to Google. This is unique key that is generated for any project that you build in Android Studio which helps Google uniquely identify your application.
3. The SHA key is verified and an API key is generated. This API key is for you to use to get the services that the respective API offers.
4. Add the map to the fragment's layout file

```
<fragment     xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:map="http://schemas.android.com/apk/res-auto"
android:id="@+id/map"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:name="com.google.android.gms.maps.MapFragment"
```

5. We can set the angle, elevation, tilt and other setting of the virtual camera that we use to view the map. We can zoom in an zoom out easily and got to different places in the city by dragging.
   We can specify the direction that the camera faces with the bearing command. (0 degree for north and 180 for south). Tilt decides the angle of the camera lens (90 degree for the camera to face directly towards the earth surface from the top)
   Zoom can take values from 0 to 21. Higher the number, closer would the camera be to the surface.

6. Next we added a traffic layer onto the map. In this layer we can see the streets in different colours depending upon their traffic conditions, red for heavy, orange for medium and blue/green for light. To add this we need to set traffic setting to 'enabled' in our map in our main activity java file.
7. Next we moved on to placing markers at important intersections in the city. Each marker, when clicked upon displays the name of the intersection and the current traffic level there depending on user input.
   To add a marker:
   Add this in the onCreate method of the activity main java file

```
XYZ= new MarkerOptions()
        .position(new LatLng(<Latitiude of the place>,<Langitude>))
        .title("string to be displayed")
        .snippet(Snippets[2]);
```

   Add this statement in the onMapReady method
   ahm_map.addMarker(**XYZ**);

User Input Integration:

We have created an activity for taking traffic input from the user. The user can choose among the three options for input. When you press one of these radio buttons your input is recorded and sent to the Firebase server where data from various users is combined to give a clear picture of the traffic condition.

Code for recording the user input:

```
public void btnClickHandler(View view) {
// Is the button now checked?
booleanchecked = ((RadioButton) view).isChecked();
intt=0;
// Check which radio button was clicked
switch(view.getId()) {
    // Go through the three cases high, medium and low in this switch case control statement.
  }
```

When any button of the appropriate traffic junction is pressed, this method is called.
The variable 't' gets an appropriate value depending on whether the user selects heavy, medium or light radio button.

Connecting the app with firebase:

The app required a database to store the level of traffic at the various traffic intersections in Ahmedabad. We used an online database named firebase for this purpose. We did the following to incorporate firebase in our app

1. Put the firebase initialisation code in the main activity java file.

   Firebase.*setAndroidContext*(**this**);
2. The traffic input taken from the user needs to be stored into the firebase database.
3. For writing into the database we need to create a firebase object.This object is tied to the specific firebase database at the mentioned URL location.
4. To write the information into the node of 'Traffic Junction XYZ' in the database,we created an object linked to that child.
5. If the user input is heavy,we increment the heavy key of the child node and decrement the medium and light key values.
6. To read the data form firebase, an object is created linked to that node and an event listener is added.

   Firebase ref1 = **new** Firebase(Constants.***FIREBASE_URL***).child(*TrafficJunctionname*);

```
    ref1.addValueEventListener(new ValueEventListener() {

    public void onDataChange(DataSnapshotdataSnapshot) {
    FireData temp = dataSnapshot.getValue(FireData.class);
    Snippets[t] = greatest(temp.h, temp.m, temp.l);   // Snippets are the marker snippet strings
    Log.e(TAG, Junctions[t] + " " + Snippets[t]);
        }


    public void onCancelled(FirebaseErrorfirebaseError) {
    Log.e(TAG, "The read failed");
        }
    });
    }
```

A datasnapshot object is created when data is modified in the database.
The getValue() method reads the value at that node and stores it in temp. This temp variable can then be used in your java file.

CONCLUSION

We developed an android application which shows traffic level at major road intersections in a particular city. The application has Google Maps API integratedinto it which is used to show the road map of the city. The traffic information is shown by the colors on the map and in snippets when the user's cursorhovers above the intersection marker.This data is stored in Firebase database from which data can be readand written easily. This real-time database is synchronized among allthe users that are connected to it.With the increasing number of cars, there is an ever increasingtraffic in the country. To select the best possible route from thepresent location to the desired destination one needs to take intoaccount the traffic at the various intersections that one needs tocross to reach that particular place. Therefore, mobile applicationsthat provide information about traffic intensity at various roadjunctions are very important in our busy lives.

FUTURE SCOPE

Further modifications that can be added to the app include deciding the time upto which the traffic input given by the user remains valid. This needs to be included because the traffic levels constantly change in the city. At one moment the traffic level at a particular junction may be heavy, but after some minutes the traffic may decrease. Thus, one needs to delete the user input after a specified amount of time for better and reliable results.
User validation can also be added to check the authenticity of the user input. An activity for user login can be included in/after the start page. This application can be easily made for many other cities with users being asked to enter the city that they are in while installing the application.

REFERENCES

1. Firebase essentials for Android – Udacity
2. Add Google Maps to your Android App – Udacity
3. Google has gotten incredibly good at predicting traffic -Tim StenovecNov. 18, 2015 - Tech Insider
4. How does Google Maps predict traffic?– Beth Brindle – HowStuffWorks
5. How Does Google Detect Traffic Congestion?- by John Machay, Demand Media
6. Benefits of Crowdsourcing—*VivekWadhwa, —Compiled by Adam Janofsky*
7. Professional JavaScript for Web Developers-Nicholas C. Zakas
8. How Google Builds Its Maps—and What It Means for the Future of Everything-ALEXIS C. MADRIGAL
9. What is crowdsourcing? *By* JENNIFER ALSEVER MONEYWATCH *March 7, 2007*