REAL-TIME TRANSPORTATION TRACKING AND MANAGEMENT SUITE

Prof. Theja Narayan
Assistant Professor,
Department of Computer Science and Engineering,
Vidya Vikas Institute of Engineering and Technology, Mysuru - 28.
thejan.cse.vviet@gmail.com

Mr. Ujwal V Urs, Mr. Shreyas D S, Mr. Darshan K S, Mr. Shreenidhi Sharma B K
Final Year UG Students, Department of Computer Science and Engineering,
Vidya Vikas Institute of Engineering and Technology, Mysuru - 28.
ujwalvurs@gmail.com, shreyasds96@gmail.com, darshanks678@gmail.com,
shrinidhi.sharma97@gmail.com

Abstract Due to the rapid increase in population, there is a need for efficient transportation management system. There is an increased burden on public transportation, like buses, because of the increase in population. To curb this, educational institutions have begun extending their own transportation facilities to their students. This paper presents an efficient, technology-driven solution to create an Android-based and Web-based, data-driven, non-device and non-browser specific tracking tool that facilitates transport tracking in an institution. The system uses various technologies like GPS (Global Positioning System), GSM (Global Standard for Mobile Communication), Geolocation, Geofencing, and so forth to implement such a system. Once deployed, this system has the potential to create a positive effect in students' lives by helping them track buses to/from college, manage wait times at their bus stops, and provide feedback about the system. The proposed system can also benefit the college management by
helping them manage their bus fleet, drivers, and routes more efficiently.

***Index Terms*—Bus tracking, real-time, geolocation, android, firebase, web, transport, software, suite, geofencing**

INTRODUCTION

The rapid development and changes in technology over the last decade has had a propounding impact on the transportation sector. Mobile technology, coupled with higher Internet bandwidth, and improved Global Positioning System (GPS) functionalities has led to the emergence of new approaches to vehicular tracking. With the recent flood of new tracking products in the marker, end users are faced with an extensive array of products that have been developed without the features that they actually seek. The challenge for transportation tracking and management is to ensure that real-time positioning is of the highest accuracy and help serve the customer base in a more efficient manner.

In present-day world, vehicles are increasing in cities around the globe. Be it working class people, students, medical professionals or common public, a majority of them own at least one vehicle today. Educational institutions with an aim of helping students commute easily to and from college provide them with transportation system. But not many students and other college-goers utilize this transport facility for many reasons; to name a few —

- They are unable to track the exact whereabouts of their daily bus (real-time position)
- They are unable to know if their bus has crossed their stop in cases where they arrive late
- They are unable to provide feedback/complaints regarding the transport service to the concerned authority due to hierarchy of officials

Considering these factors motivated us to build a system that overcomes all of these complexities with the sole purpose to create an efficient application that helps not only the students, but also the bus drivers and transport

management personnel. Increase in the usage of smartphones and an exponential growth in the Android market drove us to build an Android application that will enable students to register and bus drivers to access and use the system securely.

This suite of software applications is a sub-system of an organization, concerned with organisational activity aimed at bettering the management of their bus fleet and encouraging students and faculty to use college transport as opposed to private vehicles. Given the present-day environmental conditions prevailing around us, as responsible citizens, we are required to use collective transport options rather than private vehicles. Real Time Transportation Tracking System offers the institutions a robust set of features that will promote accurate location tracking, integrated feedback service from end-users, along with multiple other features that will benefit all classes of end users greatly.

LITERATURE REVIEW

In existing transportation management and tracking systems, each system incorporates a unique set of features, but many fails to cater to the actual needs of the end users. Many a times, the tracking is not accurate enough to predict the arrival time of the bus at a stop which depends on a multitude of factors with traffic congestion being the most dominant of them all.

In a country like India, although bus tracking systems are rarely in existence for colleges or educational institutions, there are public transportation tracking and management systems like Bengaluru City's BMTC and Mysuru's Mitra applications. With an aim to overcome the disadvantages of applications like our proposed system prevalent in the marker today, we conducted a study and reviewed a few applications. We found several shortcomings, such as —

- The applications do not provide real-time tracking of buses; they show only the designated bus stops and the time a bus on that route last crossed the stop
- The applications are not user friendly; the User Interface (UI) is unresponsive and faulty at times
- Too many errors and bugs in the code which lead to frequentapplication crashes and unresponsive device state

SYSTEM ARCHITECTURE

- Driver's android phone will be used as an on-board GPS device for the bus. Most Android phones are fitted with an Assisted GPS (AGPS) chips. The GPS transmitter on the Android device, with the help of network towers and nearby Wi-Fi Hotspots, communicates with the system of GPS satellites around the world to get the "world" address. Geocoding scheme is used to translate this "world" address into coordinates (latitude and longitude).

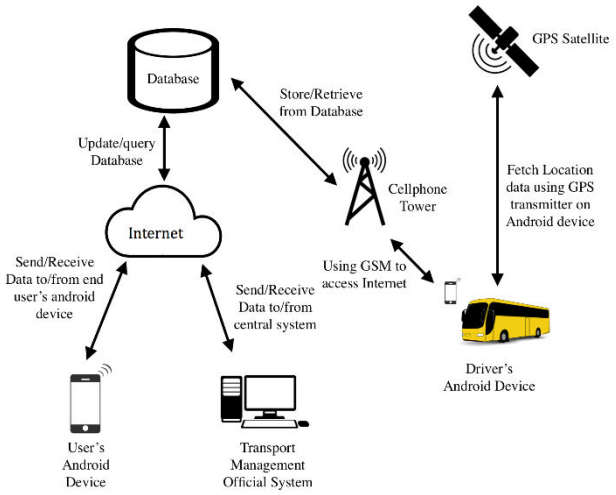

Fig. 1. System architecture of the proposed system

- Using GSM (Cellular Data), the driver's android device connects to the Internet and relays the coordinates of its current position to the cloud (database). The end users can access this real-time position of the bus from the cloud database on their Android devices.
- Transportation Management personnel at the institution can access the cloud database either through a web application developed exclusively for them.

FEATURES

- ***Real-time positioning***: Location of the bus is shown in real-time on the map with a refresh rate of 2 seconds. *Technology Used:Google Location Services.*
- ***'Wait for Me!'***: A feature that helps students alert the driver that they are in close proximity to the bus stop and will reach in a few seconds. *Technology Used:Geofencing.*
- ***Notifications***: Inform student when his/her bus is about a kilometre away from their respective stop *Technology Used:Android Notifications.*

- ***'Back Up'***: A feature that allows the driver of a bus to alert the transportation officials at college and other buses on active duty in case of emergencies or technical problems. *Technology Used:Publisher-Subscriber Scheme.*
- ***'Optimizer'***: When creating new routes, the transport management personnel can enter the bus stops randomly and the system will optimize them for creating the best-route possible. *Technology Used:Travelling Salesperson Algorithm.*
- ***'Analyzer'***: Collect data about routes, students, drivers and their reviews. Send the data to the college transportation management system for them to run data and feedback analytics in order to improve their application and service.
- ***Guest travel***: A feature that allows people not associated to the institute, such as external examiners visiting the college for examination duty, to avail the college's transportation.
- ***Parental Access***: Parents can access the application and track the whereabouts of their wards in cases of emergency or for being informed about their children's location.

IMPLEMENTATION

A. Real-time Positioning

Real-time tracking for fetching the geolocation is a core feature of our proposed web and mobile application. Real-time tracking aims at detecting and streaming location data to a live-updating map to smoothly show the movement of buses as they travel in the real world. In our system, the driver-side application (Android) is the only application that can transmit and storing geolocation to the database. The student-side application (Android) and the transport management personnel console (Web) can only fetch the stored geolocation and display it on the map.

Before implementing this feature, we need to add permissions in the application's AndroidManifest.xml file to for accessing the GPS location services on the device, i.e. *android.permission.ACCESS_FINE_LOCATION*.

We have implemented real-time positioning using Google Play Services location APIs. Location updates are retrieved by requesting the same from a method of the class FusedLocationProviderClient - requestLocationUpdates(). We then call the method *getLastLocation()* to get the device's last known position. The interval at which new location data must be fetched is set using the method setInterval() of the class LocationRequest. Since real-time tracking must be of high accuracy, we must set this using the method setPriority(PRIORITY_HIGH_ACCURACY) of class LocationRequest.

FusedLocationProviderClient does not provide only the coordinates rather, it gives a collection of values which includes speed, altitude, bearing, timestamp, accuracy, elapsedRealTimeNano, latitude, longitude, and more. These data are stored in the database. At the time of displaying the location of a bus on the map, we need to extract only the latitude and longitude of each bus from the database.

Location data in the Firebase database is stored as JavaScript Object Notation (JSON) and synchronised in real-time to every connected client [5]. The data in Firebase is structured as shown below:

rootNode:
locationUpdates:
    busRoute1:
        accuracy:
        bearing:
        speed:
        provider:
        latitude:
        longitude:
        time:
        elapsedRealTimeNano:
        . . ...
    busRoute2:
        . . .. .
    .
    .
    busRouteN:
        . . . . .

To display the location of each bus as an object of Marker class on the map of the Android application, we use HashMap<String, Object> where, String is the key of the root's child node viz. busRouteN (in our case) and Object will contain all the data under a child node. Each key-value pair within the Object can be obtained using — *objectOfHashMap.get("key")*

B. Wait for me

A geofence is a virtual perimeter set on a real geographic area. Combining a user position with a geofence perimeter enables us to know if the user is inside or outside the geofence as well as if he is exiting or entering the area.

A geofence can be marked by specifying the latitude and longitude of the location of interest. The perimeter around

the marked geofence can be set by specifying the radius. The latitude, longitude, and radius together define a geofence around a location of interest. LocationServices send entrance and exit events (also known as transitions) for each active geofence to trigger a certain event.

The active period of geofences can be limited by specifying an expiration duration in milliseconds. Any geofence that has an expiration duration is automatically removed by LocationServices on expiring. Multiple geofences can be active at any given point of time, with a limit of 100 per device user [7].

To use geofencing, we must request the permission *android.permission.ACCESS_FINE_LOCATION* in the AndroidManifest.xml file.

We create an instance of Google API Client and use GoogleApiClient.Builder to add LocationServices.

```
private GoogleApiClientgApiClient;
…….
gApiClient = new GoogleApiClient.Builder(this)
                    .addApi(LocationServies.API),
                    .addConnectionCallbacks(this),
                    .addOnConnectionFailedListener(this)
                    .build();
```

Next, we implement the callback interfaces that adds to Google API Client i.e.

GoogleApiClient.ConnectionCallbacks and *GoogleApiClient.OnConnectionFailedListener*

We start listening for location updates to check if the device/user has entered the geofence or not.

We use a HashMap<String, LatLng> data structure to store the geofences; String contains the ID for the geofence and LatLng data structure packs the latitude and longitude of the geofence. For building geofences and adding them to the Location API's builder class, we use —

```
private Geofence buildGeofence() {
  return new Geofence.Builder()
     .setRequestId(GEOFENCE_ID)
     .setExpirationDuration(Geofence.NEVER_EXPIRE)
     .setCircularRegion(latitude, longitude,
GEOFENCE_RADIUS)
     .setNotificationResponsiveness(1000)
.setTransitionTypes(Geofence.GEOFENCE_TRANSITION_
ENTER | Geofence.GEOFENCE_TRANSITION_EXIT)
     .build();
}
```

We must start monitoring the geofence and set how geofence events must be triggered by using the class GeofencingRequest along with its nested class GeofencingRequestBuilder.

```
private GeofencingRequestgetGeofencingRequest() {
GeofencingRequest.BuildergeoBuilder=
new GeofencingRequest.Builder();
    geoBuilder.setInitialTrigger(
        GeofencingRequest.INITIAL_TRIGGER_ENTER);
    geoBuilder.addGeofences(getGeofence());
    return geoBuilder.build();
}
```

On triggering of a geofence transition, an Intent is sent by the LocationServices API which can further trigger various events. In our proposed system, we use an IntentService to handle the Intent generated by the geofence transition.

```
private PendingIntentpendingIntent;
…….
private PendingIntentgetGeoPendingIntent() {
        if (pendingIntent != null) {
                return pendingIntent;
        }
        Intent intent = new Intent(
                this, GFenceBusStopService.class);
        return PendingIntent.getService(this, 0, intent,
}
```

In the *GFenceBusStopService* class, we trigger a notification to the student that he/she is within the radius of their bus stop. Also, we increment a variable count on the database that enables the driver to know how many students are waiting at a stop on the current route the driver is active on.

Before we can start monitoring the geofences, we have to add them using the method addGeofences() of the class GeofencingClient. The parameters to this method are a GeofencingRequest object and the PendingIntent object.

To help save battery power and CPU cycles on the end-users' devices, we can stop geofence monitoring when it is no longer needed. Geofence monitoring can be stopped in the main activity used to add and remove geofences; removing a geofence stops it instantly.

C. BackUp

BackUp is a feature to send alert messages in cases of emergencies, such as accidents, vehicle breakdowns and others. This is the kind of service that will make other drivers who are on active duty and the transport management personnel at the institution receive alerts on their phones and web application respectively. This feature can be used only when the bus experiences problems while on duty. Even when the display of the phone of other bus drivers are turned off or the Android application is not in the foreground, a service will constantly be running that will listen for alerts and emergency messages. On arrival of an alert, it will trigger a notification sound and the device will vibrate. On clicking the notification message, other drivers can see the route ID of the bus that has problems and can also view the location of that bus.

Even the transport management personnel can see the alerts raised by any bus driver on his/her web console with the faulty bus's location of where it broke down. To raise an emergency request, a bus driver must just press a button and choose the type of problem he/she is facing.

Using the Publisher-Subscriber scheme, all the buses (subscribers) are constantly listening to some data trigger event in the database through means of a silently running background service on their Android devices. All the bus drivers' phones also have publisher rights by which they can update some data in the database that will trigger a data change event, thereby notifying the subscribers.

D. Optimizer

Optimizer focuses on providing the best route by rearranging bus stops in a given route. This optimization is a classic implementation of the Travelling Salesperson problem. When creating a new route, the transport management personnel can enter a routeID and randomly add the bus stops to the database. Optimizer facilitates the creation of best-route, consistent with the distance and duration (in terms of traffic model) and displays it to the transport personnel.

In our system, we have selected a pessimistic traffic model for calculating the total duration of a route. The pessimistic model calculates duration by considering a time period longer than the actual travel time. The total duration and total distance of a route are calculated using the Google Directions Service API.

To find the shortest (best) path from origin to bus stop A (leg 1), from bus stop A to bus stop B (leg 2), and so on, till bus stop N to destination (leg N), the Travelling Salesperson algorithm takes the latitude and longitude of each bus stop and the distance between two stops and returns the most efficient route from the origin to the destination through all the mentioned waypoints (bus stops).

E. Analyzer

Analyzer feature focuses on extensive feedback collection and data analysis for timely updates to the application(s) and betterment of the software suite as well as the transport service itself. Data can be collected from bus drivers and students in the form of profile data as well as feedback given by them about the transport system and the application itself. The data stored in the database is plotted into different forms of graphs using the ChartJS graph plotting library for JavaScript. Some of the data that we can collect and use to run analysis are —

- Number of students using the transportation system (semester, branch, route and locality-wise)
- Increase/decrease in the number of users
- Sentiment analysis on the feedbacks obtained from students using Google Cloud NLP API
- Daily reports of effective transport system utilization

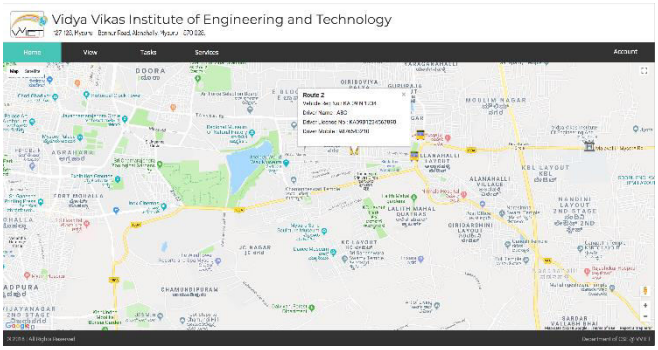

Fig. 4. Real-time tracking of buses with bus and driver details

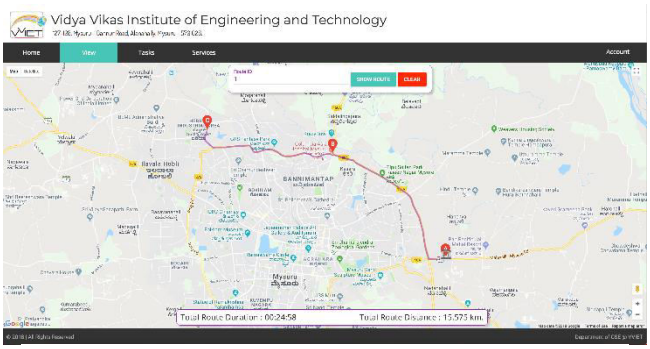

Fig. 5. Optimizer

These are screenshots of the proposed system —

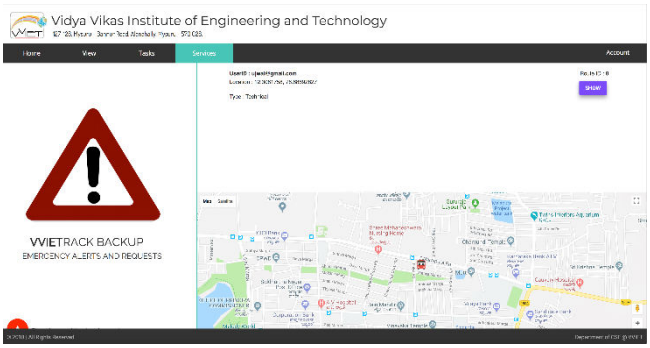

Fig. 6. BackUp - Emergency Messages and Alerts

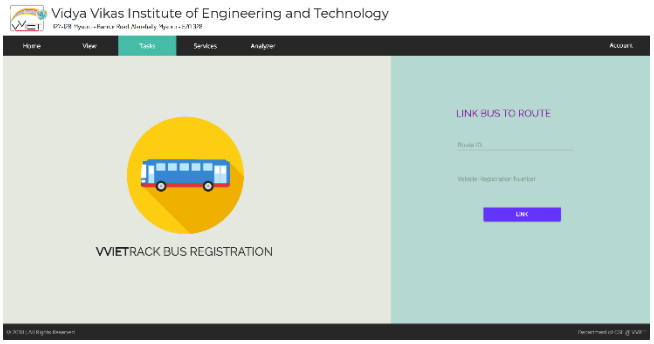

Fig. 9. Linking new bus with a route

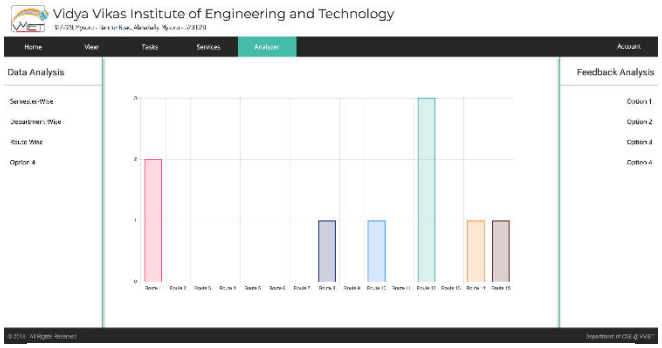

Fig. 7. Analyzer (Route-wise)

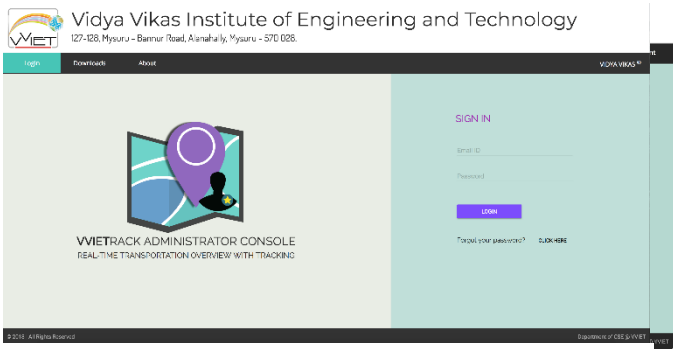

Fig. 3. Login Interface for system administrators

Fig. 8. Route Creation

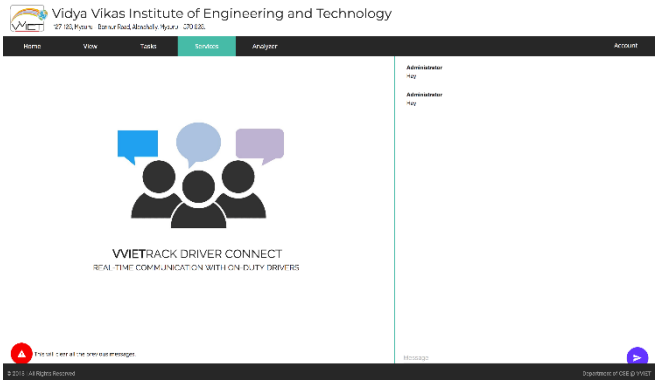

Fig. 11. DriverConnect - Real-time Intrasystem Chat

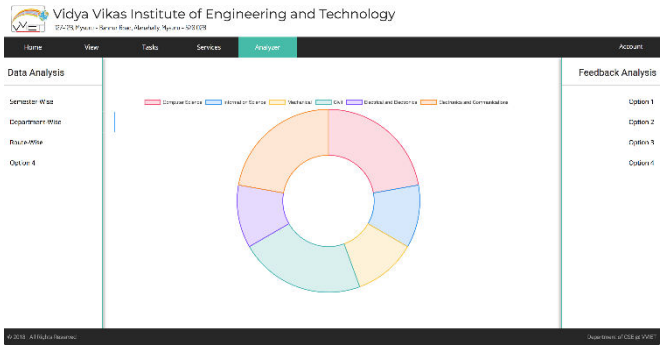

Fig. 12. Analyzer (Department-wise)

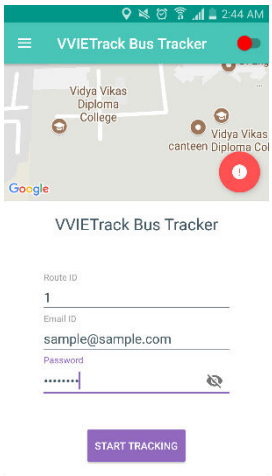

Fig. 13. Driver Application (Android)

Fig. 14. Student Application (Android)

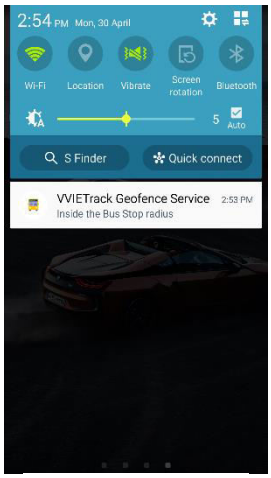

Fig. 15. Geofence transition on entering bus stop radius

CONCLUSION

One of the main reasons for why many of the students in an educational institution do not use the option of college transport is because of the additional travel and wait times. But, if users had an easy way to track the proximity of the bus to their bus stop in real time they can utilise time more efficiently by not waiting in their stops for a long duration of time.

Also, an increase in the usage of smartphones, advanced technologies and an exponential growth in the Android market can ensure that our proposed system can be used by a large percent of the targeted end-users. It will help them to use the college transport whilst providing solutions to their problems mentioned above. Meanwhile, the current users can be benefited by increased satisfaction of service. New users can be motivated to use this with a collective goal of reducing air and noise pollution levels around the city.

FUTURE ENHANCEMENTS

- The proposed system can be further developed for iOS devices to support inter-operability among cross-platform devices.
- Payment gateways can be incorporated into the system to enable the payment of bus transport fee online.
- *'Count me in'*: An automatic attendance system that tracks inflow/outflow of students and also, how many students climb onboard the bus at a particular stop.

REFERENCES

[1] ThunyasitPholprasit, SupornPongnumkul, ChalermpolSaiprasert,SarinthonMangkorn-ngam, LalidaJaritsup, "LiveBusTrack: Highfrequency location update information system for shuttle/bus riders",Communications and Information Technologies (ISCIT), 2013 13th International Symposium on, 2013.

[2] Benjamin Y.O. Low, SamsulHaimiDahlan, Mohd Helmy Abd Wahab,"Real-time bus location and arrival information system", Wireless Sensors (ICWiSE), 2016 IEEE Conference on, 2016.

[3] B. Janarthanan, T. Santhanakrishnan, "Real time metropolitan bus positioning system design using GPS and GSM", Green Computing Communication and Electrical Engineering (ICGCCEE), 2014 International Conference on, 2014

[4] Manini Kumbhar, Meghana Survase, Pratibha Mastud, Avdhut Salunke, "Real Time Web Based Bus Tracking System", International Research Journal of Engineering and Technology (IRJET), vol. 03, no. 02, Feb., pp. 632-635, 2016.

[5] https://firebase.google.com/docs/

[6] https://developers.google.com/maps/

[7] https://developer.android.com/training/location/geofencing