TABLE OF CONTENTS

ABSTRACT

This report describes my internship experience Web Sky Infotech, Indore. The report presents the tasks completed during summer internship at the company, which are listed as follows –

1. Underwent 15 day training in Android development by the company professionals.
2. Implemented the skeletal design of an android application for a local museum.
3. Prepared the proposal for an Android application project for a marketing firm.
4. The project was divided into modules and was assigned to the team.

The report is organized into four chapters. Chapter one summarizes the initial training experience for Android and Java, including the tools and software used for development. Chapter two gives an insight into the application development for the museum, the briefing and project proposal for the marketing application, roughly highlighting the project implementation. Chapter three describes the details of the project implementation and its modules. Chapter four summarizes the internship period and the current project scenario.

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Android

Android is a Linux-based open source operating system designed primarily for touchscreen mobile devices such as smartphones and tablet computers. It was initially developed by Android, Inc., which Google later bought in 2005. Android consists of a kernel based on Linux kernel version 2.6 and, from Android 4.0 Ice Cream Sandwich onwards, version 3.x, with middleware, libraries and APIs written in C, and application software running on an application framework which includes Java-compatible libraries based on Apache Harmony. Android uses the Dalvik virtual machine with just-in-time compilation to run Dalvik 'dex-code' (Dalvik Executable), which is usually translated from Java bytecode.

Android applications are developed in the Java language using the Android software development kit (SDK). The SDK includes a comprehensive set of development tools, including a debugger, software libraries, a handset emulator based on QEMU, documentation, sample code, and tutorials. The officially supported integrated development environment (IDE) is Eclipse using the Android Development Tools (ADT) plugin.

Fig 1.1 Logo of Android

The Android OS is roughly divided into five sections in four main layers:

- **Linux kernel** — Contains all the low-level device drivers for the various hardware components of an Android device.
- **Libraries** — Provides the main features of an Android OS.
- **Android Runtime** — At the same layer as the libraries, the Android runtime provides a set of core libraries that enable developers to write Android apps using the Java programming language. The Android runtime also includes the Dalvik virtual machine, which enables every Android application to run in its own process, with its own instance of the Dalvik virtual machine.
- **Application framework** — Exposes the various capabilities of the Android OS to application developers so that they can make use of them in their applications.
- **Applications** — Applications that are shipped with an Android device, as well as applications that are downloaded and installed from the Google Play Store.

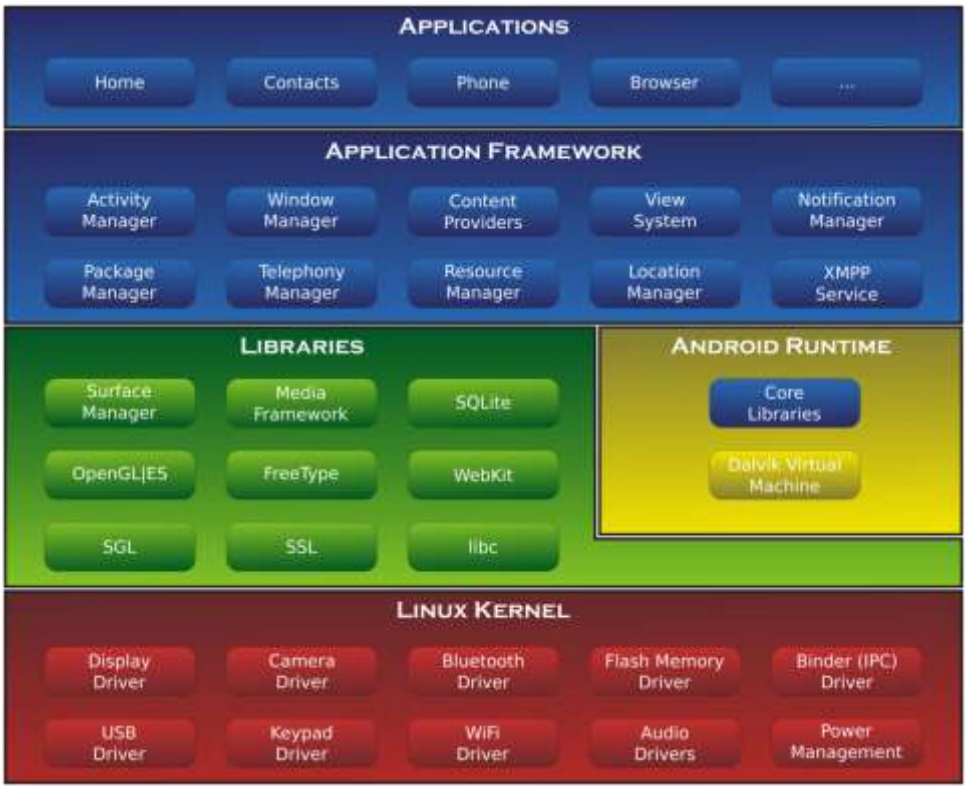

Fig 1.2 Android Architecture

1.2 Motivation

With Android based smart phone revolution, the Android Application Development has become the largest and fastest growing profit generating market. Many web development companies and mobile application development companies have moved their focus on Android App Development. The possibilities of customization in Android Application Development is the primary reason for its usage, as any level of complex application can be successfully developed and deployed. With Android being open source, building custom android Rom's for devices is permitted.

1.3 Objectives

1.3.1 To set up a development environment for java and android and to learn the basics of professional android application development.

1.3.2 To develop an interactive android application for a museum to guide the tourists and inform them about the articles that are kept in the museum.

1.3.3 To develop an android application for a marketing firm to allow the managers to track the positions of the employees under them using GPS and assign and update client appointments.

CHAPTER 2

PROJECTS

2.1 Museum Application

2.1.1 Problem Description :

To design and implement an android application that meets the requirements of a local museum described as follows:

i. To organize the articles available in the museum into categories.
ii. To describe each item.
iii. Use audio, video and images for better user interaction and easy understanding.

2.1.2 Details of the Application:

i. Displaying various categories in a grid on the homepage of the application.
ii. Navigating to the proper category page and displaying all the available items in that category.
iii. Displaying images, playing audio clips, video clips or textual information of a particular item, depending upon the users' choice.
iv. Categorizing the items based on their location in the museum.

2.2 Marketing Application

2.2.1 Problem Description :

To design and implement an Android application that meets the requirement of a marketing firm as outlined:

i. Identifying, assigning and updating clients to all the employees.
ii. To outline an optimal path for covering the assigned clients.
iii. Ability to prioritize and update client status.
iv. To track the location of the employees (privileged only to a manager).

v. To be able to communicate between employees and manager.
vi. To synchronize all relevant information between the android application and a windows application.

2.2.2 Details of the Application:

i. Live Tracking of Employees: A map view that displays the latitude and longitude of any and all employees at any given instance along with street view for conveying the information regarding the position of the employee.
ii. Ability to determine the status of a deal: Various markers will be introduced in the map view to signify the status of the deal with the client. For example green markers signifies a confirmed deal, orange signifies pending or unattended, red for deal cancelled , blue to signify an in Progress deal.
iii. Easy Updating and prioritizing of Client deals: The status of the Client's deal will be updated instantly on all the devices and prioritizing distinguishes the more promising and premium clients from the other clients.
iv. Suggesting an optimal path for the assigned clients: Once the list is assigned the map will show a path to cover all the clients with minimum cost overhead
v. Establishing communication between employees and managers: A quick messaging services that facilitates the communication between employees and managers
vi. Authentication: A login Interface to provide a secure gateway and protect the privileged function of the application against the unauthorized usage.
vii. A remote desktop application for managers, which updates the central database (with the aid of Microsoft Outlook), and shows the position of the employees on Bing Maps using GPS.

2.2.3 Problems & Constraints:

1. Android Devices supporting Google Maps API Version 2 for the employees and managers of the concerned organization.
2. Reliable Internet services and GPS facility.
3. Accessing Google Maps and Bing Maps offline.

CHAPTER 3

DESIGN AND IMPLEMENTATION

3.1 Architecture

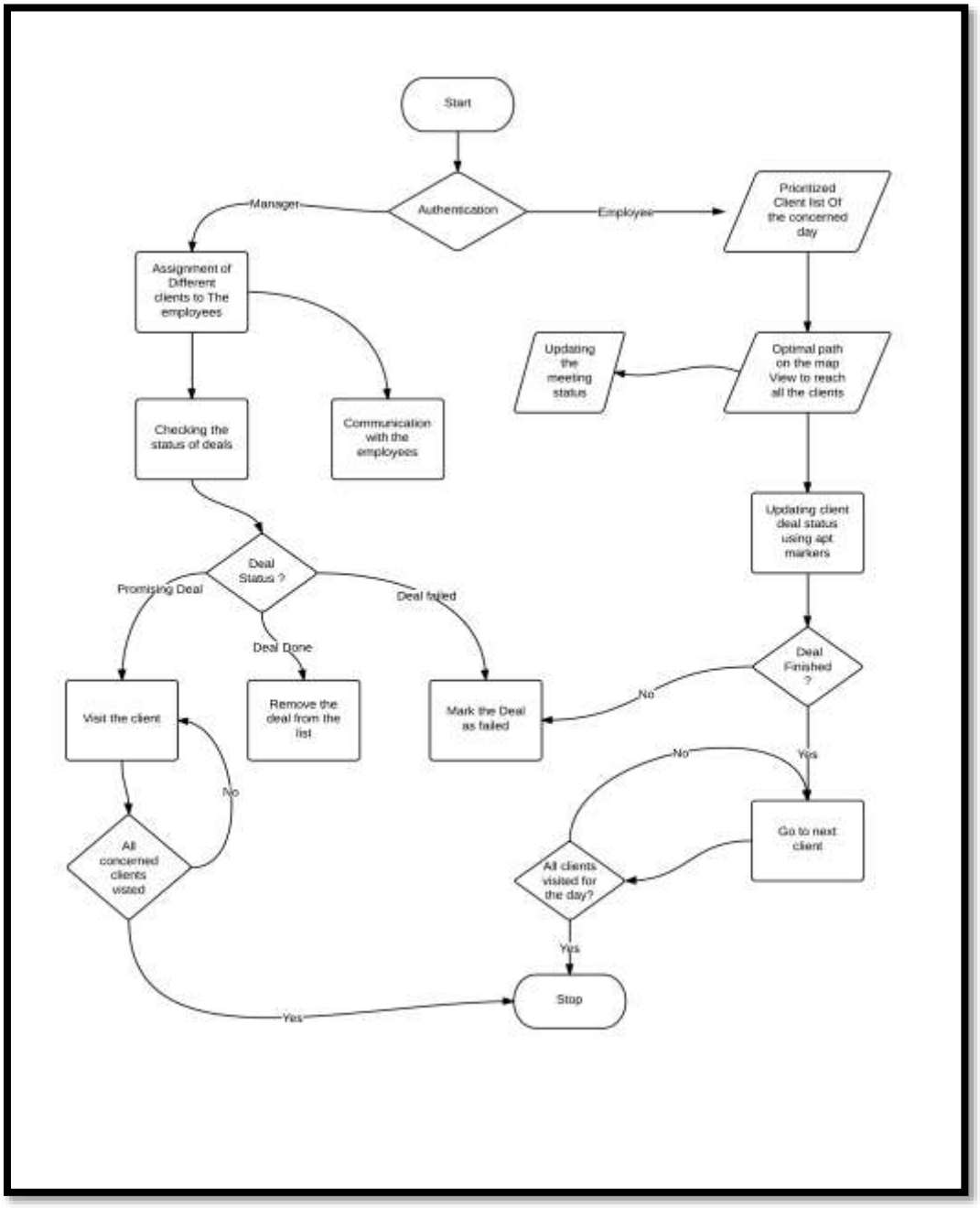

Fig 3.1 Flowchart for marketing application

3.2 Description

The application starts with an authentication page to distinguish between managers and employees. The desktop application will be used by the mangers to monitor and track the position of the employees, as well as for assigning them client deals dynamically, using Microsoft Outlook. The desktop application also enables prioritizing the deals based on their clientele.

The android application for the employees will be used for showing them the allotted client deals and possibly also for suggesting them an optimal path for the same. An employee can update a deal's status on this application, which will be instantly updated on the database, and will be available to the manager on the desktop application.

The android application runs a background service which will receive the latitude and longitude of its current position using GPS, and continuously send these coordinates to the remote desktop application, identifying itself with the help of a unique ID. The data will be sent from Android to Windows using UDP packets (or using sms, if necessary).

3.3 Design Details

The project was split into two modules, one for a Windows desktop, and another for an Android smartphone.

3.3.1 Windows Desktop Application

1. Communication: The windows application includes a chat module that will enable the managers to communicate with the employees. It implements UDP to transfer packets. It can also be used to notify the employees of a new deal or update information about an existing deal.

2. Bing Maps: An android service running on every employee's handset continuously sends its corresponding latitude and longitude, which are geocoded on a Bing Map in the windows application. The location of each connected employee is rendered as a marker of different color, and is updated as soon as a new location for that particular handset is received. A list of employees along with their corresponding color of the marker is displayed.

3. Integration with Microsoft Outlook: The managers can update or add any client deal on Outlook. The desktop application detects a change or a new addition to Outlook, and updates the database accordingly. It also sends these updates to the connected android handsets, which then notifies the employees of the changes made in the schedule.

Fig 3.2 Authentication Page for the Desktop Application

3.3.2 Android Application

1. Communication: The android application includes a chat module to communicate with the desktop application using UDP packets. It runs a service that receives packets from the desktop application. It is used for receiving updates on current

deals and information about new deals, which are forwarded by the managers. The application can also send sms, if there is no network available for the UDP packets.

2. GPS Tracking: The android application runs a background service, invisible to the user, which continuously tracks the current location of the handset using GPS and sends the latitudes and longitudes to the desktop application, at regular intervals. These coordinates are, in turn, used by the desktop application to plot the handsets location on Bing Maps.

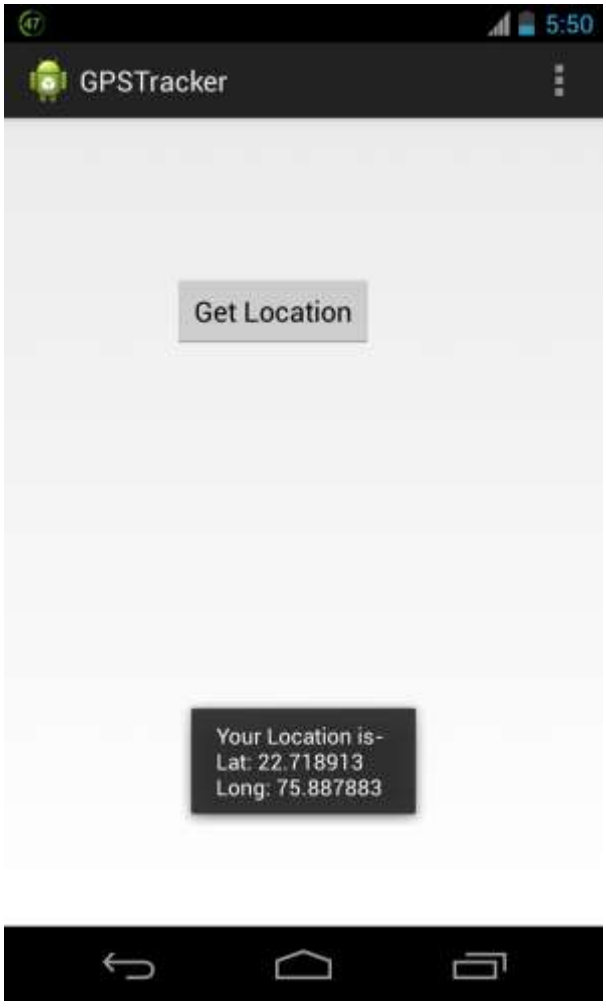

Fig 3.3 Getting Gps location from android application

3. Updating Deal Status: Location of the client deals are displayed as colored markers on map for the employees to refer. The status of a particular deal will affect the

color of its marker on the map. The status of a deal can be updated by the employees, which will change the color of its marker.

4. Google Maps: The google maps are used to indicate the location of employees. Each employee is shown with different color markers.

The map application comprises a service which at regular interval fetches the GPS coordinates of employee. If the current location and the previous location show a significant difference, the new location is indicated on the map. A line joining the current location and previous location is used to indicate the change in position of the employee.

The map is rendered on the android application using Google Console Api which provides a unique key. The key authorizes the android application to render maps. The rendering of maps on an android device require several privileges like use of internet connection, use of the Gps etc. The application uses Strict mode policy to allow those privileges.

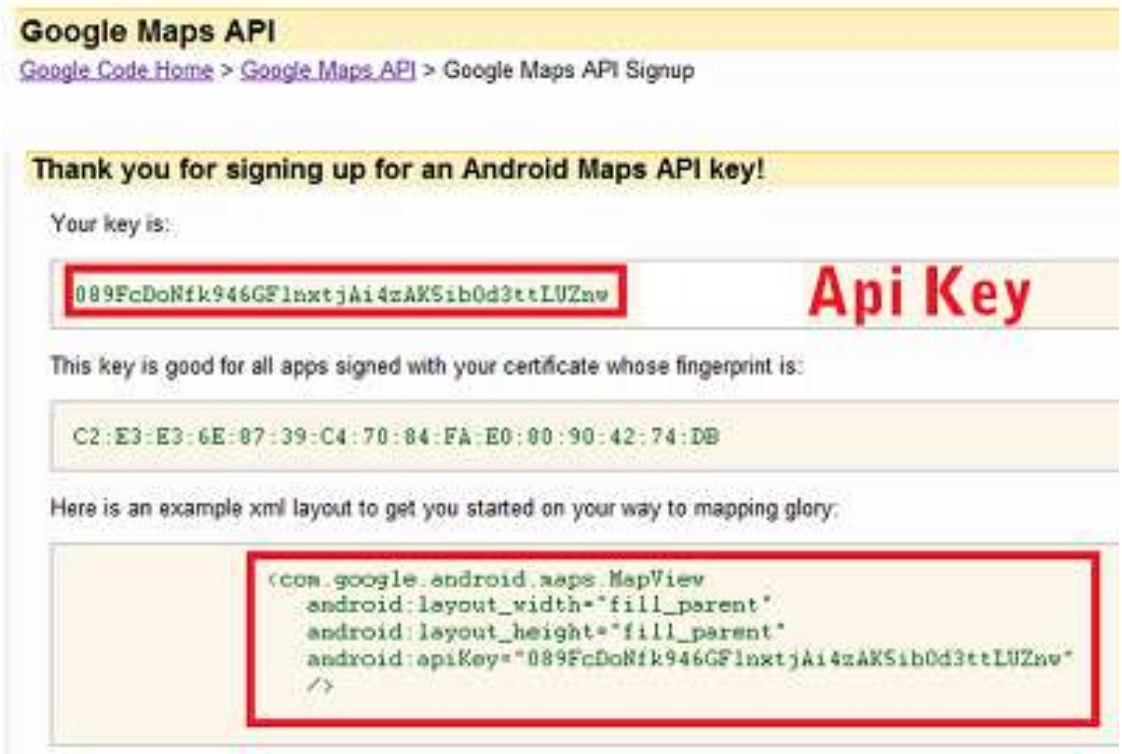

Fig 3.4 Obtaining Google Api key from Api Console

The employee can check their locations in four different views namely normal view, terrain view, satellite view and hybrid view. Each view renders a different map with the marker of employee being the same.

The map gets the present location of the employee and hovers to that location on start of the application. The starting location is shown with a red marker. The marker contains detailed information about the location. When clicked, the marker displays the location coordinates and the address.

Fig 3.5 Normal View of Map with red dot representing current location of employee

The Gps location is verified at a regular interval of 2 seconds. When a significant change in the position is noticed, a dot is plotted on the map. The marker is then updated to point to the new coordinate. Polyline control is used to plot lines indicating location updates.

Fig 3.6 Ploryline representing the path traversed by the employee

CHAPTER 4

RESULTS AND DISCUSSIONS

4.1 Modules Completed

4.1.1 GPS Tracking: Tracking the location of Android device using a background service.

4.1.2 Plotting on Bing Maps: Plotting the location of connected Android devices on Bing Maps (WPF application) running on a desktop.

4.1.3 Plotting on Google Maps: Plotting the current location of an Android device on Google Maps on the same device.

4.1.4 Transferring location coordinates from Android device to desktop: Sending the location coordinates from Android to Windows using UDP packets.

4.1.5 Real time updates from Outlook: Updating appointments on a Windows form application whenever a new entry is added to Microsoft Outlook.

4.1.6 Calendar on Android: A calendar on Android gets updated every time an entry is made on the desktop application.

4.2 Modules Under Development

4.2.1 SMS: Sending sms from Android to Windows or vice versa, whenever network is not available for updating the deals.

4.2.2 Database: Saving employee details, manager details, client deals and employee locations on a central database for future reference.

4.2.3 Distributing client deals to employees: Assigning client deals to the employees added by the managers on the desktop application.

4.2.4 Updating employee list: Changing the list of employees whenever a new employee joins or leaves.

4.2.5 Authenticating Employees and Managers: A login interface with username and password to authenticate which employee is accessing the application and showing details relevant only to the particular employee.

4.3 Performance Analysis

4.3.1 Location Coordinates:

The GPS on the Android device returns the following coordinates –

Latitude: 22.7178722 deg.

Longitude: 75.8827764 deg.

The Coordinates on Windows application returns the following coordinates –

Latitude: 22.7178722 deg.

Longitude: 75.8827764 deg.

On sending these coordinates to the Windows application, Bing Maps displayed the following output –

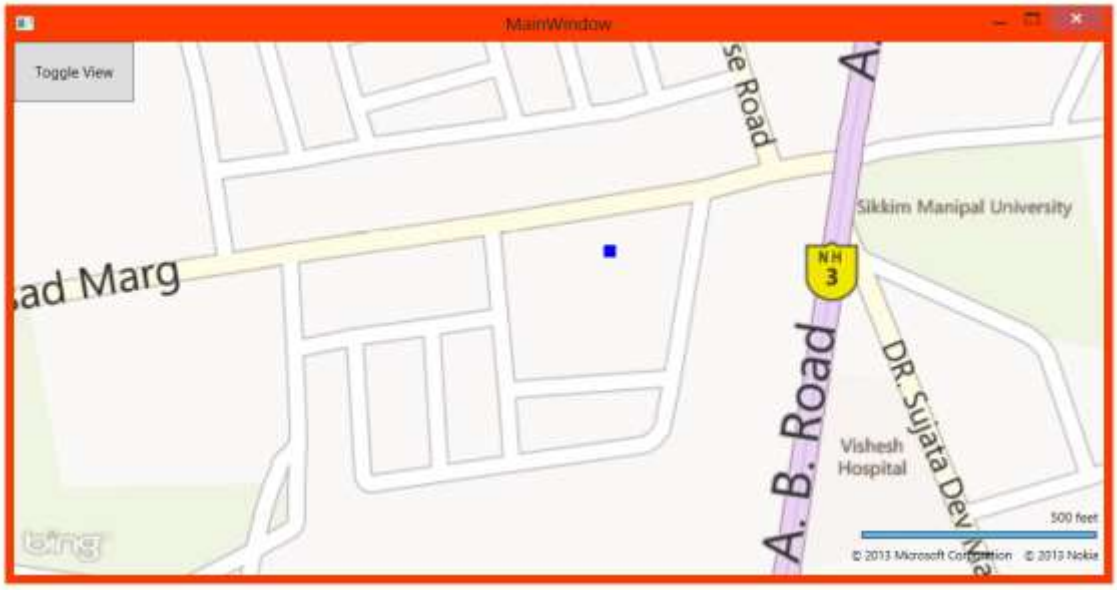

Fig 4.1 Output on Bing Maps

Google Maps on the same android device geocodes these coordinates to display the following output –

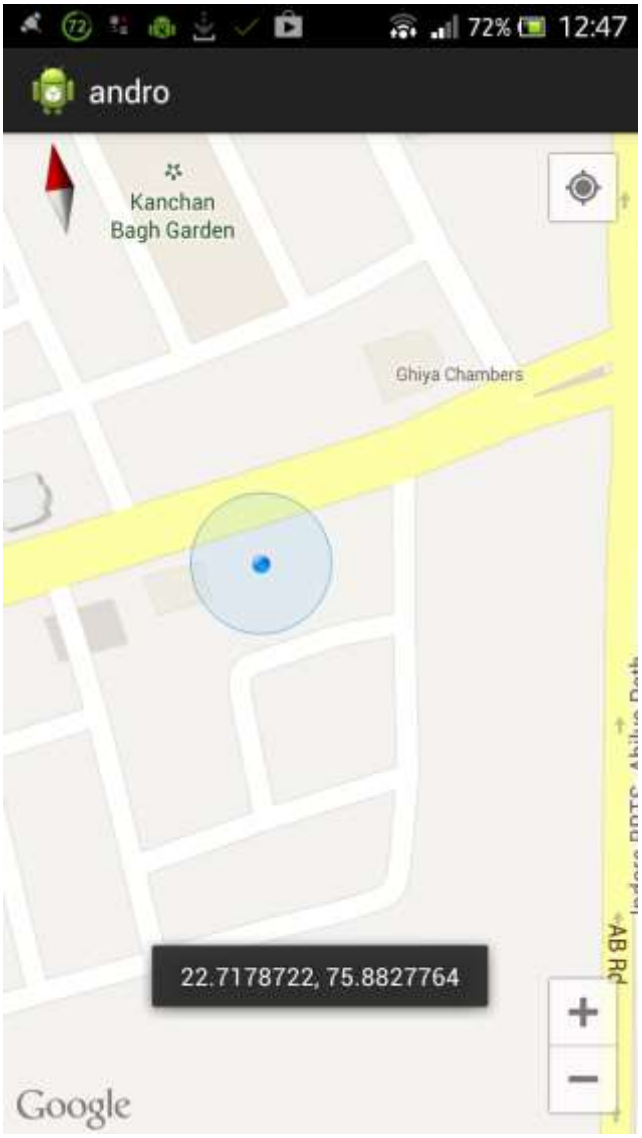

Fig 4.2 Output on Google Maps

CHAPTER 5

CONCLUSION AND FUTURE ENHANCEMENTS

5.1 Conclusion

This project could serve a very wide range of corporations, specifically marketing organizations, and aid them to elevate their productivity. The employees would also be assisted by the navigation technology in locating their clients. The managers can dynamically assign deals and discreetly track the employees.

5.2 Future Enhancements

5.2.1 Commercial Feasibility: As of now, Google Maps serve the purpose satisfactorily, but their commercial feasibility is yet to be determined. Google currently allows only 25,000 map accesses per day, which would be insufficient for a large corporation.

5.2.2 Battery Life of Android Devices: With the GPS continuously monitoring the current geographical location of the device and with the continuous transmission of the location coordinates to a remote desktop, the battery life of the device gets diminished drastically. Increasing the transmission interval to save the battery life reduces the real time tracking performance. A probable solution can be experimentally determining the time to transmit the location coordinates without affecting the tracking performance.

5.2.3 Usage of Microsoft Outlook: The module used to read the appointment cannot distinguish between personal and official appointment.

5.2.4 Visibility of Service: Making the service that tracks the employee invisible so as to prevent it accidental or deliberate termination by the employee.

REFERENCES

Books

1. Wei-Meng Lee, ‘Beginning Android Application Development’, 2011
2. Reto Meier, ‘Professional Android Application Development’, 2009
3. Satya Komatineni, Dave MacLean, ‘Pro Android 4’, 2012
4. J. F. DiMarzio, ‘Practical Android 4 Games Development’, 2012
5. Raghav Sood, ‘Pro Android Augmented Reality’, 2012

APPENDIX A

UDP (User Datagram Protocol)

The User Datagram Protocol (UDP) is one of the core members of the Internet protocol suite (the set of network protocols used for the Internet). With UDP, computer applications can send messages, in this case referred to as datagrams, to other hosts on an Internet Protocol (IP) network without prior communications to set up special transmission channels or data paths.

UDP uses a simple transmission model with a minimum of protocol mechanism. It has no handshaking dialogues, and thus exposes any unreliability of the underlying network protocol to the user's program. As this is normally IP over unreliable media, there is no guarantee of delivery, ordering or duplicate protection. UDP provides checksums for data integrity, and port numbers for addressing different functions at the source and destination of the datagram.

UDP is suitable for purposes where error checking and correction is either not necessary or performed in the application, avoiding the overhead of such processing at the network interface level. Time-sensitive applications often use UDP because dropping packets is preferable to waiting for delayed packets, which may not be an option in a real-time system.

APPENDIX B

GPS (Global Positioning System)

The Global Positioning System (GPS) is a space-based satellite navigation system that provides location and time information in all weather conditions, anywhere on or near the Earth where there is an unobstructed line of sight to four or more GPS satellites. The system provides critical capabilities to military, civil and commercial users around the world. It is maintained by the United States government and is freely accessible to anyone with a GPS receiver.

A GPS receiver calculates its position by precisely timing the signals sent by GPS satellites high above the Earth. Each satellite continually transmits messages that include

1. The time the message was transmitted
2. Satellite position at time of message transmission

The receiver uses the messages it receives to determine the transit time of each message and computes the distance to each satellite using the speed of light. Each of these distances and satellites' locations define a sphere. The receiver is on the surface of each of these spheres when the distances and the satellites' locations are correct. These distances and satellites' locations are used to compute the location of the receiver using the navigation equations. This location is then displayed, perhaps with a moving map display or latitude and longitude; elevation or altitude information may be included. Many GPS units show derived information such as direction and speed, calculated from position changes.

APPENDIX C

Code Snippets

1. App Manifest code:

   The manifest presents essential information about the application to the Android system, information the system must have before it can run any of the application's code

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="com.example.andro"
android:versionCode="1"
android:versionName="1.0" >

<uses-sdk
android:minSdkVersion="10"
android:targetSdkVersion="17" />
<uses-feature
android:glEsVersion="0x00020000"
android:required="true" />


<uses-permission android:name="android.permission.INTERNET" />
<uses-permission
android:name="android.permission.ACCESS_NETWORK_STATE" />
 <uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="com.google.android.providers.gsf.permission
                                    .READ_GSERVICES" />

<uses-permission
android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission
android:name="android.permission.ACCESS_FINE_LOCATION" />

<permission
android:name="com.example.andro.permission.MAPS_RECEIVE"
android:protectionLevel="signature" />

<uses-permission
android:name="com.example.andro.permission.MAPS_RECEIVE" />
```

```
<application
android:allowBackup="true"
android:icon="@drawable/ic_launcher"
android:label="@string/app_name"
android:theme="@style/AppTheme" >
<uses-library android:name="com.google.android.maps" />

<activity
        android:name="com.example.andro.MainActivity"
        android:label="@string/app_name" >
        <intent-filter>
        <action android:name="android.intent.action.MAIN" />

        <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
</activity>
<service android:name=".GpsClass"></service>
<meta-data
android:name="com.google.android.maps.v2.API_KEY"
android:value="AIzaSyDyU4fRVYN-FBcewE47Bw4g9ix62Oxgl64" />
</application>

</manifest>
```

2. GPS Class:

   The subroutine getLocation is used to get the Gps location of employees. If the Gps is turned off, this subroutine prompts the user of turn on the Gps. Once the Gps is turns on, the subroutine gets the location and returns it in the form of a Location.

```
public Location getLocation()
{
lm = (LocationManager)c1.getSystemService(LOCATION_SERVICE);
        isGps = lm.isProviderEnabled(lm.GPS_PROVIDER);
        isNetwork = lm.isProviderEnabled(lm.NETWORK_PROVIDER);
        if(isGps && isNetwork)
        {
                this.came = true;
                lm.requestLocationUpdates(lm.NETWORK_PROVIDER,
                meanTime, distance, this);


        if(lm !=null)
        {
                l1 = lm.getLastKnownLocation(lm.NETWORK_PROVIDER);
        }
```

```
if(l1 != null)
{
        d1= l1.getLatitude();
        d2= l1.getLongitude();
}

}

if(isGps)
{
        if(l1!= null)
        {

        }
        else
        {
                lm.requestLocationUpdates(lm.NETWORK_PROVIDER,
                meanTime, distance, this);
                if(lm !=null)
                {
                l1 =
                lm.getLastKnownLocation(lm.NETWORK_PROVIDER);
                }
                if(l1 != null)
                {
                        d1= l1.getLatitude();
                        d2= l1.getLongitude();
                }
        }

}


return l1;
}
```

3. Main Activity:

   This is the activity that serves as the main entry point of the application. The map is rendered on this activity.

```
map = ((MapFragment)
getFragmentManager().findFragmentById(R.id.map)).getMap();

//map.setMapType(GoogleMap.MAP_TYPE_HYBRID);
// map.setMapType(GoogleMap.MAP_TYPE_NONE);
 map.setMapType(GoogleMap.MAP_TYPE_NORMAL);
// map.setMapType(GoogleMap.MAP_TYPE_SATELLITE);

//map.setMapType(GoogleMap.MAP_TYPE_TERRAIN);
 map.setMyLocationEnabled(true);
//map.setTrafficEnabled(true);

GpsClass g1 = new GpsClass(this);

Toast.makeText(MainActivity.this, "Turn on the gps",
                        Toast.LENGTH_LONG).show();
Intent intent = new
Intent(android.provider.Settings.ACTION_LOCATION_SOURCE_SETTINGS);
startActivity(intent);
LocationManager lm;

lm = (LocationManager)this.getSystemService(LOCATION_SERVICE);

isGps = lm.isProviderEnabled(lm.GPS_PROVIDER);
isNetwork = lm.isProviderEnabled(lm.NETWORK_PROVIDER);


if(l1!=null)
{
d1 = l1.getLatitude();
d2 = l1.getLongitude();

mb = new LatLng(d1, d2);
CameraPosition camera = new CameraPosition.Builder().target(mb).build();

map.animateCamera(CameraUpdateFactory.newCameraPosition(camera));
                        Marker marker = map.addMarker(
                        new MarkerOptions().position(mb)
                                        );
Marker marker2 = map.addMarker(new MarkerOptions().position(ab));
}
map.addPolyline(

                new PolylineOptions().add(mb,ab).color(Color.RED).width(5)

                );
```

4. Strict Mode Access:

   This class is used to enable and enforce various policies that can be checked for and reported upon.

```
StrictMode.ThreadPolicy policy = new
StrictMode.ThreadPolicy.Builder().permitAll().build();

StrictMode.setThreadPolicy(policy);
```