Syrian Private University

Faculty of Computer and Informatics Engineering

الجامعة السورية الخاصة

كلية هندسة الحاسوب والمعلوماتية

GPS TRACKING SYSTEM

(Droid Track)

A Senior Project Presented to the Faculty of Computer and Informatics Engineering of Syrian Private University in Partial Fulfillment of the Requirements for the Degree of Bachelor of Engineering

by

Mhmad Rohy Abboud

Danah Maher Tawfiq

Under the supervision of

Dr. Salah Dowaji

August 2014

مقدمة:

الاتصالات أصبحت أمراً ضرورياً للجنس البشري. وكنتيجة لتطور العملية التكنولوجية لقد واجهنا أمكانيات خارقة للاتصال عبر المسافات. الأجهزة الذكية تمثل خياراً عظيماً وجزئاً عظيماً منها كما وتجعلها أكثر سهوله.

في هذه الظروف ظهر أندرويد كخيار أمثل للمبرمجين لينشؤوا تطبيقات معقدة بسهولة.

الهدف من هذا المشروع (Droid Track) هو إنشاء نظام موزع يدعم العديد من ميزات الأجهزة الذكية مثل ال GPS ,Maps,etc .وهي تتعامل مع تطبيق أندرويد لمشاركة معلومات المستخدمين مثل المواقع في الزمن الحقيقي باستخدام أخر وأحدث التقنيات.

وبطريقة بسيطة عن طريق خزن المواقع من جهاز الأندرويد في قاعدة بيانات على المخدم من خلال نظام موزع يحتوي على web service application وعلى android application وأخذ معلومات اخر موقع وتتبع مسارات الأصدقاء.

هذا البحث يحتوي عدة فصول وسوف نعرض لكم ملخص عنها :

الفصل الأول: يتحدث عن أنظمة التتبع وتقنياتها ومحاسنها وفوائدها وبعض الأمثلة عنها.

كما ويتحدث عن تقنية GPS لمحة تاريخية عنها ,أجهزتها, أقمارها الصناعية ,والمعادلات الرياضية لحساب الموقع بالإضافة إلى إلقاء نظرة على أنظمة المواقع الأخرى التي تشيه الGPS في عملها.

ثم يتحدث أنظمة أندرويد وتاريخ ظهورها وتطورها وما هي مكونة منه من الناحية التقنية والناحية البرمجية مع شرح لبيئة تطوير أندرويد وأدوات التطوير وطريقة تثبيتها على الحاسب الشخصي.

الفصل الثاني: يتحدث عن طريقة تصميم الserver side والخوارزميات المستخدمة لتصميم هذا النظام , الذي يهدف إلى عمل خدمة لتطبيقات الأندرويد على المخدم ,ولغات البرمجة المستخدمة ومتطلبات هذا النظام.

الفصل الثالث: يتحدث عن طريقة تصميم تطبيق الأندرويد والخوارزميات والصفوف المستخدمة في ذلك ,مع المرور على لغة XML ولغة الجافا.

الفصل الرابع: يتحدث عن اختبار التطبيقات وقياس النتائج.

الفصل الخامس: يتحدث عن العمليات الي يستطيع التطبيق تنفيذها مدعومة بالعديد من الصور كما ويحتوي على نظرة مستقبلية لتطوير هذا التطبيق.

وأخيراً نود شكر كل القائمين على الجامعة وكل من ساهم في إنجاز هذا التطبيق.

CONTENTS

PAGE

DEDICATION

We'd like to thank all those who have helped with their advice and efforts …

We'd like also to thank all the university doctors, especially our supervisor, Dr. Salah Dowaji for his valuable advices.

For our parents, friends and everyone, we offer our research.

With love and respect.

Introduction:

Communicating has ever since been a necessity for human kind. As a result of technological progress we are facing an incredible variety of possibilities to communicate regardless of the distance. Smartphones provide a great choice of features that facilitate the life for the users as well they make it more comfortable.
In this context, Android presents the best option for developers to create complex applications easily.
The aim of Droid Track is to create a distributed system that embraces many features of Smart-Phones such as GPS, Maps, etc. This transforms into a mobile application for sharing user's information like location in real time using the latest technologies.

And that in a simple way by store location from android device in database on server with distributed system consist of a web service application and an android application and take this information to the last position and tracks of our friends.

This research includes several chapters and we will offer you the bottom line

Chapter One: talking about the tracking Systems and the technologies that benefit from the tracking system and some examples about it.

Then talking about the GPS technology, its history, devices, satellites, and the math equation for calculate the position.in addition; take a look on other location systems, which work like GPS system.

Then talking about the android system its history, what Androids are made of, and the technical side of android. With explain for android developer environments and android developer tools, with how to download and install them.

Chapter Two: talking about the way that the server side design and the disciples and algorithms used to design this system, which aims to make service from the server to the android application, and the programming languages and requirement to program this service.

Chapter Three: talking about the way that the android application was design and the disciples, algorithms, and classes used to design this application, with a look about Xml and java language.

Chapter Four: talking about testing our project with two testing measures and the results of testing.

Chapter Five: talking about the operation of the project in detail and contains lots of pictures and explanations of how to use this project.

Then talking about results and future work.

LIST OF FIGURES

FIGURE

PAGE

Chapter1:

Theoretical Study

1.1Tracking Systems:

1.1.1 Introduction:

Generally, a tracking system is used for the observing of persons or objects on the move and supplying a timely ordered sequence of respective location data to a model.

1.1.2 Tracking in virtual space:

In virtual space technology, a tracking system is generally a system capable of rendering virtual space to a human observer while tracking the observer's body coordinates. For instance, in dynamic virtual auditory space simulations, a real-time head tracker provides feedback to the central processor, allowing for selection of appropriate head-related transfer functions at the estimated current position of the observer relative to the environment.

1.1.3 Tracking in real world:

There are a myriad of tracking systems. Some are 'lag time' indicators, that is, the data is collected after an item has passed a point for example a bar code or choke point or gate. Others are 'real-time' or 'near real-time' like Global Positioning Systems depending on how often the data is refreshed. There are bar-code systems, which require a person to scan items and automatic identification (RFID auto-id). For the most part, the tracking worlds are composed of discrete hardware and software systems for different applications. That is, bar-code systems are separate from Electronic Product Code (EPC) systems, GPS systems are separate from active real time locating systems or RTLS for example, a passive RFID system would be used in a warehouse to scan the boxes as they are loaded on a truck - then the truck itself is tracked on a different system using GPS with its own features and software.

1.1.4 The major technology in the supply chain are:

1.1.4.1 Distribution/Warehousing/Manufacturing:

Indoors assets are tracked repetitively reading e.g. a barcode, any passive and active RFID and feeding read data into Work in Progress models (WIP) or Warehouse Management Systems (WMS) or ERP software.

1.1.4.2 Yard management:

Outdoors mobile assets of high value are tracked by choke point. Received (RSSI), Time Delay on Arrival (TDOA), active RFID or GPS Yard Management; feeding into either third party yard management software from the provider or to an existing system. Yard Management Systems (YMS) couple location data collected by RFID and GPS systems to help Supply Chain Managers to optimize utilization of yard assets such as trailers and dock doors. YMS systems can use either active or passive RFID tags.

1.1.4.3 Fleet management:

Fleet management is applied as a tracking application using GPS and composing tracks from subsequent vehicle's positions. Each vehicle to be tracked is equipped with a GPS receiver and relays the obtained coordinates via cellular or satellite networks to a home station. Fleet management is required by:

- Large fleet operators, (vehicle/railcars/trucking/shipping).
- Forwarding operators (containers, machines, heavy cargo, valuable shipping's).
- Operators who have high equipment and/or cargo/product costs.
- Operators who have a dynamic workload.

1.1.4.4 Mobile phone services:

Location-based services or LBS is a term that is derived from the telematics and telecom world. The combination of A-GPS, newer GPS and cellular locating

technology is what has enabled the latest “LBS” for handsets and PDAs. Line of sight is not necessarily required for a location fix. This is a significant advantage in certain applications since a GPS signal can still be lost indoors. As such, A-GPS enabled cell phones, PDAs can be located indoors, and the handset may be tracked more precisely.

Currently, A-GPS enabled handsets are still highly dependent on the Location-Based Service (LBS) carrier system, so handset device choice and application requirements are still not apparent. Enterprise system integrators need the skills and knowledge to correctly choose the pieces that will fit the application and geography.

1.2 GPS (GLOBAL POSITIONING SYSTEM):

1.2.1 GPS Basics:

The Global Positioning System (GPS) is a space-based global navigation satellite system (GNSS) that provides reliable location and time information in all weather and at all times and anywhere on or near the Earth when and where there is an unobstructed line of sight to four or more GPS satellites. It is maintained by the United States government and is freely accessible by anyone with a GPS receiver. When people talk about "a GPS," they usually mean a GPS receiver. The Global Positioning System (GPS) is actually a constellation of 27 Earth-orbiting satellites (24) in operation and three extras in case one fails. The U.S. military developed and implemented this satellite network as a military navigation system, but soon opened it up to everybody else. Each of these 3,000- to 4,000-pound solar-powered satellites circles the globe at about 12,000 miles (19,300 km), making two complete rotations every day. The orbits are arranged so that at anytime, anywhere on Earth, there are at least four satellites "visible" in the sky.

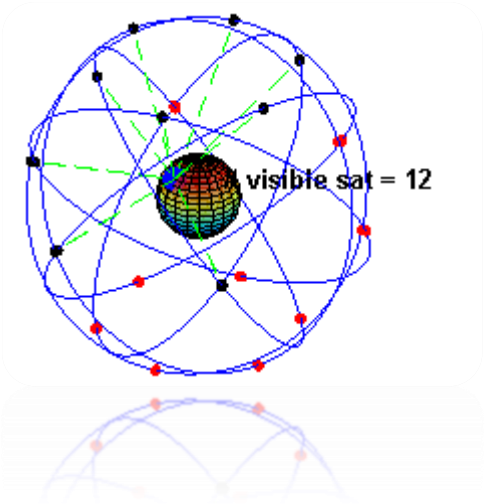


Figure (1.1) GPS satellites

In addition to GPS, other systems are in use or under development. The Russian Global Navigation Satellite System (GLONASS) was developed contemporaneously with GPS, but suffered from incomplete coverage of the globe until the mid-2000s. There are also the planned European Union Galileo positioning system, India's Indian Regional Navigational Satellite System and Chinese Compass navigation system.

1.2.2 Navigation equations:

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 the time the message was transmitted precise orbital information (the ephemeris) the general system health and rough orbits of all GPS satellites (the almanac).

The receiver uses the messages it receives to determine the transit time of each message and computes the distance to each satellite. These distances along with the satellites' locations are used with the possible aid of trilateration, depending on which algorithm is used, to compute the position of the receiver. This position is then displayed, perhaps with a moving map display or latitude and longitude; elevation information may be included. Many GPS units show derived information such as direction and speed, calculated from position changes.

Mathematical model:

i1=automatically generated source location from GPS.

i2= destination location given by user.

i3= rate per km as per city.

i4= time.

Fare = Distance * Rate

Where, distance is in km, Rate and Fare in Rupees.

Spherical law cosines formula for calculating distance between two GPS coordinates:

d=acos (sin (lat1)*sin (lat2) +cos (lat1)*cos (lat2)*cos (long2-long1))*R

Where, Lat1 is latitude of source, Lat2 is latitude of destination, Long1 is longitude of source, Long2 is longitude of destination, R is 6371 (km radius of earth) and d is the distance between source and destination.

1.2.3 Satellite frequencies:

All satellites broadcast at the same two frequencies, 1.57542 GHz (L1 signal) and 1.2276 GHz (L2 signal). The satellite network uses a CDMA spread-spectrum technique, where the low-bitrate message data is encoded with a high-rate pseudo-random (PRN) sequence that is different for each satellite. The receiver must be aware of the PRN codes for each satellite to reconstruct the actual message data. The C/A code, for civilian use, transmits data at 1.023 million chips per second, whereas the P code, for U.S. military use, transmits at 10.23 million chips per second. The actual internal reference of the satellites is 10.22999999543 MHz to compensate for relativistic effects that make observers on Earth perceive a different time reference with respect to the transmitters in orbit. The L1 carrier is modulated by both the C/A and P codes, while the L2 carrier is only modulated by the P code. The P code can be encrypted as a so-called P(Y) code that is only available to military equipment with a proper decryption key. Both the C/A and P(Y) codes impart the precise time-of-day to the user.

The L3 signal at a frequency of 1.38105 GHz is used to transmit data from the satellites to ground stations. This data is used by the United States Nuclear Detonation (NUDET) Detection System (USNDS) to detect, locate, and report nuclear detonations (NUDETs) in the Earth's atmosphere and near space. One usage is the enforcement of nuclear test ban treaties. The L4 band at 1.379913 GHz is being studied for additional ionospheres' correction. The L5 frequency band at 1.17645 GHz was added in the process of GPS modernization. This frequency falls into an internationally protected range for aeronautical navigation, promising little or no interference under all circumstances. The first Block IIF satellite that provides this signal was launched in 2010.The L5 consists of two carrier components that are in phase quadrature with each other. Each carrier component is bi-phase shift key (BPSK) modulated by a separate bit train. "L5, the third civil GPS signal, will eventually support safety-of-life applications for aviation and provide improved availability and accuracy.

GPS frequency overview

Band	Frequency	Description
L1	1575.42 MHz	Coarse-acquisition (C/A) and encrypted precision (P(Y)) codes, plus the L1 civilian (L1C) and military (M) codes on future Block III satellites.
L2	1227.60 MHz	P(Y) code, plus the L2C and military codes on the Block IIR-M and newer satellites.
L3	1381.05 MHz	Used for nuclear detonation (NUDET) detection.
L4	1379.913 MHz	Being studied for additional ionospheric correction.[citation needed]
L5	1176.45 MHz	Proposed for use as a civilian safety-of-life (SoL) signal.

Figure (1.2) GPS Frequency overview

1.2.4 Other systems:

Other satellite navigation systems in use or various states of development include:

- GLONASS – Russia's global navigation system. Fully operational worldwide.
- Galileo – a global system being developed by the European Union and other partner countries planned to be operational by 2014 (and fully deployed by 2019).
- Beidou – People's Republic of China's regional system, currently limited to Asia and the West Pacific.
- COMPASS – People's Republic of China's global system, planned to be operational by 2020.
- IRNSS – India's regional navigation system, planned to be operational by 2014, covering India and Northern Indian Ocean.
- QZSS – Japanese regional system covering Asia and Oceania.

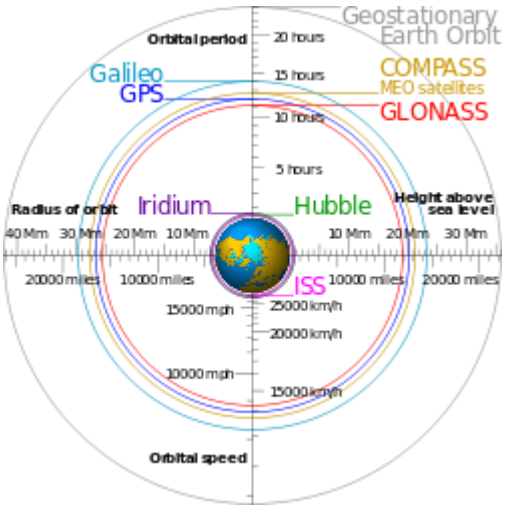


Figure (1.3) Other System satellites

1.2.5 Applications:

While originally a military project, GPS is considered a dual-use technology, meaning it has significant military and civilian applications.

GPS has become a widely deployed and useful tool for commerce, scientific uses, tracking, and surveillance. GPS's accurate time facilitates everyday activities such as banking, mobile phone operations, and even the control of power grids by allowing well synchronized hand-off switching.

Many civilian applications use one or more of GPS's three basic components: absolute location, relative movement, and time transfer.

- Astronomy: both positional and clock synchronization data is used in Astrometry and Celestial mechanics calculations. It is also used in amateur astronomy using small telescopes to professional's observatories, for example, while finding extra solar planets.
- Automated vehicle: applying location and routes for cars and trucks to function without a human driver.
- Cartography: both civilian and military cartographers use GPS extensively.
- Cellular telephony: clock synchronization enables time transfer, which is critical for synchronizing its spreading codes with other base stations to facilitate inter-cell handoff and support hybrid GPS/cellular position detection for mobile emergency calls and other applications. The first handsets with integrated GPS launched in the late 1990s. The U.S. Federal Communications Commission (FCC) mandated the feature either in the handset or in the towers (for use in triangulation) in 2002, so emergency services could locate 911 callers. Third-party software developers later gained access to GPS APIs from Nextel upon launch, followed by Sprint in 2006, and Verizon soon thereafter.
- Clock synchronization: the accuracy of GPS time signals (±10 ns) is second only to the atomic clocks upon which they are based.
- Disaster relief/emergency services: depend upon GPS for location and timing capabilities.
- Meteorology-Upper Airs: measure and calculate the atmospheric pressure, wind speed and direction up to 27 km from the earth's surface
- Fleet Tracking: the use of GPS technology to identify, locate and maintain contact reports with one or more fleet vehicles in real-time.

- Geofencing: vehicle tracking systems, person-tracking systems, and pet tracking systems use GPS to locate a vehicle, person, or pet. These devices are attached to the vehicle, person, or the pet collar. The application provides continuous tracking and mobile or Internet updates should the target leave a designated area.
- Geotagging: applying location coordinates to digital objects such as photographs (in Exif data) and other documents for purposes such as creating map overlays with devices like Nikon GP-1
- GPS Aircraft Tracking.
- GPS for Mining: the use of RTK GPS has significantly improved several mining operations such as drilling, shoveling, vehicle tracking, and surveying. RTK GPS provides centimeter-level positioning accuracy.
- GPS tours: location determines what content to display; for instance, information about an approaching point of interest.
- Navigation: navigators value digitally precise velocity and orientation measurements.
- Phasor measurements: GPS enables highly accurate time stamping of power system measurements, making it possible to compute phasors.
- Recreation: for example, geocaching, geodashing, GPS drawing and way marking.
- Robotics: self-navigating, autonomous robots using GPS sensors, which calculate latitude, longitude, time, speed, and heading.
- Surveying: surveyors use absolute locations to make maps and determine property boundaries.
- Tectonics: GPS enables direct fault motion measurement in earthquakes.
- Telematics: GPS technology integrated with computers and mobile communications technology in automotive navigation systems.
- Navigation: GPS allows soldiers to find objectives, even in the dark or in unfamiliar territory, and to coordinate troop and supply movement. In the United States armed forces, commanders use the Commanders Digital Assistant and lower ranks use the Soldier Digital Assistant.

- Target tracking: Various military weapons systems use GPS to track potential ground and air targets before flagging them as hostile. These weapon systems pass target coordinates to precision-guided munitions to allow them to engage targets accurately. Military aircraft, particularly in air-to-ground roles, use GPS to find targets.
- Missile and projectile guidance: GPS allows accurate targeting of various military weapons including ICBMs, cruise missiles, precision-guided munitions and Artillery projectiles. Embedded GPS receivers able to withstand accelerations of 12,000 g or about 118 km/s^2 have been developed for use in 155 millimeters (6.1 in) howitzers.
- Search and Rescue: Downed pilots can be located faster if their position is known.
- Reconnaissance: Patrol movement can be managed more closely.
- GPS satellites carry a set of nuclear detonation detectors consisting of an optical sensor (Y-sensor), an X-ray sensor, a dosimeter, and an electromagnetic pulse (EMP) sensor (W-sensor), that form a major portion of the System.

1.3 Android:

1.3.1 Android and the Open Handset Alliance:

Google, along with the Open Handset Alliance (OHA), launched the mobile operating system Android in the end of 2007 (DiMarzio 2008, 23.) The Open Handset Alliance is dedicated to "accelerate innovation in mobile and offer consumers a richer, less expensive, and better mobile experience" (Open Handset Alliance 2007a.) Nowadays the OHA is represented by 84 of the leading software and hardware companies, and mobile operators, for example Sprint, T-Mobile, Intel, Asus and Acer in addition to Google and others (Open Handset Alliance 2007a, Open Handset Alliance 2007b.) As the name of the Open Handset Alliance implies, the companies representing it are dedicated to contribute to the openness of the mobile world. This is why Android is an open source system bringing cheaper and more innovative products for customers and better developing platforms for programmers. As soon as Android was first released,

development tools and tutorials were made available to help new developers find the new system. (Open Handset Alliance 2007a.) The Open Handset Alliance has an advantage over other operating system developers, because Android as such is a ready package to be converted into different devices.

1.3.2 What Androids are made of:

While writing applications on desktop, you are "master of your own domain". You can launch main window and no of child windows as in dialog boxes. From our viewpoint, we are on our own, features, which are supported by OS, and mostly unaware of any other programs we can communicate with MySQL or any other database typically using API like frameworks atop it. Android has comparably same concepts packaged in a different way and structured for crash resistant. (Murphy, 2008).

1.3.2.1 Activities:

Activity can be explained as building block of user interface. We can consider activity as analogue for window or dialog in desktop in desktop application. It can even be possible that activity not having a user interface, while the code packaged in the structure of services or content providers.

1.3.2.2 Content Providers:

For any data stored in device, Content Providers offer a level of abstraction, which is accessible by various applications. In android development it encourage us to make our own data which can be accessed by other applications and even build our own content provider which gives you a complete control over how that data can be accessed.

1.3.2.3 Intents:

System messages, which run inside the device, various applications notification such as hardware changes like SD card inserted, notifications of incoming data like SMS arrived and even application events are called as Intents. It doesn't only allow you to

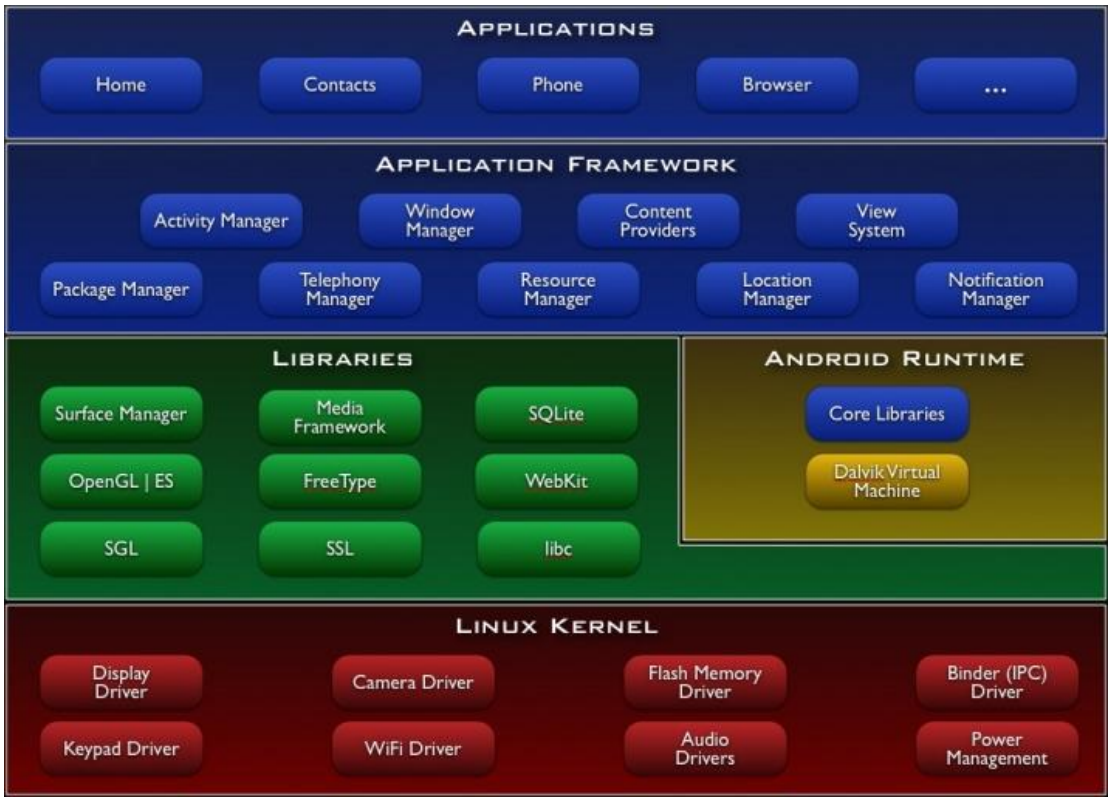
respond to such intents but also to initiate other activities or let knew when particular event occurs such as suggest WIFI availability when in range. (Murphy, 2008).

1.3.2.4 Services:

All the above stated Activities, intent receivers and content providers are all short term and can be terminated any time whereas services are intended to run continuously independent of other activities such as play music while using other applications, in here music controlling is no longer available but the service keep running in the background.

1.3.3 Android from a technical perspective:

Android is a stack of software built for mobile devices. It consists of an operating system, middleware and some key applications. Android applications can be programmed with the Java-language. (Android Developers 2011a.).



The major components of the Android OS (Android Developers 2011a.)

Figure (1.4)

1.3.3.1 Applications:

By default, all Android devices have an email client, a text messaging program, a calendar, a map software (usually Google Maps), an Internet browser, an application for managing contacts, and other core applications installed in them. More applications for Android can be downloaded from the Android Market; some of the applications there are free and some cost a low amount of money. (Android Developers 2011a.).

1.3.3.2 Application Framework:

Because of Android's open development platform, developers have free hands to build very innovative applications. It is possible for developers to take advantage of the de-vice hardware, access location information, run background services, set alarms and add notifications to the status bar, basically anything that helps in creating the ideal application.

Being an open system, Android lets developers have full access to the same framework APIs, application-programming interfaces, which are used by the core applications. (Android Developers 2011a.)

1.3.3.3 Libraries:

A set of C/C++ programming languages' libraries are included in Android. They are used by various components of the system and visible to developers through the application framework. Key libraries and their functions are described briefly below.

System C Library: a standard C system library tuned for embedded Linux-based devices.

- Media Libraries: libraries supporting playback and recording of many popular image files and audio and video formats.
- Surface Manager: unites 2D and 3D graphic layers from several applications.
- LibWebCore: a modern web browser engine powering the Android browser and the embeddable Web View described later.
- SGL: the main 2D graphics engine.

- 3D libraries: libraries using either hardware 3D acceleration or the 3D software rasterizer.
- Free Type: used to render bitmap and vector fonts to the form needed.
- SQLite: a lightweight database engine available for all applications to use.

1.3.3.4 Android Runtime:

There is a set of core libraries in android resembling the core libraries of the Java programming language and thus providing similar functionality.

Android uses a certain virtual machine known as Dalvik to run each of its applications. These applications are run in their own processes each of which having their own instance of the Dalvik virtual machine. Thanks to the way Dalvik has been written, a device can efficiently run multiple virtual machines at the same time. Applications are executed in the Dalvik Executable format in a way that minimal memory footprint is required. Dalvik relies on the Linux kernel for matters concerning threading and low-level memory management.

1.3.3.5 Linux Kernel:

The Linux kernel version 2.6 is the communicator between the device hardware and the software. The kernel also takes care of Android's core system services, which include the overall security, memory management, process management, network related systems and the driver model.

1.3.4 Android SDK:

The Android SDK provides a group of tools needed when developing applications for the Android operating system. It is recommended to use the Eclipse software development environment to run ADT, Android Development Tools, which then gives the user access to all of the SDK tools.

It is also possible to develop applications without Eclipse: this method requires the use of a text editor and the ability to use the SDK tools on the command line or with

scripts. This is not such an easy way as using Eclipse but all the same, tools are still available.

The most important SDK tools include android, which is an SDK and AVD Manager, the emulator and the Dalvik Debug Monitor Server, DDMS. In addition to these, there are also many other tools available and included in the SDK starter package.

1.3.4.1 System and software requirements for developing with Android SDK:

The system and software requirements for developing applications with Android SDK are briefly listed in this section. For the original list, please see the Android developer web guide at http://developer.android.com/sdk/requirements.html.

1.3.4.2 Supported Development Environments:

The integrated development environment best capable of running the Android SDK is Eclipse IDE. From it, you need the version 3.5 or higher. One of the following Eclipse IDE packages is recommended for Android development: Eclipse IDE for Java Developers, Eclipse Classic v.3.5.1 and higher, or Eclipse IDE for Java EE Developers. If not already included in the Eclipse IDE, it is necessary to install the appropriate JDT, Java Development Tools, plugin.

The JDK, Java Development Kit, version five or six is required. The JRE, Java Run-time Environment, alone is not enough. The installation of the Android Development Tools plugin is highly recommended.

1.4 Conclusion:

Regardless of the tracking technology, for the most part the end-users just want to locate themselves or wish to find points of interest. The reality is that there is no "one size fits all" solution with locating technology for all conditions and applications.

Application of tracking is a substantial basis for vehicle tracking in fleet management, asset management, individual navigation, social networking, asset management, or

mobile resource management and more. Company, group or individual interests can benefit from more than one of the offered technologies depending on the context.

One of these technologies is GPS receivers, for the last 30 years, GPS receivers have operated next to the Mobile Satellite Service band, and have discriminated against reception of mobile satellite services, without any issue.

So because of technological progress, we are facing an incredible variety of possibilities to communicate regardless of the distance. Smartphone's provide a great choice of features that facilitate the life for the users as well they make it more comfortable.

Many of these features are primarily integrated with the mobile operating system, which is out of reach to public, by which the users can't manipulate those features. Google came up with an innovative operation system termed as ANDROID, which is open system architecture with customizable third party development and debugging environment, which helps the users to manipulate the features and to create their own customizable applications.

1.5 General requirements of our project:

Every day the features and capabilities of Mobiles are increasing surprisingly. For this reason, we want to create an Android application, which gives us the opportunity to improve our knowledge of Mobile developing, and test all these features provides by Mobiles, due to they are turning into a vital part of human life.

We find this project a great opportunity to combine many technologies and languages in the same software system, and learn how to work in big projects as a team member.

After eliciting the requirements of the project, we divided the system in tow main parts. The first was client side (Android application). And the second part is server side (database).

Finally, we noticed that the application need to store and access the information that the users share, so we decided to make a server using web services to communicate with the applications.

Location will come from Google location, which contain many technologies like:

- GPS applications.
- Real-time Locating Systems (RTLS).

This web service is the system's main service, which means that will provide an Application Programming Interface (API for the rest of the Android application. The Web service application provides an API to allow clients to access the web services that interacts with the database.

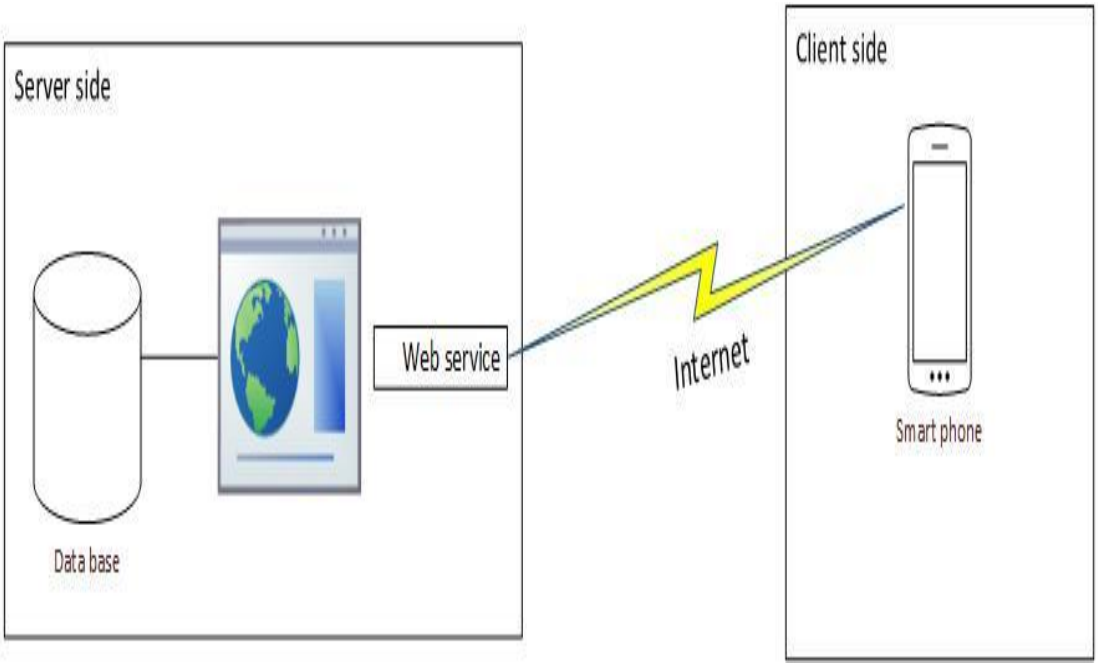


Figure (1.5) System Component

It is decided to use a web service because it will be called from different environments. The web services provide a high compatibility with any system. The reason for this is that these remote calls are serialized in XML, transferred over HTTP using json messages.

Chapter 2:

Server side

2.1 Introduction:

After analyzing the system's requirements, we decided to implement the main server using a web service. This API has to implement all the functions the Android program requires that have to do with the database.

2.2 Web service application:

A web service is typically an application-programming interface (API) or Web API that is accessed via Hypertext Transfer Protocol (HTTP) and executed on a remote system, hosting the requested service. Web services tend to fall into one of two camps: big web services and RESTful web services.

The W3C defines a "web service" as "a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-process able format. Other systems interact with the web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web related standards." The W3C also states, "We can identify two major classes of Web services, REST-compliant Web services, in which the primary purpose of the service is to manipulate XML representations of Web resources using a uniform set of "stateless" operations; and arbitrary Web services, in which the service may expose an arbitrary set of operations.

2.3 Requirements:

a. Be able to handle user's login from the android using Email and password.
b. Be able to recognize user session after login.
c. Allow to check if an Email is already being used.
d. Allow a creation of new users.
e. Allow users to get their own or their friends in the group last position.
f. Allow users to get any friend positions (truck).

g. Allow the user to send his own position.
h. Allow the user to get their friend shared with a group list.
i. Allow the user to get list of the group he is in it.
j. Allow manger to modify the group.

2.4 Use case discretion:

2.4.1 Login:

The system take the user login information and check it then make him login.

2.4.2 Regester:

A new user wants to use our system, the system take his personal data and store it in the database.

2.4.3 Regester my position:

The system reserve the location data from client side every 60s and store it in the right field in the database.

2.4.4 Make groups and edit it:

The system allow users to create new groups and edit on this groups and store the group name and the user creator name.

2.4.5 Recognize the user type:

The system may have an algorithm to recognize the user type to allow to the manger to edit the group, allow to the manger and tracker to show the friends position, and then send positions to them.

2.4.6 Get location information:

The system may have to store the time of every new position for users to send this information to the client side if needed.

2.4.7 Get group friend:

The system may have to recognize the group members and the time they are join to the group to send this information to the client side if needed.

2.5 Use case diagram:

Figure (2.1) web service use case diagram

We choose to use codgnizer platform to develop the web service with php code work in apache and my sql engine so we begin with create the data base then write our classes to control it and make the connection with the android application.

2.6 The database:

It has already been decided to use a independent database system to store the data. In this section, we have separated the database's analysis, design and implementation.

2.7 Data base Analysis:

1. Database must be able to store user with information, name, Email, mobile and password.

2. Each mobile user is going to have his own positions although the user type (manger, tracker, tracked) need all position to show the users track path in time.
3. The user has to be able to make groups edit in it and delete it. There has to be a direct relation that specifies that the user be in same group.

2.8 Data base design:

1. The is going to be a user entity with a primary ID that identifies a single user with the attributes Email, name, mobile and password.
2. Another entity is going to store all users' position in time; this means the users IDs, the time and the position. The position in GPS is composed of two float numbers, one for the Latitude and the other for the longitude.
3. Another entity is going to be a group entity with a primary ID that identifies a single group with the attributes group name and create by going to tell who make the group and can modify it.
4. One relationship between user and groups there is an entity are going to store group member. The first filed is who is in this group, the second filed tell which group is in, the third filed tell type of the first filed (manger, tracker or tracked) and the last filed tell us the time of join to the group.

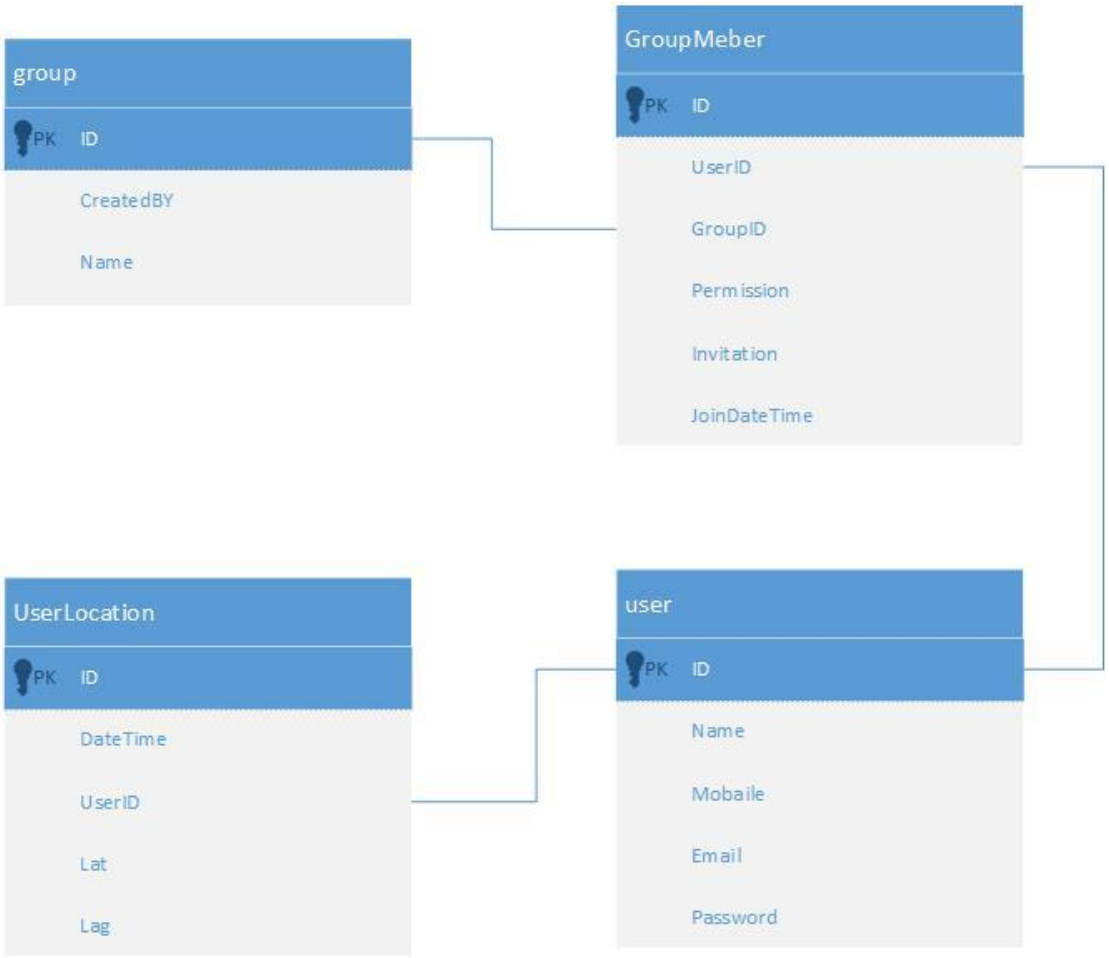


Figure (2.2) database design

2.9 Choosing software:

We have decided to use MySQL database due to the following reasons:

1. MySQL open source software is provided under the GPL License, what means it is free.
2. The whole team has previous experience with this software.
3. It has the innoDB engine, which supports foreign keys constraints.
4. It is the world's most popular open source database.
    a) Is supposed to be reliable.
    b) A large amount of information can be found on the Internet.
    c) Others standard database driver connectivity with applications and tools that are compatible with industry standards ODBC and JDBC.

2.10 Implementation:

- Decisions depending on software as the database design has relations between tables we found strongly beneficial to implement using foreign keys. Due to this fact, we decided to use the innoDB engine, as it is the only one that supports this feature natively.
- Setting up the database system at this time we decide to use phpmyadmin to handle the administration of MySQL. The main reason is that we already have installed an Apache Server with PHP5 used for the website and we have previous experience with this software. Then we creating the database using the phpmyadmin interface. We create each table with all fields and set the primary keys, index fields and we use the innoDB engine. Ones this is done we create all the constraints using the foreign key references between tables.

Note that this could be done using MySQL client instead of using phpmyadmin, for example to create the "user" table, the following SQL statement could be used:

```
CREATE TABLE IF NOT EXISTS `User` (
  `ID` int (11) NOT NULL AUTO_INCREMENT,
  `Name` varchar (50) NOT NULL,
  `Mobile` varchar (50) NOT NULL,
  `Email` varchar (50) NOT NULL,
  `Password` varchar (50) NOT NULL,
  PRIMARY KEY (`ID`)
) ENGINE=InnoDB DEFAULT CHAR SET=utf8 AUTO_INCREMENT=1;
```

2.11 Web Service:

In the codgnizer platform, we download many folder but we need the API folder, which contain:

- System folder: this folder contain the libbers that codgnizer used in the platform.
- Index.XML: the main page of my service.
- Application folder: this folder that we edit it with our service classes to programming our web service this classes will be:

1- Db_model: this class have the function to work with our sql database and store information in it or read information from it like that :

```php
<?php

class Db_model extends CI_Model
{
    function GetUserGroups($UserID)
    {
        $this->db->select("Group.ID as 'ID', Group.Name as 'Name', Group.CreatedBy, GroupMember.Permission, GroupMember.Invitation, GroupMember.JoinD
        $this->db->where("UserID", $UserID);
        $this->db->where("GroupMember.Invitation !=", "Rejected");
        $this->db->join("GroupMember", "GroupMember.GroupID = Group.ID");
        $this->db->order_by("ID", "asc");
        $q = $this->db->get("Group");

        $Groups = array();

        if($q->num_rows() > 0)
        {
            foreach ($q->result() as $row)
            {
                $Groups[] = $row;
            }
        }

        return array("Status" => "Success", "Action" => "GetUserGroups", "UserID" => $UserID, "Groups" => $Groups);
    }
```

2- V1: This class is a sub class from reset controller and have the functions to work with data in our service and functions to the other actions we want to do and it like that:

```php
<?php if ( ! defined('BASEPATH')) exit('No direct script access allowed');

require(APPPATH.'/libraries/REST_Controller.php');

class V1 extends REST_Controller
{
    public function index_get()
    {
        $this->response(array("Message" => "Not Allowed"), 403);
    }

    public function user_get()
    {
        $this->response($this->db_model->GetUser($this->get()), 200);
    }

    public function user_groups_get()
    {
        $this->response($this->db_model->GetUserGroups($this->get("UserID")), 200);
    }

    public function group_members_get()
    {
        $this->response($this->db_model->GetGroupMembers($this->get("GroupID")), 200);
    }
```

3- ROUT: this class have the function to which action our service will begin first and the home page in the controller.

Finally, we transfer our service to the server with FTP protocol and turn it on.

Chapter 3:

Client side

3.1 Introduction:

The Android application gives the opportunity to share information with other users in an intuitive way. In order to do that, the application displays a friendly interface that gives the abilities to register new users in the system, make and accept friend requests, share user's locations automatically, show the user's friends locations.

3.2 Requirements:

a. Be able to recognize a user (manger, tracker, tracked).
b. Be able to register new user.
c. Not allow two or more users with the same Email.
d. Be able to store information like name, email, phone and mobile.
e. Be able to display world map and the interface to navigate it.
f. Be able to send periodically the GPS position in order to update the user location.
g. Allow user (without tracked) to see his friends in groups if they have already apposition associated.
h. Be able to draw the user's (without tracked) friends in map using their last known position.
i. Be able to draw a friend history of previous position in a map.
j. Allow a manger to delete a user and edit in this group.
k. Allow manger to create many groups.

3.3 Use cases description:

Here we quote the different use cases that have been identified for the Android Application:

- Login: The user logs in the system.
- Register: A new user wants to use our system, he enters his personal data in the application and the system stores it.
- Update position: The system updates the user's position.
- Add group: The user create a new group in the system.
- Edit group: the manager edit on his group.
- View friends position: The system displays a map with the position of the user's friends in a group.

- Request for friend: A user sends a request for friend to another user to join his group.
- View friend requests: The system displays a list of friend requests that the user has to join a group.
- Accept a friend: The user selects a user from a list and the system stores the user is member on the group.
- Reject a friend: The user rejects the request for friend.

3.4 Use case diagram:

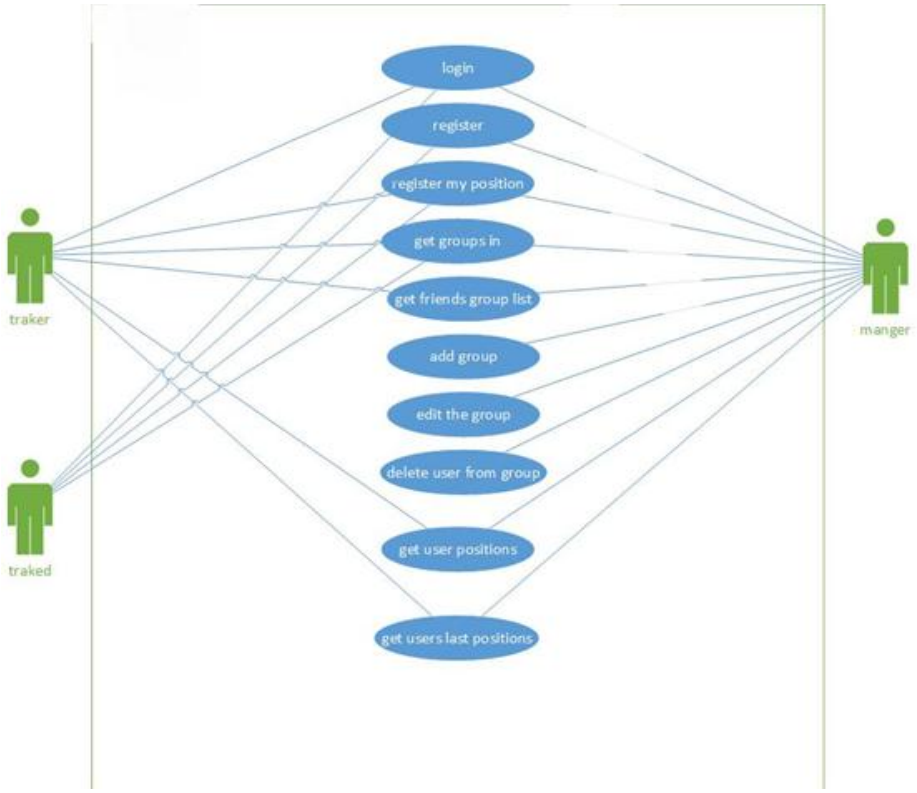


Figure (3.1) Android Use case diagram

3.5 Class Diagram:

The Android application needs to contact with the Find Friend Location main server in order to access the Web service's public methods. This is going to be done by using the json message over the HTTP protocol. The Android application also needs to send request messages with the serialized parameters and the web server is going to return the response messages with serialized values.

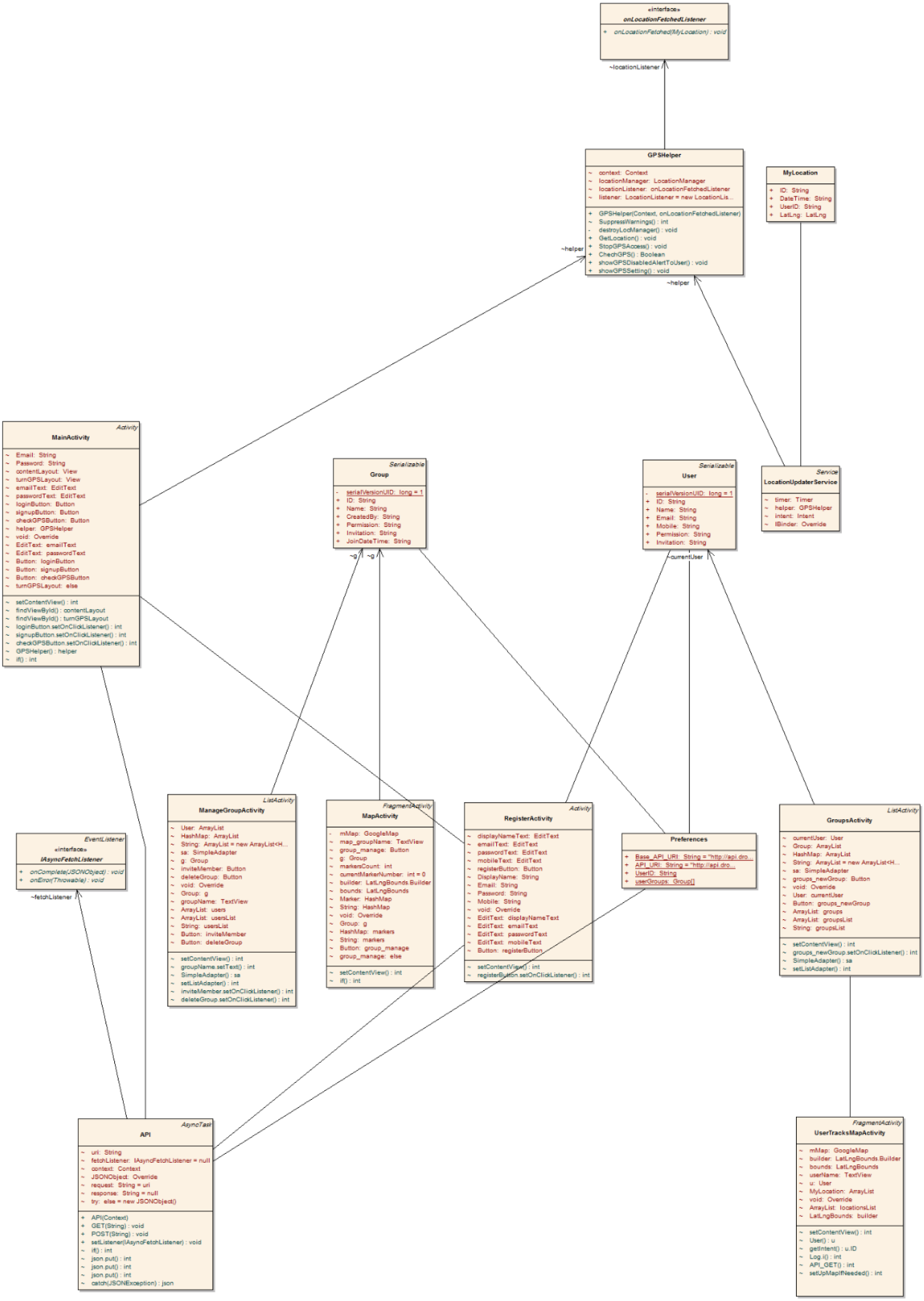


Figure (3.2) android Class diagram (classes in Appendix page iii)

3.6 Install to develop in Eclipse:

- Install the Eclipse IDE. You have to download in the next site http://www.eclipse.org/downloads/ and install it. The installation depends on the operation system that you have, but it is very easy anyway so we are not going to explain how to install.
- Once you have Eclipse installed in your computer you need to install the ADT Plugin.

  1. Start Eclipse, and then select Help! Install New Software.

  2. Click Add, in the top-right corner.

  3. In the Add Repository dialog that appears, enter "ADT Plugin" for the Name and the following URL for the Location: https://dlssl.google.com/android/eclipse/.

  4. In the Available Software dialog, select the checkbox next to Developer Tools and click next.

  5. In the next window, you will see a list of the tools to be downloaded. Click Next.

  6. Accept the license agreements, and then click Finish.

  7. When the installation completes, restart Eclipse
- Now we need to install the correct SDK.

  1. Select Window! Android SDK and AVD Manager.

  2. Select Available packages.

  3. Select the Android Repository and expand the line Third party
  Add-ons in order to select Google APIs.

  4. Press Install Selected
- Install the emulator (Android Virtual Machine).

  1. Select Window! Android SDK and AVD Manager.

  2. Select Virtual Devices.

  3. Click New.

  4. Selects a name for the AVD.

  5. Selects Google APIs (Google Inc.) in Target.

  6. Click Create AVD.

3.7 Implementation:

This section tries to explain how to develop code for the application in order to facilitate the learning of the application to new developers. We will focus especially on the most interesting parts of the Android application like maps and GPS.

3.7.1 Android's activities:

Android activities have control over the interfaces XML. Each screen in the program has an Activity. In the application are found 6 different Activities

1. Main: This Activity allows users to login. It takes the Email and password from the user and sends them to the server.

2. Register: This Activity allows users to register. It takes the information from the user and sends them to the server.

3. Groups: This Activity shows the user groups and the invitation to the groups.

4. Map: This Activity allows users to show the friends in the group last location on a Google map.

5. Manage Group: This Activity allows only the manger to edit on his groups.

6. User_tracks_map: This Activity allows only the manger and tracker to show a friend track since he registered on the group.

3.7.2 Programming:

- First, in order to add a new Activity to the program it is necessary to modify the AndroidManifest.xml file to be as this code.

```
<? Xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="tk.droidtrack.app"
   android:versionCode="1"
```

```
    android:versionName="1.0" >

    <uses-permission android:name="android.permission.READ_PHONE_STATE" />
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission
android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission
android:name="android.permission.ACCESS_COARSE_LOCATION" />
    <uses-permission
android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE" />

    <uses-sdk
        android:minSdkVersion="15"
        android:targetSdkVersion="20" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <meta-data
            android:name="com.google.android.maps.v2.API_KEY"
            android:value="AIzaSyCsWvkK0sYEuPKxkgxb419R3yXp9ZDZp7g" />
        <meta-data
            android:name="com.google.android.gms.version"
            android:value="@integer/google_play_services_version" />

        <activity
            android:name="tk.droidtrack.app.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>
    <activity
        android:name=".MapActivity"
        android:screenOrientation="portrait" />
    <activity
        android:name=".GroupsActivity"
        android:screenOrientation="portrait" />
    <activity
        android:name=".ManageGroupActivity"
        android:screenOrientation="portrait" />
    <activity
        android:name=".UserTracksMapActivity"
        android:screenOrientation="portrait" />
    <activity
        android:name=".RegisterActivity"
        android:screenOrientation="portrait" />

    <service android:name=".LocationUpdaterService" />
  </application>


</manifest>
```

- Then you can add the activities on the code with just drug the item to its place, and the eclipse will generate an XML code in the layout folder, then you will to add the java code of this activity on the src folder, like the main activity on this

main_layout.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal" >

    <RelativeLayout
        android:id="@+id/main_contentLayout"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:paddingBottom="@dimen/activity_vertical_margin"
        android:paddingLeft="@dimen/activity_horizontal_margin"
        android:paddingRight="@dimen/activity_horizontal_margin"
        android:paddingTop="@dimen/activity_vertical_margin"
        android:visibility="gone" >

        <EditText
            android:id="@+id/main_emailText"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentLeft="true"
            android:layout_alignParentRight="true"
            android:layout_alignParentTop="true"
            android:ems="10"
            android:hint="@string/main_emailHint"
            android:inputType="textEmailAddress" >
```

MainActivity.java

```
package tk.droidtrack.app;

import java.util.ArrayList;

public class MainActivity extends Activity
{
    String Email;
    String Password;

    View contentLayout;
    View turnGPSLayout;

    EditText emailText;
    EditText passwordText;
    Button loginButton;
    Button signupButton;
    Button checkGPSButton;

    GPSHelper helper;

    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main_layout);

        contentLayout = findViewById(R.id.main_contentLayout);
        turnGPSLayout = findViewById(R.id.main_turnGPSLayout);

        emailText = (EditText) findViewById(R.id.main_emailText);
        passwordText = (EditText) findViewById(R.id.main_passwordText);
        loginButton = (Button) findViewById(R.id.main_loginButton);
        signupButton = (Button) findViewById(R.id.main_signupButton);
        checkGPSButton = (Button) findViewById(R.id.main_checkGPSButton);

        loginButton.setOnClickListener(new OnClickListener()
        {
            @Override
            public void onClick(View arg0)
            {
```

- And the other classes to complete the work is:

1-user: this class to define the user type to takes the user's object from it.

2-group: this class to define the group type to takes the group's object from it.

3- Preferences: this class to make the login automatic after the first time and when we do not log out, and it's refer to a memory location when we store this data.

4- My Location: this class to define the location attributes as Google planning to take objects from it.

5- GPS Helper: this class define the work with the GPS device in the phone and check here state.

6- API:

1. The Android APIs: The core of the SDK is the Android API libraries that provide developer access to the Android stack. These are the same libraries used at Google to create native Android applications.

2. This class define to control the whole application and the way of make communication with server.

7-onLocationFetchedListener interface

```
package tk.droidtrack.app;

public interface onLocationFetchedListener
{
    public void onLocationFetched(MyLocation location);

}
```

8- IAsyncFetchListener interface

```
package tk.droidtrack.app;

import java.util.EventListener;

import org.json.JSONObject;

public interface IAsyncFetchListener extends EventListener

{

        void onComplete(JSONObject result);

        void onError(Throwable error);

}
```

9- LocationUpdaterService: this class define the service of our application, which work in background and send the user location data to the server every 60 second.

10- Google play service libraries:

it's important to work the application because the application using Google maps service's and Google Inc. store all its service's in on library to simplify the work of developer's which is Google play service .

Chapter 4:

System Testing

4.1 Introduction:

The chapter, which is presented below, deals with the various tests that have been made to the developed software to detect the failures it may have.

Along this chapter, there will be carried out two types of tests: unit tests and integration tests.

1. Unit tests try to detect if all application functions work correct individually.

2. Integration tests try to detect if all these functions are accessible in our application and they are properly integrated.

4.2 Client side test:

4.2.1 Unit tests:

4.2.1.1 Login:

- Input-> the application was running.
- Output-> the system display a dialog to turn on the GPS in your device.
- Conditions-> the GPS is turned off.

- Input-> the application was running.
- Output-> the system a massage the internet was not connected in your device.
- Conditions-> the internet connection is turned off.

- Input-> the email and password had been introduced.
- Output-> the user has been logged in.
- Conditions->the user must have been registered in the system before.

- Input ->the email and password had been introduced.
- Output -> the system displays a message stating that the email user is not registered.
- Conditions-> the user was not registering in the system before.

- Input->the email and password had been introduced.
- Output-> the system displays a message stating that the user has to enter an email or password.
- Conditions-> the user has not introduced a required field.

4.2.1.2 Register:

- Input-> all fields had been filled.
- Output-> the system displays a message stating that the user has been registered.
- Conditions-> all the fields have been filled correctly. The email and password fields have been used from another person.

- Input-> email or password had not been filled.
- Output-> the system displays a message informing about it.
- Conditions-> one of the fields was not filled.

4.2.1.3 Make a request for friends:

- Input-> it had been introduced a user to make a friend's request for another user.
- Output-> the system displays a message informing that the friend not retested yet.
- Conditions-> the friend have not an account in the system.

4.2.1.4 Open a group:

- Input->open a group which user in it.
- Output-> the system displays a message informing that you are tracked.
- Conditions-> the friend was a tracked member in the group.

4.2.2 Integration tests:

1. The application starts.
2. The login interface starts.
3. If you press Register in the Login interface, the Register interface starts.
4. Is possible to introduce values in the fields in Register's interface.
5. Is not possible introduce letters in the mobile's field in Register's interface.
6. Is not possible to show the password characters in Register's interface.
7. If you press Cancel button or Go back in your phone in Register's interface you return to the Login interface.
8. If you press Register in Register's interface the user has been stored and you go to the groups interface.
9. Is not possible to see the password characters in Login interface.
10. If you press Login in Login interface you go to the groups interface.
11. If you press create group in groups interface you add a new group by adding it's name.
12. If you press on a group name in groups interface you go to the map interface if you are manger or tracker but if you are tracked, you cannot go to the map interface.
13. If you click in one user's bubble in the map the system displays his track.
14. If you press mange group in map interface you go to the group mange interface.
15. If you press invite friend in group mange interface you can invite new friend to the group by his email.
16. If you press delete group in group mange interface you can delete the group.

17. If you press on a friend name in group mange interface you can delete or change permission of this friend in the group.

18. If you press the button Menu in your mobile the menu is displayed.

19. If you press Logout or Go back in your phone in the menu you return to login interface.

20. If you press Go back in Login interface you leaves the application.

4.3 Server side test:

4.3.1 Unit tests:

- Input-> open the home page.
- Output-> the system message this page not allowed.
- Conditions-> the android application is the only client connect with server information.

- Input-> the server have error data.
- Output-> the system display a massage there is an error.
- Conditions-> the data was not from the system requirements.

- Input-> the application was running.
- Output-> the system running correctly.

4.3.2 Integration tests:

1. The system start.
2. The system take the data from client and check it to allow the user login.
3. The system store the data in database if the action is registering.
4. The system send the groups that the user in it.
5. The system store data of anew group.
6. The system send the data of all friends last positions in the group.
7. The system send the data of all friends in the group.
8. The system update data of users or groups in database.
9. The system send a friend positions to the client to draw the track of him.
10. The system reload any action from before.

Chapter 5:

User guide

This document explains how to use the application

5.1 Main:

When you click in the application icon, you will see a simple interface (Figure 5.1)

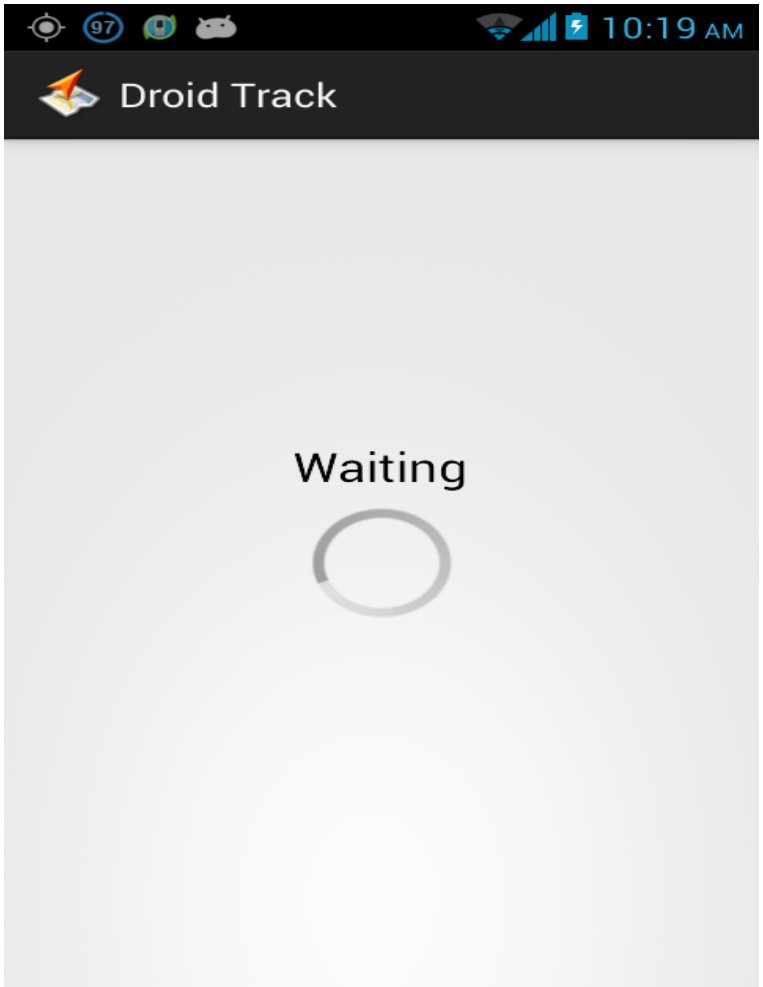


Figure (5.1) Application Main interface

If your gps not active the application will show you an interface to chick gps state and a dialog take you to the settings to turn the gps on (Figure 5.2) and (Figure 5.3)

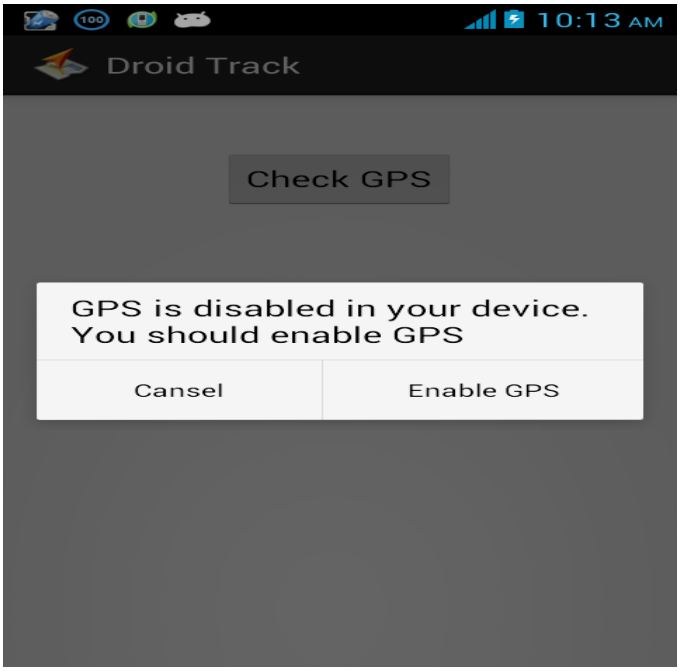


Figure (5.2) Application GPS dialog

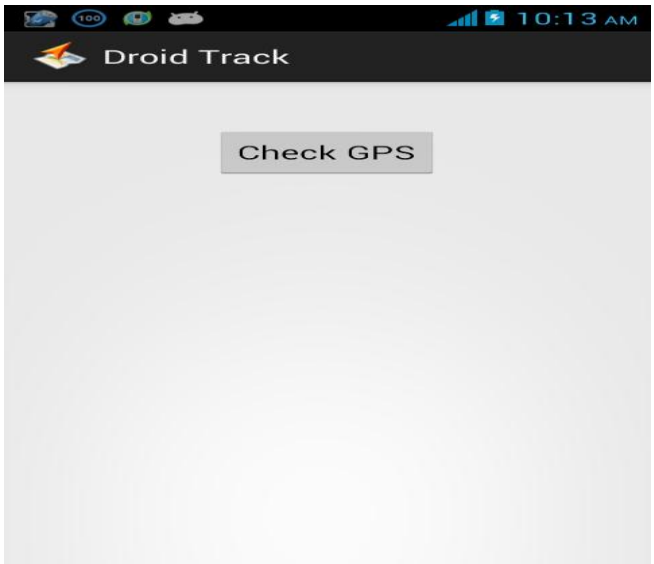


Figure (5.3) Application GPS interface

Then the application check your internet connection if you don't have it show interface like (Figure 5.4)

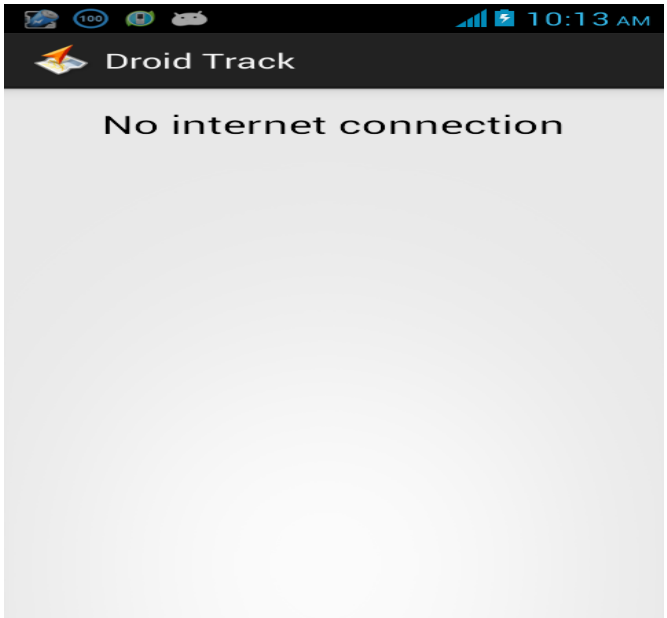


Figure (5.4) Application No Internet interface

If everything turn on you will go to the login interface.

5.2 Login:

The login page is the application's interface (Figure 5.5). Here you are able to fill your email and password and press the button login to log in the application. If you do not have an account in the system yet, you can press the button Register to go to the register interface.

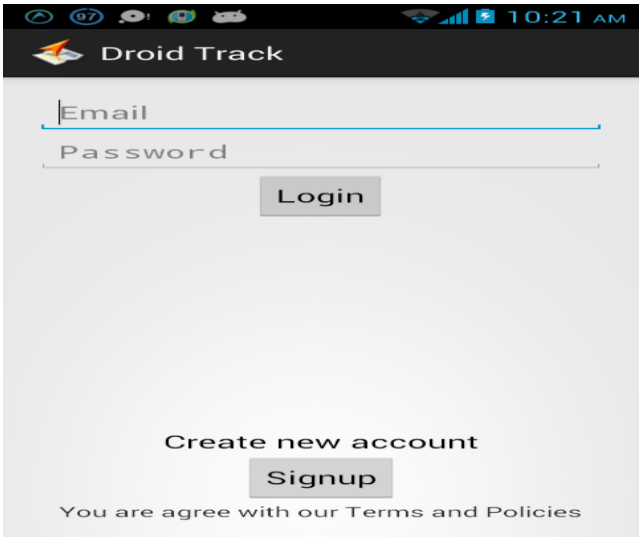


Figure (5.5) Application Login interface

5.3 Register a user:

This interface (Figure 5.6) allows you to register in the system. You do not have to fill all the fields.

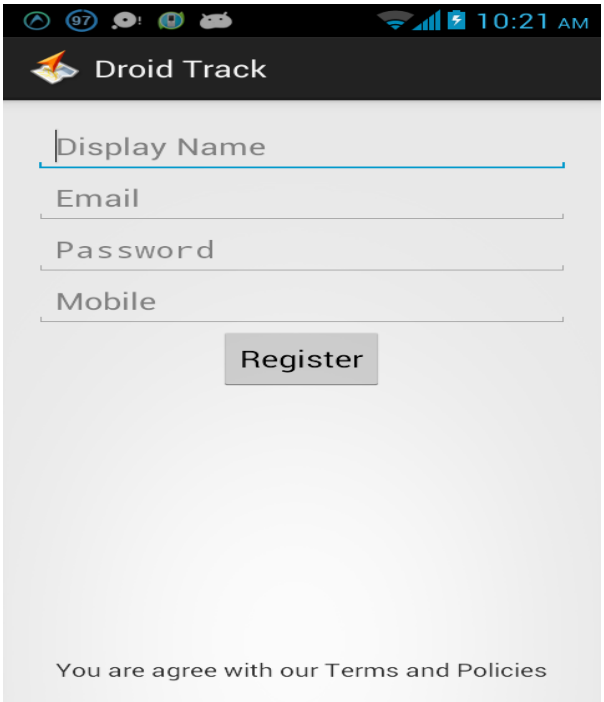


Figure (5.6) Application Register interface

When you are registered and log in you will go to groups interface.

5.4 Groups:

It's the interface which show you your groups and invitations (Figure 5.7)

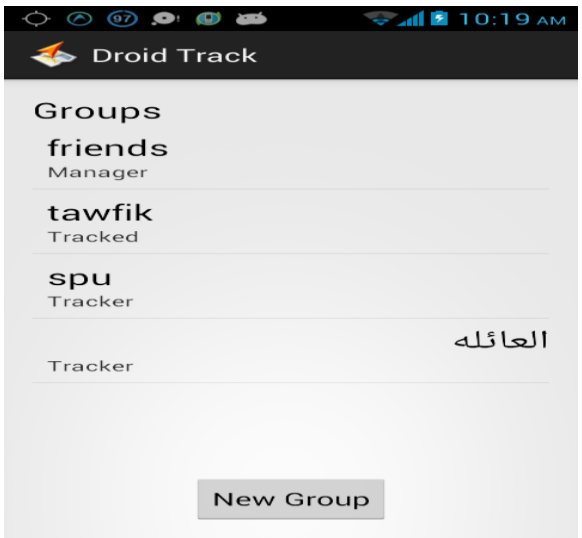


Figure (5.7) Application Groups interface

If you want to make a new group click in button new group and it will show you a dialog like (Figure 5.8)

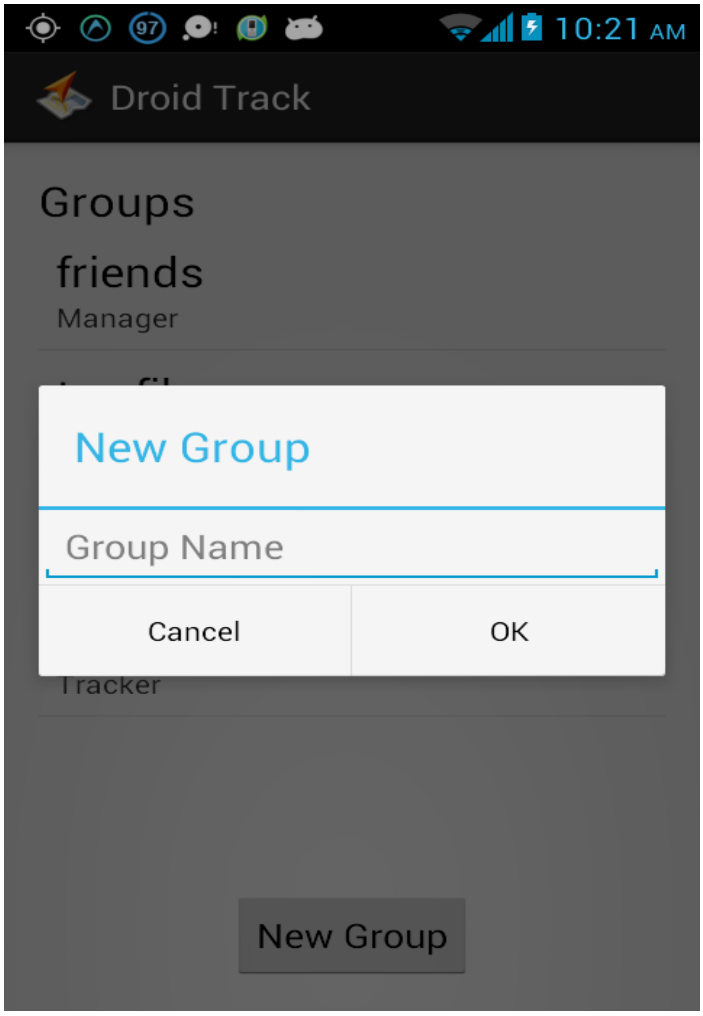


Figure (5.8) Application Add group dialog

Then when you chose a group and you are the manger or tracker you go to the map interface.

5.5 Map:

This interface show you your friends last locations (Figure 5.9) if you manger and (Figure 5.10) if you tracker.

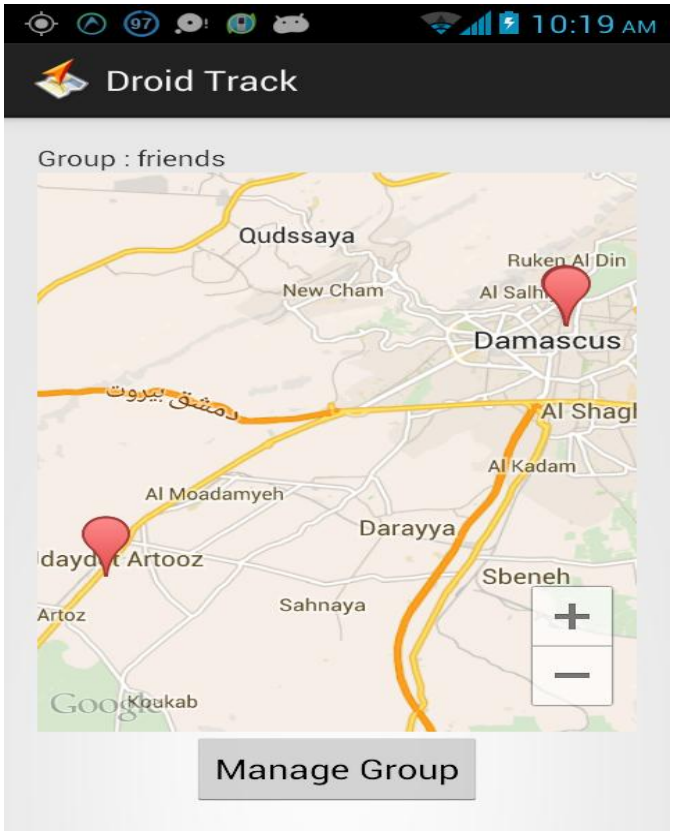


Figure (5.9) Application Map interface (manger)

Figure (5.10) Application Map interface (tracker)

If you are manager and click on manage group button you will go to the manage group interface.

5.6 Manage group:

(Figure 5.11)

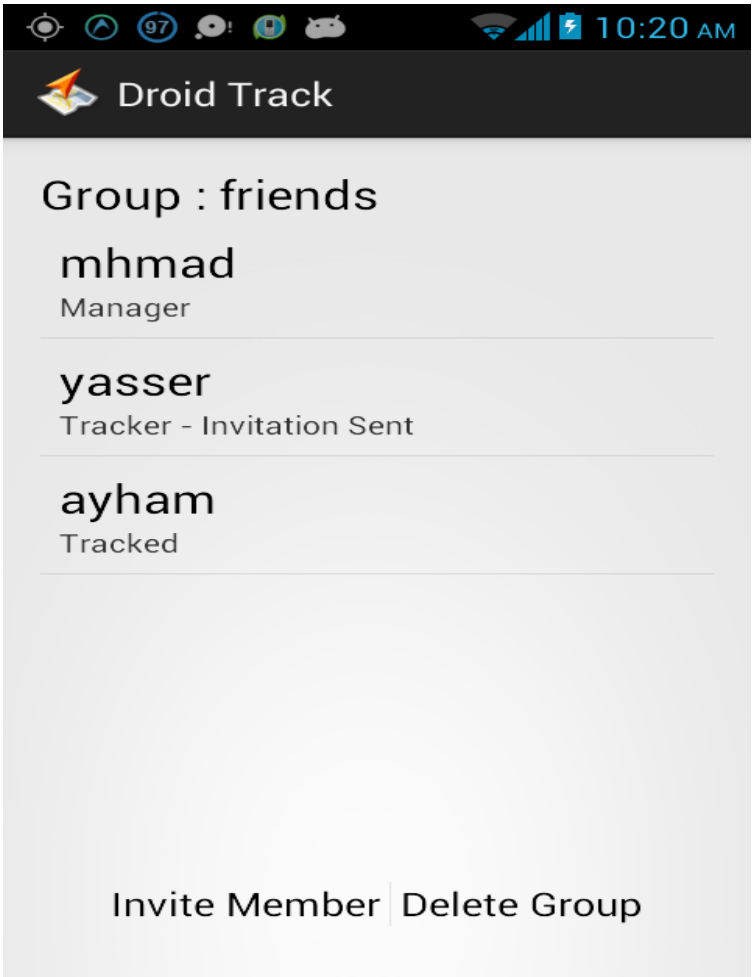


Figure (5.11) Application Group Manage interface

This interface allow the manger to invite a new friends to the group (Figure 5.12)

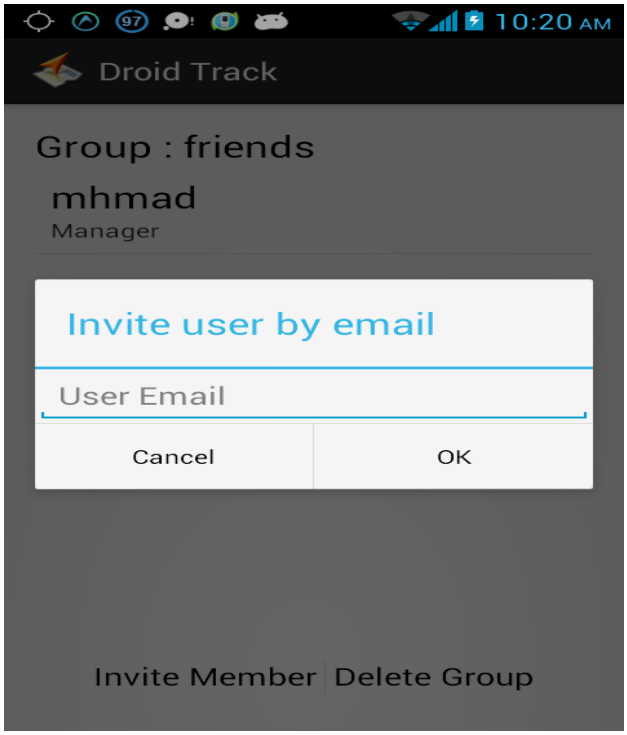


Figure (5.12) Application Invite Friend dailog

Alternatively, change a friend permission in the group (Figure 5.13)

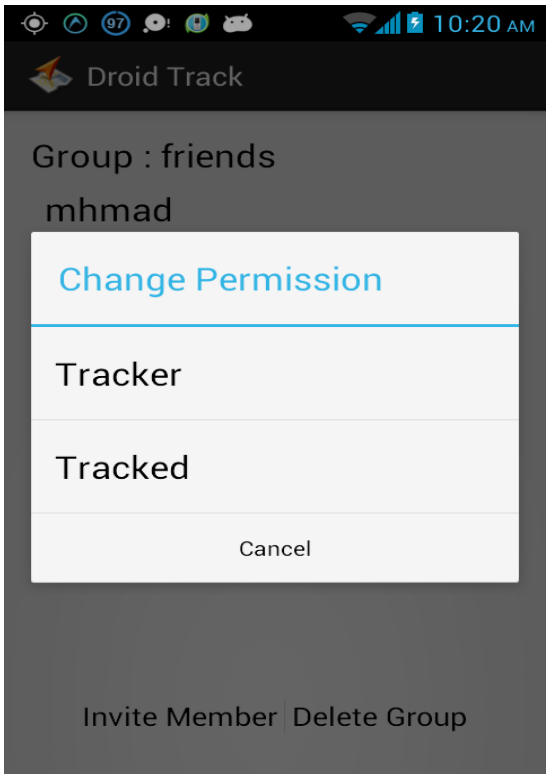


Figure (5.13) Application Change Primassion dialog

Alternatively, delete a friend from the group (Figure 5.14)

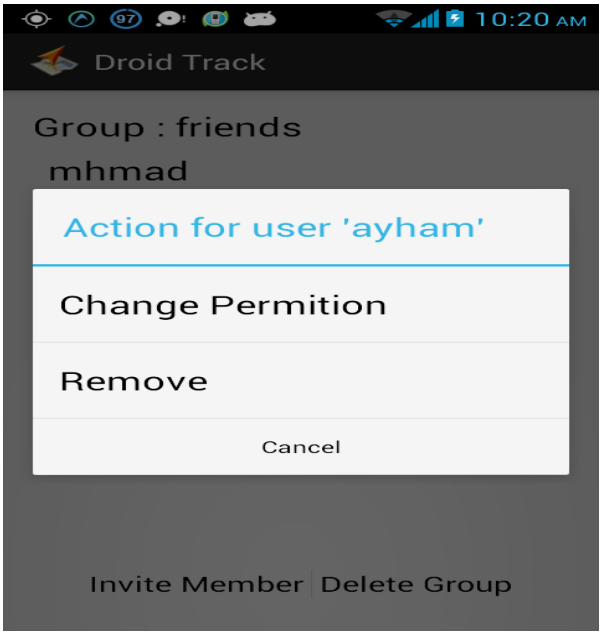


Figure (5.14) Application Delete Friend dialog

If you are manger or tracker when you click on a friend name, you go to the friend track interface.

5.7 Friend track:

This inter face show you the track of your friend with time (Figure 5.15)

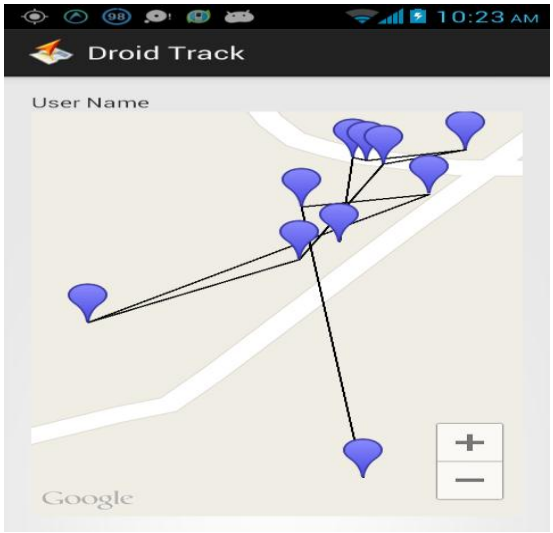


Figure (5.15) Application Map Track interface

If you are tracked when you click in a group the application show you a massage that you are tracked since date (Figure 5.16)

Figure (5.16) Application Map dialog (tracked)

And with that, we finished the work.

5.8 Conclusion and Future work:

As a result of technological progress, we are facing an incredible variety of possibilities to communicate regardless of the distance. Smartphones provide a great choice of features that facilitate the life for the users as well they make it more comfortable.

Every day the features and capabilities of Mobiles are increasing surprisingly. For this reason we want to create an Android application which gives us the opportunity to improve our knowledge of Mobile developing, and test all these features provides by Mobiles, due to they are turning into a vital part of human life.

We find this project a great opportunity to combine a lot of technologies and languages in the same software system, and learn how to work in big projects as a team member.

An important motivation for us is the challenge to solve all the problems that will arise while the project is developed. In addition, we will try to make the program portable, reliable, secure, stable, intuitive, etc.

Therefore, we make distributed system allow users to make groups from them friends by registering, login, and invite friends to the group by email, then tracking these friends' positions, and tracks with some real time techniques.

Finally as a future work, we look to update our project with some things like:

1. Make registering by activation code by SMS plane, and with phone number not email.
2. Make all the friends have phone number show as available choices to invite them to the groups.
3. Make the tracking time limited by user choice not from the time he join the group.
4. Mack the tracking action more real time tracking and more quality.
5. We can add a chat part and send photo and files to our application.

Finally, we would to thank all who contributed in the preparation of this research

Appendix

1. Server side:

1.1 Db_model: this class have the function to work with our sql database and store information in it or read information from it like that :

```
<?php

4- class Db_model extends CI_Model
{
5-    function GetUserGroups($UserID)
6-    {
7-          $this->db->select("Group.ID      as      'ID',
   Group.Name      as      'Name',      Group.CreatedBy,
   GroupMember.Permission,         GroupMember.Invitation,
   GroupMember.JoinDateTime");
8-          $this->db->where("UserID", $UserID);
9-          $this->db->where("GroupMember.Invitation   !=",
   "Rejected");
10-               $this->db->join("GroupMember",
   "GroupMember.GroupID = Group.ID");
11-               $this->db->order_by("ID", "asc");
12-               $q = $this->db->get("Group");
13-
14-               $Groups = array();
15-
16-               if($q->num_rows() > 0)
17-               {
18-                    foreach ($q->result() as $row)
19-                    {
20-                         $Groups[] = $row;
21-                    }
22-               }
23-
24-               return   array("Status"   =>   "Success",
   "Action"  =>  "GetUserGroups",  "UserID"  =>  $UserID,
   "Groups" => $Groups);
25-          }
26-
27-          function GetGroupMembers($GroupID)
28-          {
29-               $this->db->join("GroupMember",
   "GroupMember.UserID = User.ID");
30-               $this->db->where("GroupID", $GroupID);
31-               $this->db->select("UserID    as    'ID',
   User.Name,                     GroupMember.Permission,
   GroupMember.Invitation");
32-               $q = $this->db->get("User");
33-
34-               $Users = array();
35-
36-               if($q->num_rows() > 0)
37-               {
```

```
38-                     foreach ($q->result() as $row)
39-                     {
40-                          $Users[] = $row;
41-                     }
42-                }
43-
44-                return   array("Status"   =>   "Success",
   "Action" => "GetGroupMembers", "GroupID" => $GroupID,
   "Users" => $Users);
45-           }
46-
47-           function GetLastUserLocation($UserID)
48-           {
49-                //  SELECT  *  FROM  `UserLocation`  WHERE
   UserID = 1 ORDER BY `DateTime` DESC Limit 1
50-                $this->db->where("UserID", $UserID);
51-                $this->db->order_by("DateTime", "DESC");
52-                $this->db->limit(1);
53-                $q = $this->db->get("UserLocation");
54-
55-                $Location = null;
56-
57-                if($q->num_rows() > 0)
58-                {
59-                     foreach ($q->result() as $row)
60-                     {
61-                          $Location = $row;
62-                     }
63-                }
64-
65-                $User  =  $this->GetUser(array("UserID"  =>
   $UserID));
66-
67-                return   array("Status"   =>   "Success",
   "Action"    =>    "GetLastUserLocation",    "User"    =>
   $User["User"], "Location" => $Location);
68-           }
69-
70-           function GetUserLocations($UserID, $GroupID)
71-           {
72-                //   SELECT   UserLocation.ID   as   'ID',
   UserLocation.DateTime,             UserLocation.UserID,
   UserLocation.Lat, UserLocation.Lng
73-                // FROM `UserLocation`
74-                //       JOIN       `GroupMember`       ON
   `GroupMember`.`UserID` = `UserLocation`.`UserID`
75-                // WHERE `UserLocation`.`UserID` = 1
76-                // AND `GroupMember`.`GroupID` = 1
77-                //    AND    `UserLocation`.`DateTime`    >
   `GroupMember`.`JoinDateTime`
78-                // ORDER BY `DateTime` DESC
79-
80-                $this->db->join("GroupMember",
   "GroupMember.UserID = UserLocation.UserID");
81-                $this->db->where("UserLocation.UserID",
   $UserID);
```

```
82-             $this->db->where("GroupMember.GroupID",
   $GroupID);
83-             $this->db->where("UserLocation.DateTime >
   GroupMember.JoinDateTime");
84-             $this->db-
   >order_by("UserLocation.DateTime", "DESC");
85-             $this->db->select(array("UserLocation.ID
   as            'ID'",            "UserLocation.DateTime",
   "UserLocation.UserID",              "UserLocation.Lat",
   "UserLocation.Lng"));
86-             $q = $this->db->get("UserLocation");
87-
88-             $Locations = array();
89-
90-             if($q->num_rows() > 0)
91-             {
92-                  foreach ($q->result() as $row)
93-                  {
94-                       $Locations[] = $row;
95-                  }
96-             }
97-
98-             return   array("Status"   =>   "Success",
   "Action"  =>  "GetUserLocations",  "UserID"  =>  $UserID,
   "GroupID" => $GroupID, "Locations" => $Locations);
99-        }
100-
101-       function GetUser($data = null)
102-       {
103-             if(isset($data["UserID"]))
104-             {
105-                  $this->db->where("User.ID",
   $data["UserID"]);
106-             }
107-             elseif(isset($data["Email"]))
108-             {
109-                  $this->db->where("Email",
   $data["Email"]);
110-             }
111-             else
112-             {
113-                  return  array("Status"  =>  "Failed",
   "Action"  =>  "GetUser",  "Message"  =>  "No  UserID  or
   Email sended");
114-             }
115-
116-             $this->db->select(array("ID",     "Name",
   "Email", "Mobile"));
117-             $q = $this->db->get("User");
118-
119-             if($q->num_rows() > 0)
120-             {
121-                  foreach ($q->result() as $row)
122-                  {
123-                       $User = $row;
124-                  }
```

```
125-
126-                    return array("Status" => "Success",
   "Action" => "GetUser", "User" => $User);
127-             }
128-             return   array("Status"   =>   "Failed",
   "Action" => "GetUser", "Message" => "User not found");
129-       }
130-
131-       function Login($data)
132-       {
133-             if (empty($data))
134-             {
135-                    return array("Status" => "Failed",
   "Action" => "Login", "Message" => "Empty request");
136-             }
137-
138-             $this->db->where(array("Email"         =>
   $data["Email"], "Password" => $data["Password"]));
139-             $q = $this->db->get("User");
140-
141-             if($q->num_rows() > 0)
142-             {
143-                    foreach ($q->result() as $row)
144-                    {
145-                           $UserID = $row->ID;
146-                    }
147-                    $User            =            $this-
   >GetUser(array("UserID" => $UserID));
148-                    return array("Status" => "Success",
   "Action" => "Login", "User" => $User["User"]);
149-             }
150-             else
151-             {
152-                    return array("Status" => "Failed",
   "Action" => "Login", "Message" => "Email or password
   is wrong");
153-             }
154-       }
155-
156-       function Register($data)
157-       {
158-             $this->db->where("Email",
   $data["Email"]);
159-             $q = $this->db->get("User");
160-
161-             if($q->num_rows() > 0)
162-             {
163-                    $r = $q->result();
164-                    return array("Status" => "Failed",
   "Action" => "Register", "Message" => "Email already
   used");
165-             }
166-             else
167-             {
168-                    $this->db->insert("User", $data);
169-
```

```
170-                    $User            =            $this-
   >GetUser(array("UserID" => $this->db->insert_id()));
171-                    return array("Status" => "Success",
   "Action"  =>  "Register",  "User"  =>  $User["User"],
   "Activation" => Null);
172-             }
173-        }
174-
175-        function GroupInvitation($data)
176-        {
177-             if (empty($data))
178-             {
179-                  return array("Status" => "Failed",
   "Action" => "GroupInvitation", "Message" => "Empty
   request");
180-             }
181-
182-             $this->db->where("UserID",
   $data["UserID"]);
183-             $this->db->where("GroupID",
   $data["GroupID"]);
184-
185-             $record = array();
186-             $record["Invitation"]                    =
   $data["Invitation"];
187-
188-             if($data["Invitation"] == "Accepted")
189-             {
190-                  $record["JoinDateTime"]  =  date("Y-
   m-d H:i:s", $_SERVER['REQUEST_TIME']);
191-             }
192-
193-             $this->db->update("GroupMember",
   $record);
194-             return   array("Status"   =>   "Success",
   "Action"    =>    "GroupInvitation",    "UserID"    =>
   $data["UserID"],    "GroupID"    =>    $data["GroupID"],
   "Invitation" => $data["Invitation"]);
195-        }
196-
197-        function SetUserLocation($data)
198-        {
199-             if (empty($data))
200-             {
201-                  return array("Status" => "Failed",
   "Action" => "SetUserLocation", "Message" => "Empty
   request");
202-             }
203-
204-             $this->db->insert("UserLocation",
   array("UserID"    =>    $data["UserID"],    "Lat"    =>
   $data["Lat"], "Lng" => $data["Lng"]));
205-
206-             return   array("Status"   =>   "Success",
   "Action" => "SetUserLocation");
207-        }
```

```
208-
209-        function CreateGroup($data)
210-        {
211-              if (empty($data))
212-              {
213-                    return array("Status" => "Failed",
   "Action" => "SetUserLocation", "Message" => "Empty
   request");
214-              }
215-
216-              $this->db->insert("Group",
   array("CreatedBy" => $data["UserID"], "Name" =>
   $data["GroupName"]));
217-              $this->db->insert(
218-                    "GroupMember", array(
219-                          "UserID" => $data["UserID"],
220-                          "GroupID" => $this->db-
   >insert_id(),
221-                          "Permission" => "Manager",
222-                          "Invitation" => "Accepted",
223-                          "JoinDateTime" => date("Y-m-d
   H:i:s", $_SERVER['REQUEST_TIME'])
224-                    )
225-              );
226-
227-              return array("Status" => "Success",
   "Action" => "CreateGroup");
228-        }
229-
230-        function InviteToGroup($data)
231-        {
232-              if (empty($data))
233-              {
234-                    return array("Status" => "Failed",
   "Action" => "InviteToGroup", "Message" => "Empty
   request");
235-              }
236-
237-              $User = $this->GetUser(array("Email" =>
   $data["Email"]));
238-
239-              if($User["Status"] == "Success")
240-              {
241-                    $this->db->where("UserID",
   $User["User"]->ID);
242-                    $this->db->where("GroupID",
   $data["GroupID"]);
243-                    $q = $this->db->get("GroupMember");
244-
245-                    if($q->num_rows() > 0)
246-                    {
247-                          return array("Status" =>
   "Failed", "Action" => "InviteToGroup", "Message" =>
   "User already exist in group");
248-                    }
249-                    else
```

```
250-                 {
251-                      $this->db->insert(
252-                           "GroupMember", array(
253-                                "UserID"         =>
   $User["User"]->ID,
254-                                "GroupID"        =>
   $data["GroupID"],
255-                                "Permission"     =>
   "Tracked",
256-                                "Invitation"     =>
   "Sent"
257-                           )
258-                      );
259-                      return    array("Status"    =>
   "Success", "Action" => "InviteToGroup", "Message" =>
   "Invitation sent to " . $data["Email"]);
260-                 }
261-            }
262-            else
263-            {
264-                 return array("Status" => "Failed",
   "Action" => "InviteToGroup", "Message" => "User not
   registered yet!");
265-            }
266-       }
267-
268-       function DeleteGroup($data)
269-       {
270-            if (empty($data))
271-            {
272-                 return array("Status" => "Failed",
   "Action"   =>   "DeleteGroup",   "Message"   =>   "Empty
   request");
273-            }
274-
275-            $this->db->where("GroupID",
   $data["GroupID"]);
276-            $this->db->delete("GroupMember");
277-
278-            $this->db->where("ID", $data["GroupID"]);
279-            $this->db->delete("Group");
280-
281-            return   array("Status"   =>   "Success",
   "Action"   =>   "DeleteGroup",   "Message"   =>   "Group
   Deleted");
282-       }
283-
284-       function UpdateUserPermission($data)
285-       {
286-            if (empty($data))
287-            {
288-                 return array("Status" => "Failed",
   "Action"   =>   "UpdateUserPermission",   "Message"   =>
   "Empty request");
289-            }
290-
```

```
291-            $this->db->where("UserID",
   $data["UserID"]);
292-            $this->db->where("GroupID",
   $data["GroupID"]);
293-
294-            $record = array();
295-            $record["Permission"]                    =
   $data["Permission"];
296-
297-            $this->db->update("GroupMember",
   $record);
298-            return   array("Status"   =>   "Success",
   "Action" => "UpdateUserPermission", "Message" => "User
   permission updated");
299-       }
300-
301-       function RemoveUserFromGroup($data)
302-       {
303-            if (empty($data))
304-            {
305-                 return array("Status" => "Failed",
   "Action" => "RemoveUserFromGroup", "Message" => "Empty
   request");
306-            }
307-
308-            $this->db->where("UserID",
   $data["UserID"]);
309-            $this->db->where("GroupID",
   $data["GroupID"]);
310-            $this->db->delete("GroupMember");
311-
312-            return   array("Status"   =>   "Success",
   "Action" => "RemoveUserFromGroup", "Message" => "User
   removed");
313-       }
314- }
```

1.2 V1: This class is a sub class from reset controller and have the functions to work with data in our service and functions to the other actions we want to do and it like that:

```
<?php if ( ! defined('BASEPATH')) exit('No direct script
access allowed');

require(APPPATH.'/libraries/REST_Controller.php');

class V1 extends REST_Controller
{
     public function index_get()
     {
          $this->response(array("Message" => "Not Allowed"),
403);
     }

     public function user_get()
```

```
	{
		$this->response($this->db_model->GetUser($this-
>get()), 200);
	}

	public function user_groups_get()
	{
		$this->response($this->db_model-
>GetUserGroups($this->get("UserID")), 200);
	}

	public function group_members_get()
	{
		$this->response($this->db_model-
>GetGroupMembers($this->get("GroupID")), 200);
	}

	public function user_last_location_get()
	{
		$this->response($this->db_model-
>GetLastUserLocation($this->get("UserID")), 200);
	}

	public function user_locations_get()
	{
		$this->response($this->db_model-
>GetUserLocations($this->get("UserID"),	$this-
>get("GroupID")), 200);
	}

	public function server_time_get()
	{
		$this->response(date("Y-m-d	H:i:s",
$_SERVER['REQUEST_TIME']), 200);
	}

	public function login_post()
	{
		$this->response($this->db_model->Login($this-
>post()), 200);
	}

	public function register_post()
	{
		$fields = array(
			"Name" => $this->post("Name"),
			"Email" => $this->post("Email"),
			"Password" => $this->post("Password"),
			"Mobile" => $this->post("Mobile")
		);

		$this->response($this->db_model->Register($fields),
200);
	}

	public function group_invitation_post()
```

```
	{
		$this->response($this->db_model-
>GroupInvitation($this->post()), 200);
	}

	public function create_group_post()
	{
		$this->response($this->db_model->CreateGroup($this-
>post()), 200);
	}

	public function invite_to_group_post()
	{
		$this->response($this->db_model-
>InviteToGroup($this->post()), 200);
	}

	public function delete_group_post()
	{
		$this->response($this->db_model->DeleteGroup($this-
>post()), 200);
	}

	public function user_location_post()
	{
		$this->response($this->db_model-
>SetUserLocation($this->post()), 200);
	}

	public function user_permission_post()
	{
		$this->response($this->db_model-
>UpdateUserPermission($this->post()), 200);
	}

	public function remove_user_from_group_post()
	{
		$this->response($this->db_model-
>RemoveUserFromGroup($this->post()), 200);
	}
}
```

1.3 ROUT: this class have the function to which action our service will begin first and the home page in the controller.

```
<?php	if	(	!	defined('BASEPATH'))	exit('No	direct	script
access allowed');
/*
|	-------------------------------------------------------------
-------------
| URI ROUTING
```

```
|  -------------------------------------------------------------
-------------
|  This file lets you re-map URI requests to specific
controller functions.
|
| Typically there is a one-to-one relationship between a URL
string
| and its corresponding controller class/method. The segments
in a
| URL normally follow this pattern:
|
|    example.com/class/method/id/
|
|  In some instances, however, you may want to remap this
relationship
| so that a different class/function is called than the one
| corresponding to the URL.
|
| Please see the user guide for complete details:
|
|    http://codeigniter.com/user_guide/general/routing.html
|
|  -------------------------------------------------------------
-------------
| RESERVED ROUTES
|  -------------------------------------------------------------
-------------
|
| There area two reserved routes:
|
|    $route['default_controller'] = 'welcome';
|
| This route indicates which controller class should be loaded
if the
| URI contains no data. In the above example, the "welcome"
class
| would be loaded.
|
|    $route['404_override'] = 'errors/page_missing';
|
| This route will tell the Router what URI segments to use if
those provided
| in the URL cannot be matched to a valid route.
|
*/

$route['default_controller'] = "home";
$route['404_override'] = '';


/* End of file routes.php */
/* Location: ./application/config/routes.php */
```

2 Client side

2.1 user: this class to define the user type to takes the user's object from it:

```
package tk.droidtrack.app;

import java.io.Serializable;

public class User implements Serializable
{
      private static final long serialVersionUID = 1;

      public String ID;
      public String Name;
      public String Email;
      public String Mobile;
      public String Permission;
      public String Invitation;
}
```

2.2 group: this class to define the group type to takes the group's object from it:

```
package tk.droidtrack.app;

import java.io.Serializable;

public class Group implements Serializable
{
      private static final long serialVersionUID = 1;
      public String ID;
      public String Name;
      public String CreatedBy;
      public String Permission;
      public String Invitation;
      public String JoinDateTime;
}
```

2.3 Preferences: this class to make the login automatic after the first time and when we do not log out, and it's refer to a memory location when we store this data:

```
package tk.droidtrack.app;

public class Preferences
{
```

```
	public	static	final	String	Base_API_URI	=
"http://api.droidtrack.tk/v1";
	public	static	String	API_URI	=
"http://api.droidtrack.tk/v1";
	public static String UserID;
	public static Group[] userGroups;
}
```

2.4 My Location: this class to define the location attributes as Google planning to take objects from it:

```
package tk.droidtrack.app;

import com.google.android.gms.maps.model.LatLng;

public class MyLocation
{
	public String ID;
	public String DateTime;
	public String UserID;
	public LatLng LatLng;
}
```

2.5GPS Helper: this class define the work with the GPS device in the phone and check here state:

```
package tk.droidtrack.app;

import com.google.android.gms.maps.model.LatLng;

import android.app.AlertDialog;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
import android.os.Bundle;
import android.util.Log;

public class GPSHelper
{
	Context context;
	LocationManager locationManager;
	onLocationFetchedListener locationListener;

	LocationListener listener = new LocationListener()
	{
		@Override
		public void onStatusChanged(String arg0, int arg1,
Bundle arg2)
		{
		}
```

```
            @Override
            public void onProviderEnabled(String arg0)
            {
            }

            @Override
            public void onProviderDisabled(String arg0)
            {
            }

            @Override
            public void onLocationChanged(Location location)
            {
                  MyLocation l = new MyLocation();
                  l.LatLng  =  new  LatLng(location.getLatitude(),
location.getLongitude());

                  locationListener.onLocationFetched(l);
                  destroyLocManager();
            }
      };

      public              GPSHelper(Context              context,
onLocationFetchedListener locationListener)
      {
            this.context = context;
            this.locationListener = locationListener;
      }

      @SuppressWarnings("static-access")
      private void initLocManager()
      {
            locationManager          =          (LocationManager)
context.getSystemService(context.LOCATION_SERVICE);
      }

      private void destroyLocManager()
      {
            initLocManager();
            locationManager.removeUpdates(listener);
      }

      public void GetLocation()
      {
            initLocManager();

      locationManager.requestLocationUpdates(LocationManager.GP
S_PROVIDER, 3000, 10, listener);
      }

      public void StopGPSAccess()
      {
            destroyLocManager();
      }

      public Boolean ChechGPS()
```

```
	{
		initLocManager();
		if
(locationManager.isProviderEnabled(LocationManager.GPS_PROVIDE
R))
		{
			Log.i("GetLocation", "GPS is enabled");
			return true;
		}
		else
		{
			Log.i("GetLocation", "GPS is disabled");
		 showGPSDisabledAlertToUser();
		return false;
		}
	}

	public void showGPSDisabledAlertToUser()
	{
		AlertDialog.Builder	alertDialogBuilder	=	new
AlertDialog.Builder(context);
		alertDialogBuilder.setMessage("GPS	is	disabled	in
your device. You should enable GPS")
		.setCancelable(false)
		.setPositiveButton("Enable GPS",
			new DialogInterface.OnClickListener()
			{
				public	void	onClick(DialogInterface
dialog, int id)
				{
					showGPSSetting();
				}
			})
		.setNegativeButton("Cansel",
			new DialogInterface.OnClickListener()
			{
				public	void	onClick(DialogInterface
dialog, int id)
				{
				dialog.dismiss();
				}
			});
		AlertDialog alert = alertDialogBuilder.create();
		alert.show();
	}

	public void showGPSSetting()
	{
		Intent	callGPSSettingIntent	=	new
Intent(android.provider.Settings.ACTION_LOCATION_SOURCE_SETTIN
GS);
		context.startActivity(callGPSSettingIntent);
	}
}
```

2.6 API: This class define to control the whole application and the way of make communication with server:

```
package tk.droidtrack.app;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.util.List;

import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.NameValuePair;
import org.apache.http.client.ClientProtocolException;
import org.apache.http.client.HttpClient;
import org.apache.http.client.entity.UrlEncodedFormEntity;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.protocol.BasicHttpContext;
import org.apache.http.protocol.HTTP;
import org.apache.http.protocol.HttpContext;
import org.json.JSONException;
import org.json.JSONObject;

import android.content.Context;
import android.net.ConnectivityManager;
import android.os.AsyncTask;
import android.util.Log;

public class API extends AsyncTask<Object, Void, JSONObject>
{
	String uri;

	IAsyncFetchListener fetchListener = null;
	Context context;

	public API(Context context)
	{
		this.context = context;

		uri = Preferences.API_URI;
	}

	public void GET(String request)
	{
		this.execute("GET", request);
	}

	public void POST(String request, List<NameValuePair>
list)
	{
		this.execute("POST", request, list);
```

```
      }

    public void setListener(IAsyncFetchListener listener)
    {
        this.fetchListener = listener;
    }

      @Override
      protected JSONObject doInBackground(Object... args)
      {
            Log.i("args", args.toString());
            String request = uri + (String)args[1];
            String response = null;

            if(CheckInternetConnection(context))
            {
                  if(((String)(args[0])).compareTo("GET") == 0)
                  {
                        try
                        {
                              Log.i("HTTP_GET_URI", request);
                              response = APIGET(request);
                              Log.i("HTTP_GET_RESPONSE",
response);
                        }
                        catch (Exception e)
                        {
                              Log.e("Exception", e.getMessage());
                              fetchListener.onError(e);
                        }
                  }
                  else
                  {
                        try
                        {
                              @SuppressWarnings("unchecked")
                              List<NameValuePair>      list      =
(List<NameValuePair>)args[2];
                              Log.i("HTTP_POST_URI", request);
                              Log.i("HTTP_POST_PAYLOAD",
list.toString());
                              response = APIPOST(request, list);
                              Log.i("HTTP_POST_RESPONSE",
response);
                        }
                        catch (Exception e)
                        {
                              Log.e("Exception", e.getMessage());
                              fetchListener.onError(e);
                        }
                  }
            }
            else
            {
                  try
                  {
```

```
                    JSONObject json = new JSONObject();
                    json.put("Status", "Failed");
                    json.put("Internet", false);
                    json.put("Message",     "No     Internet
Connection");
                    return json;
               }
               catch (JSONException e)
               {
                    Log.e("Exception", e.getMessage());
                    fetchListener.onError(e);
               }
          }

          try
          {
               return new JSONObject(response);
          }
          catch (JSONException e)
          {
               Log.e("Exception", e.getMessage());
               fetchListener.onError(e);
          }
          return null;
     }

     @Override
     protected void onPostExecute(JSONObject result)
     {
          try
          {
               this.fetchListener.onComplete(result);
          }
          catch (Exception e)
          {
               Log.e("Exception", e.getMessage());
          }
     }

     private     String     APIGET(String     s1)     throws
ClientProtocolException, IOException
     {
          String result = null;
          try
          {
               HttpClient       httpclient       =       new
DefaultHttpClient(); // Create HTTP Client
               HttpGet httpget = new HttpGet(s1); // Set the
action you want to do
               HttpResponse            response            =
httpclient.execute(httpget); // Executeit
               HttpEntity entity = response.getEntity();
               InputStream   is   =   entity.getContent();   //
Create an InputStream with the response
               BufferedReader reader = new BufferedReader(new
InputStreamReader(is, "utf8"), 8);
```

```
                StringBuilder sb = new StringBuilder();
                String line = null;
                while ((line = reader.readLine()) != null) //
Read line by line
                      sb.append(line + "\n");

                result = sb.toString(); // Result is here

                is.close(); // Close the stream
           }
           catch (Exception e)
           {
                Log.e("Exception", e.getMessage());
                fetchListener.onError(e);
           }
           return result;
      }

      private String APIPOST(String uri, List<NameValuePair>
list) throws ClientProtocolException, IOException
      {
           String result = null;
           try
           {
                HttpClient       httpClient       =       new
DefaultHttpClient();
                HttpContext      localContext      =      new
BasicHttpContext();
                HttpPost request = new HttpPost(uri); // Set
Method
                request.setEntity(new
UrlEncodedFormEntity(list,HTTP.UTF_8));
                HttpResponse            response            =
httpClient.execute(request, localContext);
                BufferedReader reader = new BufferedReader(new
InputStreamReader( response.getEntity().getContent()));
                StringBuilder sb = new StringBuilder();
                String line = null;
                while ((line = reader.readLine()) != null) //
Read line by line
                      sb.append(line + "\n");

                result = sb.toString(); // Result is here
           }
           catch (Exception e)
           {
                Log.e("Exception", e.getMessage());
                fetchListener.onError(e);
           }

           Log.i("API_POST_ROW_RESPONSE", result);
           return result;
      }

      public static boolean CheckInternetConnection(Context
context)
```

```
	{
		ConnectivityManager conMgr = (ConnectivityManager)
context.getSystemService(Context.CONNECTIVITY_SERVICE);
		if (conMgr.getActiveNetworkInfo() != null
				&&
conMgr.getActiveNetworkInfo().isAvailable()
				&&
conMgr.getActiveNetworkInfo().isConnected())
		{
			return true;
		}
		else
		{
			return false;
		}
	}
}
```

2.7 onLocationFetchedListener interface:

```
package tk.droidtrack.app;

public interface onLocationFetchedListener
{
	public void onLocationFetched(MyLocation location);
}
```

2.8 IAsyncFetchListener interface:

```
package tk.droidtrack.app;

import java.util.EventListener;

import org.json.JSONObject;

public interface IAsyncFetchListener extends EventListener
{
	void onComplete(JSONObject result);
	void onError(Throwable error);
}
```

2.9 LocationUpdaterService: this class define the service of our application, which work in background and send the user location data to the server every 60 second:

```
package tk.droidtrack.app;

import java.util.ArrayList;
import java.util.List;
import java.util.Timer;
```

```
import java.util.TimerTask;

import org.apache.http.NameValuePair;
import org.apache.http.message.BasicNameValuePair;
import org.json.JSONObject;

import android.app.Service;
import android.content.Intent;
import android.os.Handler;
import android.os.IBinder;
import android.util.Log;

public class LocationUpdaterService extends Service
{
	Timer timer;
	GPSHelper helper;
	Intent intent;

	@Override
	public IBinder onBind(Intent intent)
	{
		return null;
	}

	@Override
	public void onCreate()
	{
		super.onCreate();
		Log.i("onCreate", "Start");

		helper	=	new	GPSHelper(this,	new
onLocationFetchedListener()
		{
			@Override
			public	void	onLocationFetched(MyLocation
location)
			{
				SendLocation(location);
			}
		});

		timer = new Timer();
	}

	@Override
	public int onStartCommand(Intent intent, int flags, int
startId)
	{
		Log.i("onStartCommand", "Start");

		int userGroups_length;
		try
		{
			userGroups_length	=
Preferences.userGroups.length;
		}
```

```
        catch (Exception e)
        {
                userGroups_length = 0;
        }

        if(userGroups_length > 0)
        {
                final Handler handler = new Handler();
                TimerTask task = new TimerTask()
                {
                        @Override
                        public void run()
                        {
                                handler.post(new Runnable()
                                {
                                        public void run()
                                        {
                                                Log.i("Timer", "Tick");
                                                try
                                                {

helper.GetLocation();
                                                }
                                                catch (Exception e)
                                                {

helper.StopGPSAccess();
                                                }
                                        }
                                });
                        }
                };

                timer.scheduleAtFixedRate(task, 0, 60000);
        }

        return super.onStartCommand(intent, flags, startId);
}

@Override
public void onDestroy()
{
        Log.i("onDestroy", "Start");

        timer.cancel();
        timer.purge();

        helper.StopGPSAccess();

        this.stopSelf();
        super.onDestroy();
}

private void SendLocation(MyLocation location)
{
```

```
            Log.i("GPSLatLng", location.LatLng.latitude + "," +
location.LatLng.longitude);
            Log.i("SERVICE_UserID", Preferences.UserID);

            List<NameValuePair>        list        =        new
ArrayList<NameValuePair>();

            list.add(new            BasicNameValuePair("UserID",
Preferences.UserID));
            list.add(new               BasicNameValuePair("Lat",
location.LatLng.latitude + ""));
            list.add(new               BasicNameValuePair("Lng",
location.LatLng.longitude + ""));

            Log.i("List", list.toString());
            API_POST("/user_location", list);
      }

    private   void   API_POST(String   arg,   List<NameValuePair>
list)
    {
        API api = new API(this);

        Log.i("API_Obj", "new API(this)");

        try
           {
                 api.setListener(new IAsyncFetchListener()
                 {
                       @Override
                       public void onError(Throwable error)
                       {
                             Log.e("Error", error.getMessage());
                       }

                       @Override
                       public void onComplete(JSONObject result)
                       {
                             Log.i("Complete",
result.names().toString());
                       Log.i("JSON_Result", result.toString());
                       }
                 });
           }
           catch (Exception e)
           {
                 Log.e("Exception", e.getMessage());
           }

        api.POST(arg, list);
      }
}
```

2.10 Main: This Activity allows users to login. It takes the Email and password from the user and sends them to the server:

```
package tk.droidtrack.app;

import java.util.ArrayList;
import java.util.List;

import org.apache.http.NameValuePair;
import org.apache.http.message.BasicNameValuePair;
import org.json.JSONObject;

import android.app.Activity;
import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

public class MainActivity extends Activity
{
      String Email;
      String Password;

      View contentLayout;
      View turnGPSLayout;

      EditText emailText;
      EditText passwordText;
      Button loginButton;
      Button signupButton;
      Button checkGPSButton;

      GPSHelper helper;

      @Override
      protected void onCreate(Bundle savedInstanceState)
      {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.main_layout);

            contentLayout                                        =
findViewById(R.id.main_contentLayout);
            turnGPSLayout                                        =
findViewById(R.id.main_turnGPSLayout);

            emailText                =                 (EditText)
findViewById(R.id.main_emailText);
```

```
            passwordText                =                (EditText)
findViewById(R.id.main_passwordText);
            loginButton                 =                  (Button)
findViewById(R.id.main_loginButton);
            signupButton                =                  (Button)
findViewById(R.id.main_signupButton);
            checkGPSButton              =                  (Button)
findViewById(R.id.main_checkGPSButton);

            loginButton.setOnClickListener(new OnClickListener()
            {
                  @Override
                  public void onClick(View arg0)
                  {
                        Email = emailText.getText().toString();
                        Log.i("Email", Email);
                        Password                                =
passwordText.getText().toString();
                        Log.i("Password", Password);

                        if          (Email.matches("")          ||
Password.matches(""))
                        {
                              Toast.makeText(MainActivity.this,
"Enter Email and Password", Toast.LENGTH_SHORT).show();
                        }
                        else
                        {
                              Login();
                        }
                  }
            });

            signupButton.setOnClickListener(new
OnClickListener()
            {
                  @Override
                  public void onClick(View arg0)
                  {
                        Log.i("signupButton", "clicked");
                        Intent        intent        =        new
Intent(MainActivity.this, RegisterActivity.class);
                        startActivity(intent);
                        MainActivity.this.finish();
                  }
            });

            checkGPSButton.setOnClickListener(new
OnClickListener()
            {
                  @Override
                  public void onClick(View arg0)
                  {
                        if(helper.ChechGPS())
                        {
                              startApp();
```

```
                        }
                }
            });

            helper = new GPSHelper(this, null);

            if(helper.ChechGPS())
            {
                startApp();
            }
            else
            {
                turnGPSLayout.setVisibility(View.VISIBLE);
            }
        }

        private void startApp()
        {
            turnGPSLayout.setVisibility(View.GONE);

            if(API.CheckInternetConnection(this))
            {
                SharedPreferences          settings          =
getSharedPreferences("DroidTrack", 0);
                if(settings.getBoolean("LoggedIn", false))
                {

        emailText.setText(settings.getString("UserEmail", ""));

        passwordText.setText(settings.getString("UserPassword",
""));

                        Preferences.API_URI                    =
settings.getString("API_URI", Preferences.Base_API_URI);
                        Login();
                }
                else
                {

        contentLayout.setVisibility(View.VISIBLE);
                }
            }
            else
            {

        ((TextView)findViewById(R.id.main_waiting)).setText("No
internet connection");

        findViewById(R.id.progressBar1).setVisibility(View.GONE);
            }
        }

        private void Login()
        {
            Email = emailText.getText().toString();
            Password = passwordText.getText().toString();
```

```
            List<NameValuePair>        list        =        new
ArrayList<NameValuePair>();

            list.add(new BasicNameValuePair("Email", Email));
            list.add(new          BasicNameValuePair("Password",
Password));

            Log.i("List", list.toString());

            API_POST("/login", list);
      }

    private   void   API_POST(String   arg,   List<NameValuePair>
list)
    {
        API api = new API(this);

        Log.i("API_Obj", "new API(this)");

        try
           {
                 api.setListener(new IAsyncFetchListener()
                 {
                       @Override
                       public void onError(Throwable error)
                       {
                             Log.e("Error", error.getMessage());
                       }

                       @Override
                       public void onComplete(JSONObject result)
                       {
                             Log.i("Complete",
result.names().toString());
                             ParseJson(result);
                       }
                 });
           }
           catch (Exception e)
           {
                 Log.e("Exception", e.getMessage());
           }

        api.POST(arg, list);
      }

    private void ParseJson(JSONObject json)
    {
      Log.i("JSON_Result", json.toString());
      try
      {
            if(json.getString("Status").compareTo("Success")  ==
0)
            {
                  User user = new User();
```

```
                user.ID                                         =
json.getJSONObject("User").getString("ID");
                user.Name                                       =
json.getJSONObject("User").getString("Name");
                user.Email                                      =
json.getJSONObject("User").getString("Email");
                user.Mobile                                     =
json.getJSONObject("User").getString("Mobile");

                Preferences.UserID = user.ID;

                SharedPreferences          settings          =
getSharedPreferences("DroidTrack", 0);
                SharedPreferences.Editor        editor        =
settings.edit();

                editor.putBoolean("LoggedIn", true);
                editor.putString("UserID", user.ID);
                editor.putString("UserEmail", user.Email);
                editor.putString("UserPassword", Password);
                editor.putString("UserName", user.Name);
                editor.putString("UserMobile", user.Mobile);

                editor.commit();

                Intent    intent    =    new    Intent(this,
GroupsActivity.class);
                intent.putExtra("User", user);
                startActivity(intent);

                Intent    intent2    =    new    Intent(this,
LocationUpdaterService.class);
               startService(intent2);

               this.finish();
           }
           else
           {
                if(!json.has("Internet"))
                {
                      emailText.setError(null);
                      passwordText.setError(null);
                }
                Toast.makeText(this,
json.getString("Message"), Toast.LENGTH_LONG).show();

                if(json.getString("Action").compareTo("Login")
== 0)
                {
                      SharedPreferences        settings        =
getSharedPreferences("DroidTrack", 0);
                      SharedPreferences.Editor     editor     =
settings.edit();

                      editor.putBoolean("LoggedIn", false);
```

```
                    editor.commit();


      contentLayout.setVisibility(View.VISIBLE);
                  }
            }
            }
      catch (Exception e)
      {
            Log.e("Exception", e.getMessage());
            }
      }
}
```

2.11 Register: This Activity allows users to register. It takes the information from the user and sends them to the server:

```
package tk.droidtrack.app;

import java.util.ArrayList;
import java.util.List;

import org.apache.http.NameValuePair;
import org.apache.http.message.BasicNameValuePair;
import org.json.JSONObject;

import android.app.Activity;
import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class RegisterActivity extends Activity
{
      EditText displayNameText;
      EditText emailText;
      EditText passwordText;
      EditText mobileText;

      Button registerButton;

      String DisplayName, Email, Password, Mobile;

      @Override
      protected void onCreate(Bundle savedInstanceState)
      {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.register_layout);


            displayNameText             =             (EditText)
findViewById(R.id.register_displayNameText);
```

```
            emailText                =                (EditText)
findViewById(R.id.register_emailText);
            passwordText              =               (EditText)
findViewById(R.id.register_passwordText);
            mobileText                =               (EditText)
findViewById(R.id.register_mobileText);

            registerButton              =               (Button)
findViewById(R.id.register_registerButton);

            registerButton.setOnClickListener(new
OnClickListener()
            {
                  @Override
                  public void onClick(View arg0)
                  {
                        Email = emailText.getText().toString();
                        Password                                =
passwordText.getText().toString();
                        if          (Email.matches("")          ||
Password.matches(""))
                        {

      Toast.makeText(RegisterActivity.this,  "Enter  Email  and
Password", Toast.LENGTH_SHORT).show();
                        }
                        else
                        {
                              Register();
                        }
                  }
            });
      }

      private void Register()
      {
            DisplayName = displayNameText.getText().toString();
            Email = emailText.getText().toString();
            Password = passwordText.getText().toString();
            Mobile = mobileText.getText().toString();

            List<NameValuePair>        list        =        new
ArrayList<NameValuePair>();

            list.add(new              BasicNameValuePair("Name",
DisplayName));
            list.add(new BasicNameValuePair("Email", Email));
            list.add(new          BasicNameValuePair("Password",
Password));
            list.add(new BasicNameValuePair("Mobile", Mobile));

            Log.i("List", list.toString());

            API_POST("/register", list);
      }
```

```
    private  void  API_POST(String  arg,  List<NameValuePair>
list)
    {
        API api = new API(this);

        Log.i("API_Obj", "new API(this)");

        try
           {
                 api.setListener(new IAsyncFetchListener()
                 {
                       @Override
                       public void onError(Throwable error)
                       {
                             Log.e("Error", error.getMessage());
                       }

                       @Override
                       public void onComplete(JSONObject result)
                       {
                             Log.i("Complete",
result.names().toString());
                             ParseJson(result);
                       }
                 });
           }
           catch (Exception e)
           {
                 Log.e("Exception", e.getMessage());
           }

        api.POST(arg, list);
      }

    private void ParseJson(JSONObject json)
    {
      Log.i("JSON_Result", json.toString());
      try
      {
           if(json.getString("Status").compareTo("Success")  ==
0)
           {

      if(json.getString("Action").compareTo("Register") == 0)
                 {
                       User user = new User();
                       user.ID                                =
json.getJSONObject("User").getString("ID");
                       user.Name                              =
json.getJSONObject("User").getString("Name");
                       user.Email                             =
json.getJSONObject("User").getString("Email");
                       user.Mobile                            =
json.getJSONObject("User").getString("Mobile");

                       Preferences.UserID = user.ID;
```

```
                    SharedPreferences       settings       =
getSharedPreferences("DroidTrack", 0);
                    SharedPreferences.Editor     editor     =
settings.edit();

                    editor.putBoolean("LoggedIn", true);
                    editor.putString("UserID", user.ID);
                    editor.putString("UserEmail",
user.Email);
                    editor.putString("UserPassword",
Password);
                    editor.putString("UserName", user.Name);
                    editor.putString("UserMobile",
user.Mobile);

                    editor.commit();

                    Intent   intent   =   new   Intent(this,
GroupsActivity.class);
                    intent.putExtra("User", user);
                    startActivity(intent);

                    Intent   intent2   =   new   Intent(this,
LocationUpdaterService.class);
                   startService(intent2);

                   this.finish();
              }
        }
        else
        {
              if(!json.has("Internet"))
              {
                    emailText.setError(null);
                    passwordText.setError(null);
              }
              Toast.makeText(this,
json.getString("Message"), Toast.LENGTH_LONG).show();
        }
        }
    catch (Exception e)
    {
        Log.e("Exception", e.getMessage());
        }
    }
}
```

2.12 Groups: This Activity shows the user groups and the invitation to the groups:

```
package tk.droidtrack.app;

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

```
import org.apache.http.NameValuePair;
import org.apache.http.message.BasicNameValuePair;
import org.json.JSONArray;
import org.json.JSONObject;

import android.app.AlertDialog;
import android.app.ListActivity;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.util.Log;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ListView;
import android.widget.SimpleAdapter;
import android.widget.Toast;

public class GroupsActivity extends ListActivity
{
	User currentUser;
	ArrayList<Group> groups;
	ArrayList<HashMap<String,String>>	groupsList	=	new
ArrayList<HashMap<String,String>>();

	SimpleAdapter sa;

	Button groups_newGroup;

	@Override
	protected void onCreate(Bundle savedInstanceState)
	{
		super.onCreate(savedInstanceState);
		setContentView(R.layout.groups_layout);

		currentUser	=
(User)getIntent().getExtras().getSerializable("User");

		groups_newGroup	=
(Button)findViewById(R.id.groups_newGroup);
		groups_newGroup.setOnClickListener(new
OnClickListener()
		{
			@Override
			public void onClick(View v)
			{
			AlertDialog.Builder	builder	=	new
AlertDialog.Builder(GroupsActivity.this);
			builder.setTitle("New Group");
			final	EditText	input	=	new
EditText(GroupsActivity.this);
			input.setHint("Group Name");
```

```
                builder.setView(input);

                builder.setNegativeButton("Cancel",         new
DialogInterface.OnClickListener()
                {
                 @Override
                 public  void  onClick(DialogInterface  dialog,
int which)
                 {
                      dialog.dismiss();
                 }
                });
                builder.setPositiveButton("OK",             new
DialogInterface.OnClickListener()
                {
                 @Override
                 public  void  onClick(DialogInterface  dialog,
int which)
                 {
                      String              value              =
input.getText().toString();
                      if(value.compareTo("") != 0)
                      {
                           List<NameValuePair>   list   =   new
ArrayList<NameValuePair>();
                           list.add(new
BasicNameValuePair("UserID", currentUser.ID));
                           list.add(new
BasicNameValuePair("GroupName", value));

                           API_POST("/create_group", list);
                      }
                      else
                      {
                           Toast.makeText(GroupsActivity.this,
"Enter group name!", Toast.LENGTH_LONG).show();
                      }
                 }
                });
                builder.show();
                 }
           });

           groups = new ArrayList<Group>();

           groupsList                  =                   new
ArrayList<HashMap<String,String>>();

           sa = new SimpleAdapter(this, groupsList,
                      android.R.layout.simple_list_item_2 ,
                      new String[] { "line1","line2" },
                      new       int[]       {android.R.id.text1,
android.R.id.text2});

        setListAdapter(sa);
        //initGroups();
```

```
      }

      @Override
      protected void onResume()
      {
            Log.i("GroupsActivity.onResume", "Start");
            super.onResume();
            InitGroups();
      }

      @Override
      protected void onListItemClick(ListView l, View v, int
position, long id)
      {
            super.onListItemClick(l, v, position, id);
            final Group g = groups.get(position);
            Log.i("ListItemClicked", g.Name);
            Log.i("Group_Invitation", g.Invitation);

            if(g.Invitation.compareTo("Sent") == 0)
            {
                  DialogInterface.OnClickListener
dialogClickListener = new DialogInterface.OnClickListener()
                  {
                      @Override
                      public     void     onClick(DialogInterface
dialog, int which)
                      {
                       List<NameValuePair>     list     =     new
ArrayList<NameValuePair>();
                       list.add(new BasicNameValuePair("UserID",
currentUser.ID));
                       list.add(new
BasicNameValuePair("GroupID", g.ID));

                         switch (which)
                         {
                         case DialogInterface.BUTTON_POSITIVE:
                            list.add(new
BasicNameValuePair("Invitation", "Accepted"));


      Toast.makeText((GroupsActivity.this),          "Accepted",
Toast.LENGTH_LONG).show();
                             break;

                         case DialogInterface.BUTTON_NEGATIVE:
                            list.add(new
BasicNameValuePair("Invitation", "Rejected"));


      Toast.makeText((GroupsActivity.this),          "Rejected",
Toast.LENGTH_LONG).show();
                             break;
                         }
```

```
                        Log.i("POST_Payload",
list.toString());
                      API_POST("/group_invitation", list);
                    }
                };

                AlertDialog.Builder     builder     =     new
AlertDialog.Builder(this);
                builder.setMessage("Accept                group
invitation?").setPositiveButton("Yes", dialogClickListener)
                    .setNegativeButton("No",
dialogClickListener).show();
           }
           else if(g.Invitation.compareTo("Accepted") == 0)
           {
                if(g.Permission.compareTo("Tracked") == 0)
                {
                      Toast.makeText(this,  "Your  are  tracked
since " + g.JoinDateTime, Toast.LENGTH_LONG).show();
                }
                else
                {
                      Intent   intent   =   new   Intent(this,
MapActivity.class);

                      Bundle bundle = new Bundle();
                      bundle.putSerializable("Group", g);

                      intent.putExtras(bundle);

                      startActivity(intent);
                }
           }
      }

      @Override
      public boolean onCreateOptionsMenu(Menu menu)
      {
           getMenuInflater().inflate(R.menu.groups_menu, menu);
           return true;
      }

      @Override
      public boolean onOptionsItemSelected(MenuItem item)
      {
           switch (item.getItemId())
           {
           case R.id.groups_menu_refresh:
                InitGroups();
                break;
           case R.id.groups_menu_logout:
                Logout();
                break;
           default:
                break;
           }
```

```
            return super.onOptionsItemSelected(item);
      }

      private void Logout()
      {
            SharedPreferences              settings              =
getSharedPreferences("DroidTrack", 0);
            SharedPreferences.Editor editor = settings.edit();

            editor.putBoolean("LoggedIn", false);

            editor.commit();

            Intent      intent      =      new      Intent(this,
LocationUpdaterService.class);
            Preferences.UserID = "";
         stopService(intent);

         Intent intent2 = new Intent(this, MainActivity.class);
         startActivity(intent2);

            this.finish();
      }

      public void InitGroups()
      {
            Log.i("InitGroups", "Start Method");
            groups.clear();
            groupsList.clear();
            sa.notifyDataSetChanged();

            try
            {
                  API api = new API(this);
                  api.setListener(new IAsyncFetchListener()
                  {
                        @Override
                        public void onError(Throwable error)
                        {
                              Log.e("Error", error.getMessage());
                        }

                        @Override
                        public void onComplete(JSONObject result)
                        {
                              Log.i("Complete",
result.names().toString());
                              try
                              {
                                    JSONArray       Groups       =
result.getJSONArray("Groups");
                                    Log.i("Groups",
Groups.length() + "");

                                    for   (int   i   =   0;   i   <
Groups.length(); i++)
```

```
                                    {
                                          Group g = new Group();
                                          g.ID                    =
((JSONObject)Groups.get(i)).getString("ID");
                                          g.Name                  =
((JSONObject)Groups.get(i)).getString("Name");
                                          g.CreatedBy             =
((JSONObject)Groups.get(i)).getString("CreatedBy");
                                          g.Permission            =
((JSONObject)Groups.get(i)).getString("Permission");
                                          g.Invitation            =
((JSONObject)Groups.get(i)).getString("Invitation");
                                          g.JoinDateTime          =
((JSONObject)Groups.get(i)).getString("JoinDateTime");
                                          groups.add(g);

                                          HashMap<String,String>
item = new HashMap<String, String>();
                                          item.put("line1",
g.Name);
                                          item.put("line2",
g.Invitation.compareTo("Sent")   ==   0   ?   "Invited   as   "   +
g.Permission : g.Permission);
                                          groupsList.add(item);
                                    }
                                    sa.notifyDataSetChanged();

                                    Preferences.userGroups   =   new
Group[groups.size()];
                                    Preferences.userGroups       =
(Group[]) groups.toArray(Preferences.userGroups);
                              }
                              catch (Exception e)
                              {
                                    Log.e("Exception",
e.getMessage());
                              }
                        }
                  });

                  api.GET("/user_groups?UserID="                +
currentUser.ID);
            }
            catch (Exception e)
            {
                  Log.e("Exception", e.getMessage());
            }
      }

    private   void   API_POST(String   arg,   List<NameValuePair>
list)
    {
        API api = new API(this);

        Log.i("API_Obj", "new API(this)");
```

```
        try
           {
                 api.setListener(new IAsyncFetchListener()
                 {
                       @Override
                       public void onError(Throwable error)
                       {
                             Log.e("Error", error.getMessage());
                       }

                       @Override
                       public void onComplete(JSONObject result)
                       {
                             ParceJSON(result);
                       }
                 });
           }
           catch (Exception e)
           {
                 Log.e("Exception", e.getMessage());
           }

        api.POST(arg, list);
      }

      private void ParceJSON(JSONObject json)
      {
      Log.i("JSON_Result", json.toString());
      try
      {
            if(json.getString("Status").compareTo("Success")  ==
0)
            {

      if(json.getString("Action").compareTo("CreateGroup")   ==
0)
                 {
                       Toast.makeText(this,   "Group   Created",
Toast.LENGTH_SHORT).show();
                 }
            }
            }
      catch (Exception e)
      {
            Log.e("Exception", e.getMessage());
            }

            InitGroups();
      }
}
```

2.13 Map: This Activity allows users to show the friends in the group last location on a Google map:

```
package tk.droidtrack.app;

import java.util.HashMap;
```

```
import org.json.JSONArray;
import org.json.JSONObject;

import tk.droidtrack.app.R;

import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import
com.google.android.gms.maps.GoogleMap.OnInfoWindowClickListene
r;
import com.google.android.gms.maps.SupportMapFragment;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.LatLngBounds;
import com.google.android.gms.maps.model.Marker;
import com.google.android.gms.maps.model.MarkerOptions;

import android.content.Intent;
import android.os.Bundle;
import android.support.v4.app.FragmentActivity;
import android.util.Log;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.TextView;

public class MapActivity extends FragmentActivity
{
    private GoogleMap mMap;
    TextView map_groupName;
    Button group_manage;
    Group g;

    int markersCount = -1, currentMarkerNumber = 0;

      LatLngBounds.Builder builder;
      LatLngBounds bounds;

      HashMap<Marker, String> markers;

    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.map_layout);

      g                                                    =
(Group)getIntent().getExtras().getSerializable("Group");

      markers = new HashMap<Marker, String>();

        group_manage                 =                 (Button)
findViewById(R.id.map_group_manage);

        if (g.Permission.compareTo("Manager") == 0)
        {
```

```
            group_manage.setOnClickListener(new
OnClickListener()
                {
                        @Override
                        public void onClick(View arg0)
                        {
                              Intent      intent      =      new
Intent(MapActivity.this, ManageGroupActivity.class);

                              intent.putExtra("Group", g);

                              startActivity(intent);
                              finish();
                        }
                  });
            }
         else
         {
            group_manage.setVisibility(View.GONE);
         }

         setUpMapIfNeeded();

         map_groupName                                          =
(TextView)findViewById(R.id.map_groupName);
         map_groupName.setText("Group : " + g.Name);
         Log.i("map_groupName.text",
map_groupName.getText().toString());

      builder = new LatLngBounds.Builder();
     }

     @Override
     protected void onResume()
     {
         super.onResume();
         setUpMapIfNeeded();
     }

     private void setUpMapIfNeeded()
     {
         if (mMap == null)
         {
             mMap            =            ((SupportMapFragment)
getSupportFragmentManager().findFragmentById(R.id.map)).getMap
();
             if (mMap != null)
             {
                  mMap.setOnInfoWindowClickListener(new
OnInfoWindowClickListener()
                  {
                              @Override
                              public                          void
onInfoWindowClick(Marker marker)
                              {
```

```
                            Log.i("onInfoWindowClick",
marker.getTitle());

                               Intent      intent      =      new
Intent(MapActivity.this, UserTracksMapActivity.class);
                               intent.putExtra("UserID",
markers.get(marker));
                               intent.putExtra("GroupID", g.ID);
                               startActivity(intent);
                            }
                        });
                 setUpMap();
             }
         }
     }

     private void setUpMap()
     {
         Log.i("api.GET", "/group_members?GroupID=" + g.ID);
         try
         {
            API_GET("/group_members?GroupID=" + g.ID);
             Log.i("api.GET?Done", "Yes");
            }
         catch (Exception e)
         {
             Log.e("Exception", e.getMessage());
            }
     }

     private void addMarker(String UserID, String Name, LatLng
latLng)
     {
      markers.put(mMap.addMarker(new
MarkerOptions().position(latLng).title(Name)), UserID);
     }

     private void API_GET(String arg)
     {
         Log.i("API_GET_arg", arg);
         API api = new API(this);

         Log.i("API_Obj", "new API(this)");

         try
            {
                  api.setListener(new IAsyncFetchListener()
                  {
                        @Override
                        public void onError(Throwable error)
                        {
                              Log.e("Error", error.getMessage());
                        }

                        @Override
                        public void onComplete(JSONObject result)
```

```
                        {
                              Log.i("Complete",
result.names().toString());
                              ParseJson(result);
                        }
                  });
            }
            catch (Exception e)
            {
                  Log.e("Exception", e.getMessage());
            }

        api.GET(arg);
      }

    private void ParseJson(JSONObject json)
    {
      Log.i("JSON_Result", json.toString());
      try
      {

      if(json.getString("Action").compareTo("GetGroupMembers")
== 0)
            {
                  JSONArray users = json.getJSONArray("Users");

                  markersCount = users.length();

                  for (int i = 0; i < users.length(); i++)
                  {
                        currentMarkerNumber = i;
                        API_GET("/user_last_location?UserID="   +
((JSONObject)users.get(i)).getString("ID"));
                  }
            }
            else
if(json.getString("Action").compareTo("GetLastUserLocation")
== 0)
            {
                  LatLng latLng = new LatLng(

      Double.parseDouble(json.getJSONObject("Location").getStri
ng("Lat")),

      Double.parseDouble(json.getJSONObject("Location").getStri
ng("Lng"))
                                    );

      addMarker(json.getJSONObject("Location").getString("UserI
D"),json.getJSONObject("User").getString("Name"), latLng);
                  builder.include(latLng);

                  if(currentMarkerNumber == markersCount - 1)
                  {
                  bounds = builder.build();
```

```
      mMap.animateCamera(CameraUpdateFactory.newLatLngBounds(bo
unds, 50));
                  }
            }
            }
      catch (Exception e)
      {
            Log.e("Exception", e.getMessage());
            }
      }
}
```

2.14 Manage Group: This Activity allows only the manger to edit on his groups:

```
package tk.droidtrack.app;

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

import org.apache.http.NameValuePair;
import org.apache.http.message.BasicNameValuePair;
import org.json.JSONArray;
import org.json.JSONObject;

import android.app.AlertDialog;
import android.app.ListActivity;
import android.content.DialogInterface;
import android.os.Bundle;
import android.text.InputType;
import android.util.Log;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ListView;
import android.widget.SimpleAdapter;
import android.widget.TextView;
import android.widget.Toast;

public class ManageGroupActivity extends ListActivity
{
      ArrayList<User> users;
      ArrayList<HashMap<String,String>>    usersList    =    new
ArrayList<HashMap<String,String>>();

      SimpleAdapter sa;

      Group g;

      Button inviteMember, deleteGroup;
```

```
      @Override
      protected void onCreate(Bundle savedInstanceState)
      {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.manage_group_layout);

            g                     =                     (Group)
getIntent().getExtras().getSerializable("Group");
            TextView        groupName        =        (TextView)
findViewById(R.id.manage_group_groupName);
            groupName.setText("Group : " + g.Name);

            users = new ArrayList<User>();

            usersList = new ArrayList<HashMap<String,String>>();

            sa = new SimpleAdapter(this, usersList,
                        android.R.layout.simple_list_item_2,
                        new String[] {"line1", "line2"},
                        new       int[]       {android.R.id.text1,
android.R.id.text2});

        setListAdapter(sa);

        inviteMember                 =                 (Button)
findViewById(R.id.manage_group_inviteMember);
        deleteGroup                  =                 (Button)
findViewById(R.id.manage_group_deleteGroup);

        inviteMember.setOnClickListener(new OnClickListener()
        {
            @Override
                  public void onClick(View arg0)
            {
                  AlertDialog.Builder     builder     =     new
AlertDialog.Builder(ManageGroupActivity.this);
                  builder.setTitle("Invite user by email");

                  final      EditText      input      =      new
EditText(ManageGroupActivity.this);

input.setInputType(InputType.TYPE_TEXT_VARIATION_EMAIL_ADDRESS
);
                  input.setHint("User Email");

                  builder.setView(input);

                  builder.setNegativeButton("Cancel",         new
DialogInterface.OnClickListener()
                  {
                  @Override
                  public  void  onClick(DialogInterface  dialog,
int which)
                  {
                        dialog.dismiss();
                  }
```

```
                });
                builder.setPositiveButton("OK",             new
DialogInterface.OnClickListener()
                {
                @Override
                public  void  onClick(DialogInterface  dialog,
int which)
                {
                      String              value              =
input.getText().toString();

                      if(value.compareTo("") != 0)
                      {
                            List<NameValuePair>   list   =   new
ArrayList<NameValuePair>();
                            list.add(new
BasicNameValuePair("GroupID", g.ID));
                            list.add(new
BasicNameValuePair("Email", value));

                            API_POST("/invite_to_group", list);
                      }
                      else
                      {

     Toast.makeText(ManageGroupActivity.this,   "Enter   email
address!", Toast.LENGTH_LONG).show();
                      }

                }
                });
                builder.show();
                }
           });

        deleteGroup.setOnClickListener(new OnClickListener()
        {
           @Override
                public void onClick(View arg0)
           {
                AlertDialog.Builder     builder     =     new
AlertDialog.Builder(ManageGroupActivity.this);
                builder.setTitle("Delete group : " + g.Name);
                builder.setMessage("Are  you  sure  to  delete
group : '" + g.Name + "' ?");

                builder.setNegativeButton("No",             new
DialogInterface.OnClickListener()
                {
                @Override
                public  void  onClick(DialogInterface  dialog,
int which)
                {
                      dialog.dismiss();
                }
                });
```

```
                builder.setPositiveButton("Yes",            new
DialogInterface.OnClickListener()
                {
                @Override
                public  void  onClick(DialogInterface  dialog,
int which)
                {
                      List<NameValuePair>     list     =     new
ArrayList<NameValuePair>();
                      list.add(new
BasicNameValuePair("GroupID", g.ID));

                      API_POST("/delete_group", list);
                }
                });
                builder.show();
                }
           });
      }

      @Override
      protected void onResume()
      {
           Log.i("GroupsActivity.onResume", "Start");
           super.onResume();
           InitUsers();
      }

      @Override
      protected  void  onListItemClick(ListView  l,  View  v,  int
position, long id)
      {
           super.onListItemClick(l, v, position, id);
           final User u = users.get(position);

           if (u.Permission.compareTo("Manager") == 0)
           {
                Toast.makeText(this,    "You    are    manager",
Toast.LENGTH_LONG).show();
           }
           else
           {
                AlertDialog.Builder    builderSingle    =    new
AlertDialog.Builder(this);
             builderSingle.setTitle("Action   for   user   '"   +
u.Name + "'");

             final  ArrayAdapter<String>  arrayAdapter  =  new
ArrayAdapter<String>(this,
                      android.R.layout.select_dialog_item);
             arrayAdapter.add("Change Permition");
             arrayAdapter.add("Remove");

             final  ArrayAdapter<String>  innerArrayAdapter  =
new ArrayAdapter<String>(this,
                      android.R.layout.select_dialog_item);
```

```
            innerArrayAdapter.add("Tracker");
            innerArrayAdapter.add("Tracked");

            builderSingle.setNegativeButton("Cancel",      new
DialogInterface.OnClickListener()
            {
               @Override
               public  void  onClick(DialogInterface  dialog,
int which)
               {
                     dialog.dismiss();
               }
            });

            builderSingle.setAdapter(arrayAdapter,         new
DialogInterface.OnClickListener()
            {
               @Override
               public  void  onClick(DialogInterface  dialog,
int which)
               {
                     if(which == 0)
                     {
                           AlertDialog.Builder  builderInner  =
new AlertDialog.Builder(ManageGroupActivity.this);
                          builderInner.setTitle("Change
Permission");

builderInner.setNegativeButton("Cancel",                    new
DialogInterface.OnClickListener()
                          {
                           @Override
                           public void onClick(DialogInterface
dialog, int which)
                           {
                                 dialog.dismiss();
                           }
                          });

builderInner.setAdapter(innerArrayAdapter,                  new
DialogInterface.OnClickListener()
                          {
                           @Override
                           public void onClick(DialogInterface
dialog, int which)
                           {
                                 List<NameValuePair> list = new
ArrayList<NameValuePair>();
                                 list.add(new
BasicNameValuePair("UserID", u.ID));
                                 list.add(new
BasicNameValuePair("GroupID", g.ID));
                                 list.add(new
BasicNameValuePair("Permission",
innerArrayAdapter.getItem(which)));
```

```
                                    API_POST("/user_permission",
list);
                              }
                              });
                              builderInner.show();
                        }
                        else
                        {
                              AlertDialog.Builder  builderInner  =
new AlertDialog.Builder(ManageGroupActivity.this);
                              builderInner.setTitle("Remove  '"  +
u.Name + "' ?");

builderInner.setPositiveButton("Yes",                       new
DialogInterface.OnClickListener()
                              {
                              @Override
                              public  void  onClick(DialogInterface
dialog, int which)
                              {
                                    List<NameValuePair> list = new
ArrayList<NameValuePair>();
                                    list.add(new
BasicNameValuePair("UserID", u.ID));
                                    list.add(new
BasicNameValuePair("GroupID", g.ID));


      API_POST("/remove_user_from_group", list);
                              }
                              });
                              builderInner.setNegativeButton("No",
new DialogInterface.OnClickListener()
                              {
                              @Override
                              public  void  onClick(DialogInterface
dialog, int which)
                              {
                                    dialog.dismiss();
                              }
                              });
                              builderInner.show();
                        }
                  }
            });
            builderSingle.show();
      }
}

@Override
public boolean onCreateOptionsMenu(Menu menu)
{
      getMenuInflater().inflate(R.menu.groups_menu, menu);
      return true;
}
```

```
	@Override
	public boolean onOptionsItemSelected(MenuItem item)
	{
		int id = item.getItemId();
		if (id == R.id.groups_menu_refresh)
		{
			InitUsers();
			return true;
		}
		if (id == R.id.action_settings)
		{
			return true;
		}
		return super.onOptionsItemSelected(item);
	}

	public void InitUsers()
	{
		users.clear();
		usersList.clear();
		sa.notifyDataSetChanged();

		try
		{
			API api = new API(this);
			api.setListener(new IAsyncFetchListener()
			{
				@Override
				public void onError(Throwable error)
				{
					Log.e("Error", error.getMessage());
				}

				@Override
				public void onComplete(JSONObject result)
				{
					Log.i("Complete",
result.names().toString());
					try
					{
						JSONArray	Users	=
result.getJSONArray("Users");
						Log.i("Users",  Users.length()
+ "");

						for  (int  i  =  0;  i  <
Users.length(); i++)
						{
							User u = new User();
							u.ID	=
((JSONObject)Users.get(i)).getString("ID");
							u.Name	=
((JSONObject)Users.get(i)).getString("Name");
							u.Permission	=
((JSONObject)Users.get(i)).getString("Permission");
```

```
                                        u.Invitation            =
((JSONObject)Users.get(i)).getString("Invitation");
                                        users.add(u);

                                        HashMap<String,String>
item = new HashMap<String, String>();
                                        item.put("line1",
u.Name);
                                        item.put("line2",
u.Permission + (u.Invitation.compareTo("Accepted") == 0 ? "" :
" - Invitation " + u.Invitation));
                                        usersList.add(item);
                                  }
                                  sa.notifyDataSetChanged();
                            }
                            catch (Exception e)
                            {
                                  Log.e("Exception",
e.getMessage());
                            }
                      }
                });

                api.GET("/group_members?GroupID=" + g.ID);
          }
          catch (Exception e)
          {
                Log.e("Exception", e.getMessage());
          }
     }

    private  void  API_POST(String  arg,  List<NameValuePair>
list)
    {
        API api = new API(this);

        Log.i("API_Obj", "new API(this)");

        try
           {
                api.setListener(new IAsyncFetchListener()
                {
                      @Override
                      public void onError(Throwable error)
                      {
                            Log.e("Error", error.getMessage());
                      }

                      @Override
                      public void onComplete(JSONObject result)
                      {
                            ParceJSON(result);
                      }
                });
           }
           catch (Exception e)
```

```
            {
                  Log.e("Exception", e.getMessage());
            }

        api.POST(arg, list);
      }

      private void ParceJSON(JSONObject json)
      {
      Log.i("JSON_Result", json.toString());
      try
      {
            if(json.getString("Status").compareTo("Success")  ==
0)
            {
                  Toast.makeText(this,
json.getString("Message"), Toast.LENGTH_LONG).show();


      if(json.getString("Action").compareTo("DeleteGroup")   ==
0)
                  {
                        this.finish();
                  }
            }
            else if(json.getString("Status").compareTo("Failed")
== 0)
            {

      if(json.getString("Action").compareTo("InviteToGroup")  ==
0)
                  {
                        Toast.makeText(this,
json.getString("Message"), Toast.LENGTH_LONG).show();
                  }
            }
            }
      catch (Exception e)
      {
            Log.e("Exception", e.getMessage());
            }

      InitUsers();
      }
}
```

2.15 User_tracks_map: This Activity allows only the manger and tracker to show a friend track since he registered on the group:

```
package tk.droidtrack.app;

import java.util.ArrayList;
```

```
import org.json.JSONArray;
import org.json.JSONObject;

import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
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.LatLngBounds;
import com.google.android.gms.maps.model.MarkerOptions;
import com.google.android.gms.maps.model.PolylineOptions;

import android.os.Bundle;
import android.support.v4.app.FragmentActivity;
import android.util.Log;
import android.widget.TextView;

public class UserTracksMapActivity extends FragmentActivity
{
    GoogleMap mMap;
     LatLngBounds.Builder builder;
     LatLngBounds bounds;

    TextView userName;
    User u;

    ArrayList<MyLocation> locationsList;

    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.user_tracks_map_layout);

        locationsList = new ArrayList<MyLocation>();
     builder = new LatLngBounds.Builder();

        u = new User();
        u.ID = getIntent().getExtras().getString("UserID");
        Log.i("UserTracksMapActivity_UserID", u.ID);
        Log.i("Line", "----------------------------");
        Log.i("Line", "----------------------------");
        Log.i("Line", "----------------------------");
        Log.i("Line", "----------------------------");
        Log.i("Line", "----------------------------");
        Log.i("Line", "----------------------------");
        Log.i("Line", "----------------------------");
        Log.i("Line", "----------------------------");
        Log.i("Line", "----------------------------");
        Log.i("Line", "----------------------------");
        Log.i("Line", "----------------------------");

        API_GET("/user?UserID=" + u.ID);

        setUpMapIfNeeded();
```

```
    }

    @Override
    protected void onResume()
    {
        super.onResume();
        setUpMapIfNeeded();
    }


    private void setUpMapIfNeeded()
    {
        if (mMap == null)
        {
            mMap             =             ((SupportMapFragment)
getSupportFragmentManager().findFragmentById(R.id.user_tracks_
map)).getMap();
            if (mMap != null)
            {
                setUpMap();
            }
        }
    }

    private void setUpMap()
    {
        try
        {
           API_GET("/user_locations?UserID="    +    u.ID    +
"&GroupID=" + getIntent().getExtras().getString("GroupID"));
           }
        catch (Exception e)
        {
            Log.e("Exception", e.getMessage());
           }
    }

    private void API_GET(String arg)
    {
        API api = new API(this);

        try
           {
                 api.setListener(new IAsyncFetchListener()
                 {
                       @Override
                       public void onError(Throwable error)
                       {
                             Log.e("Error", error.getMessage());
                       }

                       @Override
                       public void onComplete(JSONObject result)
                       {
                             Log.i("Complete",
result.names().toString());
```

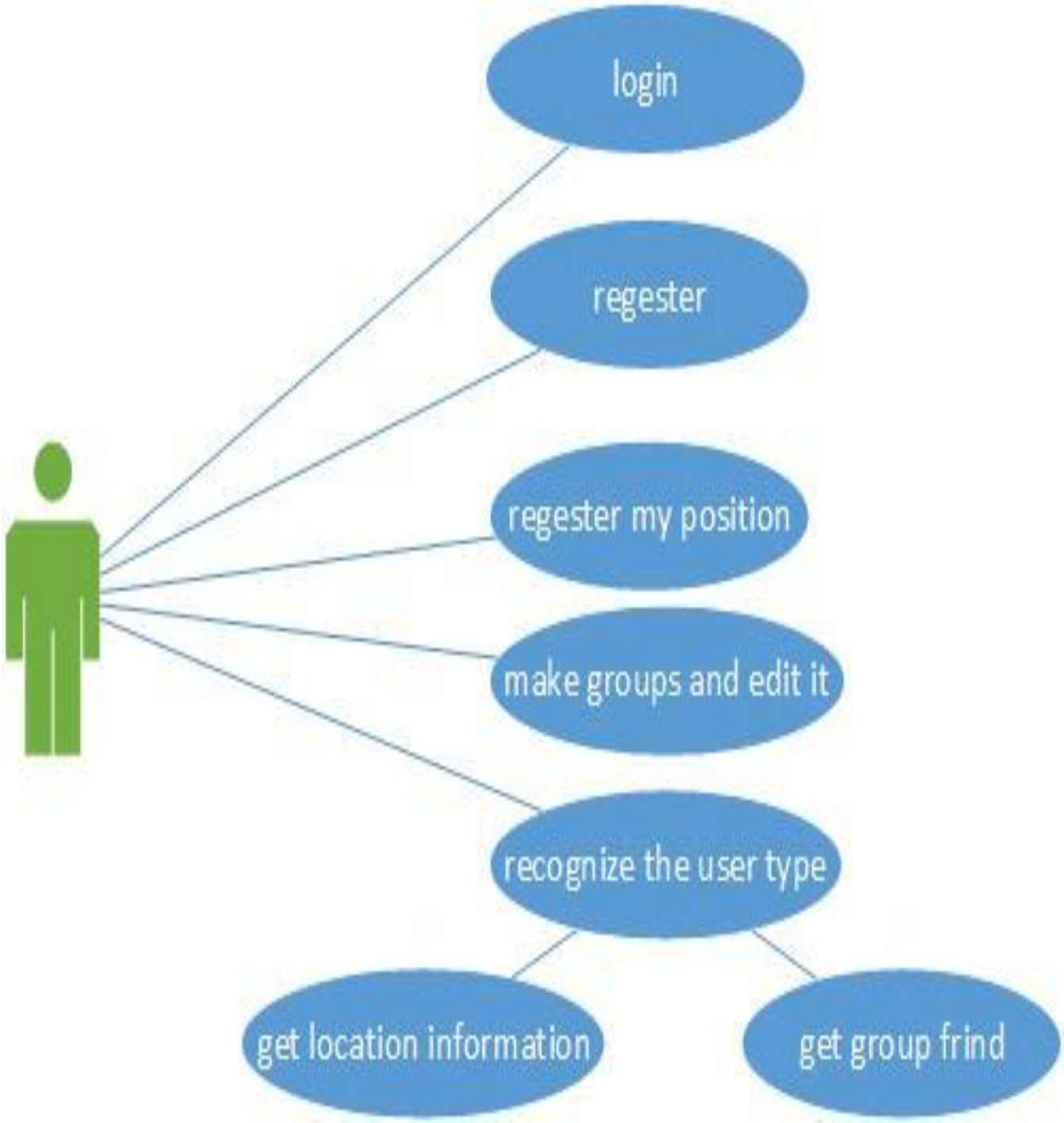
```
                        ParseJson(result);
                    }
                });
            }
            catch (Exception e)
            {
                Log.e("Exception", e.getMessage());
            }

        api.GET(arg);
    }

    private void ParseJson(JSONObject json)
    {
     try
     {
            if(json.getString("Action").compareTo("GetUser")  ==
0)
            {
                u.Name                                          =
json.getJSONObject("User").getString("Name");
              userName.setText("User : " + u.Name);
              Log.i("userName", u.Name);
            }
            else
if(json.getString("Action").compareTo("GetUserLocations")    ==
0)
            {
                JSONArray              locations              =
json.getJSONArray("Locations");

                for (int i = 0; i < locations.length(); i++)
                {
                    MyLocation l = new MyLocation();
                    l.ID                                        =
((JSONObject)locations.get(i)).getString("ID");
                    l.DateTime                                  =
((JSONObject)locations.get(i)).getString("DateTime");
                    l.UserID                                    =
((JSONObject)locations.get(i)).getString("UserID");
                    l.LatLng = new LatLng(

     Double.parseDouble(((JSONObject)locations.get(i)).getStri
ng("Lat")),

     Double.parseDouble(((JSONObject)locations.get(i)).getStri
ng("Lng")));
                    locationsList.add(l);
                }
                setUpTracks();
            }
            }
     catch (Exception e)
     {
            Log.e("Exception", e.getMessage());
            }
```

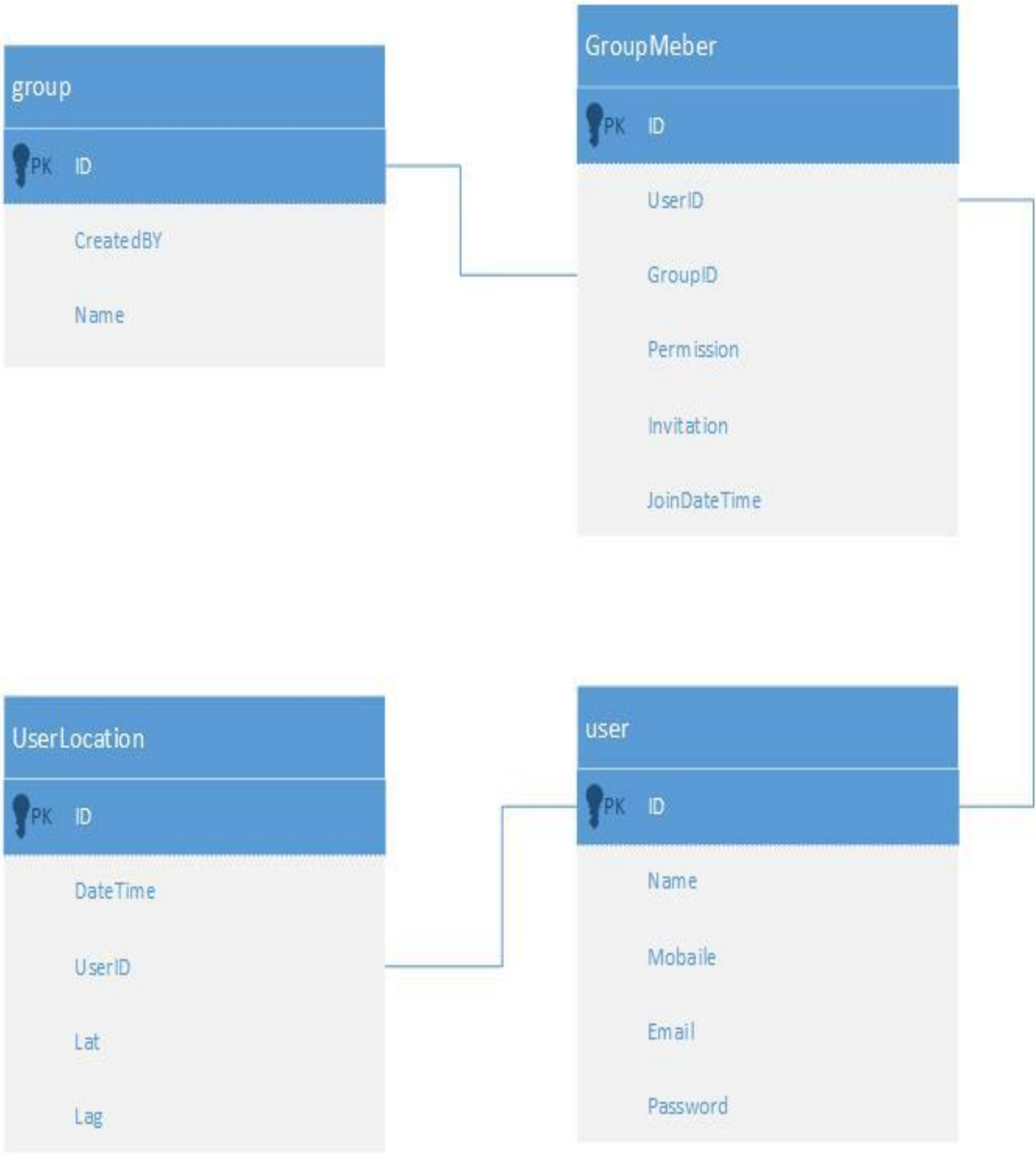
```
      }

    private void setUpTracks()
    {
      PolylineOptions rectOptions = new PolylineOptions();

      rectOptions.width(2);

      for (int i = 0; i < locationsList.size(); i++)
      {
            LatLng latlng = locationsList.get(i).LatLng;
            rectOptions.add(latlng);
            builder.include(latlng);
            mMap.addMarker(
                        new MarkerOptions()
                        .position(latlng)
                        .title(locationsList.get(i).DateTime)

      .icon(BitmapDescriptorFactory.defaultMarker(BitmapDescrip
torFactory.HUE_BLUE))
                        );
            }

      bounds = builder.build();


      mMap.animateCamera(CameraUpdateFactory.newLatLngBounds(bo
unds, 50));

            mMap.addPolyline(rectOptions);
      }
}
```

3. Diagrams:

3.1 Server side:

3.1.1 Use case diagram:

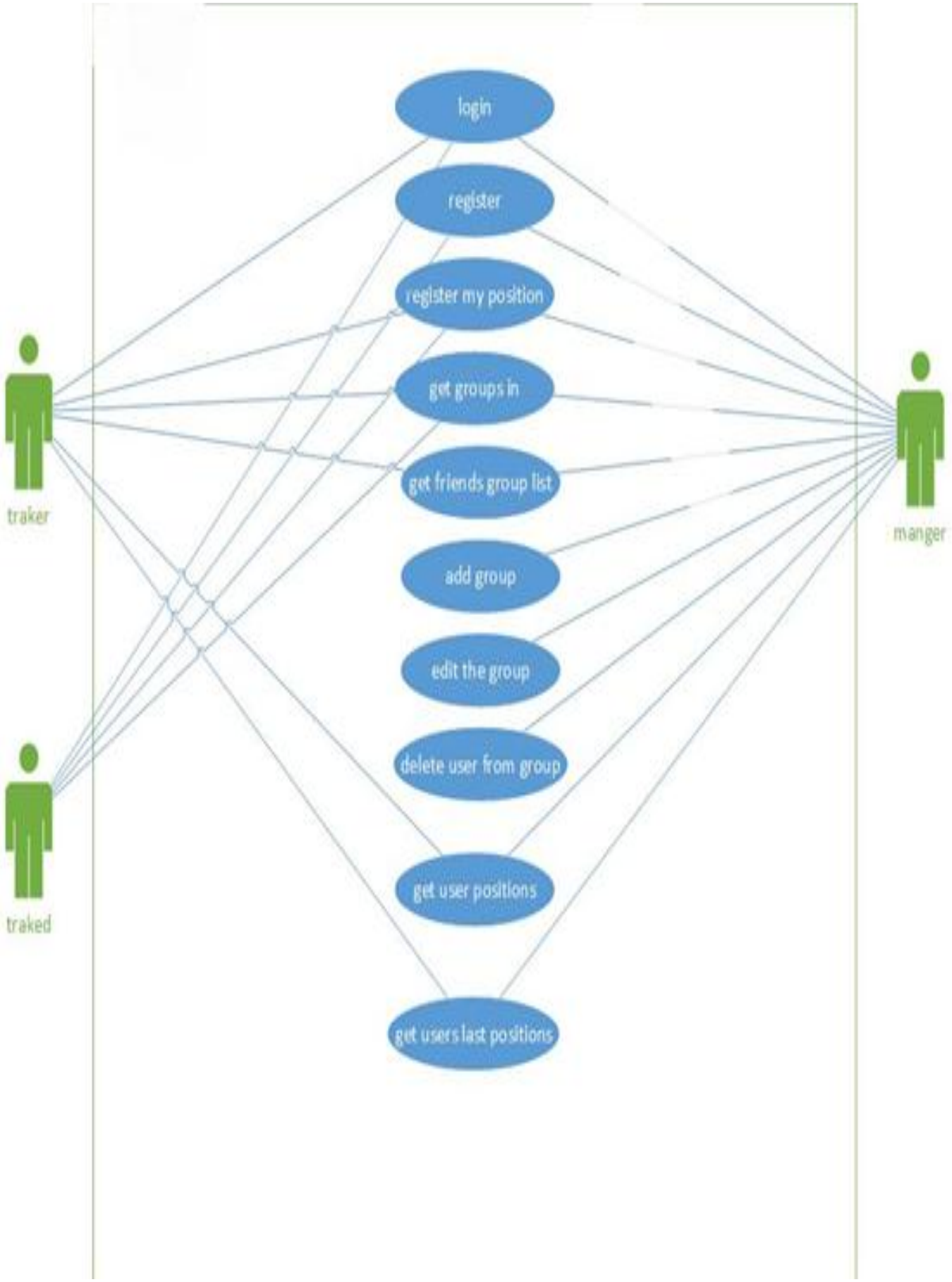




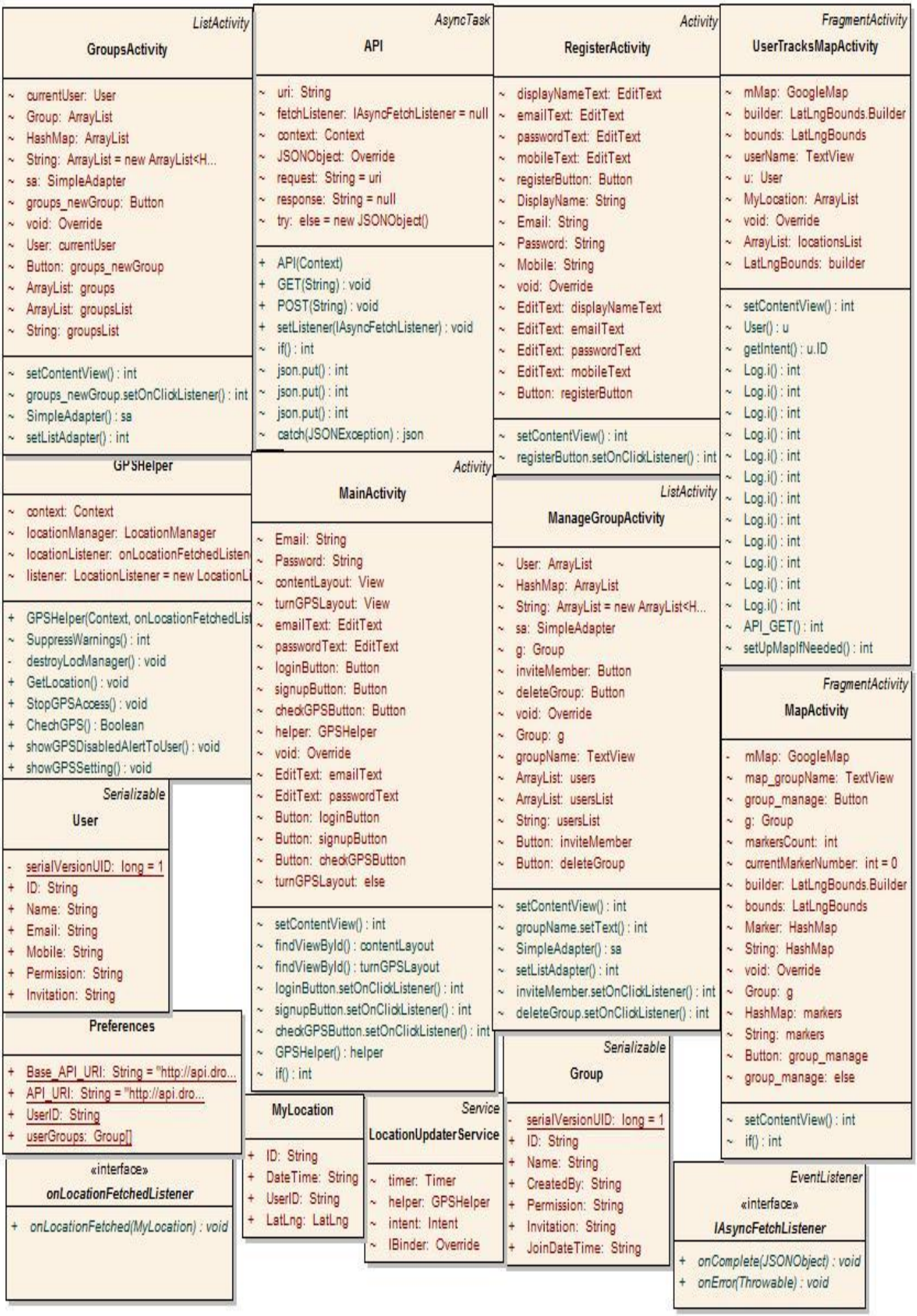
3.1.2 Database design:

3.2 Client side:

3.2.1 Use case diagram:



3.2.2 Classes Diagrams:


ListActivity
GroupsActivity
~ currentUser: User
~ Group: ArrayList
~ HashMap: ArrayList
~ String: ArrayList = new ArrayList<H...
~ sa: SimpleAdapter
~ groups_newGroup: Button
~ void: Override
~ User: currentUser
~ Button: groups_newGroup
~ ArrayList: groups
~ ArrayList: groupsList
~ String: groupsList
~ setContentView() : int
~ groups_newGroup.setOnClickListener() : int
~ SimpleAdapter() : sa
~ setListAdapter() : int
AsyncTask
API
~ uri: String
~ fetchListener: IAsyncFetchListener = null
~ context: Context
~ JSONObject: Override
~ request: String = uri
~ response: String = null
~ try: else = new JSONObject()
+ API(Context)
+ GET(String) : void
+ POST(String) : void
+ setListener(IAsyncFetchListener) : void
~ if() : int
~ json.put() : int
~ json.put() : int
~ json.put() : int
~ catch(JSONException) : json
Activity
RegisterActivity
~ displayNameText: EditText
~ emailText: EditText
~ passwordText: EditText
~ mobileText: EditText
~ registerButton: Button
~ DisplayName: String
~ Email: String
~ Password: String
~ Mobile: String
~ void: Override
~ EditText: displayNameText
~ EditText: emailText
~ EditText: passwordText
~ EditText: mobileText
~ Button: registerButton
~ setContentView() : int
~ registerButton.setOnClickListener() : int
FragmentActivity
UserTracksMapActivity
~ mMap: GoogleMap
~ builder: LatLngBounds.Builder
~ bounds: LatLngBounds
~ userName: TextView
~ u: User
~ MyLocation: ArrayList
~ void: Override
~ ArrayList: locationsList
~ LatLngBounds: builder
~ setContentView() : int
~ User() : u
~ getIntent() : u.ID
~ Log.i() : int
~ Log.i() : int
~ Log.i() : int
~ Log.i() : int
~ Log.i() : int
~ Log.i() : int
~ Log.i() : int
~ Log.i() : int
~ Log.i() : int
~ Log.i() : int
~ Log.i() : int
~ Log.i() : int
~ API_GET() : int
~ setUpMapIfNeeded() : int
GPSHelper
~ context: Context
~ locationManager: LocationManager
~ locationListener: onLocationFetchedListen
~ listener: LocationListener = new LocationLi
+ GPSHelper(Context, onLocationFetchedList
~ SuppressWarnings() : int
- destroyLocManager() : void
+ GetLocation() : void
+ StopGPSAccess() : void
+ ChechGPS() : Boolean
+ showGPSDisabledAlertToUser() : void
+ showGPSSetting() : void
Activity
MainActivity
~ Email: String
~ Password: String
~ contentLayout: View
~ turnGPSLayout: View
~ emailText: EditText
~ passwordText: EditText
~ loginButton: Button
~ signupButton: Button
~ checkGPSButton: Button
~ helper: GPSHelper
~ void: Override
~ EditText: emailText
~ EditText: passwordText
~ Button: loginButton
~ Button: signupButton
~ Button: checkGPSButton
~ turnGPSLayout: else
~ setContentView() : int
~ findViewById() : contentLayout
~ findViewById() : turnGPSLayout
~ loginButton.setOnClickListener() : int
~ signupButton.setOnClickListener() : int
~ checkGPSButton.setOnClickListener() : int
~ GPSHelper() : helper
~ if() : int
ListActivity
ManageGroupActivity
~ User: ArrayList
~ HashMap: ArrayList
~ String: ArrayList = new ArrayList<H...
~ sa: SimpleAdapter
~ g: Group
~ inviteMember: Button
~ deleteGroup: Button
~ void: Override
~ Group: g
~ groupName: TextView
~ ArrayList: users
~ ArrayList: usersList
~ String: usersList
~ Button: inviteMember
~ Button: deleteGroup
~ setContentView() : int
~ groupName.setText() : int
~ SimpleAdapter() : sa
~ setListAdapter() : int
~ inviteMember.setOnClickListener() : int
~ deleteGroup.setOnClickListener() : int
FragmentActivity
MapActivity
- mMap: GoogleMap
~ map_groupName: TextView
~ group_manage: Button
~ g: Group
~ markersCount: int
~ currentMarkerNumber: int = 0
~ builder: LatLngBounds.Builder
~ bounds: LatLngBounds
~ Marker: HashMap
~ String: HashMap
~ void: Override
~ Group: g
~ HashMap: markers
~ String: markers
~ Button: group_manage
~ group_manage: else
~ setContentView() : int
~ if() : int
Serializable
User
- serialVersionUID: long = 1
+ ID: String
+ Name: String
+ Email: String
+ Mobile: String
+ Permission: String
+ Invitation: String
Preferences
+ Base_API_URI: String = "http://api.dro...
+ API_URI: String = "http://api.dro...
+ UserID: String
+ userGroups: Group[]
«interface»
onLocationFetchedListener
+ onLocationFetched(MyLocation) : void
MyLocation
+ ID: String
+ DateTime: String
+ UserID: String
+ LatLng: LatLng
Service
LocationUpdaterService
~ timer: Timer
~ helper: GPSHelper
~ intent: Intent
~ IBinder: Override
Serializable
Group
- serialVersionUID: long = 1
+ ID: String
+ Name: String
+ CreatedBy: String
+ Permission: String
+ Invitation: String
+ JoinDateTime: String
EventListener
«interface»
IAsyncFetchListener
+ onComplete(JSONObject) : void
+ onError(Throwable) : void

REFERENCES

1. Clancy, Heather. "California Security Company uses barcodes to help track assets". CBS Interactive. Retrieved February 9, 2012.
2. "Cisco Unveils Wireless Location Solution and New Unified Wireless Network Software Release". CISCO. Retrieved May 22, 2007.
3. "10 tips for selecting a GPS fleet management solution". Ph.C. News. Retrieved August 2011.
4. Had lock, Charles (Oct 14, 2012). "RFID chips let schools track students -- and retain funding -- but some parents object". NBC News. Retrieved 22 April 2014.
5. "RFID". RFID Journal LLC.
6. "GPS and Relativity". Astronomy.ohio-state.edu. Retrieved November 6, 2011.
7. Project Report - GPS Tracker Pascal Bragger MSc Course- Ubiquitous Computing University of Fribourg, Switzerland March 9, 2006.
8. Welderufael Berhane Tesfay, Todd Booth, and Karl Andersson, Reputation Based Security Model for Android Applications‘, 2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications.
9. Location Manager APIs– Android Developer: http://developer.android.com/reference/android/location/LocationManager.html
10. Schwinger, W., Grin, C., Prll1, B., and Retschitzegger, W. A lightweight framework for location-based services. In Lecture Notes in Computer Science (Berlin, 2005), Springer, pp. 206_210.
11. MySQL - The world's most popular open source database: http://www.mysql.com .
12. James Fuller, Professional PHP Web Services, Wrox Press Inc., 2003.

13. Dave W. Mercer, Allan Kent, and others, Beginning PHP5 Wrox Press Inc., 2005.
14. Anthony T. Holdener III, Ajax: The Denitive Guide Interactive Applications for the Web, O'Reilly Media, 2008.
15. PHP tutorial: http://www.w3schools.com/php/default.asp.
16. Andi Gutmans, Stig Bakken, Derick Rethans, PHP 5 Power Programming, Pearson, 2004.
17. Quickie tutorial: http://www.quackit.com.
18. Wikipedia: http://en.wikipedia.org.
19. W3Schools: http://www.w3schools.com.
20. Mark L. Murphy, Beginning Android 2. Apress.
21. Ed Lecky-Thompson, Professional PHP5 Wrox Press Inc., 2004.
22. Android. http://www.android.com .
23. Android Market. http://market.android.com .