GEO LINKED ATTACHMENTS AND TAGS

A PROJECT REPORT

submitted by

Akshat Sharma (13BCE1009)

Antariksh Narain (13BCE1017)

Niranj Jyothish (13BCE1085)

in partial fulfillment for the award

of

B. Tech

degree in

Computer Science and Engineering

School of Computing Science and Engineering

VIT®

UNIVERSITY

(Estd. u/s 3 of UGC Act 1956)

www.vit.ac.in

Vellore ▪ Chennai

CHENNAI CAMPUS

Vandalur - Kelambakkam Road, Chennai - 600127

May 2017

GEO LINKED ATTACHMENTS AND TAGS

A PROJECT REPORT

submitted by

Akshat Sharma (13BCE1009)

Antariksh Narain (13BCE1017)

Niranj Jyothish (13BCE1085)

in partial fulfillment for the award

of

B. Tech

degree in

Computer Science and Engineering

School of Computing Science and Engineering

VIT UNIVERSITY

(Estd. u/s 3 of UGC Act 1956)

www.vit.ac.in

Vellore ▪ Chennai

CHENNAI CAMPUS

Vandalur - Kelambakkam Road, Chennai - 600127

May 2017

VIT UNIVERSITY
(Estd. u/s 3 of UGC Act 1956)
www.vit.ac.in
Vellore ▪ Chennai
CHENNAI CAMPUS
Vandalur - Kelambakkam Road, Chennai - 600127

School of Computing Science and Engineering

DECLARATION

We hereby declare that the project entitled "**Geo Linked Attachments and Tags (GLAT)**" submitted by us to the School of Computing Science and Engineering, VIT University, Chennai Campus, Chennai (600127) in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology in Computer Science and Engineering** is a record of bonafide work carried out by us under the supervision of **Prof. Ramesh Ragala, Assistant Professor (Sr.) VIT University Chennai**. We further declare that the work reported in this project has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma of this institute or of any institute or university.

Akshat Sharma (13BCE1009)

Antariksh Narain (13BCE1017)

Niranj Jyothish (13BCE1085)

VIT®
UNIVERSITY
(Estd. u/s 3 of UGC Act 1956)
www.vit.ac.in
Vellore ▪ Chennai
CHENNAI CAMPUS
Vandalur - Kelambakkam Road, Chennai - 600127

School of Computing Science and Engineering

CERTIFICATE

The project report entitled "**Geo Linked Attachments and Tags**" is prepared and submitted by **Akshat Sharma (13BCE1009), Antariksh Narain (13BCE1017) and Niranj Jyothish (13BCE1085).** It has been found satisfactory in terms of scope, quality and presentation as partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology in Computer Science and Engineering** at VIT University, Chennai Campus, Chennai, India.

Prof. Ramesh Ragala

Examined by:

Examiner 1 **Examiner 2**

VIT®
UNIVERSITY
(Estd. u/s 3 of UGC Act 1956)
www.vit.ac.in
Vellore ▪ Chennai
CHENNAI CAMPUS
Vandalur - Kelambakkam Road, Chennai - 600127

School of Computing Science and Engineering

ACKNOWLEDGEMENT

We would like to express our gratitude to all those who provided us the possibility to complete this project. We give our special gratitude to our faculty advisor, **Prof. Ramesh Ragala**, whose contribution is stimulating suggestions and encouragement, helped us to coordinate the project and writing this project.

Furthermore, we would also like to acknowledge with much appreciation the crucial role of **Dr. Ganesan R. (**Program Chair), whose encouraging words enabled us to complete this project successfully. We would also like to acknowledge **Dr. Vaidehi Vijaykumar** (Dean, SCSE) whose constant word of encouragement helped us immensely during the course of this Project. Lastly, we take this opportunity to thank **Dr. Bhargavi R.**, whose timely emails and proper communication facilitated smooth operation and completion of the project in stipulated time.

CONTENT

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATION

Abbreviation	Expansion
SIFT	Scale Invariant Feature Transform
SURF	Speeded Up Robust Transform
TLD	Tracking Learning and Detection
GPS	Global Positioning System
JSON	Javascript Object Notation
IoT	Internet of Things
PSQL	Postgres Structured Query Language
Redis	Remote Dictionary Server
IDE	Integrated Development Environment

ABSTRACT

In today's world, social networks of all kinds are found – professional, educational, media, informational and even simplistic messengers and blog based networks – each of these capitalizing on some kind of communication method unique to themselves. Messages and posts can be tagged to timestamps, pictures, locations and even people, thanks to these networks. However, with the coming of the next decade where, augmented and virtual reality seem to be big players, the evolution of different kinds of communication methods seems inevitable. This project aims at exploring one such potential method while economically connecting devices virtually to the internet.

Smartphones in the global market today come inbuilt with enough sensors (Gyroscope, accelerometer and compass) to pinpoint a person's geolocation and the orientation and altitude of the phone. Also, they come equipped with quality cameras that can be used in conjunction with these sensors to map static objects in the environment virtually. With the boom in the demand for computer vision products, better and faster object detection algorithms are being developed. Combining the object identifiers returned from any such detection algorithm with the sensor data available through the smartphone, we can effectively provide a unique identity to any object in the environment. Using this very simple idea, object detection algorithms and a modest stack of technologies, in this project we attempt to build a messenger for augmented reality.

Every message comprises of the content, sensor data, latitude, longitude and object identifiers pertaining to what object the message is tagged to. This message is then visible to only a subset of users that the creator of the message wishes to share the message with. Keeping consistency with other messengers, this subset is called a group. The basic functionalities of any messenger along with features that such a messenger should have been provided. Keeping in mind the heavy nature of image processing, performance has been improved by using a diversity of tools.

1. INTRODUCTION

1.1 Motivation

With the increasing demand for connecting things to the Internet and enabling a more accessible world, the need for specialized network-enabled sensors specific to classes of things has arisen. This project tries to solve this problem by proposing a method to virtually connect stationary objects to the Internet without the inherent need to attach to them anything physically.

The proposed method is used to implement a simple messenger like application where messages can be tagged onto any object in the environment. A tag would essentially comprise of the geolocation, object identifiers and other sensor data retrieved from the user's smartphone. This tag information along with the content of the message is stored on the server and can be used to re-identify this object by any intended recipient.

1.2 Problem Statement

The messenger provides the basic functionality that any messenger based application provides today. Users are allowed to choose a display name, upload a display picture, make groups, add group display pictures and add messages. The messages that can be added are broken into two categories: location based and object based messages. Further enhancement of user experience has been attempted by allowing them to comment on other people's messages in the form of a feed.

The problem this project is trying to solve can be thus broken into the following modules:

- To design a messenger with the basic functionalities of creating a profile and groups.
- To design a system that gives identity to (stationary) objects in the environment using sensor data and object identifiers.
- To allow users to share the tags and messages with specific groups of people.
- To interpret these tags correctly to link messages back to objects they were tagged on at the recipients' side.

1.3 Background and Related Work

The first internet connected appliance was a modified the Coke machine at Carnegie Mellon University (1982), which could report its inventory and tell if the drinks were cold or not. Since then the vision of Internet of Things has evolved with the convergence of technologies in the field of wireless communication, sensors and embedded systems. The IoT allows the object to be sensed, creating opportunities for more direct integration of the physical world into a computer-based system. This results in an efficient, improved and economically beneficial system reducing human intervention.

Many companies are working towards Augmented Reality, that is overlaying of data on top of the current view of the world. Microsoft has developed HoloLens which provides a platform for experiencing mixed reality. People are developing applications for 3D visualization of the object on HoloLens. These devices work by combining sensor data and computer vision with machine learning to provide a magnificent experience.

There are algorithms like SURF (Speeded Up Robust Transform) et al. [1] and SIFT (Scale Invariant Feature Transform) et al. [2] which work on feature based descriptors for object detection, hence they do not need any training time overhead. As these rely on features of objects, orientation and scale of the object to be detected do not hinder in the process of object detection.

Another commonly used object detection method is the use of HAAR Cascades et al. [3] which require training with positive and negative sample images to generate a HAAR Cascade. This algorithm may require training time, but it is efficient when it comes to detecting multiple instances of an object in a given frame.

Some use an algorithm which works on the principle of tracking and learning for generating an object detector like TLD (Tracking Learning and Detection) et al. [4]. This algorithm learns about the object from live video, by tracking object marked in the first frame of the video. It learns about the features of the object frame by frame and identifies unique features. These features can be loaded later to detect the object in a given frame.

Geo Tagging is a concept wherein one can tag an image with location information. A similar concept can be applied to objects in the environment, by tagging these objects with location information using GPS (Global Positioning System) and sensor data from compass and gyroscope to uniquely identify these object in the three-dimensional space. Also, digital information like text or multimedia can be stored on these tagged objects.

1.4 Objective

The objective of this project is to design a social networking system on the lines of augmented reality. The basic idea is to develop a messaging application where a user can create message from their location and share it among the members of the group. These messages can be given additional information by tagging the message to an object. Now when the user receives the message they can only view it when he/she is near the location or looking at an object through a camera. These messages are stored with sensor data from GPS, magnetic sensor and gyroscope. The end product enclosing the above ideas to be an Android application supporting multiple CPU architecture.

2. DESIGN STANDARDS WITH CONSTRAINTS

2.1 Hardware

The system can be developed on a custom hardware using microprocessors like Raspberry-PI or Beagle bone and using a combination of sensors like GPS, camera, accelerometer and gyroscope. But a tradeoff had to be made in the quality of sensors and to provide a user experience a high-resolution touch screen for interaction with the application. A power source would be required to run the system for at least an hour. Also, the system would have been bulky for practical applications.

To overcome these issues, we implemented the system on a platform having industrial grade quality sensors, which is easy to carry, use and is easily accessible. Therefore, we implemented the system as an Application on an Android phone (54% of the world population uses it) [5], an average android phone has all the required sensors embedded in them. An android phone has higher clocked processor with high virtual memory in comparison to a Raspberry PI. Also in today's world, everyone carry's a mobile phone with them hence, it automatically becomes portable. Being on a mobile platform it is accessible to more people with no overhead charges.

As android operating system supports a wide range of CPU architectures instruction sets like (armeabi, armeabi-v7a, arm64-v8a, MIPS, x86 and x86_64) [6], helps in making cross-platform application. A cost analysis of both the above described system can be seen in Table 1 and Table 2, making the system to be developed on Android platform a better choice.

Quantity	Description	Unit Price	Total
1	Raspberry Pi 3 • 1.2 GHz 64-bit quad-core ARMv8 CPU • 802.11n Wireless LAN • Bluetooth 4.1 • 1 GB RAM	2999	2999
1	Raspberry Pi Camera V2 – 8MP	2160	2160
1	Raspberry Pi 7” Touch Screen LCD	7949	7949
1	U-BLOX NEO-6M GPS Module	1183	1183
1	GY-521 MPU-6050	209	209
1	GPRS GSM Module with Antenna	3539	3539
1	Lenovo PB410 5000mAh	945	945
		Subtotal	18984
		Mass Production @ 40%	-7593
		Software Cost @ 10%	1898
		Total Due By [Date]	**13289**

Table 1: Cost Analysis of Systems (Separate Modules)

Quantity	Description	Unit Price	Total
1	Samsung Galaxy On7 Pro • 2 GB RAM • 5.5” Touchscreen • 16 GB internal • Quad-core 1.2 GHz Cortex-A53 • Camera: 13MP • Sensors: GPS, Accelerometer, Gyroscope • Wi-Fi 802.11 b/g/n • Bluetooth v4.1	9490	8303.75
		Subtotal	8303.75
		Sales Tax @ 12.5%	1186.25
		Total Due By [Date]	**9490**

Table 2: Cost Analysis of Systems (Integrated Modules)

2.2 Software

The project requires an android application and an IDE with up-to-date support like Android Studio which supports native libraries and external modules. Xamarin is also an IDE for mobile application development, with support all the java libraries and native C libraries to run C++ language scripts on android under development. An alternative is Android Eclipse, but the support has been deprecated.

Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient. These design choices aim to optimize throughput and scalability in Web applications with many input/output operations, as well as for real-time Web applications.

Our software pool used for the development of the system can be seen in figure 2.

Figure 1: Software Pool

2.3 Standards used in the Project

We have used the below standards for the project:

- Hyper Text Transfer Protocol (HTTP) is used as an Internet standard through which requests are sent and responses are received from the APIs. HTTP is the underlying protocol used by the World Wide Web and this protocol defines how messages are formatted and transmitted, and what actions Web servers and browsers should take in response to various commands.
- IEEE 802.11 b/g/n standard is used for Wi-Fi communication. This standard is widely used to provide WLAN solutions both for temporary connections in hotspots in cafes, airports, hotels and similar places as well as within office scenarios
- RGB and Gray color standards are used for the images used. RGB standard uses primary colors like Red, Green and Blue to represent images on computer display. Whereas, grayscale image uses only two variations of pixel intensities, i.e. Black and White.
- For object detection, SIFT, SURF and HAAR algorithms are applied. SIFT and SURF are image processing algorithms which work on feature based descriptors for object detection, hence they do not need any training time overhead. As these rely on features of objects, orientation and scale of the object to be detected do not hinder in the process of object detection. On the other hand, HAAR algorithm require training with positive and negative sample images to generate a HAAR Cascade. This algorithm may require training time, but it is efficient when it comes to detecting multiple instances of an object in a given frame.
- IEEE 830 – 1998 SRS is the standard used for the project documentation. This practice is used to specify the requirements of a software in an orderly way. It gives an overall idea about the project like its background, objective, definitions and scope.

3. SOCIETAL ISSUES

3.1 Ethical

The application uses location service, that updates users' location every 5 minutes to provide better user experience by gathering data about nearby places. Though the data secure on the server, but in the case of a cyber-attack on the server the data may be compromised, exposing the location of the user to the attacker. When it comes to location security within the application only users within the group can see the messages.

3.2 Social

As the application is location aware, it can provide user's a personalized experience by showing a message of interest based on their location. People can share messages by tagging them on objects, therefore these messages can act as reminders to like, one can tag the refrigerator with the message "Buy milk from the market" and another person can be prompted with this message when they are near the object or looking at it.

3.3 Economic

Additional hardware is not required to run this application. The application can be easily executed on an Android phone, having GPS, camera and basic sensors support. The user does not need to buy an additional hardware to make it run. Also, if a separate hardware is designed for this purpose it will cost twice as buying a phone as per Table 1 and Table 2 There will be charges of using the internet (GPRS/ Wi-Fi) which will vary among different network providers.

3.4 Sustainability

The first version that has developed is mostly error free and can be used by the public when the server is up and running. The application is secure and the user conversation on the application are private. With the advent of IoT when every object needs an identity, this system can act as an interface to give these non-IoT devices into devices a digital identity. Also, information can be stored on these object in digital format.

3.5 Environmental Impact

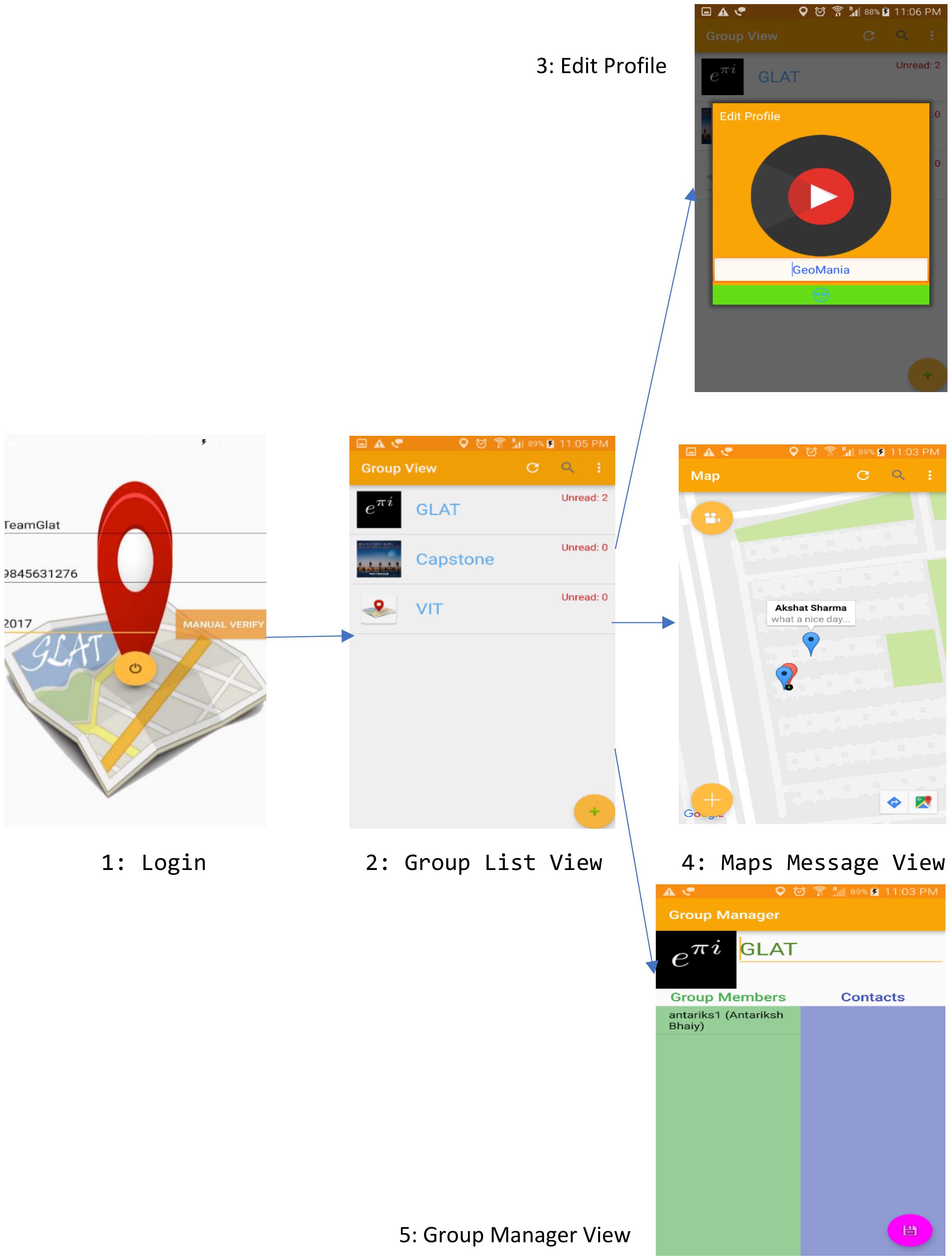
As the application run on the mobile phone (embedded system), no additional hardware is required, therefore reducing electronic waste. When the application is running at its full capacity, it will drain the phone battery at a higher rate. Also, the device may get heated up, reducing the battery performance.

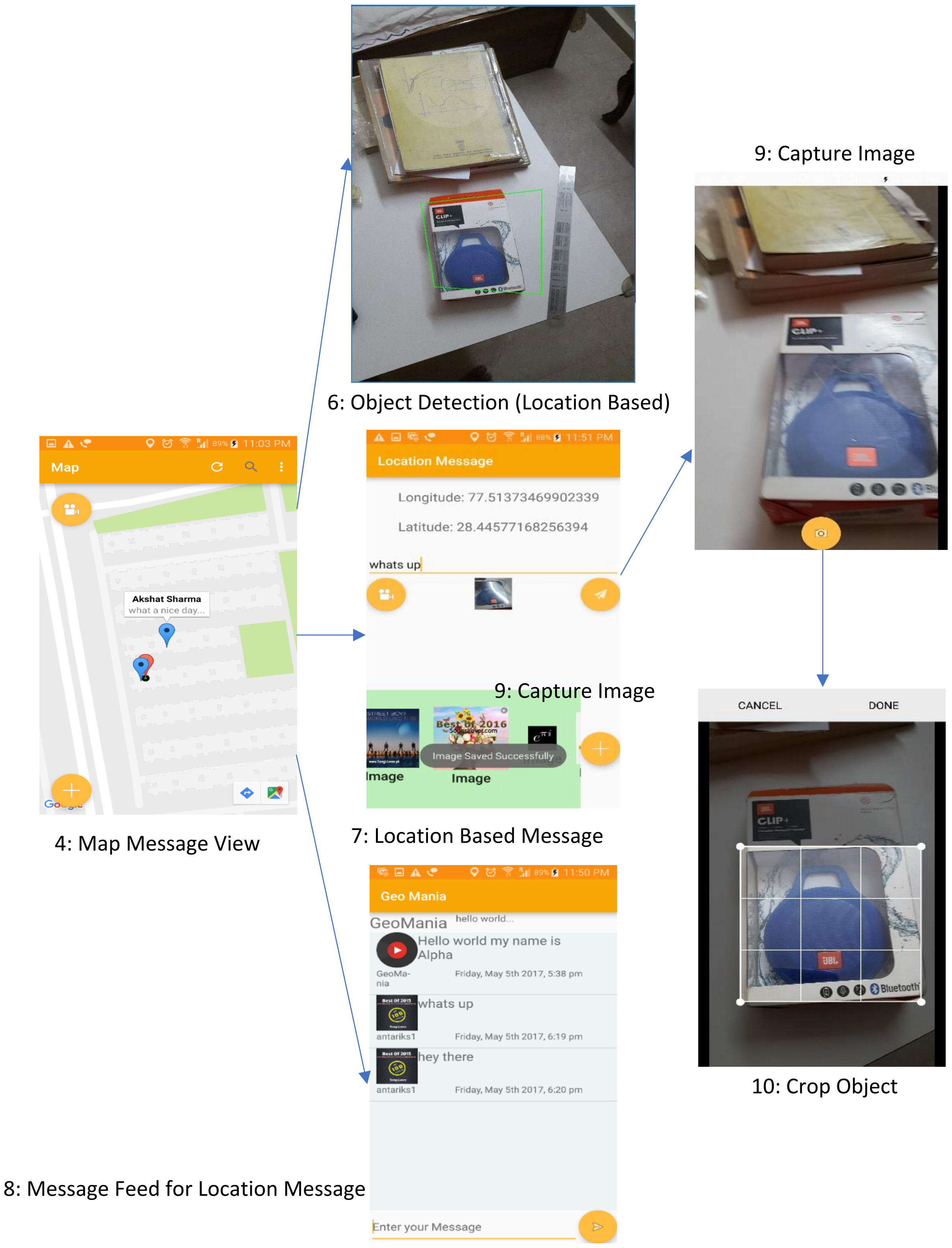
3.6 Usability

This software application has been developed by keeping user experience in mind. Therefore, the user interface is easy to use and people can share the application and build a network. The UI has been developed based on the user experience standards provided by Google. The mobile interface allows the user to generate location-based messages or tag messages to objects and share it with the group. Also, they can view messages at a given location or information tagged on an object.

4. IMPLEMENTATION

4.1 High Level Design





Akshat Sharma
what a nice day...

Longitude: 77.51373469902339

Latitude: 28.44577168256394

whats up

Image Saved Successfully

GeoMania hello world...

Hello world my name is Alpha
GeoMania Friday, May 5th 2017, 5:38 pm

whats up
antariks1 Friday, May 5th 2017, 6:19 pm

hey there
antariks1 Friday, May 5th 2017, 6:20 pm

Figure 2: Application Flow and Screenshots

4.2 Low Level Design

4.2.1 Sequence Diagram

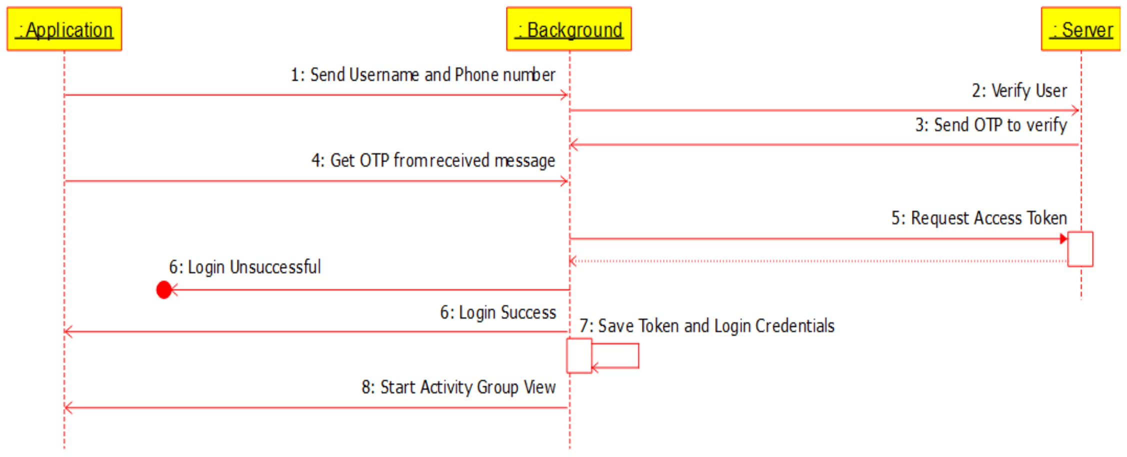

: Application
: Background
: Server
1: Send Username and Phone number
2: Verify User
3: Send OTP to verify
4: Get OTP from received message
5: Request Access Token
6: Login Unsuccessful
6: Login Success
7: Save Token and Login Credentials
8: Start Activity Group View

Figure 3: Login

To securely login and authenticate the user on the application.

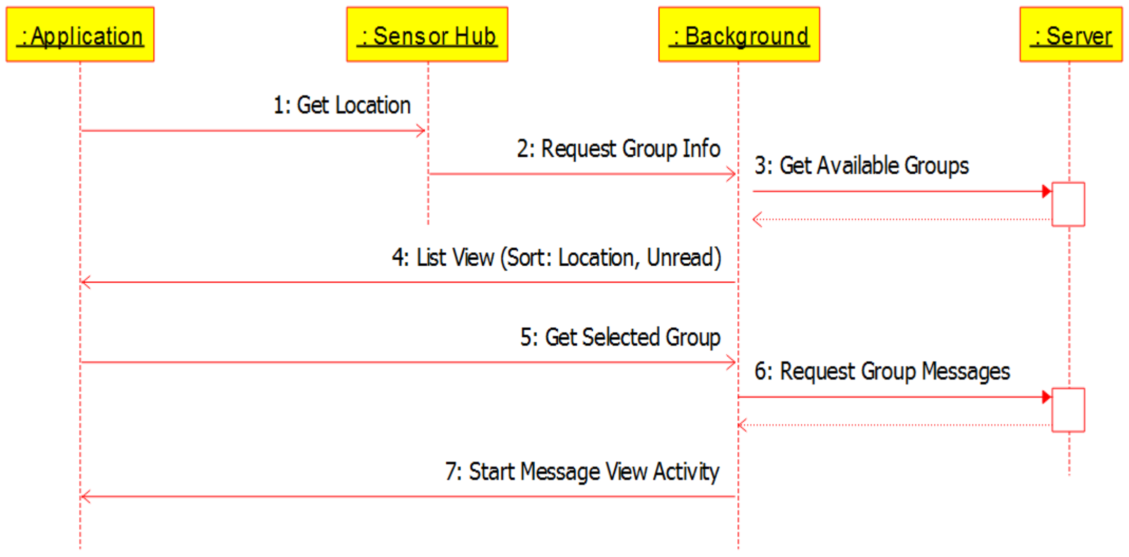


Figure 4: Group View

Retrieve user group's information, view them as a list.

.

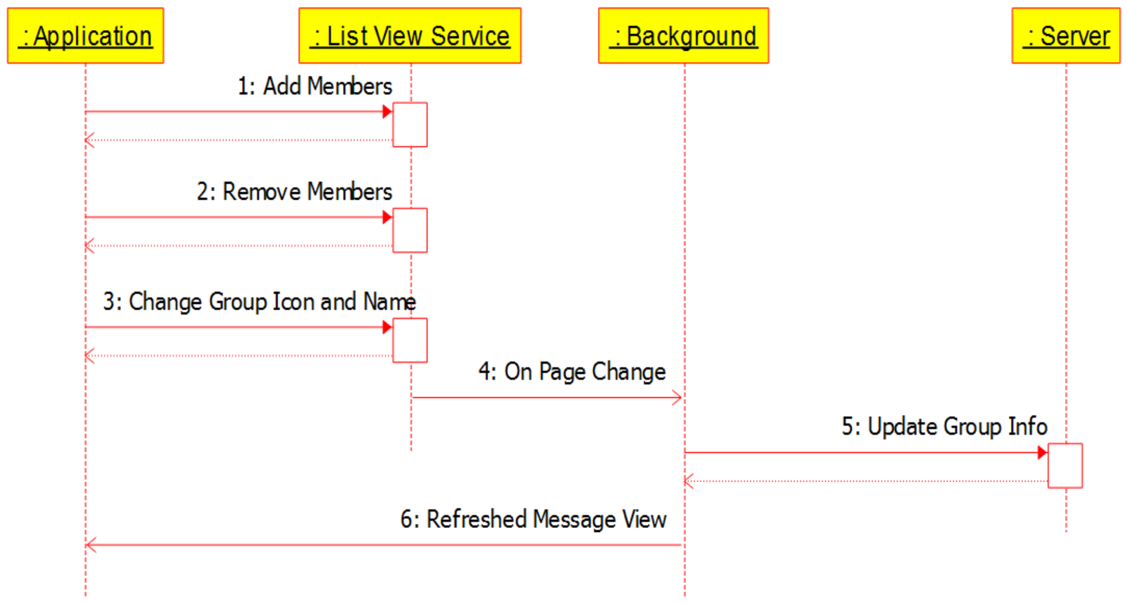


Figure 5: Group Manager

Manage members in the group between contact list (members) and existing members

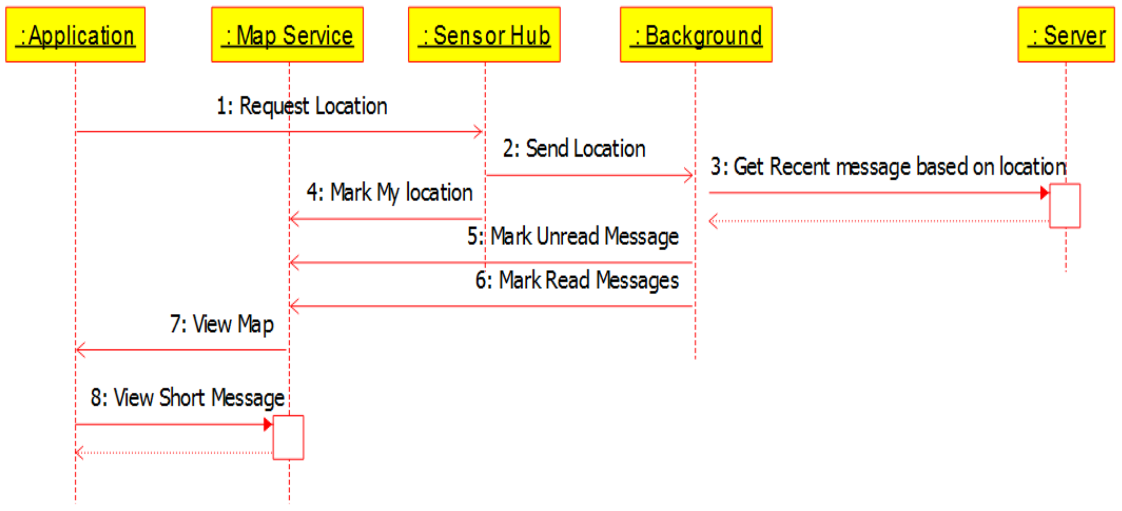


Figure 6: Message View

To view messages of the selected group on a Google Map with markers showing the location of messages. Different markers represent read, unread and your own messages.

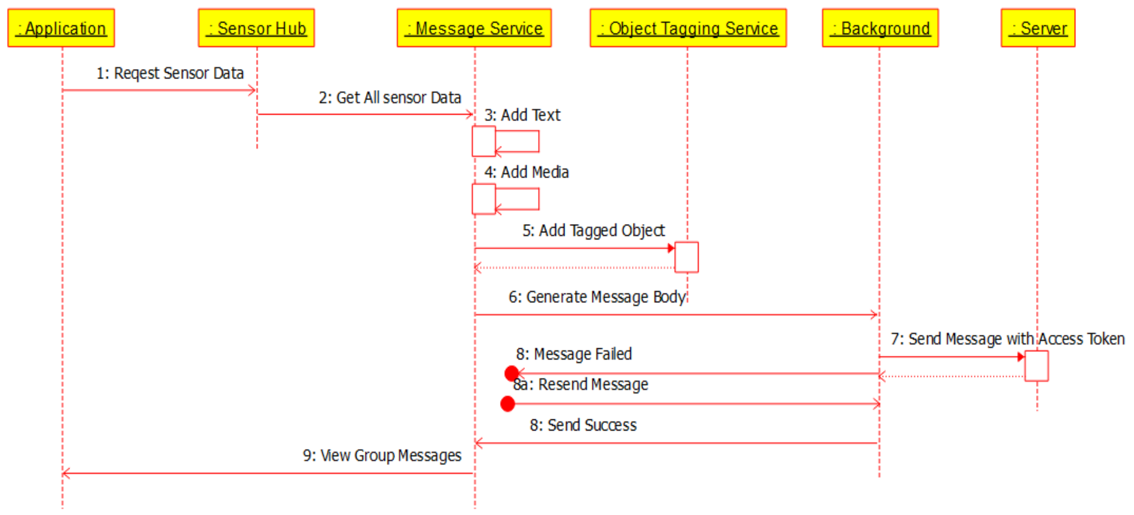


Figure 7: Create Message

To make a new message in the group with text, images or tagged objects on the given location.

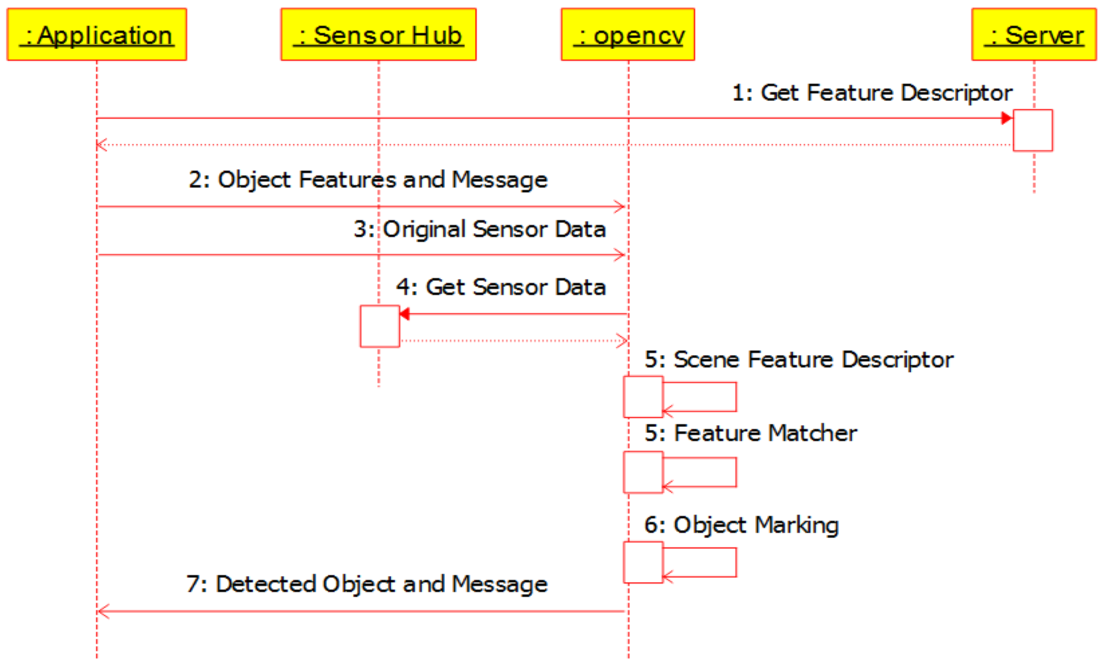


Figure 8: Object Based Message

Detecting object in the environment on which message is stored. This involves image processing in the background.

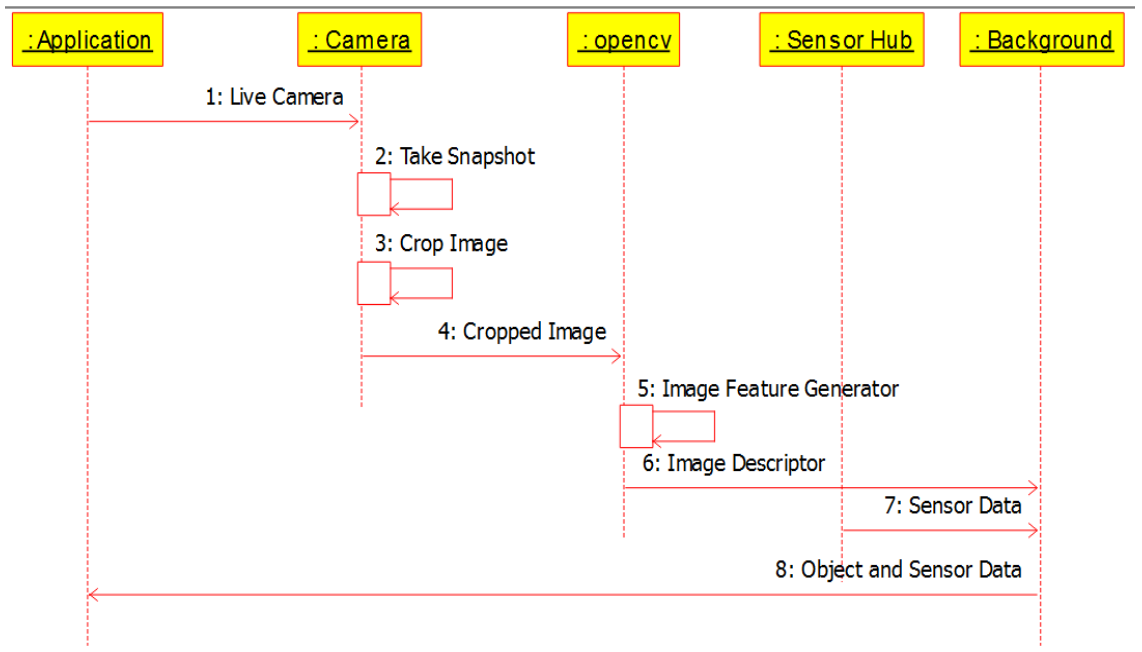


Figure 9: Object Creation

Creating an object for tagging a message. Sending information related to the object like sensor data, location and key points to the server.

4.2.2 Application and Data Flow

The application has seven activities which interact with the server to retrieve or update information.

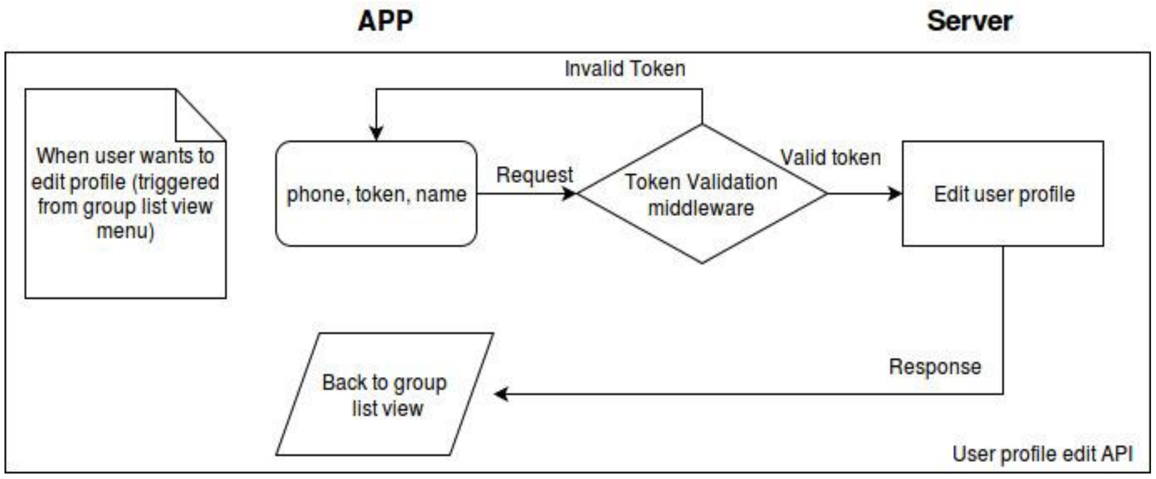


Figure 10: User Profile Edit API
API flow for editing the profile of a user

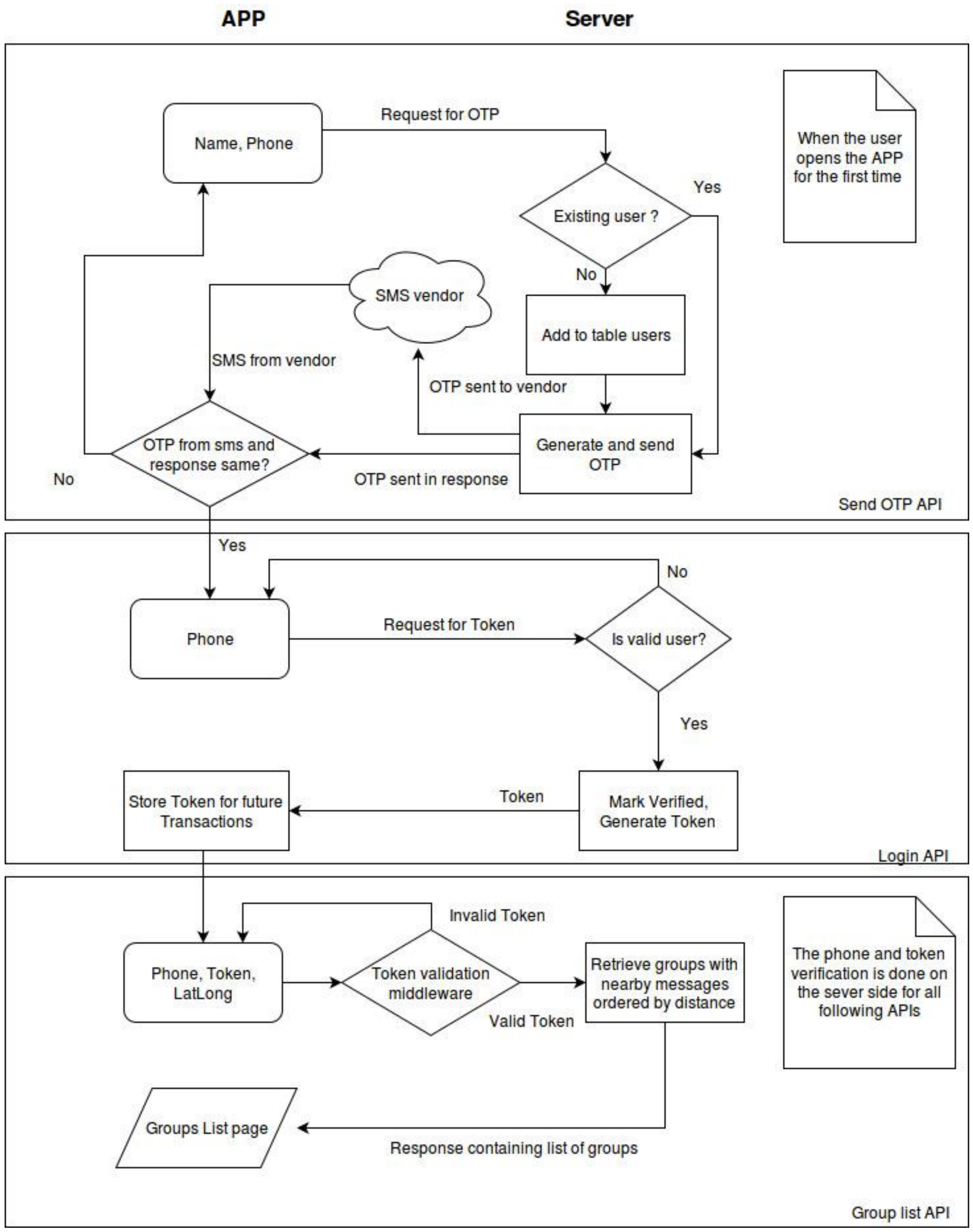

APP
Server
Name, Phone
Request for OTP
When the user opens the APP for the first time
Existing user ?
Yes
No
SMS vendor
Add to table users
SMS from vendor
OTP sent to vendor
OTP from sms and response same?
Generate and send OTP
No
OTP sent in response
Send OTP API
Yes
No
Phone
Request for Token
Is valid user?
Yes
Store Token for future Transactions
Token
Mark Verified, Generate Token
Login API
Invalid Token
Phone, Token, LatLong
Token validation middleware
Retrieve groups with nearby messages ordered by distance
Valid Token
The phone and token verification is done on the sever side for all following APIs
Groups List page
Response containing list of groups
Group list API

Figure 11: Group API

(i) SendOTP API: To verify the user's OTP,

(ii) Login API: To login the user and generate its token

(iii) GroupList API: To view group messages nearby based on user's location

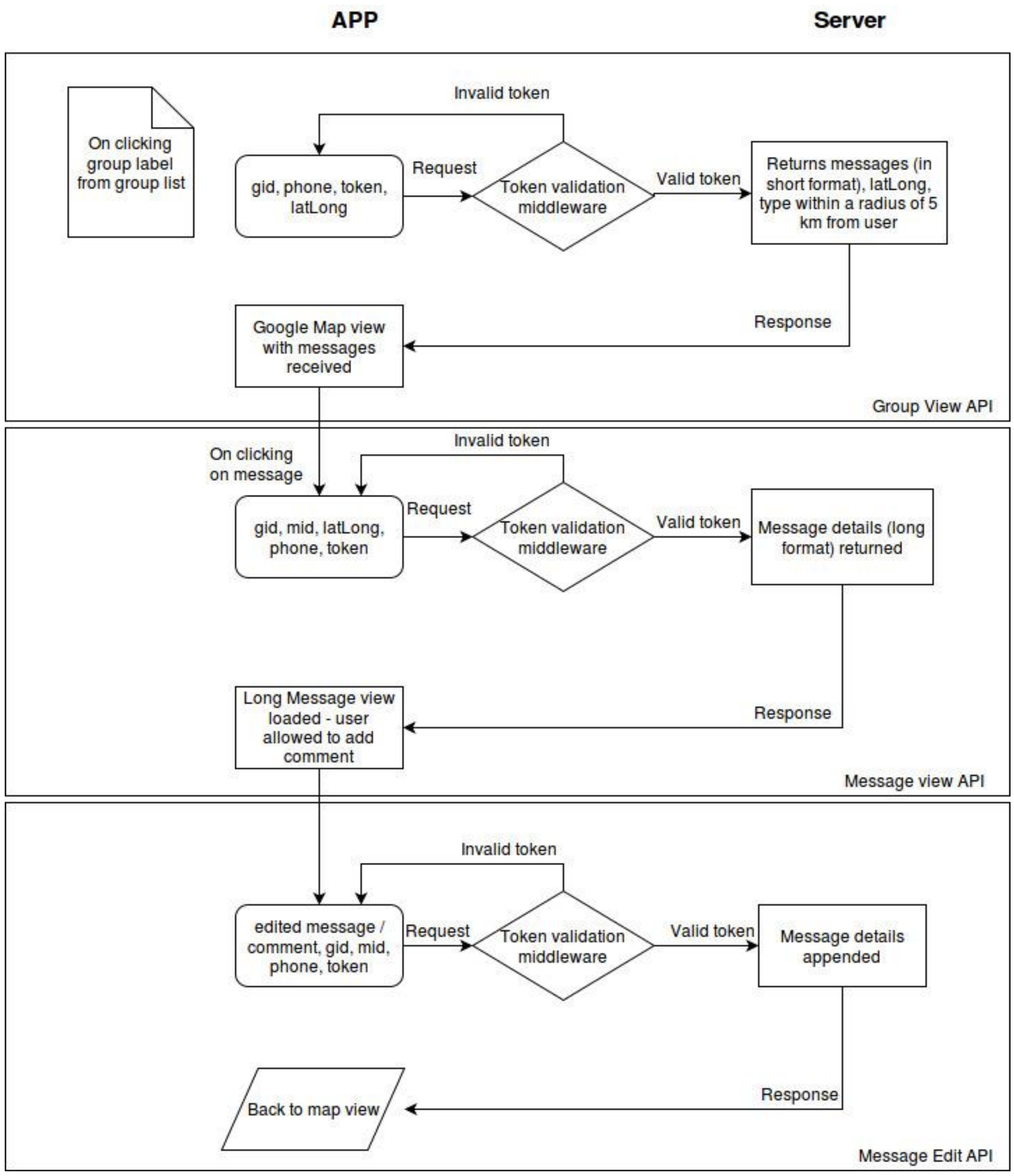

APP
Server
On clicking group label from group list
Invalid token
gid, phone, token, latLong
Request
Token validation middleware
Valid token
Returns messages (in short format), latLong, type within a radius of 5 km from user
Response
Google Map view with messages received
Group View API
On clicking on message
Invalid token
gid, mid, latLong, phone, token
Request
Token validation middleware
Valid token
Message details (long format) returned
Response
Long Message view loaded - user allowed to add comment
Message view API
Invalid token
edited message / comment, gid, mid, phone, token
Request
Token validation middleware
Valid token
Message details appended
Response
Back to map view
Message Edit API

Figure 12: Message API

(i) GroupView API: To retrieve nearby group messages in Google Maps

(ii) MessageView API: To view the message based on user's selection

(iii) MessageEdit API: To edit a particular message's details

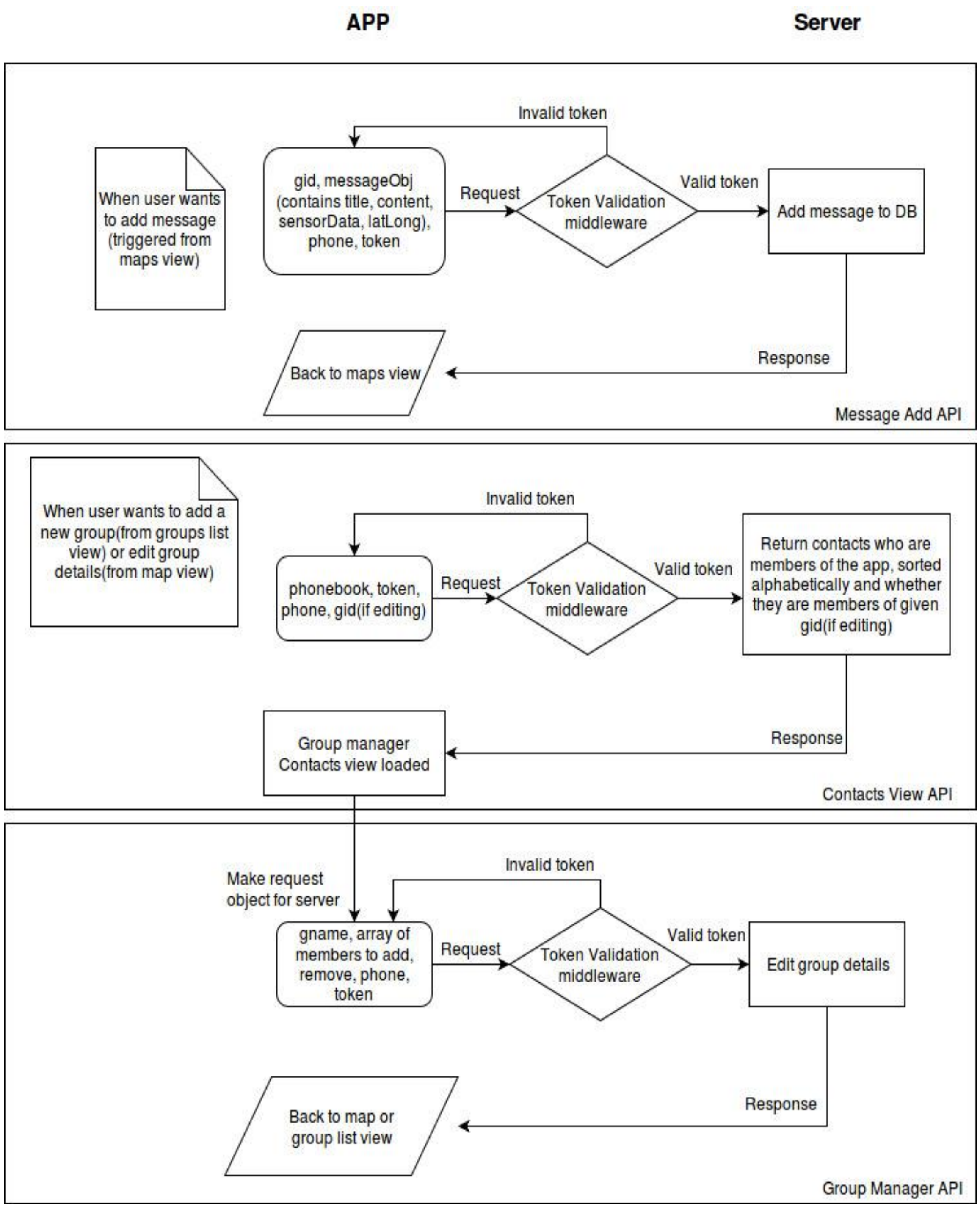

APP
Server
When user wants to add message (triggered from maps view)
gid, messageObj (contains title, content, sensorData, latLong), phone, token
Request
Invalid token
Token Validation middleware
Valid token
Add message to DB
Response
Back to maps view
Message Add API
When user wants to add a new group(from groups list view) or edit group details(from map view)
phonebook, token, phone, gid(if editing)
Request
Invalid token
Token Validation middleware
Valid token
Return contacts who are members of the app, sorted alphabetically and whether they are members of given gid(if editing)
Response
Group manager Contacts view loaded
Contacts View API
Make request object for server
gname, array of members to add, remove, phone, token
Request
Invalid token
Token Validation middleware
Valid token
Edit group details
Response
Back to map or group list view
Group Manager API

Figure 13: Manager API

(i) MessageAdd API: To add the message object into the database

(ii) ContactsView API: To view all the contacts who are members of the app

(iii) GroupManager API: To edit the name and members of the group

5. SUMMARY AND CONCLUSION

5.1 Summary

Through this project, an attempt has been made to implement a method to virtually connect stationary objects in the environment to the internet. The same concept has been used to experiment with a potential means of communication via augmented reality and in the process different object detection algorithms have been explored. In an attempt to make the messenger work efficiently the asynchronous, lightweight NodeJs has been used to implement the server. To make data access fast different databases have been used – Postgres SQL as the primary RDBMS and SOLR as the primary NOSQL database. Redis has been used to cache relevant data in memory for faster access and also for maintaining a job priority queue.

The User Interface has been attempted to be made as intuitive as possible by making use of google maps as the primary backdrop over which messages are displayed. To make the user interface better, proper drawing board designs were made and long discussions differentiating between what the user expected and what was needed were held. The massive nature of the project and a clear line of division between front-end and back-end served dual purposes – it allowed work to be modularly distributed and an understanding of how real life projects work out was developed in the team.

In spite of the time and commitment dedicated to this project there are still many areas lacking. The application is not a messenger per se since the messages are not sent to the user instantaneously, a request needs to be initiated from the client side – there is still scope for improvement in the actual user experience. Also, the object detection algorithms used work slower than acceptable, thus making tag detection a painful experience for the user.

Due to lack of time, the end product may not look market ready, but it clearly showcases that such a concept can be marketable if worked on vigorously. Given the time constraints and the fact that most of these technologies were new to the team at the beginning, large gains in terms of skill set and experience have been felt by the team.

5.2 Advantages and Disadvantages

There are numerous features of the project. The user is given provision of forming groups and updating the profile, where his name or display image can be changed just like a messenger. Apart from that, the user may create location or object based messages and share them among groups. They are also given facility of adding comments feed in a message. All the features of the project are available in form of a user-friendly Android app.

The project when made public would be available in Android Play Store free of cost. There would no extra charges nor any additional hardware. Anyone who owns a smartphone with a camera can experience the likes of the app.   Since most of the people are turning into smartphones users this project would be easily accessible. The ease of use and portability of these devices are an added advantage to the project.

There are a few disadvantages of the app as well. The app exhaustively uses the camera and the GPS of the mobile. Thus, it can lead to a faster battery drainage and the device might get heated up. Also, the object detection algorithms used are not as accurate as the user would expect.

5.3 Future Works

According to the Gartner's Hype Cycle of Innovative technologies of 2016, augmented reality is one of the critical technologies to be considered for the future. This project attempts to bring this technology in hands of the general public. For a better user experience, the app can be used along with AR headsets like the HoloLens, Sony PlayStation VR and so on. Thus, it would lead to lesser and easier user-interaction than it does while handling a handset.

5.4 Scope

The object detection algorithm used in the existing system is slow in performance. Also, it has a limited accuracy. To resolve the given issue, more advanced techniques of training thousands of images in a neural network can be used as an alternative.

6. TIMELINE

Tasks	Starting Date	Completion Date	Assignee	Description
• Brainstorming for project	November 2016	December 2016	Team	Devising ideas to use augmented reality with IoT
• Literature Survey	November 2016	December 2016	Team	Looking into object detection algorithms like SIFT,SURF and HAAR
• Learning the technology	January 17	February 17	Team	Study of basics of Android development , NodeJS, PostgresSQL and Solr
• Backend setup • Database models added	26 February	4 March	Akshat	Installation and Setup of the above technologies .Design of database schema
• User handler API structure • APIs: send_otp, login, contacts_view	5 March	11 March	Akshat	Backend development of the APIs for verifying OTP, authenticated login of the user and view of members of the app.
• Group handler API to create/update • Message handler API to create	12 March	18 March	Akshat	Backend development of the APIs for editing the group’s name and members. Also, edition of messages.

• Creation of KueJobs	19 March	25 March	Akshat	Priority job queue: optimize response time
• File transfer of images	26 March	1 April	Akshat	Image uploading SMS sending from Web service
• Custom List View Adapter • Designing Layout for Adapter	26 February	4 March	Antariksh	Design of a generic Adapter to view ListView for properties like messages,groups
• Google Map API • Acquiring location data • GPS with multiple providers • Google Map with Multiple Markers	5 March	11 March	Antariksh	Learning about GoogleMap API and its implementation and using Location Manager library
• Sensor Data Optimization	12 March –	18 March	Antariksh	
• SIFT implementation in C++	19 March	25 March	Antariksh	Implementation of basic object detection
• Standard template for UI of Android App. • UI design for Group Message page	26 March	1 April	Antariksh	UI design
• Accessing user contacts • Sending Contacts to the server	26 February	4 March	Niranj	Understanding the Contacts Content Library of Android and using it to

				retrieve and send contacts of the user.
• UI Design to view messages. • Searching messages. • UI Design for managing groups.	5 March	11 March	Niranj	Understading use of Adapters and use it to customise view in the UI
• Activity to edit group information	12 March	18 March	Niranj	UI design
• UI design of profile page. • Activity to edit user information	19 March	25 March	Niranj	UI design
• Sending updated information to the server. • Bug fixing.	26 March	8 April	Niranj	Understanding to connect the server using HTTP requests and responses. Testing for the same
• File transfer of Images	26 March	8 April	Akshat	Image uploading. SMS sending from Web service
• Group List View API • General Media upload and download	2 April	15 April	Akshat	Media storage and retrieval. Send Group List based on location.Manag-ing registered members in groups.
• Standard template for UI of Android App.	26 March	8 April	Antariksh	UI design

• UI design for Group Message page				
• Combining Modules • Horizontal List View for Gallery. • Camera feed data access	2 April	15 April	Antariksh	Integration of modules and design of a horizontal View for contacts.Getting data from camera feed.
• Group View • Add and removing members from group. • Profile Editor	2 April	15 April	Niranj	UI design of group view module. Developing an interactive UI for addition and deletion of group members.
• Capturing and cropping image to generate SURF keypoints Object Detection using SURF and boundary marking.	8 April	15 April	Antariksh	Implementing basic object detection and boundary making
• Bug Fixing • Managing groups of users	8 April	22 April	Niranj	Testing of the App.
• Indexing data on Solr • Retrieve nearby messages and groups.	8 April	15 April	Akshat	Understanding Solr and implementing the group List API
• Message Feed API • User related functions • Bug Fix for UpdateCreate API	15 April	1 May	Akshat	Developing the addMessageFeed API to add comment to messages

• MarkMessage API • Update Profile API • ViewAllMessages API	15 April	1 May	Niranj	Developing to mark the read status of message
• Server Side Object Detection.	15 April	1 May	Akshat/Antarish	Object detection in C++ using SURF (Antariksh) ,Backend Related APIs fir the same (Akshat)
• Front End Integration and model redesign	15 April	1 May	Antariksh	UI integration

7. REFERENCES AND CITATIONS

[1]. SURF: H. Bay, T. Tuytelaars, L. V. Gool, "SURF: Speeded up robust features", European Conference of Computer vision ECCV 2006, pp. 404-417, 2006.

[2]. SIFT: D. G. Lowe, "Distinctive image features from scale-invariant key-points", International Journal of computer vision, vol. 60, no. 2, pp. 91-110, 2004.

[3]. Haar Cascades: P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in IEEE Comp. Soc. Conf. on Computer Vision and Pattern Recognition. CVPR 2001, vol. 1. IEEE, 2001, pp. 1-511

[4]. TLD: W. Hailong, W. Guangyu and L. Jianxun, "An improved tracking-learning-detection method," 2015 34th Chinese Control Conference (CCC), Hangzhou, 2015, pp. 3858-3863.

[5]. https://www.theverge.com/2017/2/16/14634656/android-ios-market-share-blackberry-2016

[6]. https://developer.android.com/ndk/guides/abis.html

8. APPENDICES

8.1 System Setup

8.1.1 Software Requirements:

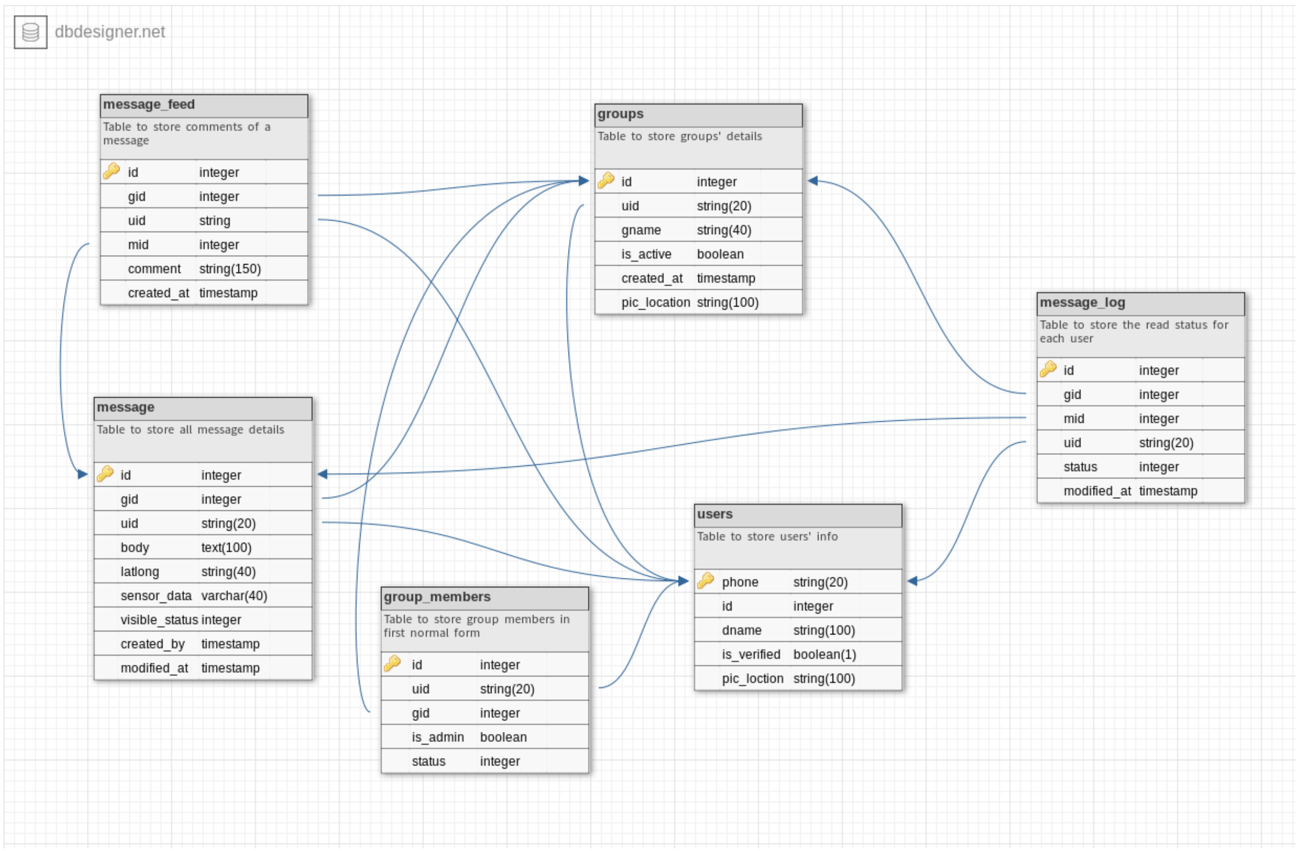
- Server
    - Operating System: Linux Debian [Ubuntu 16.04 LTS or Mint 18]
    - Solr 5.4.1 or better (for inverted geo indexing)
    - PostgreSQL 9.5.5 or better (as primary DBMS)
    - NodeJS 4.2.6 or better (as request handler)
    - Redis-server 3.0.6 or better (for caching relevant data)
- Application Development
    - Operating System: Windows 10
    - Visual Studio 2015
    - Software Development: Android Studio 2.3.1
    - OpenCV 3.2.0
    - OpenCV 3.2.0 Android SDK
    - OpenCV 3.2.0 Extra Modules
- Android Phone
    - Operating System: Android 5.0+

8.1.2 Hardware Requirements:

- Server:
    - 40 GB HDD
    - 4 core processor @ 2.60 GHz
    - 8 GB RAM

- Application Development:
    - 40 GB HDD
    - Intel i5-3230M @ 2.60 GHz
    - 8 GB RAM

8.2 Database Schema



message_feed
Table to store comments of a message

id	integer
gid	integer
uid	string
mid	integer
comment	string(150)
created_at	timestamp

groups
Table to store groups' details

id	integer
uid	string(20)
gname	string(40)
is_active	boolean
created_at	timestamp
pic_location	string(100)

message_log
Table to store the read status for each user

id	integer
gid	integer
mid	integer
uid	string(20)
status	integer
modified_at	timestamp

message
Table to store all message details

id	integer
gid	integer
uid	string(20)
body	text(100)
latlong	string(40)
sensor_data	varchar(40)
visible_status	integer
created_by	timestamp
modified_at	timestamp

users
Table to store users' info

phone	string(20)
id	integer
dname	string(100)
is_verified	boolean(1)
pic_loction	string(100)

group_members
Table to store group members in first normal form

id	integer
uid	string(20)
gid	integer
is_admin	boolean
status	integer

Figure 14: Database Schema

8.3 Source Code (Frontend)

The complete code can be found at https://github.com/antarikshnarain/GLAT

Here are some important android activities and java classes that helped in making the project a success.

8.3.1 nodeHttpRequest.java:

```
public class nodeHttpRequest extends AsyncTask<URLDataHash, Void, JSONObject> {

    private Context context;
    public String requestData;
    public nodeHttpRequest(Context context) {
        this.context = context;
    }

    protected void onPreExecute() {

    }

    @Override
    protected JSONObject doInBackground(URLDataHash... mydata) {
        //"192.168.43.231"
        /*
        Organizing data from mydata
         */
        String link="http://"+mydata[0].url+":8080"+"/"+mydata[0].apicall;
        // Sending data over HTTP
        BufferedReader bufferedReader;
        String result;
        try {
            String message = mydata[0].jsonData.toString();
            URL url = new URL(link);
            HttpURLConnection con = (HttpURLConnection) url.openConnection();
            con.setReadTimeout(10000);
            con.setConnectTimeout(15000);
            con.setRequestMethod("POST");
            con.setDoInput(true);
            con.setDoOutput(true);
            con.setChunkedStreamingMode(0);// Size of data unknown
            // making http header
            con.setRequestProperty("Content-Type",
"application/json;charset=utf-8");
            con.setRequestProperty("X-Requested-With", "XMLHttpRequest");
            //Connect
            con.connect();
            //setup send
            OutputStream os = new BufferedOutputStream(con.getOutputStream());
            os.write(message.getBytes());
            os.flush();
            bufferedReader = new BufferedReader(new
InputStreamReader(con.getInputStream()));
            result = bufferedReader.readLine();
            //con.disconnect();
            // Parsing String to JSON
            // note: all data received is in text format,
            Log.d("MYAPP",result);
            JSONObject myobj = new JSONObject(result);
            return myobj;

        } catch (Exception e) {
            //Toast.makeText(context,"Server Not 
Found",Toast.LENGTH_SHORT).show();
            e.printStackTrace();
            return null;
        }

    }

    private String getStringFromBitmap(Bitmap bitmap){
        // Image to String
        final int COMPRESSION_QUALITY = 100;
        ByteArrayOutputStream baos = new ByteArrayOutputStream();
        bitmap.compress(Bitmap.CompressFormat.JPEG, COMPRESSION_QUALITY, baos);
        byte[] mBytes = baos.toByteArray();
        return Base64.encodeToString(mBytes,Base64.DEFAULT);
    }
    private Bitmap getBitmapFromString(String imageString){
        // String to Image
        byte[] decodeString = Base64.decode(imageString,Base64.DEFAULT);
        return 
BitmapFactory.decodeByteArray(decodeString,0,decodeString.length);
    }
    private String getStringFromFile(String path){
        final StringBuilder stringBuilder = new StringBuilder();
        try{
            File file = new File(path);
            final InputStream inputStream = new FileInputStream(file);
            final BufferedReader reader = new BufferedReader(new 
InputStreamReader(inputStream));
            String line = reader.readLine();
            while(line != null){
                stringBuilder.append(line+'\n');
                line = reader.readLine();
            }
            reader.close();
            inputStream.close();
        }
        catch (FileNotFoundException e){
            e.printStackTrace();
        }
        catch (IOException e){
            e.printStackTrace();
        }

        Log.d("MYAPP: File Data",stringBuilder.toString());
        return stringBuilder.toString();
    }
    @Override
    protected void onPostExecute(JSONObject result) {
        super.onPostExecute(result);
        //this.requestData = result;
        //Log.d("Data",result);
    }
}
```

8.3.2 IncomingSms.java

```
public class IncomingSms extends BroadcastReceiver {
    public static final String SMS_BUNDLE = "pdus";
    @Override
    public void onReceive(Context context, Intent intent) {
        Bundle intentExtras = intent.getExtras();
        if (intentExtras != null) {
            Object[] sms = (Object[]) intentExtras.get(SMS_BUNDLE);
            String smsMessageStr = "";
            String smsBody = "";
            String smsAddress = "";
            for (int i = 0; i < sms.length; ++i) {
                SmsMessage smsMessage = SmsMessage.createFromPdu((byte[])
sms[i]);

                smsBody = smsMessage.getMessageBody().toString();
                smsAddress = smsMessage.getOriginatingAddress();
            }
            Log.d("MYAPP","Live Cap:"+smsAddress);
            //this will update the UI with message
            Login inst = Login.instance();
            inst.updateList(smsBody,smsAddress);
        }
    }
}
```

8.3.3 SensorData.java

```
public class SensorData implements SensorEventListener{
    Context mContext;

    SensorManager sensorManager;
    Sensor sensor_proximity, sensor_accelerometer, sensor_gyroscope,
sensor_magnetic, sensor_pressure;
    public float accelerometer[] = new float[3], gyroscope[] = new float[3],
pressure, magnetic[]= new float[3], proximity;

    public SensorData(Context context) {
        this.mContext = context;
        // Sensor Data
        sensorManager = (SensorManager)
mContext.getSystemService(SENSOR_SERVICE);
        sensor_proximity =
sensorManager.getDefaultSensor(Sensor.TYPE_PROXIMITY);
        sensorManager.registerListener(this, sensor_proximity,
SensorManager.SENSOR_DELAY_NORMAL);
        sensor_accelerometer =
sensorManager.getDefaultSensor(Sensor.TYPE_LINEAR_ACCELERATION);
        sensorManager.registerListener(this, sensor_accelerometer,
SensorManager.SENSOR_DELAY_NORMAL);
        sensor_gyroscope =
sensorManager.getDefaultSensor(Sensor.TYPE_GYROSCOPE);
        sensorManager.registerListener(this, sensor_gyroscope,
SensorManager.SENSOR_DELAY_NORMAL);
        sensor_magnetic =
sensorManager.getDefaultSensor(Sensor.TYPE_MAGNETIC_FIELD);
        sensorManager.registerListener(this, sensor_magnetic,
SensorManager.SENSOR_DELAY_NORMAL);
        sensor_pressure = sensorManager.getDefaultSensor(Sensor.TYPE_PRESSURE);
        sensorManager.registerListener(this, sensor_pressure,
SensorManager.SENSOR_DELAY_NORMAL);
        Log.d("MYAPP","Successfully initialized all the sensors!");
    }
    public void Unregister(){
        sensorManager.unregisterListener(this,sensor_accelerometer);
        sensorManager.unregisterListener(this,sensor_gyroscope);
        sensorManager.unregisterListener(this,sensor_magnetic);
        sensorManager.unregisterListener(this,sensor_pressure);
        sensorManager.unregisterListener(this,sensor_proximity);
        Log.d("MYAPP","Unregistered Sensors!");
    }
    @Override
    public void onSensorChanged(SensorEvent sensorEvent) {
        if (sensorEvent.sensor.getType() == Sensor.TYPE_LINEAR_ACCELERATION){
            accelerometer[0] = sensorEvent.values[0];
            accelerometer[1] = sensorEvent.values[1];
            accelerometer[2] = sensorEvent.values[2];
        }
        if (sensorEvent.sensor.getType() == Sensor.TYPE_PROXIMITY){
            proximity = sensorEvent.values[0];
        }
        if (sensorEvent.sensor.getType() == Sensor.TYPE_GYROSCOPE){
            gyroscope[0] = sensorEvent.values[0];
            gyroscope[1] = sensorEvent.values[1];
            gyroscope[2] = sensorEvent.values[2];
        }
        if (sensorEvent.sensor.getType() == Sensor.TYPE_MAGNETIC_FIELD){
            magnetic[0] = sensorEvent.values[0];
            magnetic[1] = sensorEvent.values[1];
            magnetic[2] = sensorEvent.values[2];
        }
        if (sensorEvent.sensor.getType() == Sensor.TYPE_PRESSURE) {
            pressure = sensorEvent.values[0];
        }

//if(mContext.checkSelfPermission(Manifest.permission.ACCESS_FINE_LOCATION)==
PackageManager.PERMISSION_GRANTED)
        //    location =
locationManager.getLastKnownLocation(LocationManager.GPS_PROVIDER);
    }



}
```

8.3.4 Custom Data Structure: URLDataHash.java

```
public class URLDataHash {
    public String url;
    public String apicall;
    public JSONObject jsonData;
    public URLDataHash(){
        url = apicall = "";
        jsonData = new JSONObject();
    }
}
```

8.3.5 Custom Data Structure: MapMessages.java

```
public class MapMessages {
    public double latitude;
    public double longitude;
    public String summary;
    public int gid;
    public int mid;
    public String createdby;
    public int message_state;
    public MapMessages(){
        latitude = longitude = 0.0;
        createdby = summary = "";
        message_state = gid = mid = 0;
    }


}
```

8.3.6 SharedFunctions.java:

```
public class SharedFunctions {

    private Context context;
    //Shared Preferences
    private SharedPreferences sharedPreferences;
    private SharedPreferences.Editor sharedPrefEditor;
    public String user, token, phone, user_profile_pic;

    // Bitmap Variables
    private final int imgWidth = 120;
    private final int imgHeight = 120;

    // Server Constants
    public final String serverUrl = "52.172.193.163";//"192.168.43.231";
    // External Storage Directory
    public final String root_path;
    // Request Codes
    public final static int GALLERY_IMAGE = 102;
    public final static int IMAGE_CLICK = 401;
    public final static int SUCCESS= 500;
    // Constructor
    public SharedFunctions(Context mContext){
        context = mContext;
        root_path =
Environment.getExternalStorageDirectory().getPath()+"/GeoMania/";
        String[] localdirs = {"temp","profile_pic","group_icon","object_file"};
        for (String a: localdirs
             ) {
            File mydir = new File(root_path+a);
            if(!mydir.isDirectory())
                mydir.mkdirs();
        }
        sharedPreferences = context.getSharedPreferences("userdata",
Context.MODE_PRIVATE);
```

```
        sharedPrefEditor = sharedPreferences.edit();
        user = sharedPreferences.getString("user","");
        phone = sharedPreferences.getString("phone","");
        token = sharedPreferences.getString("token","");
        user_profile_pic = sharedPreferences.getString("user_profile_pic","");
    }
    // Update Shared Preferences (user, phone, token)
    public void updateSharedPreference(String key, String value){
        sharedPrefEditor.putString(key,value);
        sharedPrefEditor.commit();
        if(key.equals("user"))
            user = value;
        else if(key.equals("phone"))
            phone = value;
        else if(key.equals("token"))
            token = value;
        else if(key.equals("user_profile_pic"))
            user_profile_pic = value;
    }
    // Function to resize Bitmap from File for the Application
    public Bitmap resizeBitmap(String filePath){
        BitmapFactory.Options bmOptions = new BitmapFactory.Options();
        bmOptions.inJustDecodeBounds = true;
        BitmapFactory.decodeFile(filePath, bmOptions);
        int photoW = bmOptions.outWidth;
        int photoH = bmOptions.outHeight;
        int scaleFactor = 1;
        if ((imgWidth > 0) || (imgHeight > 0)) {
            scaleFactor = Math.max(photoW/imgWidth, photoH/imgHeight);
        }
        bmOptions.inJustDecodeBounds = false;
        bmOptions.inSampleSize = scaleFactor;
        return BitmapFactory.decodeFile(filePath, bmOptions);
    }
    // Function get Get Path from Uri
    public String getRealPathFromURI(Uri uri) {
        String[] projection = { MediaStore.Images.Media.DATA };
        @SuppressWarnings("deprecation")
        Cursor cursor = context.getContentResolver().query(uri, projection,
null, null, null);
        int column_index = cursor
                .getColumnIndexOrThrow(MediaStore.Images.Media.DATA);
        cursor.moveToFirst();
        return cursor.getString(column_index);
    }
    // File to String
    public String FileToBase64(File filePath ){
        try {
            Log.d("MYAPP","Starting Image to Base64");
            InputStream inputStream = new FileInputStream(filePath);
            byte[] bytes;
            byte[] buffer = new byte[8192];
            int bytesRead;
            ByteArrayOutputStream output = new ByteArrayOutputStream();
            while((bytesRead = inputStream.read(buffer)) !=-1){
                output.write(buffer,0,bytesRead);
            }
            bytes = output.toByteArray();
            Log.d("MYAPP","Starting Image to Base64 Sending Data");
            return Base64.encodeToString(bytes,Base64.DEFAULT);
        }
        catch(FileNotFoundException e){
            e.printStackTrace();
        }
        catch (IOException e){
            e.printStackTrace();
        }
        return "";
    }
    // String to File
    public void Base64ToFile(String data, File filename){
        try {
            Log.d("MYAPP","Starting Base64 to File");
            byte[] bytes = Base64.decode(data, Base64.DEFAULT);
            FileOutputStream out = new FileOutputStream(filename);
            out.write(bytes);
            Log.d("MYAPP","File Written");
            out.close();
        }
        catch(IOException e){
            e.printStackTrace();
        }
    }
    // Get ProfilePic from folder, missing then request
    public Bitmap setPicture(String filename, int category){
        if(filename.equals("")){
            if(category == 1)
                return
BitmapFactory.decodeResource(context.getResources(),R.drawable.profile_icon);
            else if(category == 2)
                return BitmapFactory.decodeResource(context.getResources(),
R.mipmap.ic_launcher);
            else
                return BitmapFactory.decodeResource(context.getResources(),
R.mipmap.ic_launcher);
        }
        String imagePath = "";
        if(category == 1)
            imagePath = root_path + "profile_pic/" + filename;
        else if (category == 2)
            imagePath = root_path + "group_icon/" + filename;
        else
            return null;
        File mFile = new File(imagePath);
        if(mFile.isFile())
            return resizeBitmap(imagePath);
        // File Doesnot Exist Request Server for download
        else {
            if (downloadFile(imagePath, filename))
                return resizeBitmap(imagePath);
        }
        return null;
    }
    // Download file(filename) to file_path
    public boolean downloadFile(String file_path, String filename){
        // Request Server
        try {
            URLDataHash mydata = new URLDataHash();
            mydata.jsonData.put("fileName", filename);
            mydata.jsonData.put("phone", phone);
            mydata.jsonData.put("token", token);
            mydata.url = serverUrl;
            mydata.apicall = "file/download";
            JSONObject data = new
nodeHttpRequest(context).execute(mydata).get();
            Toast.makeText(context, "Data Send to Server!",
Toast.LENGTH_SHORT).show();
            if (data.getString("status").equals("success")) {
                Toast.makeText(context, "File Downloaded Successfully",
Toast.LENGTH_SHORT).show();
                // Saving File after download
                Base64ToFile(data.getString("resp"), new File(file_path));
                return true;
            }
            else{
                Toast.makeText(context, "Failed To Download!",
Toast.LENGTH_SHORT).show();
            }
        }
        catch (JSONException e){ e.printStackTrace(); }
        catch (InterruptedException e){ e.printStackTrace(); }
        catch (ExecutionException e) { e.printStackTrace(); }
        return false;
    }
    // Upload File(temp_path) and save it to file_path
    public String uploadFile(String temp_path, String file_path){
        try{
            URLDataHash mydata = new URLDataHash();
            mydata.jsonData.put("file",FileToBase64(new File(temp_path)));
            mydata.jsonData.put("phone",phone);
            mydata.jsonData.put("token",token);
            mydata.url=serverUrl;
            mydata.apicall="file/upload";
            JSONObject data = new
nodeHttpRequest(context).execute(mydata).get();
            Toast.makeText(context,"Data Send to
Server!",Toast.LENGTH_SHORT).show();
            if (data.getString("status").equals("success")) {
                Toast.makeText(context, "File Uploaded Successfully",
Toast.LENGTH_SHORT).show();
                // Saving File after download resp="filename"
                file_path += data.getString("resp");
                Base64ToFile(FileToBase64(new File(temp_path)), new
File(file_path));
                Log.d("MYAPP: Upload", "File Uploaded "+file_path);
                return data.getString("resp");
            }
            else{
                Toast.makeText(context, "Failed To Upload!",
Toast.LENGTH_SHORT).show();
            }
        }
        catch (JSONException e){ e.printStackTrace(); }
        catch (InterruptedException e){ e.printStackTrace(); }
        catch (ExecutionException e) { e.printStackTrace(); }
        return null;
    }
    public void askForPermission(String permission, Integer requestCode){
        if(ContextCompat.checkSelfPermission(context,permission) !=
PackageManager.PERMISSION_GRANTED){
            ActivityCompat.requestPermissions(null,new String[]{permission},
requestCode);
        }
    }
}
```

8.3.7 MyCamera.java

```
public class MyCamera extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_my_camera);
        if (null == savedInstanceState) {
            getFragmentManager().beginTransaction()
                    .replace(R.id.container_camera,
Camera2BasicFragment.newInstance())
                    .commit();
        }
    }
}
```

8.3.8 GroupManager.java

```
public class GroupManager extends AppCompatActivity {
 ListView listview;
 ListView listview1;
 EditText groupName;
 ImageView groupIcon;
 String group_name, group_icon, group_id;

ProgressDialog progressDialog;

SharedFunctions myfunction;
 ArrayList<String> latestNumbers = new ArrayList<>();
 ArrayList<String> latestMembers = new ArrayList<String>();
 JSONObject NumbersHash = new JSONObject();
 int GALLERY_IMAGE = 102;
 @Override
 protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 Intent intent = getIntent();
 group_name = intent.getStringExtra("title");
 group_icon = intent.getStringExtra("icon");
 group_id = intent.getStringExtra("gid");
 setContentView(R.layout.activity_group_manager);

groupName = (EditText) findViewById(R.id.groupName);
 listview = (ListView) findViewById(R.id.contactsView);
 listview1 = (ListView) findViewById(R.id.membersView);
 groupIcon = (ImageView) findViewById(R.id.image_group_icon);
 // Get Read Contacts Permission
 if(checkSelfPermission(Manifest.permission.READ_CONTACTS) !=
PackageManager.PERMISSION_GRANTED){
 ActivityCompat.requestPermissions(this, new
String[]{Manifest.permission.READ_CONTACTS}, 203);
 }
 progressDialog = new ProgressDialog(this);
 myfunction = new SharedFunctions(this);
 groupName.setText(group_name);
 if(group_icon.equals("") || group_icon.equals("null")){
 groupIcon.setImageResource(R.mipmap.ic_launcher);
 }
 else{
 groupIcon.setImageBitmap(myfunction.resizeBitmap(group_icon));
 }
 progressDialog.setTitle(":((())):");
 progressDialog.setMessage("Retrieving Group Information ...");
 progressDialog.show();
 showContacts();
 progressDialog.dismiss();
 // getUserPhoneNumber();
 }
 @Override
 public void onActivityResult(int requestCode, int resultCode, Intent data){
 super.onActivityResult(requestCode,resultCode,data);
 if(requestCode == GALLERY_IMAGE){
 // Gallery Images
 if (resultCode == Activity.RESULT_OK){
 Uri selectedImage = data.getData();
 String filePath = getRealPathFromURI(selectedImage);
 groupIcon.setImageBitmap(myfunction.resizeBitmap(filePath));
 }
 }
 }
 public String getRealPathFromURI(Uri uri) {
 String[] projection = { MediaStore.Images.Media.DATA };
 @SuppressWarnings("deprecation")
 Cursor cursor = managedQuery(uri, projection, null, null, null);
 int column_index = cursor
 .getColumnIndexOrThrow(MediaStore.Images.Media.DATA);
 cursor.moveToFirst();
 return cursor.getString(column_index);
 }
 // OnClick Function
 public void changeGroupIcon(View v){
 Intent intent = new Intent();
 intent.setType("image/*");
 //intent.putExtra(Intent.EXTRA_ALLOW_MULTIPLE,true);
 intent.setAction(Intent.ACTION_GET_CONTENT);
 startActivityForResult(Intent.createChooser(intent,"Select
Images"),GALLERY_IMAGE);
 }
 // Get all the phone book contacts
 private JSONArray getContactNames() throws JSONException {
 Cursor contacts =
getContentResolver().query(ContactsContract.CommonDataKinds.Phone.CONTENT_URI,
null, null, null, null);
 String aNameFromContacts[] = new String[contacts.getCount()];
 String aNumberFromContacts[] = new String[contacts.getCount()];
 int i = 0;
 int nameFieldColumnIndex =
contacts.getColumnIndex(ContactsContract.CommonDataKinds.Phone.DISPLAY_NAME);
 int numberFieldColumnIndex =
contacts.getColumnIndex(ContactsContract.CommonDataKinds.Phone.NUMBER);

JSONArray returnContacts = new JSONArray();
 while (contacts.moveToNext()) {
 JSONObject currJsonContact = new JSONObject();
 String contactName = contacts.getString(nameFieldColumnIndex);
 aNameFromContacts[i] = contactName;
 contactName = contactName.replaceAll("'", "\'");
 currJsonContact.put("name", contactName);
 String number = contacts.getString(numberFieldColumnIndex);
 aNumberFromContacts[i] = number;
 number = number.replaceAll("'", "\'");
 currJsonContact.put("phone", number);
 i++;
 returnContacts.put(currJsonContact);
 }
 contacts.close();
 return returnContacts;
 }
 // Show members and non members of the Group
 private void showContacts() {
 try {
 JSONArray contactObjects = getContactNames();
 JSONObject requestMap = new JSONObject();
 requestMap.put("phone", myfunction.phone);
 requestMap.put("token", myfunction.token);
 requestMap.put("contacts", contactObjects);
 //Log.d("MYAPP Contacts:", contactObjects.toString());
 URLDataHash mydata = new URLDataHash();
 mydata.url = myfunction.serverUrl;
 mydata.apicall = "user/contacts/view";
 mydata.jsonData = requestMap;
 // Sending Request to server
 final JSONObject data = new nodeHttpRequest(this).execute(mydata).get();
 if (data == null) {
 // No Data found
 return;
 }
 JSONArray members = data.getJSONArray("resp");
 JSONObject currentObj;

final ArrayList<String> membersGroup = new ArrayList<String>();
 for (int i = 0; i < members.length(); i++) {
 currentObj = members.getJSONObject(i);
 //Log.d("MYAPP: Json Parse", currentObj.toString());
 membersGroup.add(currentObj.getString("dname"));
 NumbersHash.put(currentObj.getString("dname"), currentObj.getString("phone"));
 //Log.d("MYAPP: members group", membersGroup.toString());
 }
 Log.d("MYAPP: Members", "Total Members: " + membersGroup.size());

// membersGroup.add("Praful");
 final ArrayAdapter adapter = new ArrayAdapter<String>(this,
android.R.layout.simple_list_item_1, membersGroup);
 listview.setAdapter(adapter);
 final ArrayAdapter adaptermembers = new ArrayAdapter<String>(this,
android.R.layout.simple_list_item_1, latestMembers);
 listview1.setAdapter(adaptermembers);
 listview.setOnItemClickListener(new AdapterView.OnItemClickListener() {
 @Override
 public void onItemClick(AdapterView<?> parent, View view, int position, long
id) {
 String item = ((TextView) view).getText().toString();
 membersGroup.remove(item);
 latestMembers.add(item);
 adapter.notifyDataSetChanged();
 adaptermembers.notifyDataSetChanged();
 }
 });
 listview1.setOnItemClickListener(new AdapterView.OnItemClickListener() {
 @Override
 public void onItemClick(AdapterView<?> parent, View view, int position, long
id) {
 String item = ((TextView) view).getText().toString();
 membersGroup.add(item);
 latestMembers.remove(item);
 adapter.notifyDataSetChanged();
 adaptermembers.notifyDataSetChanged();
 }
 });





}
 catch (JSONException e) { e.printStackTrace(); }
 catch (InterruptedException e) { e.printStackTrace(); }
 catch (ExecutionException e) { e.printStackTrace(); }
 }
 // OnClick UpdateGroupInformation
 public void UpdateGroupInformation(View v) {
 try {
 // To handle change of Group name
 group_name = groupName.getText().toString();
 JSONObject requestMap = new JSONObject();
 requestMap.put("phone", myfunction.phone);
 requestMap.put("token", myfunction.token);
 requestMap.put("gid", group_id);
 requestMap.put("gname", group_name);
 for (String current : latestMembers)
 latestNumbers.add(NumbersHash.getString(current));
 Log.d("Latest members earlier",""+latestNumbers.size());
 Log.d("Latest members ssize",""+latestNumbers.size());
 requestMap.put("mems", new JSONArray(latestNumbers));
 URLDataHash mydata = new URLDataHash();
 mydata.url = myfunction.serverUrl;
 mydata.apicall = "group/updateOrCreate";
 mydata.jsonData = requestMap;

// Making request to server
 JSONObject data = new
nodeHttpRequest(getApplicationContext()).execute(mydata).get();
 Log.d("Response Group ", data.toString());
 Toast.makeText(getApplicationContext(), "Group has been Created / Updated",
Toast.LENGTH_LONG).show();
 finish();
 }
 catch (JSONException e) { e.printStackTrace(); }
 catch (InterruptedException e) { e.printStackTrace(); }
 catch (ExecutionException e) { e.printStackTrace(); }
 }
 @Override
 public void onRequestPermissionsResult(int requestCode, String[] permissions,
 int[] grantResults) {
```

```
 if (requestCode == 203) {
 if (grantResults[0] == PackageManager.PERMISSION_GRANTED) {
 // Permission is granted
 showContacts();
 } else {
 Toast.makeText(this, "Until you grant the permission, we cannot display the
names", Toast.LENGTH_SHORT).show();
 }
 }
 }

}
```

8.3.9 LocationMessage.java

```
public class LocationMessage extends AppCompatActivity {

TextView tv_latitude, tv_longitude;
 EditText user_message;
 double gps_longitude, gps_latitude;
 int TYPE;
 UserSendMessage message;
 String imageData;
 int GALLERY_IMAGE = 102;
 int CAPTURE_IMAGE = 105;
 int CROP_IMAGE = 106;
 Uri picUri;

SharedFunctions myfunction;

String phone, token, group_id;
 int type;
 // View
 LinearLayout media_list;
 CustomGroupListAdapter_MediaList adapter;
 ArrayList<String> itemTitle = new ArrayList<String>();
 ArrayList<String> itemImage = new ArrayList<String>();
 int adapterCount;
 @Override
 protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.activity_location_message);
 // Getting Data from MapsActivity
 Intent data = getIntent();
 gps_latitude = data.getDoubleExtra("latitude",0.0);
 gps_longitude = data.getDoubleExtra("longitude",0.0);
 group_id = data.getStringExtra("gid");
 type = data.getIntExtra("type",0);
 // Horizontal List View
 adapter = new CustomGroupListAdapter_MediaList(this,itemTitle,itemImage);
 adapterCount = adapter.getCount();
 media_list = (LinearLayout) findViewById(R.id.horizontal_list_view);
 tv_latitude = (TextView) findViewById(R.id.my_latitude);
 tv_longitude = (TextView) findViewById(R.id.my_longitude);
 tv_latitude.setText("Latitude: "+gps_latitude);
 tv_longitude.setText("Longitude: "+gps_longitude);
 }
 @Override
 public void onActivityResult(int requestCode, int resultCode, Intent data){
 super.onActivityResult(requestCode,resultCode,data);
 if(requestCode == GALLERY_IMAGE){
 // Gallery Images
 if (resultCode == Activity.RESULT_OK){
 Uri selectedImage = data.getData();
 String filePath = getRealPathFromURI(selectedImage);
 Log.d("MYAPP: FilePath",filePath);
 itemTitle.add("Image");
 itemImage.add(filePath);
 media_list.addView(adapter.getView(adapterCount++,null,null));
 }
 }
 else if(requestCode == CAPTURE_IMAGE){
 picUri = data.getData();
 performCrop();
 }
 else if(requestCode == CROP_IMAGE){
 Bundle extras = data.getExtras();
 Bitmap bitmap = (Bitmap) extras.get("data");
 saveCroppedImage(bitmap);
 ImageView mImage = (ImageView) findViewById(R.id.imageView2);
 mImage.setImageBitmap(bitmap);

}
 }
 public String getRealPathFromURI(Uri uri) {
 String[] projection = { MediaStore.Images.Media.DATA };
 @SuppressWarnings("deprecation")
 Cursor cursor = managedQuery(uri, projection, null, null, null);
 int column_index = cursor
 .getColumnIndexOrThrow(MediaStore.Images.Media.DATA);
 cursor.moveToFirst();
 return cursor.getString(column_index);
 }
 public void performCrop(){
 try{
 Intent cropImageIntent = new Intent("com.android.camera.action.CROP");
 cropImageIntent.setDataAndType(picUri, "image/*");
 cropImageIntent.putExtra("crop","true");
 cropImageIntent.putExtra("aspectX",1);
 cropImageIntent.putExtra("aspectY",1);
 cropImageIntent.putExtra("outputX",1024);
 cropImageIntent.putExtra("outputY",1024);
 //cropImageIntent.putExtra("scale",false);
 cropImageIntent.putExtra("return-data",true);
 startActivityForResult(cropImageIntent,CROP_IMAGE);
 }
 catch (ActivityNotFoundException e){
 e.printStackTrace();
 }
 }
 private void saveCroppedImage(Bitmap mBitmap){
 FileOutputStream fileOutputStream = null;
 try{
 fileOutputStream = new
FileOutputStream(Environment.getExternalStorageDirectory().getPath()+"/GeoMania
/image123.png");
```

```
 mBitmap.compress(Bitmap.CompressFormat.PNG,100, fileOutputStream);
 Toast.makeText(getApplicationContext(),"Image Saved
Successfully",Toast.LENGTH_SHORT).show();
 }
 catch (Exception e){
 e.printStackTrace();
 }
 finally {
 try{
 if(fileOutputStream != null){
 fileOutputStream.close();
 }
 }catch (IOException e){
 e.printStackTrace();
 }
 }
 }
 public void send_geo_message(View v){
 //message.msg.text = user_message.getText().toString();
 //setResult(101,new Intent().putExtra("LocationMessageData",message));
 //finish();
 // Get image from Gallary
 //Intent photoPickerIntent = new Intent(Intent.ACTION_PICK);
 //photoPickerIntent.setType("image/*");
 //startActivityForResult(photoPickerIntent,102);
 File sdcard = Environment.getExternalStorageDirectory();
 // to this path add a new directory path
 File dir = new File(sdcard.getAbsolutePath() + "/GeoMania/"+"image1.jpg");
 File dir2 = new File(sdcard.getAbsolutePath() + "/GeoMania/"+"image3.jpg");
 //Base64ToImage(ImageToBase64(dir),dir2);
 try{
 sendDataToServer(dir,dir2,"image1.jpg");
 }
 catch(JSONException e){
 e.printStackTrace();
 }
 }
 public void object_generator(View v){
 Intent captureImageIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
 startActivityForResult(captureImageIntent,CAPTURE_IMAGE);
 }
 public void addMedia(View v){
 // Getting multiple images from gallery
 Intent intent = new Intent();
 intent.setType("image/*");
 //intent.putExtra(Intent.EXTRA_ALLOW_MULTIPLE,true);
 intent.setAction(Intent.ACTION_GET_CONTENT);
 startActivityForResult(Intent.createChooser(intent,"Select
Images"),GALLERY_IMAGE);
 }

public void sendDataToServer(File dir1, File dir2,String filename) throws
JSONException{
 URLDataHash mydata = new URLDataHash();
 mydata.jsonData.put("image",myfunction.FileToBase64(dir1));
 mydata.jsonData.put("fileName",filename);
 mydata.url="192.168.43.231";
 mydata.apicall="imageTest";//"user/signup";
 //mydata.hashMap=hashMap;
 try {
 JSONObject data = new nodeHttpRequest(this).execute(mydata).get();
 Toast.makeText(getApplicationContext(),"Data Send to
Server!",Toast.LENGTH_SHORT).show();
 String fileData = data.getString("image");
 String filename_new = data.getString("fileName");
 myfunction.Base64ToFile(fileData,dir2);
 }
 catch (JSONException e){
 e.printStackTrace();
 }
 catch (InterruptedException e){
 e.printStackTrace();
 }
 catch (ExecutionException e) {
 e.printStackTrace();
 }
 }
 }
```

8.3.10 MapsActivity.java:

```
public class MapsActivity extends AppCompatActivity implements
OnMapReadyCallback, LocationListener, GoogleMap.OnInfoWindowClickListener {

private GoogleMap mMap;
 private Marker myMarker;
 boolean isGPSEnabled = false,isNetworkEnabled = false, canGetLocation = false;
 Location location;
 public double gps_longitude=0.0, gps_latitude=0.0;
 public String provider;
 final long MIN_DISTANCE_CHANGE_FOR_UPDATES = 2; // 10 Meters
 final long MIN_TIME_BW_UPDATES = 1000 * 2 * 1; // 1 minute
 protected LocationManager locationManager;

// Map Markers
 public ArrayList<Marker> map_messages;
 public ArrayList<MapMessages> messages;
 // Writing to File
 String filename = "MyGPSData.csv";
 FileOutputStream outputStream;
 File sdcard = Environment.getExternalStorageDirectory();
 // to this path add a new directory path
 File dir = new File(sdcard.getAbsolutePath() + "/GeoMania/");

// Layout variables
 SearchView mSearchView;
 ImageButton imgButton_location, imgButton_no_location, imgButton_location_tag,
imgButton_no_location_tag;
 TextView tv_message_type;
 ProgressDialog progressDialog;

// Data control variables
 String group_name, group_icon, group_id;
 boolean first_flag = true;
 SharedFunctions myfunction;
```

```
// Intents
 Intent location_message_intent;
 @Override
 protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.activity_maps);
 Intent intent = getIntent();
 group_name = intent.getStringExtra("title");
 group_icon = intent.getStringExtra("icon");
 group_id = intent.getStringExtra("gid");
 myfunction = new SharedFunctions(this);
 // Obtain the SupportMapFragment and get notified when the map is ready to be
used.
 progressDialog = new ProgressDialog(this);
 progressDialog.setTitle(":((())):");
 progressDialog.setMessage("Getting Location");
 progressDialog.show();
 // Progress required to get group data
 SupportMapFragment mapFragment = (SupportMapFragment)
getSupportFragmentManager()
 .findFragmentById(R.id.map);
 getLocation();
 mapFragment.getMapAsync(this);

// Declaring Layout variables
 mSearchView = (SearchView) findViewById(R.id.searchView);
 imgButton_location = (ImageButton) findViewById(R.id.imageButton_location);
 imgButton_location_tag = (ImageButton)
findViewById(R.id.imageButton_location_tag);
 imgButton_no_location = (ImageButton)
findViewById(R.id.imageButton_no_location);
 imgButton_no_location_tag = (ImageButton)
findViewById(R.id.imageButton_no_location_tag);
 tv_message_type = (TextView) findViewById(R.id.textView_message_type);

// Initializing messages variable for map
 location_message_intent = new Intent().setClass(this,LocationMessage.class);
 location_message_intent.putExtra("gid", group_id);
 map_messages = new ArrayList<Marker>();
 messages = new ArrayList<MapMessages>();
 }

 @Override
 public void onMapReady(GoogleMap googleMap) {
 // Add a marker in Sydney, Australia,
 // and move the map's camera to the same location.
 mMap = googleMap;
 mMap.setOnInfoWindowClickListener(this);
 loadGroupMessages();
 }
 @Override
 public void onLocationChanged(Location location) {
 if (location != null) {
 if(first_flag){
 progressDialog.dismiss();
 first_flag = false;
 }
 gps_latitude = location.getLatitude();
 gps_longitude = location.getLongitude();
 updateMap(gps_latitude,gps_longitude);
 }
 Log.d("MYAPP", "Updating My Location!");
 }

@Override
 public void onProviderDisabled(String provider) {
 }

@Override
 public void onProviderEnabled(String provider) {
 }

@Override
 public void onStatusChanged(String provider, int status, Bundle extras) {
 }
 @Override
 public void onInfoWindowClick(Marker marker){
 int id = (int) marker.getTag();
 // Start Message Feed Activity
 Intent message_feed_intent = new
Intent().setClass(this,MessageViewFeed.class);
 message_feed_intent.putExtra("gid", messages.get(id).gid);
 message_feed_intent.putExtra("mid", messages.get(id).mid);
 startActivity(message_feed_intent);
 }
 public void getLocation() {
 try {
 locationManager = (LocationManager) this.getSystemService(LOCATION_SERVICE);
 // getting GPS status
 isGPSEnabled =
locationManager.isProviderEnabled(LocationManager.GPS_PROVIDER);
 // getting network status
 isNetworkEnabled =
locationManager.isProviderEnabled(LocationManager.NETWORK_PROVIDER);
 if (this.checkSelfPermission(android.Manifest.permission.ACCESS_FINE_LOCATION)
== PackageManager.PERMISSION_GRANTED) {
 Log.d("MYAPP", "Location Manager Successfully Created");
 }
 if (!isGPSEnabled && !isNetworkEnabled) {
 // no network provider is enabled
 } else {
 // First get location from Network Provider
 //locationManager.requestLocationUpdates(LocationManager.PASSIVE_PROVIDER, 0,
0, this);
 locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER,1000, 5,
this);
 locationManager.requestLocationUpdates(LocationManager.NETWORK_PROVIDER, 1000,
5, this);
 Criteria criteria = new Criteria();
 //criteria.setAccuracy(Criteria.ACCURACY_FINE);
 //criteria.setPowerRequirement(Criteria.POWER_HIGH);
 //criteria.setAltitudeRequired(true);
 String provider = locationManager.getBestProvider(criteria, false);
 Toast.makeText(this.getApplicationContext(), "Current Provider: " + provider,
Toast.LENGTH_SHORT).show();
 //locationManager.requestLocationUpdates(provider, MIN_TIME_BW_UPDATES,
MIN_DISTANCE_CHANGE_FOR_UPDATES, this);
 if (locationManager != null) {
 location = locationManager
 .getLastKnownLocation(provider);
 if (location != null) {
 gps_latitude = location.getLatitude();
 gps_longitude = location.getLongitude();
 } else {
 Toast.makeText(this, "Last Location Unknown", Toast.LENGTH_SHORT).show();
 }
 } else {
 Log.d("MYAPP", "Location Manager NULL");
 }
 Log.d("MYAPP", "Location Manager Initialized!");
 }
 } catch (Exception e) {
 e.printStackTrace();
 }
 }


public void addMessageToMap(int id, double latitude, double longitude, int
message_state, String summary, int gid, int mid, String createdby ) {
 MapMessages messageFeed = new MapMessages();
 messageFeed.latitude = latitude;
 messageFeed.longitude = longitude;
 messageFeed.summary = summary;
 messageFeed.gid = gid;
 messageFeed.mid = mid;
 messageFeed.createdby = createdby;
 messageFeed.message_state = message_state;
 // 0-> unread, 1-> read, 2-> mine
 Marker marker = null;
 Log.d("MYAPP: state", ""+message_state);
 if (message_state == 1) {
 marker = mMap.addMarker(new MarkerOptions().position(new LatLng(latitude,
longitude)).icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.
HUE_GREEN)));
 }
 else if (message_state == 0) {
 marker = mMap.addMarker(new MarkerOptions().position(new LatLng(latitude,
longitude)).icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.
HUE_YELLOW)));
 }
 else if (message_state == 2) {
 marker = mMap.addMarker(new MarkerOptions().position(new LatLng(latitude,
longitude)).icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.
HUE_AZURE)));
 }
 else{
 return;
 }
 marker.setTag(id);
 marker.setTitle(createdby);
 if (summary.length()>40)
 marker.setSnippet(summary.substring(0,40)+"...");
 else
 marker.setSnippet(summary+"...");
 map_messages.add(marker);
 messages.add(id,messageFeed);
 Log.d("MYAPP: marker", marker.getTitle() + marker.getPosition().toString());
 }

public void loadGroupMessages(){
 try{
 JSONObject requestMap = new JSONObject();
 requestMap.put("gid", group_id);
 requestMap.put("phone", myfunction.phone);
 requestMap.put("token", myfunction.token);
 requestMap.put("lat",Double.toString(gps_latitude));
 requestMap.put("long",Double.toString(gps_longitude));
 Log.d("MYAPP: RequestData",requestMap.toString());
 URLDataHash mydata = new URLDataHash();
 mydata.url = "192.168.43.231";
 mydata.apicall = "group/showAllMessages";
 mydata.jsonData=requestMap;
 // Request the server
 JSONObject data = new nodeHttpRequest(this).execute(mydata).get();
 if(data == null){
 Log.d("MYAPP: ServerResp","Error during server request");
 ArrayList<String> t_name, t_icon, t_unread;
 t_name = new ArrayList<String>();
 t_icon = new ArrayList<String>();
 t_unread = new ArrayList<String>();
 t_name.add("Alpha");
 t_icon.add("null");
 t_unread.add("Unread: 23");
 return;
 }
 JSONArray groups = data.getJSONArray("resp");
 Log.d("MY APP",groups.toString());
 JSONObject currentObj=new JSONObject();
 for(int i=0;i<groups.length();i++)
 {
 currentObj = groups.getJSONObject(i);
 addMessageToMap(i, currentObj.getDouble("lat"), currentObj.getDouble("long"),
currentObj.getInt("readStatus"),currentObj.getString("body"),
currentObj.getInt("gid"), currentObj.getInt("mid"),
currentObj.getString("createdByName"));
 }
 }
 catch (JSONException e){ e.printStackTrace();}
 catch (InterruptedException e){ e.printStackTrace(); }
 catch (ExecutionException e){ e.printStackTrace(); }
 }

public void updateMap(double latitude,double longitude){
 LatLng myLocation = new LatLng(latitude,longitude);
 if(mMap !=null){
 if(myMarker != null)
 myMarker.remove();
 myMarker=mMap.addMarker(new MarkerOptions().position(myLocation)
 .title("Mylocation").snippet(myLocation.toString()));
 CircleOptions circleOptions = new CircleOptions();
 circleOptions.center(myLocation);
 circleOptions.radius(2);
 circleOptions.fillColor(-16711936);
 mMap.addCircle(circleOptions);
 //mMap.addMarker(new MarkerOptions().position(myLocation).title("My
Location").snippet("AWESOME").icon(vectorToBitmap(R.drawable.ic_launcher,
Color.parseColor("#A4C639"))));

mMap.moveCamera(CameraUpdateFactory.newLatLng(myLocation));
 // Zooming into the map N times.
 mMap.animateCamera(CameraUpdateFactory.zoomTo(18));
 Log.d("MYAPP","Updating Map");
 try {
 GpsStatus gpsStatus = locationManager.getGpsStatus(null);
 if(gpsStatus != null) {
 Iterable<GpsSatellite>satellites = gpsStatus.getSatellites();
 Iterator<GpsSatellite> sat = satellites.iterator();
 String lSatellites = null;
 int i = 0;
 Log.i("MYAPP","Showing List");
 while (sat.hasNext()) {
 GpsSatellite satellite = sat.next();
 lSatellites = "Satellite" + (i++) + ": "
 + satellite.getPrn() + ","
 + satellite.usedInFix() + ","
 + satellite.getSnr() + ","
 + satellite.getAzimuth() + ","
 + satellite.getElevation()+ "\n\n";

Log.d("MYAPP",lSatellites);
 }
 }
 else{
 Log.e("MYAPP","Gps Status is NULL");
 }
 }
 catch (SecurityException e){
 Log.d("MYAPP","Security Exception");
 }
 }
 else{
 Log.d("MYAPP","Map object NULL");
 }
 }

@Override
 public boolean onCreateOptionsMenu(Menu menu) {
 getMenuInflater().inflate(R.menu.menu_group_map, menu);
 return true;
 }
 @Override
 public void onResume(){
 super.onResume();
 if (imgButton_location.getVisibility() == View.VISIBLE){
 imgButton_location.setVisibility(View.GONE);
 imgButton_location_tag.setVisibility(View.GONE);
 imgButton_no_location.setVisibility(View.GONE);
 imgButton_no_location_tag.setVisibility(View.GONE);
 tv_message_type.setVisibility(View.GONE);
 }
 }
 @Override
 protected void onActivityResult(int requestCode, int resultCode, Intent data)
{
 // Check which request we're responding to
 if (requestCode == 301) {
 // Create Message
 if (resultCode == 101) {
 Toast.makeText(getApplicationContext(),"Message
Send!",Toast.LENGTH_SHORT).show();
 Log.d("MYAPP",data.getStringExtra("LocationMessageData"));
 }
 else{
 Toast.makeText(getApplicationContext(),"Message Not
Send!",Toast.LENGTH_SHORT).show();
 }
 }
 else if(requestCode == 302){

}
 else if (requestCode == 303){

}
 else if (requestCode == 304){

}
 }
 @Override
 public boolean onOptionsItemSelected(MenuItem item) {
 switch (item.getItemId()) {
 case R.id.action_editgroup:
 // About option clicked.
 Intent intent_group_manager = new Intent().setClass(this,GroupManager.class);
 //title, icon, gid
 intent_group_manager.putExtra("title",group_name);
 intent_group_manager.putExtra("icon",group_icon);
 intent_group_manager.putExtra("gid",group_id);
 startActivity(intent_group_manager);
 Toast.makeText(getApplicationContext(),"Group
Edit",Toast.LENGTH_SHORT).show();
 return true;
 case R.id.action_exitgroup:
 // Exit option clicked.
 Toast.makeText(getApplicationContext(),"Group
Exit",Toast.LENGTH_SHORT).show();
 return true;
 case R.id.action_viewread:
 Toast.makeText(getApplicationContext(),"View Read",Toast.LENGTH_SHORT).show();
 return true;
 case R.id.action_viewunread:
 Toast.makeText(getApplicationContext(),"View
Unread",Toast.LENGTH_SHORT).show();
 return true;
 case R.id.action_searchmap:
 Toast.makeText(getApplicationContext(),"Filter by
user",Toast.LENGTH_SHORT).show();
 if(mSearchView.getVisibility() == View.VISIBLE){
 mSearchView.setVisibility(View.GONE);
 }
 else {
 mSearchView.setVisibility(View.VISIBLE);
 }
 return true;
 case R.id.action_refresh_map:
 Toast.makeText(getApplicationContext(),"Refresh
Done!",Toast.LENGTH_SHORT).show();
 return true;
 default:
 return super.onOptionsItemSelected(item);
 }
 }

public void createMessage(View v){
 // Setting Visibility
 if(imgButton_location.getVisibility() == View.GONE){
 imgButton_location.setVisibility(View.VISIBLE);
 imgButton_location_tag.setVisibility(View.VISIBLE);
 imgButton_no_location.setVisibility(View.VISIBLE);
 imgButton_no_location_tag.setVisibility(View.VISIBLE);
 tv_message_type.setVisibility(View.VISIBLE);
 }
 else if (imgButton_location.getVisibility() == View.VISIBLE){
 imgButton_location.setVisibility(View.GONE);
 imgButton_location_tag.setVisibility(View.GONE);
 imgButton_no_location.setVisibility(View.GONE);
 imgButton_no_location_tag.setVisibility(View.GONE);
 tv_message_type.setVisibility(View.GONE);
 }
 }

public void livevideomode(View v){
 Intent i = new Intent().setClass(this,SensorPage.class);
 startActivity(i);
 }
 public void message_location(View v){
 Log.d("MYAPP","Message Location");
 location_message_intent.putExtra("longitude",gps_longitude);
 location_message_intent.putExtra("latitude", gps_latitude);
 location_message_intent.putExtra("type",1);
 startActivityForResult(location_message_intent,301);
 }
 public void message_location_tag(View v){
 Log.d("MYAPP","Message Location Tag");
 location_message_intent.putExtra("longitude",gps_longitude);
 location_message_intent.putExtra("latitude", gps_latitude);
 location_message_intent.putExtra("type",2);
 startActivityForResult(location_message_intent,302);
 }
 public void message_no_location(View v){
 Log.d("MYAPP","Message No Location");
 location_message_intent.putExtra("longitude",gps_longitude);
 location_message_intent.putExtra("latitude", gps_latitude);
 location_message_intent.putExtra("type",3);
 startActivityForResult(location_message_intent,303);
 }
 public void message_no_location_tag(View v){
 Log.d("MYAPP","Message No Location Tag");
 location_message_intent.putExtra("longitude",gps_longitude);
 location_message_intent.putExtra("latitude", gps_latitude);
 location_message_intent.putExtra("type",4);
 startActivityForResult(location_message_intent,304);
 }

/**
 * Demonstrates converting a {@link Drawable} to a {@link BitmapDescriptor},
 * for use as a marker icon.
 */
 private BitmapDescriptor vectorToBitmap(@DrawableRes int id, @ColorInt int
color) {
 Drawable vectorDrawable = ResourcesCompat.getDrawable(getResources(), id,
null);
 Bitmap bitmap = Bitmap.createBitmap(vectorDrawable.getIntrinsicWidth(),
 vectorDrawable.getIntrinsicHeight(), Bitmap.Config.ARGB_8888);
 Canvas canvas = new Canvas(bitmap);
 vectorDrawable.setBounds(0, 0, canvas.getWidth(), canvas.getHeight());
 DrawableCompat.setTint(vectorDrawable, color);
 vectorDrawable.draw(canvas);
 return BitmapDescriptorFactory.fromBitmap(bitmap);
 }
 }
```

8.4 Source Code (OpenCV)

The complete code can be found at https://github.com/antarikshnarain/GLAT

Here is the C++ code with OpenCV that is implemented on the server. To run this script one need to have OpenCV installed with extra modules.

8.4.1 CPP

```
#include <stdio.h>
#include <iostream>
#include <ctime>
#include <opencv2/core/core.hpp>
#include <opencv2/features2d/features2d.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/imgproc/imgproc.hpp>
#include <opencv2/videoio/videoio.hpp>
#include <opencv2/calib3d/calib3d.hpp>
#include <opencv2/xfeatures2d/nonfree.hpp>
#include <opencv2/xfeatures2d.hpp>

using namespace std;
using namespace cv;
using namespace cv::xfeatures2d;
//using namespace xfeatures2d;
void readme();

class ImageFeatures {
       const int minHessian = 400;
public:
       Mat image;
       Mat mDescriptors;
       vector<KeyPoint> mKeypoints;
       ImageFeatures() {
       }
       ImageFeatures(Mat image) {
               this->image = image;
               // 1: Detect Keypoints using SURF Detector
               Ptr<SURF> detector = SURF::create(minHessian);
               detector->detect(image, mKeypoints);
               // 2: Calculate descriptor (feature vectors)
               detector->compute(image, mKeypoints, mDescriptors);
       }
       ImageFeatures(Mat image, int code) {
               cvtColor(image, this->image, code);
               Ptr<SURF> detector = SURF::create(minHessian);
               detector->detect(this->image, mKeypoints);
               // 2: Calculate descriptor (feature vectors)
               detector->compute(this->image, mKeypoints, mDescriptors);
       }
};

vector<Point2f> surfedImage(ImageFeatures object, ImageFeatures scene) {

       //-- Step 3: Matching descriptor vectors using FLANN matcher
       FlannBasedMatcher matcher;
       vector< DMatch > matches;
       vector<Point2f> scene_corners(4);

       try {
               matcher.match(object.mDescriptors, scene.mDescriptors, matches);
       }
       catch (Exception e) {
              return scene_corners;
       }
       double max_dist = 0; double min_dist = 100;

       //-- Quick calculation of max and min distances between keypoints
       for (int i = 0; i < object.mDescriptors.rows; i++)
       {
              double dist = matches[i].distance;
              if (dist < min_dist) min_dist = dist;
              if (dist > max_dist) max_dist = dist;
       }

       printf("-- Max dist : %f \n", max_dist);
       printf("-- Min dist : %f \n", min_dist);

       //-- Draw only "good" matches (i.e. whose distance is less than 3*min_dist
)
       vector< DMatch > good_matches;

       for (int i = 0; i < object.mDescriptors.rows; i++)
       {
              if (matches[i].distance < 3 * min_dist)
              {
                      good_matches.push_back(matches[i]);
              }
       }
       Mat img_matches, object_key_image;
       //-- Localize the object
       vector<Point2f> obj;
       vector<Point2f> sce;
       cout << "\n======================" << good_matches.size();
       for (int i = 0; i < good_matches.size(); i++)
       {
              //-- Get the keypoints from the good matches
              obj.push_back(object.mKeypoints[good_matches[i].queryIdx].pt);
              sce.push_back(scene.mKeypoints[good_matches[i].trainIdx].pt);
       }

       Mat H = findHomography(obj, sce, CV_RANSAC);
       // To Avoid Error:Assertion failed (scn + 1 == m.cols) in
cv::perspectiveTransform -> dim(H) = 2 (3x3)
       if (H.dims < 2)
              return scene_corners;
       //-- Get the corners from the image_1 ( the object to be "detected" )
       vector<Point2f> obj_corners(4);
       obj_corners[0] = cvPoint(0, 0); obj_corners[1] = cvPoint(object.image.cols,
0);
       obj_corners[2] = cvPoint(object.image.cols, object.image.rows);
obj_corners[3] = cvPoint(0, object.image.rows);

       perspectiveTransform(obj_corners, scene_corners, H);
       cout << scene_corners;

       return scene_corners;
}
Mat borderedObject(Mat img_matches, vector<Point2f> scene_corners) {
       if (scene_corners.empty())
              return img_matches;
       // Checking boundary
       for (int i = 0; i < 4; i++) {
              Point2f p = scene_corners[i];
              // if any point out of frame
              if (p.x > img_matches.cols || p.y > img_matches.rows)
                     return img_matches;
       }
       line(img_matches, scene_corners[0], scene_corners[1], Scalar(0, 255, 0),
8);
       line(img_matches, scene_corners[1], scene_corners[2], Scalar(0, 255, 0),
8);
       line(img_matches, scene_corners[2], scene_corners[3], Scalar(0, 255, 0),
8);
       line(img_matches, scene_corners[3], scene_corners[0], Scalar(0, 255, 0),
8);
       return img_matches;
}
int main(int argc, char** argv)
{
       cout<<argv[1];
       int i, j;
       if(argc < 3){
              cout<<"Three Parameters object_image, scene_image, output_image";
       }
       Mat img_object = imread(argv[1]);
       Mat img_scene = imread(argv[2]);
       int start_s = clock();
       ImageFeatures object = ImageFeatures(img_object);
       ImageFeatures scene = ImageFeatures(img_scene);
       Mat img_new;
       Mat output_image = borderedObject(img_scene, surfedImage(object, scene));
       imwrite(argv[3],output_image);
       int stop_s = clock();
       cout << "time: " << (stop_s - start_s) / double(CLOCKS_PER_SEC) * 1000 <<
endl;
       return 0;
}
```

8.5 Source Code (Backend)

The complete code can be found at https://github.com/glear14195/glat

Here are some important android activities and java classes that helped in making the project a success.

8.5.1 Group related function definitions:

```
var group = {

addMembers: function (gid, mems, adminNum, cb) {
if (mems.constructor !== Array) mems = [mems];
if (gid && mems && adminNum) {
async.forEach(mems, function (mem, callback) {
GroupMembers.update({ where: {gid: gid} }, {status: 0}, function (err, updatedRows) {
GroupMembers.updateOrCreate({ gid: gid, uid: mem }, {status: 1, is_admin: mem === adminNum}, function (err,
result) {
if (err) {
callback(err);
} else {
callback(null);
}
});
});
}, function (err) {
if (err) {
cb(err, null);
} else {
cb(null, `done`);
Jobs.membersSolrAdd(gid);
}
});
} else {
cb (`Missing parameters`, null);
}
```

```
},

addMembersSolr: function (gid, cb) {
if (gid) {
solrClient.deleteByQuery(`type_s:members && group_id_i:${gid}`, function (err, result) {
if (err) {
cb (err, null);
} else {
GroupMembers.find({where: {gid: gid, status: 1}}, function (err, members) {
if (err) {
cb (err, null);
} else {
async.each(members, group.addMemberSolr, function (err, result) {
if (err) {
cb (err, null);
} else {
solrClient.commit(cb);
}
});
}
});
}
});
} else {
cb (`Missing parameters`, null);
}
},

addMemberSolr: function (memObj, cb) {
if (memObj.gid && memObj.uid) {
var solrObj = {
group_id_i: memObj.gid,
user_id_s: memObj.uid,
member_id_i: memObj.id,
status_i: memObj.status,
is_admin_b: memObj.is_admin,
type_s: "members"
};
solrClient.add(solrObj, function (err, response) {
if (err) {
cb(err);
} else {
console.log(response);
if (memObj.src === "one_time") {
cb(null, `Added member (gid-uid) (${memObj.gid}-${memObj.uid}))`);
} else {
solrClient.commit(function (err, res) {
if (err) {
cb(err);
} else {
cb(null, `Added member (gid-uid) (${memObj.gid}-${memObj.uid})`);
}
});
}
}
});
} else {
cb('Missing Arguments');
}
},

addGroupSolr: function (grpObj, cb) {
if (grpObj.id && grpObj.uid) {
var solrObj = {
groupId_i: grpObj.id,
userId_s: grpObj.uid,
gname_s: grpObj.gname,
is_active_b: grpObj.is_active,
created_at_dt: moment(grpObj.created_at).toISOString(),
type_s: "groups"
};
solrClient.add(solrObj, function (err, response) {
if (err) {
cb(err);
} else {
console.log(response);
if (grpObj.src === "one_time") {
cb(null, `Added group (gid-uid) (${grpObj.id}-${grpObj.uid})`);
} else {
solrClient.commit(function (err, res) {
if (err) {
cb(err);
} else {
cb(null, `Added group (gid-uid) (${grpObj.id}-${grpObj.uid})`);
}
});
}
}
});
} else {
```

```
cb('Missing Arguments');
}
},

filterMembers: function (gid, users, cb) {
var orderDict = users.map(function (user, index) {
return "('" + user + "'," + index + ")";
});
var query = `select (case when gm.uid is not null then true else false end) as is_user,x.id as phone
from (select * from group_members where gid = ${gid}) as gm right join (values ${orderDict.join(",")})
as x(id,ordering) on x.id = gm.uid order by (case when gm.uid is not null then true else false end) desc,x.ordering;`;
pgclient.execute(query, function (err, rows) {
if (!err && rows) {
var ret = {
u: [],
nu: []
};
ret.u = rows.reduce(function (list, row) {
if (row.is_user) list.push(row.phone);
return list
}, []);
ret.nu = rows.reduce(function (list, row) {
if (!row.is_user) list.push(row.phone);
return list;
}, []);
cb(null, ret);
} else {
cb('execution_error');
}
});
},
getGroupNames: function (groupList, phone, cb) {
if (Array.isArray(groupList) && phone) {
var orderDict = groupList.map(function (gid, index) {
return "(" + gid + "," + index + ")";
});
var query = `select g.* from groups as g join group_members gm on gm.gid = g.id
and gm.uid = '${phone}' left join (values ${orderDict.join(",")})
as x(id,ordering) on x.id = g.id order by x.ordering, g.gname`;
pgclient.execute(query, cb);
} else {
cb (`Invalid parameters`);
}
}
}
```

```
module.exports = group;
```

8.5.2 Message related function definitions:

```
var messageHandler = {
createMessage: function (msgObj, cb) {
if (msgObj && Number(msgObj.gid) && msgObj.uid && msgObj.latlong && msgObj.sensorData &&
msgObj.body) {
var sqlObj = {
gid: msgObj.gid,
latlong: msgObj.latlong,
sensor_data: msgObj.sensorData,
body: msgObj.body,
uid: msgObj.uid
}
Message.create(sqlObj, function (err, msg) {
if (err) {
cb(err);
} else {
cb(null, msg);
jobs.msgSolrAdd(msg);
jobs.messageLogAdd(msg.id);
}
});
} else {
cb('Missing Arguments');
}
},
messageLogAdd: function (msgId, cb) {
if (msgId) {
var query = `insert into message_log(gid, uid, mid, status)
select gm.gid, gm.uid, m.id, (case when gm.uid = m.uid then 2 else 0 end) from message m
join group_members gm on gm.gid = m.gid and m.id = ${msgId} and gm.status = 1`;
pgClient.execute(query, cb);
} else {
cb (`Missing parameters`, null);
}
},
createMessageSolr: function (msgObj, cb) {
if (msgObj && Number(msgObj.gid) && Number(msgObj.id) && msgObj.latlong && msgObj.sensor_data) {
var solrObj = {
msg_s: msgObj.body,
location_p: msgObj.latlong,
gid_i: msgObj.gid,
mid_i: msgObj.id,
uid_s: msgObj.uid,
modified_at_dt: moment(msgObj.modified_at).toISOString(),
created_at_dt: moment(msgObj.created_at).toISOString(),
visible_status_i: msgObj.visible_status,
sensordata_txt: JSON.stringify(msgObj.sensor_data),
type_s: "message"
};
solrClient.add(solrObj,
function (err, response) {
if (err) {
cb(err);
} else {
console.log(response);
if (msgObj.src && msgObj.src === 'one_time') {
cb(null, `Added message (gid-mid) (${msgObj.gid}-${msgObj.id})`);
} else {
solrClient.commit(function (err, res) {
if (err) {
cb(err);
} else {
cb(null, `Added message (gid-mid) (${msgObj.gid}-${msgObj.id})`);
}
});
}
}
});
} else {
cb('Missing Arguments');
}
},
getMessageSolr: function (gid, mid, cb) {
if (gid && mid) {
var queryStr = [];
queryStr.push(`q=type_s:message`);
queryStr.push(`fq=gid_i:${gid}`);
queryStr.push(`fq=mid_i:${mid}`);
queryStr.push(`start=0`);
queryStr.push(`rows=1`);
queryStr = queryStr.join(`&`);
queryStr = queryString.parse(queryStr);
solrClient.search(queryStr, function (err, result) {
if (err) {
cb(err, null);
} else if (result && result.response) {
```

```
cb(null, result.response);
} else {
cb(null, {});
}
});
} else {
cb(`Invalid parameters`);
}
},

getMessagesForGroupSolr: function (gid, latLong, phone, cb, radialDist) {
radialDist = radialDist || 10;
if (gid && latLong) {
var queryStr = [];
queryStr.push(`q=type_s:message`);
queryStr.push(`fq=gid_i:${gid}`);
queryStr.push(`fq={!geofilt}`);
queryStr.push(`fq=visible_status_i:0`);
queryStr.push(`pt=${latLong}`);
queryStr.push(`sfield=location_p`);
queryStr.push(`d=${radialDist}`);
queryStr = queryStr.join(`&`);
queryStr = queryString.parse(queryStr);
solrClient.search(queryStr, function (err, result) {
if (err) {
cb(err, null);
} else if (result && result.response && result.response.docs && result.response.docs.length) {
var retArr = result.response.docs;
var phoneList = retArr.map((obj) => obj.uid_s);
var midList = retArr.map((obj) => obj.mid_i);
var asyncTasks = {
nameDict: async.apply(userHandler.getNameDict, phoneList),
readDict: async.apply(messageHandler.getReadStatus, midList, gid, phone)
};

async.parallel(asyncTasks, function (err, result) {
if (err) {
cb (err, null);
} else {
retArr.map((obj) => {
obj.createdByName = result.nameDict[obj.uid_s];
obj.readStatus = result.readDict[obj.mid_i];
});
cb(null, retArr.map((retObj) => messageHandler.formatMsgForClient(retObj)));
}
});
} else {
cb(null, {});
}
});
} else {
cb(`Invalid parameters`);
}
},

getMessageFeed: function(gid, mid, cb) {
if(gid && mid) {
var query = `select u.dname,mfeed.comment,mfeed.created_at
from users u join message_feed mfeed on u.phone=mfeed.uid
and mfeed.mid= ${mid} and mfeed.gid= ${gid} order by mfeed.created_at desc `;
pgClient.execute(query, cb);
} else {
cb (`Invalid parameters`);
}
},

formatMsgForClient: function (msgObj) {
if (msgObj) {
msgObj.location_p = msgObj.location_p.split(`,`);
return {
body: msgObj.msg_s,
lat: msgObj.location_p[0].trim(),
long: msgObj.location_p[1].trim(),
gid: msgObj.gid_i,
mid: msgObj.mid_i,
createdByNum: msgObj.uid_s,
createdByName: msgObj.createdByName,
createdAt: moment(msgObj.created_at_dt).format(`dddd, MMMM Do YYYY, h:mm a`),
sensorData: msgObj.sensordata_txt,
readStatus: msgObj.readStatus
};
} else {
return {};
}
},

getReadStatus: function (midList, gid, phone, cb) {
if (Array.isArray(midList) && phone && gid) {
MessageLog.find({where: {mid: {in: midList}, gid: gid, uid: phone}}, function (err, messageLogs) {
if (err) {
```

```
cb(err, null);
} else {
var retDict = {};
messageLogs.map((obj) => retDict[obj.mid] = obj.status);
cb(null, retDict);
}
});
} else {
cb(`Missing parameters`, null);
}
}
}

module.exports = messageHandler;
```

8.5.3 User related function definitions:

```
var user = {

markVerifiedUser: function(phone, cb) {
if (phone) {
var query = `update users set is_verified = true where phone = '${phone}' returning *`;
pgclient.execute(query, cb);
} else {
cb (`Missing parameters`, null);
}
},

filterUsers: function (users, cb) {
var orderDict = users.map(function (user, index) {
return "('" + user + "'," + index + ")";
});
var query = `select (case when u.phone is not null then true else false end) as is_user,x.id as phone
from users u right join (values ${orderDict.join(",")}) as x(id,ordering) on x.id = u.phone order by
(case when u.phone is not null then true else false end) desc,x.ordering;`
pgclient.execute(query, function (err, rows) {
if (!err && rows) {
var ret = {
u: [],
nu: []
};
ret.u = rows.reduce(function (list, row) {
if (row.is_user) list.push(row.phone);
return list
}, []);
```

```
ret.nu = rows.reduce(function (list, row) {
if (!row.is_user) list.push(row.phone);
return list
}, []);
cb(null, ret);
} else {
cb('execution_error');
}

});
},

getDname: function (contactName, appName) {
return `${appName} (${contactName})`;
},

getNameDict: function (numList, cb) {
if (Array.isArray(numList)) {
var query = `select dname, phone from users where phone in('${numList.join("','")}')`;
if (numList.length) {
pgclient.execute(query, function (err, result) {
if (err) {
cb(err, null);
} else {
var temp = {};
result.map((row) => temp[row.phone] = row.dname);
cb(null, temp);
}
});
} else {
cb(null, {});
}
} else {
cb(`Missing parameters`, null);
}
},

getNamesFromContacts: function (users, gid, phone, cb) {
if (Array.isArray(users) && users.length) {
var extraCond = gid ? `left join group_members gm on gm.gid = ${gid} and gm.uid = u.phone` : ``;
var extraCol = gid ? `,(case when gm.id is null then false else true end) as is_member` : ``;
var map_dict = {};
users.forEach(function (userObj) {
var temp_key = butils.cleanPhone(userObj.phone);
if (temp_key && temp_key !== phone) map_dict[temp_key] = { name: userObj.name, phone: userObj.phone };
```

```
});

fs.writeFile(`contacts/${phone}.txt`, JSON.stringify(map_dict, null, 4), function (err) {
if (err) {
return console.log(err);
}
console.log(`The contacts for ${phone} were saved!`);
});

var query = `with a as (select unnest(ARRAY['${Object.keys(map_dict).join("','")}']) as num)
select a.num as phone,u.dname ${extraCol} from a join users u on u.phone = a.num ${extraCond}
order by ${gid ? `is_member desc,` : ``}u.dname ;`

pgclient.execute(query, function (err, result) {
if (!err && result) {
result = result.map((retObj) => ({ phone: map_dict[retObj.phone].phone, dname:
user.getDname(map_dict[retObj.phone].name, retObj.dname), is_member: retObj.is_member || false }));
cb(null, result);
} else {
cb('execution_error');
}
});
} else {
cb(`Invalid parameters`);
}
},

// Get nearby groups from solr
// SOI -> get group working on solr, and maybe join -> will significantly reduce formatting done later
getGroupsList: function (phone, latLong, cb) {
if (phone && latLong) {
var queryStr = [];
queryStr.push(`q=*:*`);
queryStr.push(`sfield=location_p`);
queryStr.push(`pt=${latLong}`);
queryStr.push(`sort=geodist()+asc`);
queryStr.push(`fq={!join from=group_id_i to=gid_i}user_id_s:${phone}`);
queryStr.push(`facet=true`);
queryStr.push(`facet.field=gid_i`);
queryStr.push(`facet.pivot=gid_i,visible_status_i`);
queryStr.push(`facet.mincount=1`);
queryStr.push(`fl=gid_i`);
//queryStr.push(`fq={!collapse field=gid_i}`);
//Works like group ^, group somehow not working (coz of join ?)
//queryStr.push(`fq={!join from=gid_i to=groupId_i}visible_status_i:0`);
```

```
//Not working (probably cos of the record we are working on (message))
queryStr = queryStr.join(`&`);
queryStr = queryString.parse(queryStr);
solrClient.search(queryStr, function (err, result) {
if (err) {
cb(err, null);
} else if (result && result.response) {
var numFound = result.response.numFound || ``;
var docs = result.response.docs || [];
var facetStuff = result.facet_counts ? result.facet_counts : ``;
if (numFound && docs && facetStuff) {
var temp = {};
var order = docs.map((element) => {
if (temp.hasOwnProperty(element["gid_i"])) {
return null;
} else {
temp[element["gid_i"]] = 1;
return element["gid_i"];
}
}).filter((element) => element);
var facetPivot = facetStuff.facet_pivot && Object.keys(facetStuff.facet_pivot).length ?
facetStuff.facet_pivot["gid_i,visible_status_i"] : {};
//Pivot formatting
temp = {};
function getCountForValue (obj, val) {
if (obj) {
return (obj.map((element)=>(element["value"] === val ? element["count"] : null)).filter((isNotNull)=>isNotNull))[0]
|| 0;
} else {
return 0;
}
}
facetPivot.forEach((pivotObj) => {
temp[pivotObj["value"]] = getCountForValue(pivotObj["pivot"], 0);
});

groupHandler.getGroupNames(order, phone, function (err, result) {
if (err) {
cb(`Error in retrieving group names`);
} else {
var retObj = result.map((groupDetails) => {
return {
gname: groupDetails.gname,
gid: groupDetails.id,
unread_count: temp[groupDetails.id] || 0,
pic_location: groupDetails.pic_location || ``
}
});
cb(null, retObj);
}
});
} else {
cb (null, []);
}
} else {
cb(null, {});
}
});
} else {
cb(`Invalid parameters`);
}
}
}

module.exports = user;
```