Novel Approach to Identify and Explore Under Mapped Locations

A dissertation report submitted in partial fulfillment of the requirements

for the award of degree

of

Master of Technology

in

Computer Science & Engineering

with specialization in

Artificial Intelligence and Neural Networks

By

Souvik Samanta

R102212010

Under the Esteemed Guidance of

Mr. Pankaj Badoni
Assistant Professor
CIT-COES, UPES

Mr. Vinay Rao
Senior Technical Supervisor
I2India Pvt limited

Centre for Information Technology

University of Petroleum & Energy Studies

Bidholi, Via Prem Nagar, Dehradun, UK

April – 2014

CERTIFICATE

This is to certify that the project work entitled **Novel Approach to Identify and Explore Under Mapped Locations** done by **Souvik Samanta, R102212010,** for partial fulfillment of the requirements for the award of the Degree of Masters of Technology in Computer Science & Engineering With Specialization in Artificial Intelligence and Neural Networks, to Centre For Information Technology, University of Petroleum & Energy Studies is a bonafide report of the work carried by them under our guidance and supervision.

To my best of knowledge the literature embodied in this project work has not been submitted to any other University/Institute for the award of any degree or diploma.

Pankaj Badoni 11/5/2014

Mr. Pankaj Badoni
Assistant Professor
CIT-COES
UPES, Dehradun

11/5/14

Dr. Manish Prateek
HOD, Associate Dean
CIT-COES
UPES, Dehradun

CANDIDATE'S DECLARATION

I hereby certify that the work which is being presented in this thesis entitled **Novel Approach to Identify and Explore Under Mapped Locations** in partial fulfillment of the requirements for the award of the Degree of Master of Technology in Computer Science & Engineering With Specialization in Artificial Intelligence and Neural Networks and submitted in the Department of CIT, University of Petroleum & Energy Studies, Dehradun, is an authentic record of my own work carried out during a period from January 2014 to April 2014 under the supervision of Mr. Pankaj Badoni, Assistant Professor, CIT,COES.

The matter presented in this thesis has not been submitted by me for the award of any other degree of this or any other Institute.

(SOUVIK SAMANTA)

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Date: 24/4/2014

PANKAJ BADONI Pankaj Badoni 11/5/2014

Supervisor name and signature

The Dissertation. Viva-Voce Examination of **Mr Souvik Samanta,**Roll No R102212010 .has been held on 23/4/2014.

11/5/14

Official/Department Stamp with signature head.

ACKNOWLEDGEMENT

We wish to express our deep gratitude to our guide **Mr. Pankaj Badoni and Mr. Vinay Rao** for all advice, encouragement and constant support he has given us throughout our project work. This work would not have been possible without his support and valuable suggestions.

We are grateful to I2INDIA private limited for giving us the platform and supporting technical staff that helped me to successfully accomplish the project.

We sincerely thank to our respected Program Head of the Department, **Dr. Manish Prateek**, for his great support in doing our project in **working Industry.**

We are also grateful to **Dr. Manish Prateek , Associate Dean** and **Dr. Kamal Bansal, Director** CoES, UPES for giving us the necessary facilities to carry out our project work successfully.

We would like to thank all our **friends** for their help and constructive criticism during our project work. Finally we have no words to express our sincere gratitude to our **parents** who have shown us this world and for every support they have given us.

Name **Souvik Samanta**

Roll No. **R102212010**

ABSTRACT

A novel concept of custom map-tags and cellular-phone service platform is proposed for countries like India, to enable efficient, automated, route-finding directions for mobile phone users with or without access to internet and efficiently manages last mile directions even in locations that are poorly mapped or with poor signage. The basic concept is to provide a convenient and faster one step/click solution to help locate places even in challenging situations such as but not limited to firstly when mobile/internet connectivity is not available second where locations are poorly mapped and marked (i.e. where road names are missing or buildings are not numbered properly), third where new/temporary construction has changed the directions/map from what is available on the Internet, and finally where the person looking for direction is too short on time or in an inconvenient location/situation to undergo map finding on the mobile phone etc. This solution for the above mentioned problem is enabled through three optional methods first is Short Messaging Service (SMS) on a mobile phone second through a custom application designed for a conventional smart phone or tablet and third through a conventional personal-computer connected to the internet.

The application primarily focuses on getting point to point direction for under mapped location for countries like India. Since most of the places or residential addresses in India are not mapped in the present existing maps, this application provides better way to reach the under mapped location using Google navigation services or by using the internet services.

Keywords:- Navigation, under mapped locations, Android navigation

CONTENTS

	Certificate	**i**
	Candidates Declaration	**ii**
	Acknowledgement	**iii**
	Abstract	**iv**
	Index	**v**
	List of figures	**vii**
1.	**Introduction**	**1**
2.	**Literature Review**	**5**
	2.1. Research and study	6
	2.2. Challenges in mobile application development	8
	2.3 Market Research	9
3.	**Frame used for development**	**12**
	3.1. Mobile Platform Architecture	13
	3.2 Anatomy of android app	15
	3.3 Google map API and Geocoding API	17
4.	**Overview of proposed system**	**20**
	4.1. Fast map tag system	21
	4.2. Fast map tag modules	22
	4.2.1. Fast map tag website	22
	4.2.2 Fast map tag mobile application	25
	4.2.3 Fast map tag web server	28
	4.2.4 Fast-Map Tag SMS-based Address Search Service	29
	4.3. Fast-map tag system process flow	29
	4.3.1. Findee's dataflow	30
	4.3.2. Finders dataflow	33
5.	**Prototype Implementation**	**36**
	5.1. Design analysis	37

5.2. User interface design 40
5.3. Implementation 42
5.3.1. Tag creation 42
5.3.2. Getting Directions 47
6. Application **51**
7. Result and discussion **55**
8. Conclusion and future work **57**
9. References **59**

List of figures

S.No.	Figure	Page No
1.	**Chapter 2**	
	Fig 2.1 graph of different mobile users	7
	Fig 2.2 Percentage user of different platform mobiles	8
	Fig 2.3 Fragmentation and cost of application development	9
2.	**Chapter 3**	
	Fig. 3.1 Android Architecture	13
	Fig. 3.2 Activity flow	16
	Fig. 3.3 Map API	18
	Fig. 3.4 Geocoding Process	19
3.	**Chapter 4**	
	Fig. 4.1 Fast map tag system	22
	Fig. 4.2 Fast map tag creating module	25
	Fig. 4.3 Fast map tag searching module	26
	Fig. 4.4 Fast map tag system flow	30
	Fig. 4.5 Findee's data flow	32
	Fig. 4.6 Finders data flow	35
4.	**Chapter 5**	
	Fig. 5.1 Abstraction of Fast map Tag	37
	Fig. 5.2 Map Activity	38
	Fig. 5.3 Map View	39
	Fig. 5.4 Model-view-controller concept	41
	Fig. 5.5 User interface design	41
	Fig. 5.6 Tag creation	42
	Fig. 5.7 Tag creation choice	44
	Fig. 5.8 Checking address in the map	46
	Fig. 5.9 Getting Direction View	47

Fig. 5.10 Tag details 48
Fig. 5.11 Pointing Tag's address into map 49
Fig. 5.12 . Directions options 50

Chapter 1

Introduction

1. Introduction

With changing times, the mobile technology has changed a lot and in the last few years we have seen the arrival of various new kinds of gadgets in the form of smart phone, camera-phone, Android and tablet phones. In fact, the handset industry has turned from simple budget handsets to ultra-modern high end mobile phones [1]. Today's device is almost everything - it is fashionable, innovative, appealing, high-performing, durable, stylish and multi-tasking. Latest gadgets can be used for various purposes like browsing mobile, internet, navigation, playing games, emailing, blogging, messaging and accessing all popular social networking sites like YouTube, Google search, Gmail and more.
Similarly for navigating to different places various smart phone are using different navigation apps that are using different types of maps like Nokia maps, Google maps, Apple maps, open street maps etc.
Above said maps are used all over the world for getting direction to different places. In India also the same maps are used for getting directions. But the problem with India is that, it is a developing country and not all places of the country are marked in Google or any other maps. Unlike other developed country where all places are mapped in the existing maps.
So in India if one searches for commercial or famous places in the map it gives the correct location and correct direction to reach the destination. But if a person enters a full address in the map and searches the person will get the result as "place not found" or it will give the direction to any other place, it will not give exact location of the address. There are also some cases we get the difficulties on existing mobile application maps:-

- Typically the user has to first launch the mapping application from the mobile phone, which in itself takes some time to load.
- The next step is to enter detailed location address, which is also cumbersome and prone to spelling errors especially on a mobile phone keyboard. In non-English native countries/locations, names

can often have multiple and difficult spellings, making the task even more challenging.

- The third step is to select from various options or the closest option that the mapping application can find, which itself can take a bit of waiting time.
- The fourth step is to set starting location either through GPS (current location) or by laboriously entering the starting address with similar issues as in the previous step.
- The fifth step is to wait for the directions to load. This is often the longest step as it requires loading of the map with annotated directions, before the text directions can be reviewed.
- The final and sixth step is to select between map view and step-by-step direction view.

The entire process can take several minutes, may often not work at all and often does not provide the correct destination. In locations such as several Indian cities and villages, the roads/landmarks are not signed/labelled either in the maps and/or in the physical world, making direction finding a real hassle. The Finder has to resort to making calls to the Findee or asking bystanders or both [2]. This whole process wastes time, costs extra money, wastes mobile battery power, and is greatly inefficient. The web-mapping systems in vogue today were developed with a developed country like environment in mind, where broadband mobile internet and labelled maps are taken for granted today. The systems do not fully address Indian requirements. For all above cases one need some sort of abbreviation type thing for the addresses which are easy to remember and easy to type on mobile phone while travelling also. The solution of this situation is that "TAG" system, also called FAST MAP tag system. By using the map tag the whole address can be replaced by a simple tag which will be convenient to the user and easy to remember also [3].

In this report the design and implementation of whole application is given and shown how easy the Indian addresses can be easily located and can be navigate to that address accurately [4]. The idea is to make the Indian address easily recognizable by different maps and also used for the commercial purpose also. The problem mentioned above related to maps is fully solved using this application which is described fully in other chapters. In this application a layer has been made between Google default navigation applications which provides more refined location addresses to the maps, and helps in providing exact navigation to the destined address.

Chapter 2

Literature Review

- Research and study
- Challenges in mobile application development
- Market Research

2. Literature review

Android is a relatively new platform. It is produced by Google, Inc., and its first release was presented in 2007[1]. Android is installed on many different mobile devices and its users can download Android apps and other content through Google Play service, which replaced the old Android Market [2]. This thesis discusses technologies incorporated in Android application development and how they apply to the research problem. Google claims that "Android powers millions of phones, tablets and other devices." Phones and tablets are mobile devices that can have Android applications installed on them. These applications are written in Java programming language [3] and they are called mobile device applications or apps. Development techniques for apps are structured sets of Java code focused on implementing particular task that provides content for a mobile device application. Although Java programming language includes a broad variety of topics, this thesis focuses on development techniques required for successful implementation of Android Mobile FAST MAP TAG. However, tools and techniques that have long been refined for creating successful desktop computing environments do not translate well to the mobile environment. The focus of our proposal is on developing the next generation of mobile computing applications, which will incorporate a near-constant internet connection, novel interactions (e.g., multi-touch), sensors, and machine learning (e.g., activity inference) to provide rich, interactive experiences

2.1. Research and study

Over the past year, Google's Android and Apple's iOS have been strong competitors in the Mobile app market. According to comScore, iOS mobile devices captured 30% of the market in February 2014. That's up only slightly from November 2013, despite the introduction of the iPhone on Verizon's network. On the other hand, iOS' biggest competitor (in the eyes of many),

Google's Android, has grown 7% since November 2013, and now commands 33% of the smart phone subscriber market in the United

Smart phones have surpassed feature phones in number of units shipped, the move being driven by cheap Android devices which took 75% of the smart phone market in 2013, while iOS claimed 18%. And others is still struggling with only 3% market share [4].

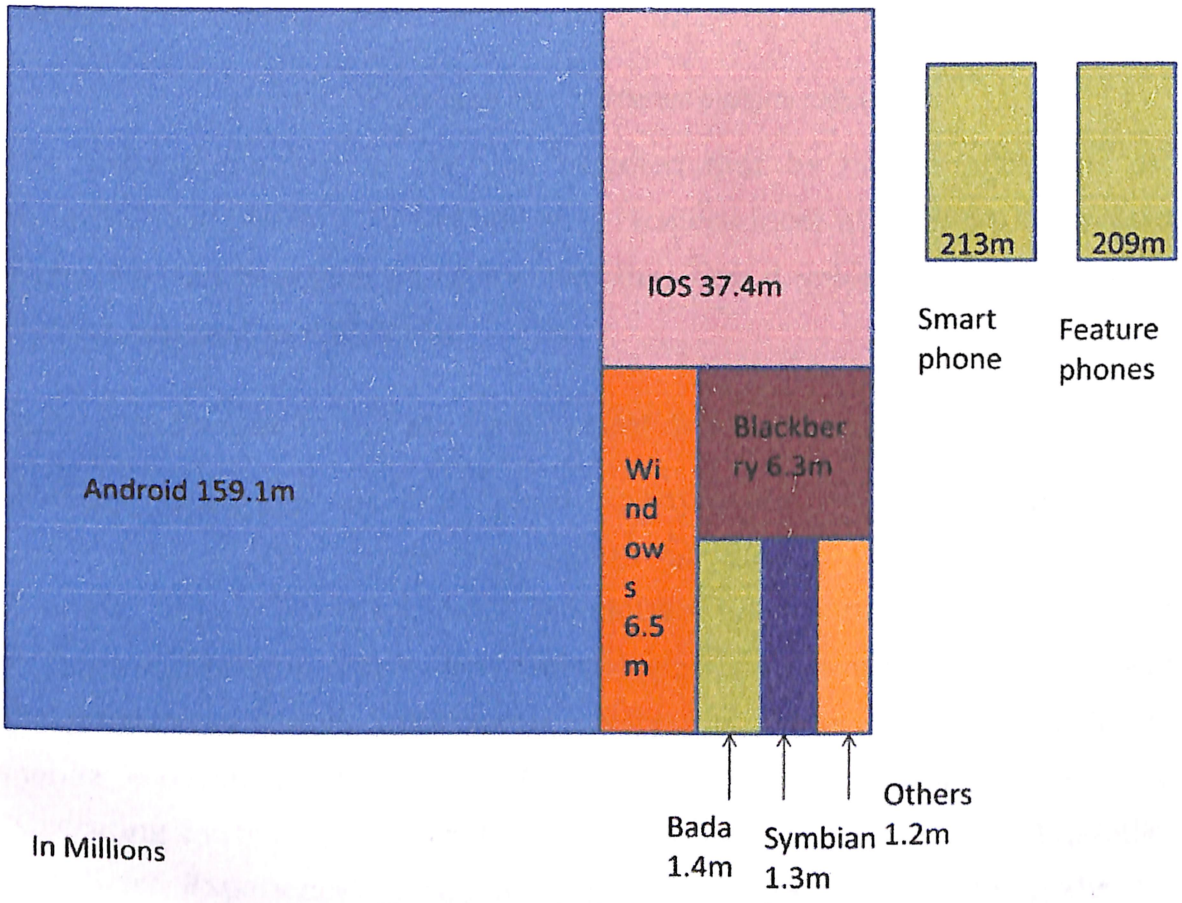


Figure 2.1 graph of different mobile users

So the maximum developers are using android platform to build their application on the platform. According to our survey, developers use 2.9 platforms on average and they choose between them based on revenue, reach, delivery Speed, costs, app discovery and development environment. The preferred (primary) platforms are Android (34.4%), iOS (32.7) and HTML(17.3). The graph has been shown below:-

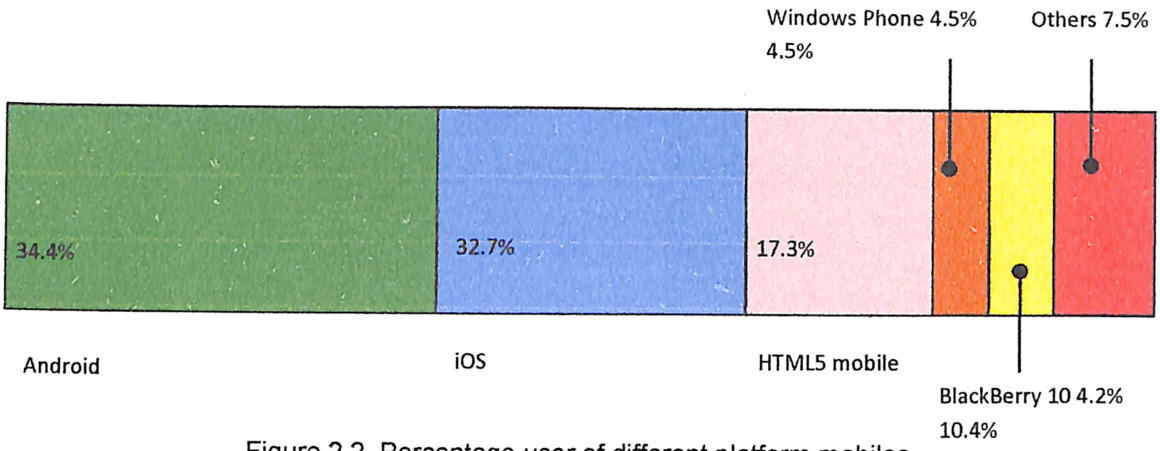


Figure 2.2. Percentage user of different platform mobiles

The Android platform is the one targeted first by new applications or new features. This platform is the one that receives most attention and investment. Android's lead is explained by the fact that new developers prefer this platform (40%) compared to iOS (21%).

2.2. Challenges in mobile application development

Mobile phones are not just "phones" anymore with only "voice and SMS" functionalities. The continuing spread of mobile technology will have a dramatic impact on the lives of individuals and institutions. Convergence of internet and telecommunication technologies is increasing rapidly. However, in the progress of mobile application development, due to complex structure of mobile ecosystem, there is a fragmentation in terms of different mobile "Operating Systems", "Screen Resolutions", "Device Models and Capabilities, " and "User Experience". Fragmentation is the word that defines the biggest barrier. Fragmentation increases the cost and the time to develop mobile applications. Figure 2.3 shows the fragmentation/cost of application development [5].

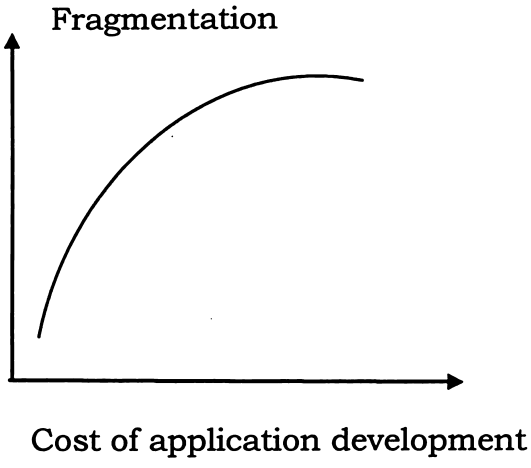


Figure 2.3. Fragmentation and cost of application development

Different device models support different functionalities, such as touch screen, gravity censors, camera flash, etc. Hardware performances also vary between devices. In addition to that, some applications need to support external device functionalities [5]. Screen resolution is crucially important in application development. Ongoing trend is to have bigger screen resolutions for expanded multimedia support, data presentation, and browsing. However, device manufacturers tend to support multi-range of resolutions to address the needs of different user segments [5].

User experience, which is simply the way the users behave while using the keyboard, screen, entry exit functions, number of clicks, etc., is different across device models. Usage scenarios and actions are different between device models and these differences should be realized and taken into consideration in order to deploy successful mobile applications [5].

2.3. Market research (existing maps and analysis)

Due to mapping and direction issues, lot of new systems have been developed to address the challenges. Most of these systems/applications try to avoid

using Web based mapping service like Google Maps®, replacing them with custom-created local content. Some of the mobile applications are as follows:-

- **Wonobo:-**

  Provides 360 degree street view of all places in India. It's for people travelling in India having no clue about the destination. Mobile version is yet to be available but it might involve several steps in locating the destination before having the final 360 degree view [6]. It will require broadband mobile internet and a significant amount of download time. It will probably still not address the issue of locating last-mile directions in a poorly annotated world.

- **Location Finder:-**

  Favours Finder, provided they have saved locations beforehand. Location finder helps people save their location with a subject, notice and an expiry date. You will have many such entries which in future with one click, help you to find the same location on Google maps. This still requires Finder to navigate through Google maps to find locations.

- **Quick place finder:-**

  Favours Finders, tapering around Google maps. Quick Place Finder purpose is to simplify place searches via useful and some popular keywords around current location. They have features like popular quick search categories, storing favourite places into local phone database, User managing search radius for filtering the results, user sharing place details via email to her friends etc.

- **Find place:-**

  Helps find anything about places. They find places nearby or by location. It can find restaurants, malls, Entertainment services etc along with the radius distance and Google map attached to it. But it does not provide exact navigating directions to the last mile and search involves several steps. Might require turning on GPS for navigation.

The above said application uses the maps depending upon their flexibility and usefulness. Like Wonobu application uses open street map for getting the destination direction.The solutions listed above, still involve use of a broadband internet connection, to navigate directions, involve multiple steps before obtaining the last mile directions, and in many cases may not even be able to find select locations. The proposed solution enables a unique method for removing all aforementioned challenges faced by current systems, while enabling faster turnaround time. This proposed solution will be very useful for individual users (Finders and Findees) as well as for businesses that involve delivery services e.g., pizza/food delivery, couriers, postal services, bank couriers etc. It is also very helpful for business Findees, especially restaurants and shopping establishments, who might register themselves with Fast-MapTag and get their unique tag to help Finders locate them without having to attend lots of phone calls.

Chapter 3

Frame work used for development

- Mobile Platform Architecture
- Anatomy of android app
- Google map API and Geocoding API

3. Frame work used for development

3.1. Android platform Architecture

Android is an open-source software platform developed by Google, for mobile app development on devices powered by the Android OS. It is a complete software stack that provides all the middleware needed to run end-user applications on mobile devices such as: device drivers, OS, core libraries, an optimized virtual machine, Java Native Interface (JNI), and a complete Java development environment. This section provides a detailed introduction to the Android framework and describes the platform architecture, execution model, and key concepts pertinent to the design of the Place Me app, which are more generally applicable to other apps as well. As shown in the figure below, the Android software stack is a tiered architecture that consists of 5 principle layers [7].

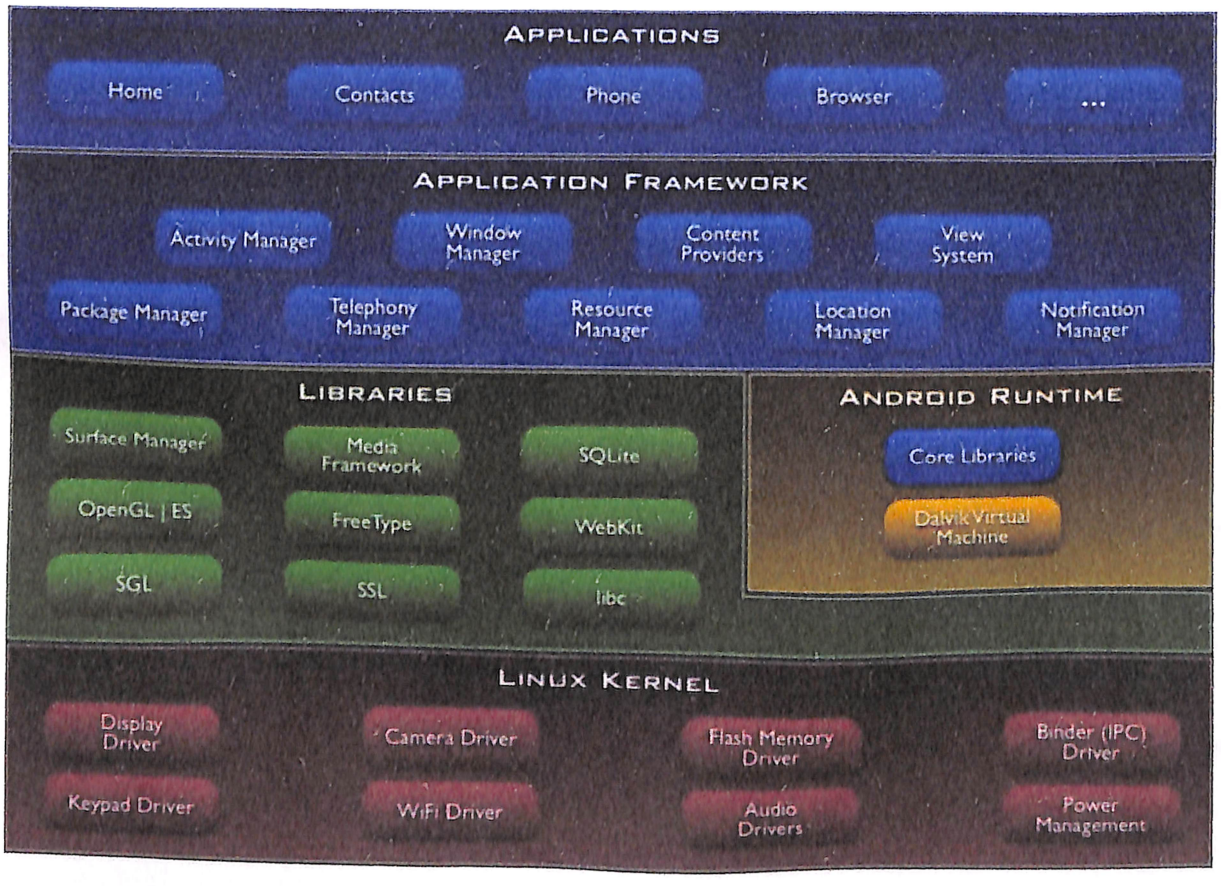


Figure 3.1 Android Architecture [(*Source, courtesy http://developer.android.com/guide/basics/what-isandroid.html*)]

Kernel

Android is built upon Linux 2.6 kernel which serves as the hardware abstraction layer. Linux is used since it provides proven and robust, low-level system infrastructure components such as memory and process management, security, network stack and hardware driver model. Original Equipment Manufacturers (OEMs) can thus bring-up Linux on their system and have the drivers running before loading the other components of the stack [7].

Libraries

On top of the kernel layer are the native libraries written in C and C++, which provide most of the real power of the Android platform. The surface manager is responsible for composing, coordinating and rendering surfaces on the screen from windows owned by different applications, running in different processes in tandem, and ensuring the pixels show up correctly on the screen. OpenGL/ES and SGL are the core 3-D and 2-D graphics libraries respectively. The 3-D graphics can be accelerated in hardware if a 3-D chip is present. Most of the applications commonly use 2-D graphics; however, the platform allows combining 2-D and 3-D graphics as well [8]. The media framework provides all of the audio and video codecs responsible for delivering a rich media experience.

Android Runtime

On the same level as the libraries discussed above is the Android runtime, which is designed for running Java programs in resource constrained, embedded environments with limited computational power, battery life, and memory. One of the main components of the Android Runtime is the Dalvik virtual machine [9]. The Dalvik VM is an optimized byte-code interpreter for efficient byte code execution on small-scale processors used in mobile devices. The Java class and JAR [10] files are translated into ".dex" 25 files at build time, for execution on the Dalvik VM.

Application framework and applications:-

This layer consists of a set of tools and APIs written in the Java programming language which are used by the application developers. Different framework manager is responsible for different activities. And the top-most layer in the stack is the Applications layer. All applications are written in Java and used the same set of APIs provided by the Application Framework. This included applications that are shipped with the phone such as Home, Dialer, Contacts, Browser, etc. as well as those developed by the programmers [8].

3.2. Anatomy of android app

This chapter briefly discusses the basic components of an Android app and defines the key concepts and vocabulary needed to understand the design of Fast map tag app [11].

View

Views are the fundamental building blocks for creating user-interfaces. A View typically consists of the content visible to the user on the screen such as a button, text field, etc [12]. It is the point of user interaction and handles UI events such as a button press. Views are grouped into a hierarchical structure to form different layout schemes such as a lists, tables, etc which organize the Views into specific pattern for rendering [8]. Layouts and Views are typically specified in XML files.

Activity

An Activity is essentially a piece of user-interface that consists of a set of related tasks a user can do in one screen. For example, a mail app can be divided into 3 basic activities: (1) Mail list Activity that shows all received mails (2) Mail view Activity that shows a single mail message (3) Compose Activity that allows creating and sending outgoing messages [8].

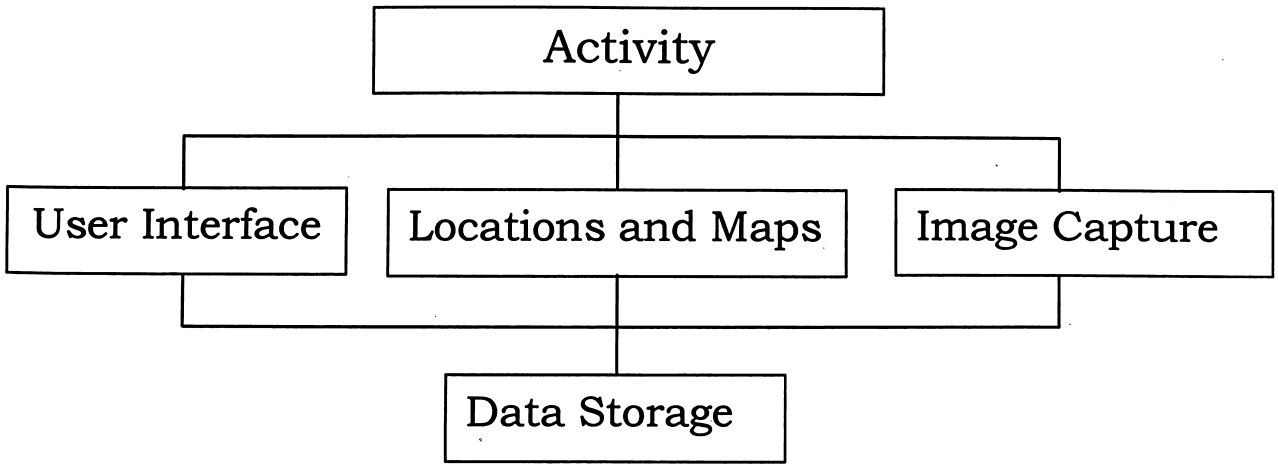


Figure 3.2 Activity flow

Intents

Intents are the fundamental message passing constructs in Android which allow communication of data and action between and among different system components such as: Applications, Activities, Services, etc. For example, when a new email is received, Intents are fired from the mail listener service to update the mail list screen to show the newly received messages [13]. Apps can also register to receive specific kinds of Intents (generated internally or externally) in order to wake up and execute code when the appropriate Intent is received.

Services

Services are background processes launched from Activities that typically perform long-running tasks and have no user interface. For example, a music player can be started as a service and keep playing music while the user may be checking emails. Other Activities or Applications can also bind to the service for performing specific tasks such as pausing, rewinding or fast-forwarding the music [8].

Data Storage

The data storage options which the Android supports: Shared Preferences, Internal Storage, External Storage, SQLite Databases and Network Connection. Trip tracker uses internal storage, External Storage and SQLite databases for storing persistent application data [14].

By default, the application once installed is stored in the internal storage of the Android system. Android also provides full support to SQLite databases. All databases that are created in the application are accessible by name to any class in the application but none outside [15]. The Android SDK includes a sqlite3 database tool which is required to browse the table contents, run SQL commands and perform other SQL functions. Executing an SQL query returns a Cursor which stores the result set pointing to all the rows found by the query. In this application, relational database is used to create two tables for storing tag names, tag details: location name [16], location description, image URI and geo points of the location.

3.3. Google Maps API and Geocoding API

Google Maps API

Android application can access the location services supported by the device through the classes in the android.location package and the Google Maps external library. The main component of the location framework is the LocationManager [17] system service, which provides APIs to determine location and bearing of the underlying device. Google provides a Maps external library that includes the com.google.android.maps package. The com.google.android.maps package used in this application offered built-in downloading, rendering, and caching of Maps tiles, as well as a variety of display options and controls. The key class in the Maps package is com.google.android.maps.MapView, a subclass of ViewGroup.

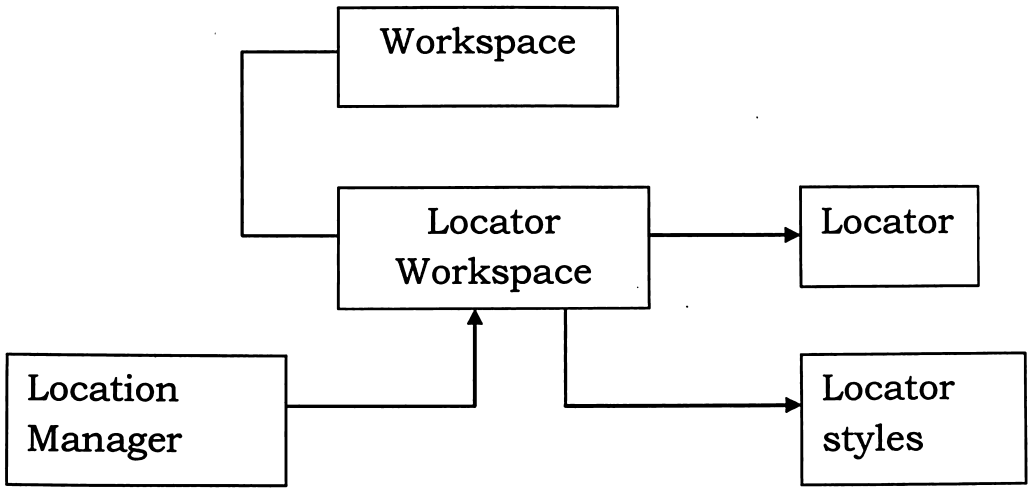


Figure 3.3 Map API

A MapView displays a map with data obtained from the Google Maps service. When the MapView has focus, it will capture key presses and touch gestures to pan and zoom the map automatically; including handling network requests for additional maps tiles. It also provides all of the UI elements necessary for users to control the map. A LocationListener [18] is the interface implemented to receive location updates. To use Google Maps in the application, a Maps API key had to be obtained to register with the service and Android system had to be notified that the application wishes to implement the add-on Google APIs which are external to the base APIs. This was done by using the uses library element in the Android manifest file, informing Android that the application used classes from the com.google.android.maps package.

Geocoding API

Geocoding is the process of creating geometric representations for locations (such as point features) from descriptions of locations (such as addresses). It is the process of assigning locations to addresses to that they can be placed as points on a map, similar to putting pins on a paper map, and analyzed with other spatial data. The process assigns geographic coordinates to the original

data, hence the name geocoding. It is also called address-matching. In a typical geocoding process, the data list might include an address like 508 W. 5th St. and a street centerline GIS data layer would have a street segment corresponding to the 500 block of West 5th Street. The result of geocoding would be a point placed somewhere along the even-address side of that street segment [19].

Two sets of data are needed for the geocoding process - the **address data** that you want to place on a map, e.g., a list of addresses, and the GIS data layer that you will use as the geographic **reference layer**, e.g., a city's street centerlines layer or a parcel address point layer. Both data sets need to be prepared prior to geocoding

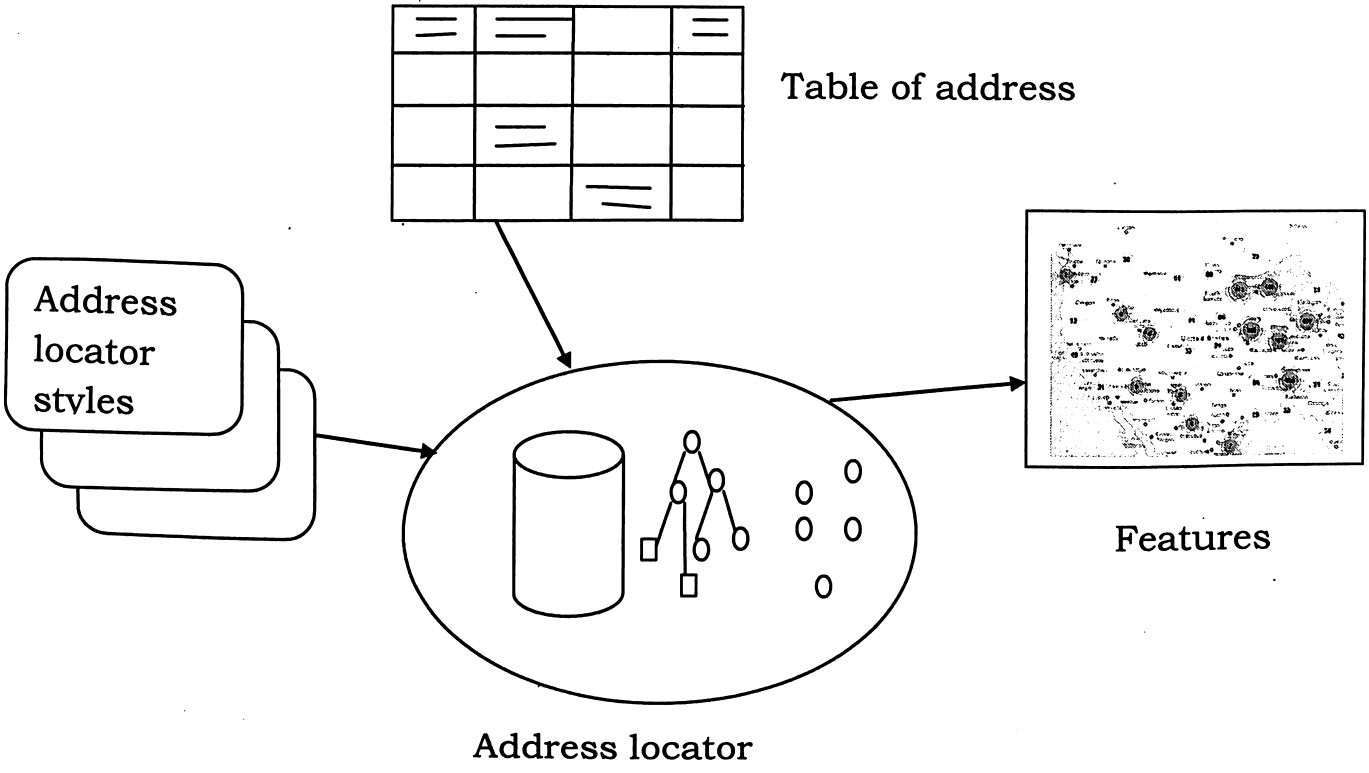


Figure 3.4 Geocoding Proces

Chapter 4

Overview of Proposed System

- Fast map tag system
- Fast map tag modules
- Process flow of Proposed system

4. Overview of Proposed System

4.1. The Proposed tag system

The system can be divided into the following modules and sub-systems namely:

1. **Fast-MapTag Website** a custom-created website with the following modules
   - Fast-MapTag Creation and Editing Module
   - Fast-MapTag Based Address Search Module

2. **Fast-MapTag Mobile Application**
   - Fast-MapTag Creation and Editing Module
   - Fast-MapTag Based Address Search Module
   - Find-Me-Now Module – for instant 1-click location messaging without knowledge of location or having to provide additional details

3. **Fast-MapTag Web-Server** with following components
   - Map Management Module – to manage queries to internal or external mapping and direction creation service such as Google Maps®.
   - Cloud Database Module – to store all Fast-MapTag relation information

4. **SMS-based Direction Query Service** with following components
   - SMS Server – to manage queries to internal or external mapping and direction creation service such as Google Maps®.
   - Business logic unit
   - Network Interface with Fast-MapTag Sever

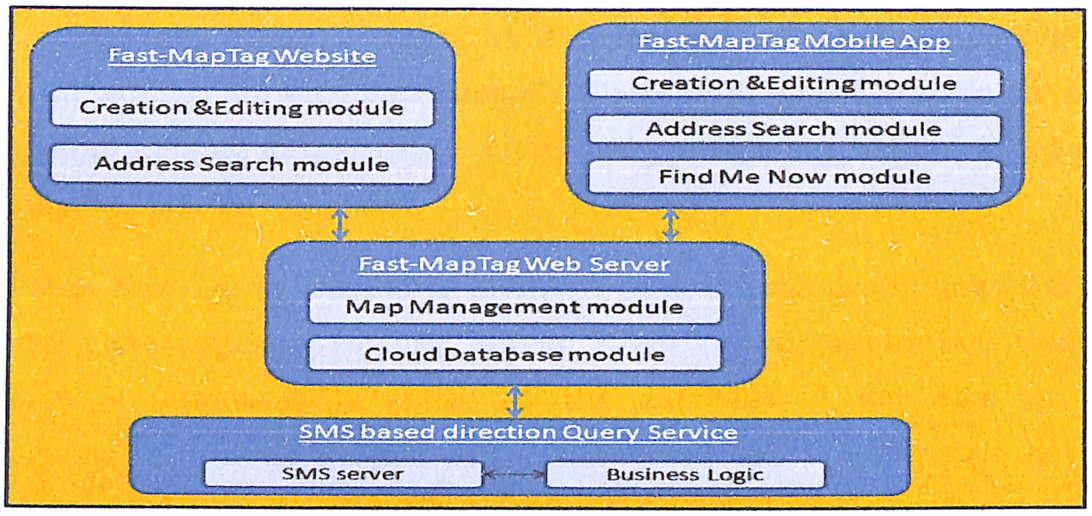


Figure 4.1. Fast map tag system

4.2. Fast map tag modules

4.2.1. Fast-MapTag Website

a. Fast-MapTag Creation and Editing Module

Findee can create their unique location tags using the Fast-MapTag application via internet. The user who needs to create a Fast-MapTag for their address location using Fast-MapTag application via internet first navigates to the Fast-MapTag website using a conventional web browser. The creation process is described as below:

1. The Fast-MapTag creation module comprises of
   a. User Direction input module,
   b. Interface with Fast-MapTag database,
   c. Interface with web based mapping service providers,
   d. Business logic and
   e. Output module

   Using this Fast-MapTag Creation Module, the Findee creates a Fast-MapTag for their location by entering the full address in the input module and locating their site using one or more underlying web-based mapping services (e.g. GoogleMaps®). The Findee is also given the option

of suggesting a natural language description for their Fast-MapTag e.g. "Anil Gupta Koramangala Home". These inputs get translated to a custom maptag format using business logic with the help of web based mapping service

2. The Fast-MapTag so created is internally and automatically linked to the precise geo-positioning-satellite system (GPS) co-ordinations and this co-relation is automatically stored in the database. A key step in this innovation is to avoid the use of the numerical GPS co-ordinates that are hard to recall precisely and no fault-tolerant (i.e. even small numerical errors in them can cause big variations in physical location of the landmark).
3. Fast-MapTags application helps people to create a unique Fast-MapTag using their conventional English language or in their vernacular language which can be easily communicated to others in a much faster way than providing their complete address every time.
4. The business logic includes the process of creating corresponding navigation direction by connecting internally to one/more web based mapping services like Google maps and storing it in database, doing redundancy check on maptags, validating the user input address, pushing the valid maptags to Google maps etc.
5. Creation module recommends Findee to provide meta-data including but not limited to navigation landmarks, driving assistance directions and photographs, individually identified with multiple incoming directions.
6. Database management module will store the user's Fast-MapTag, its corresponding physical address, any relevant landmark(s) provided by Findee, any direction-finding metadata provided by Findee and any optional pictures uploaded by Findee.
7. The output module display Fast-MapTag in a simple conventional English or vernacular language which is easily to read and remember.

8. Creation Module is login and password protected by the Findee to allow only the said Findee to change/manage the maptags and associated meta-data.

b. Fast-MapTag Based Address Search Module

The user who needs direction to navigate from one location to another can get navigation directions through an internet connected communication device, if they have the desired destination maptag. The user first navigates to a custom-created Fast-MapTag service website using a conventional web browser. The Finding module which is associated with finding the directions is described as below:

1. The Finding module comprises of user
a. Fast-MapTag input module,
b. Interface with Fast-MapTag database search,
c. Business logic module
d. Interface with Mapping Service module
e. Output module
2. The Finder inputs Fast-MapTag in a simple conventional English or vernacular language which is a definite hard-coded format. The Fast-MapTag is first tested via Business logic for possible entry errors such as spelling errors and thereupon the data is queried from the database to obtain the corresponding physical address. The address is sent to Mapping Service module to obtain navigation directions and associated details including destination map. These details are displayed using the Output module.
3. The output module will display the user with clear navigation path for the destination in a short format which is easily understood by the user.

4.2.2. Fast-MapTag Mobile Application

a. Fast-MapTag Creation and Editing Module

The user who needs to use Fast-MapTags application via mobile device will first install Fast-MapTagApp in their communication device like mobile phone or tablet, which supports installing mobile based application in the device. The creating and editing module is same as that of creating tags in fast map tag website.

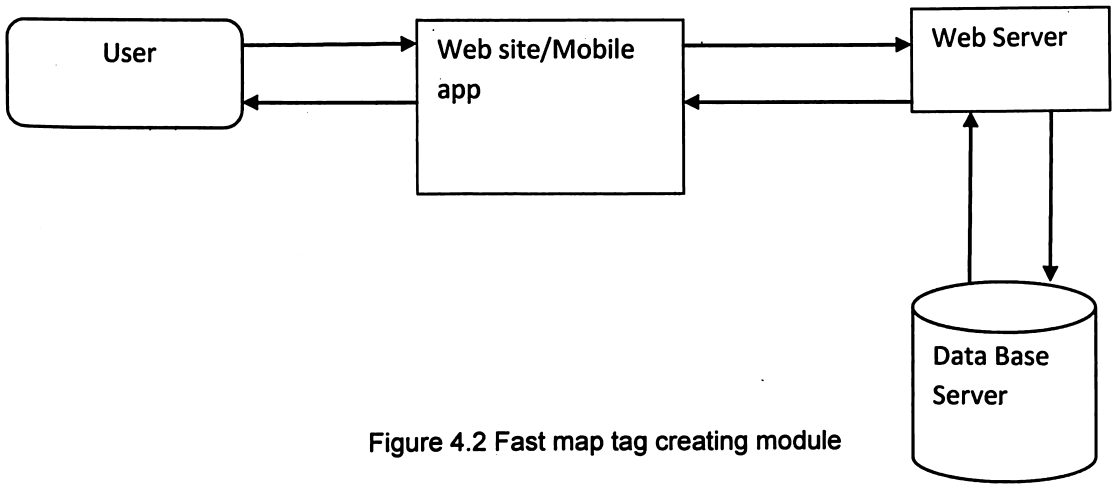


Figure 4.2 Fast map tag creating module

b. **Fast-MapTag Based Address Searching Module**

Finders can obtain navigation directions for their destinations using Fast-MapTag services through a mobile-phone application. The direction finding process involves [20] the Finder interacting with the Fast-MapTag Finding Module as described below.

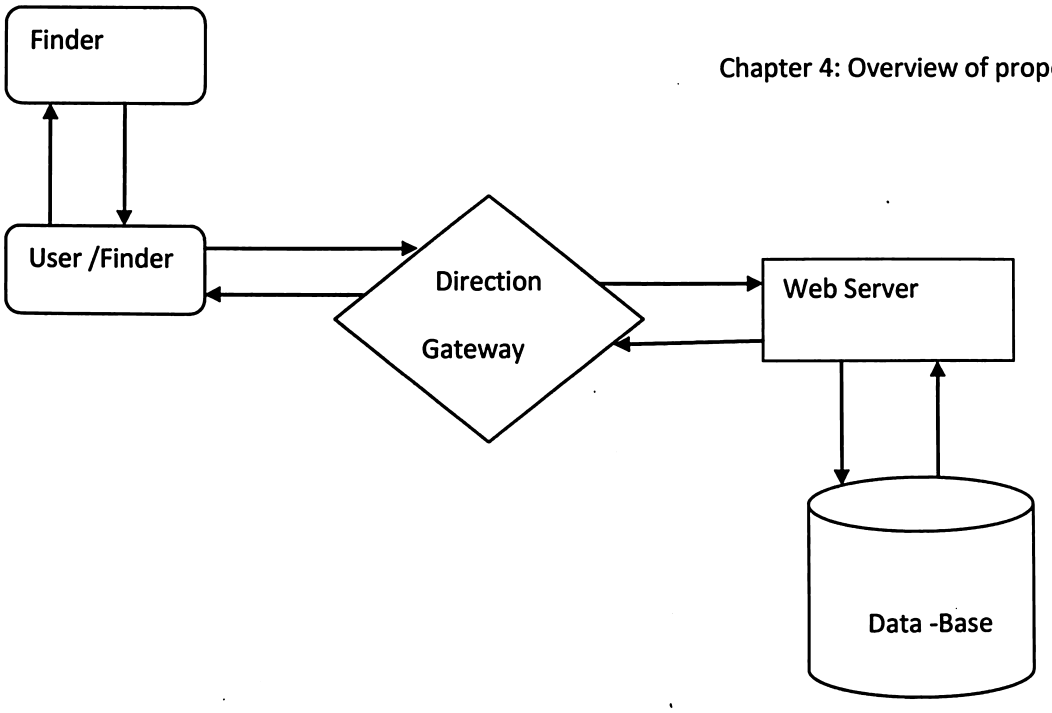


Figure 4.3 Fast map tag searching module

The Finder can get directions through mobile device if they have the desired destination Fast-MapTag. The Finding module which is associated in finding directions is described as below:

1. The user enters the destination Fast-MapTag in the Input Module, which in turn queries the Database Search Module to retrieve the custom directions. These directions are processed through the Business Logic Module and displayed using the Output Display Module.
2. The user can enter the starting location as present location or by using a different fast map tag.
3. The Input Module validates the Fast-MapTag, clarifies and/or corrects any errors or incomplete details and thereon, pushes the Fast-MapTag to the Database Search Module for querying the corresponding navigation path.
4. The output module will display the user with clear navigation path for the destination in a short format which is easily understood by the user.
5. To economize the service time, extend usage/battery of the mobile device, and other Finder-driven customizations, the Address Finding Module enables several short-cuts and options as follows:

a. Optional download of text-only directions, image-based map-data, meta-data, landmark data, Findee provided photographs etc. Default options can be adjusted by Finder based on their preferences.
b. Optional storage of recently used Directions, including Source and Destination pairs, so that no processing is required to review the record.
c. Optional storage of recently used Destination or Source Fast-MapTag much like in an address-book form. This allows quick retrieval for finding directions in future.
d. Save last direction or activity on default screen so as to keep it visible until user removes it
e. User preferences for not storing any history for privacy or other reasons.

4.2.3. Fast-MapTag Web Server

a. **Map Management Module** – to manage queries to internal or external mapping and direction creation service such as Google Maps®.

b. **Cloud Database Module** – to store all Fast-MapTag relation information

a. Map Management Module

This module interfaces with one or multiple conventional web-based or similar mapping services such as Google Maps® and can provide one or more of the following but not limited to these information

- Map details of destination
- Point to point directions from a source point to a destination point
- Satellite images of the desired location, as an option
- Traffic information
- Landmark information

b. Cloud Database Module

This module stores the Fast-MapTag and associated data required by the Fast-MapTag Server. This includes, but not limited to the following featuers:

- Stores all user account details
- Stores all Fast-MapTags – both user created, system created and public
- Stores all temporarily created maptags
- Stores all user provided meta-data associated

4.2.4. Fast-MapTag SMS-based Address Search Service

Even if the user does not have or wish to use the Fast-MapTag mobile application or if the user is using a phone which does not support the same, the Finder can get directions through their mobile device if they have the desired destination Fast-MapTag. This system will have 2 sub-components

i. Fast-MapTag SMS Server
   - The system consists of an SMS Server that receives SMS messages sent on a pre-defined phone number.
   - The SMS Server passes on the information to the Business Logic Unit

ii. Business Logic Unit
   - This module interprets the request before sending a query to Fast-MapTag Server
   - The SMS text is converted into a data-query in the form required by the Fast-MapTag Server

iii. Network Interface with Fast-MagTag Web Server
   - The system is networked over the internet, with a Fast-MapTag Server
   - The Fast-MapTag Server sends a response to the SMS Server, which in turn responds to the Finder looking for details.

4.3. Fast map tag system process flow

The concept of creating Fast-MapTag is discussed in the picture below and described through the following steps:

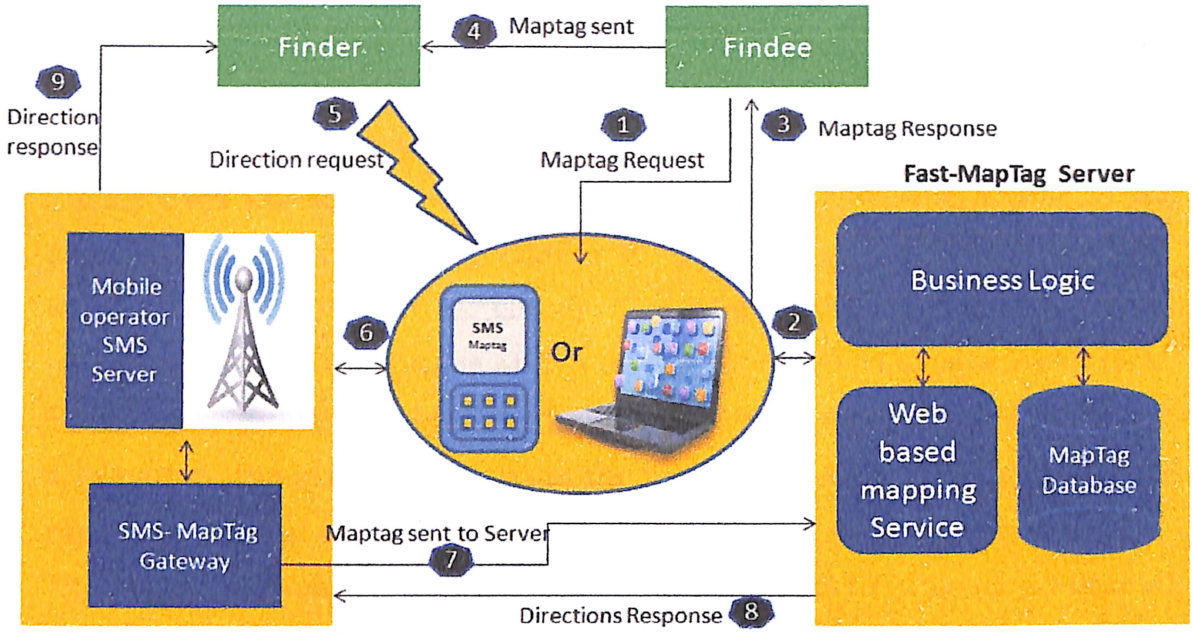


Figure 4.4. Fast map tag system flow

The Fast-MapTagApp offer people to create and link their location tags to any web based mapping service such as Google maps. This custom Fast-MapTag is created as a unique and definite code-word or code-phrase that people use to uniquely tag locations that need to be searched via digital mapping services. Without losing the generality, we refer to the people residing in these destinations/ location as 'Findee' and the person who searches for destination/location as 'Finder' in this document.

4.3.1 Findee's dataflow

1. The Findee first uses the Fast-MapTag Creation Module to create a maptag in Fast- MapTagApp application for which they use a custom-created website using a web browser in one instance else, or their mobile phone to install a custom-created mobile phone application in another instance.
2. The Findee enters his/her desired address in the address bar of the website or the mobile application to create their Fast-MapTag.

3. Fast-MapTagApp or web-service is connected to one or more new or existing web based mapping service such as the Google map via the Internet to fetch address map to the Findee.
4. Findee uses the Maptag creation module enables to create a unique tag for their location in manner similar to creating email addresses on free web-mail services such as Gmail. The Maptag is a unique language phrase chosen by the Findee and is either not reserved or already in use.
5. Once the Findee submits the Fast-MapTag in Fast-MapTagApp application, the instance is sent to the maptag database which checks for the availability of the tag.
6. The MapTag Creation Module allows the Findee to upload additional information or MapTag - Metadata such as nearby landmarks, photographs of their location and/or nearby locations, nearby Fast-MapTag references etc. while creating their maptags with maptag repository.
7. The custom MapTags so created, can be optionally communicated to 3rd party conventional mapping services such as GoogleMaps® so as to help improve the level of information available with such services.

The Fast-MapTag Creation Module provides for a hierarchical model of providing direction-finding meta-data, which is wholly customizable by the Findee. The core concept is to build a hierarchy of increasing detail and data-size, for helping find the destination. The basic idea is that the Finder can obtain more and more details from an automated system, as long as they have uncertainty about their destination. The system is designed to provide the minimal amount of information that can be delivered most speed and cost-effectively. The process flow for findees is given below

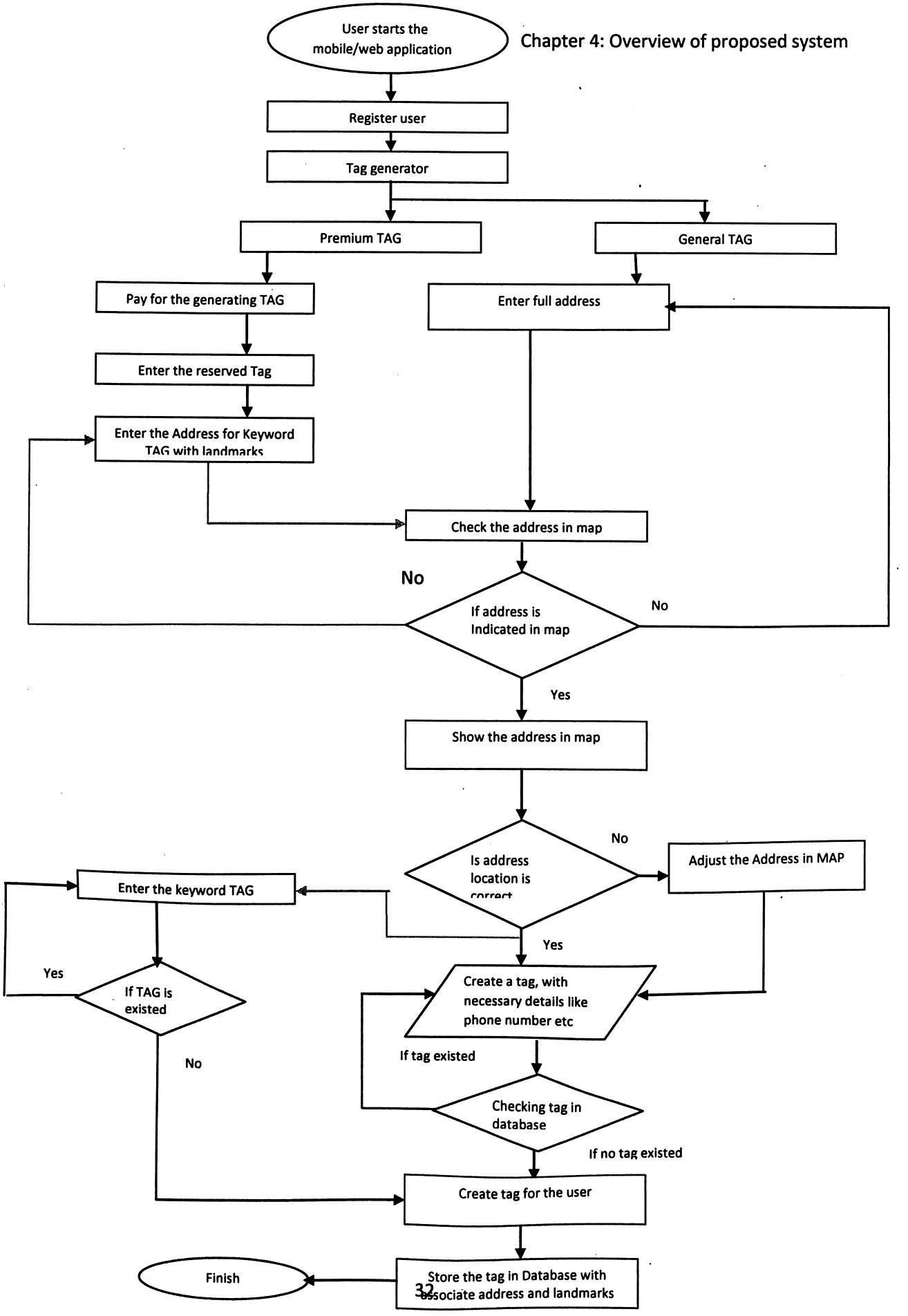


Figure 4.5. Findees Data flow

4.3.2 Finder's dataflow

1. The Finder who needs to locate a particular Findee obtains the Findee's Fast-MapTag separately (outside the use of this system). The Finder uses the Fast-MapTag Finding Module either on the custom-Fast-MapTag website or through the custom Fast-MapTag mobile phone application or through SMS service on a mobile phone.
2. While using the custom Fast-MapTag Application, the Finder uses the App to request directions to a destination Fast-MapTag, from either the current location or from another user-assigned location.
3. In both above instances (2 and 3), Fast-MapTag server parses the request syntax and retrieves the directions from a combination of conventional mapping services as well as custom-created Fast-MapTag algorithms.
4. In one instance, if only one destination MapTag is provided, the Fast-MapTag provides directions from the current location of the mobile phone being used by the Finder. The location of the mobile phone is obtained through conventional methods such as mobile phone built-in GPS or through Wi-Fi and/or cellular tower location triangulation.
5. In another instance, where 2 Fast-MapTags are provided, the Fast-MapTag Server provides directions from the second (origin) tag to the first (destination).
6. In serving all direction requests, the Fast-MapTag Server uses a standardized shorthand format to minimize the time/size/cost of the message transmitted.
7. In mobile SMS MaptagApp application mode, the Finder sends requests through text inputs instead of using an SMS message.
8. The Fast-MapTag location requests are designed for intelligent and simple user interactions. The Fast-MapTag Server is capable of natural language processing similar to text-based internet searches, where common English spelling errors and order of words can be

accommodated without requiring re-entry of complete information. For example if the word "Home" or "Road" is mis-spelt, the system is designed in an intelligent enough manner to avoid an error or require re-entry.

9. The Fast-MapTag Server is also designed to intelligently and automatically interpret Finder's situational requirements. In one instance the Finder can simply enter one Fast-MapTag to fetch directions from current location to destination. In another instance, if Finder enters two Fast-MapTags the Server will provide directions from first Fast-MapTag location to second Fast-MapTag location. If the Finder sends multiple requests to same MapTag within a short period of time, the Server will interpret it as requirement for successively more details.

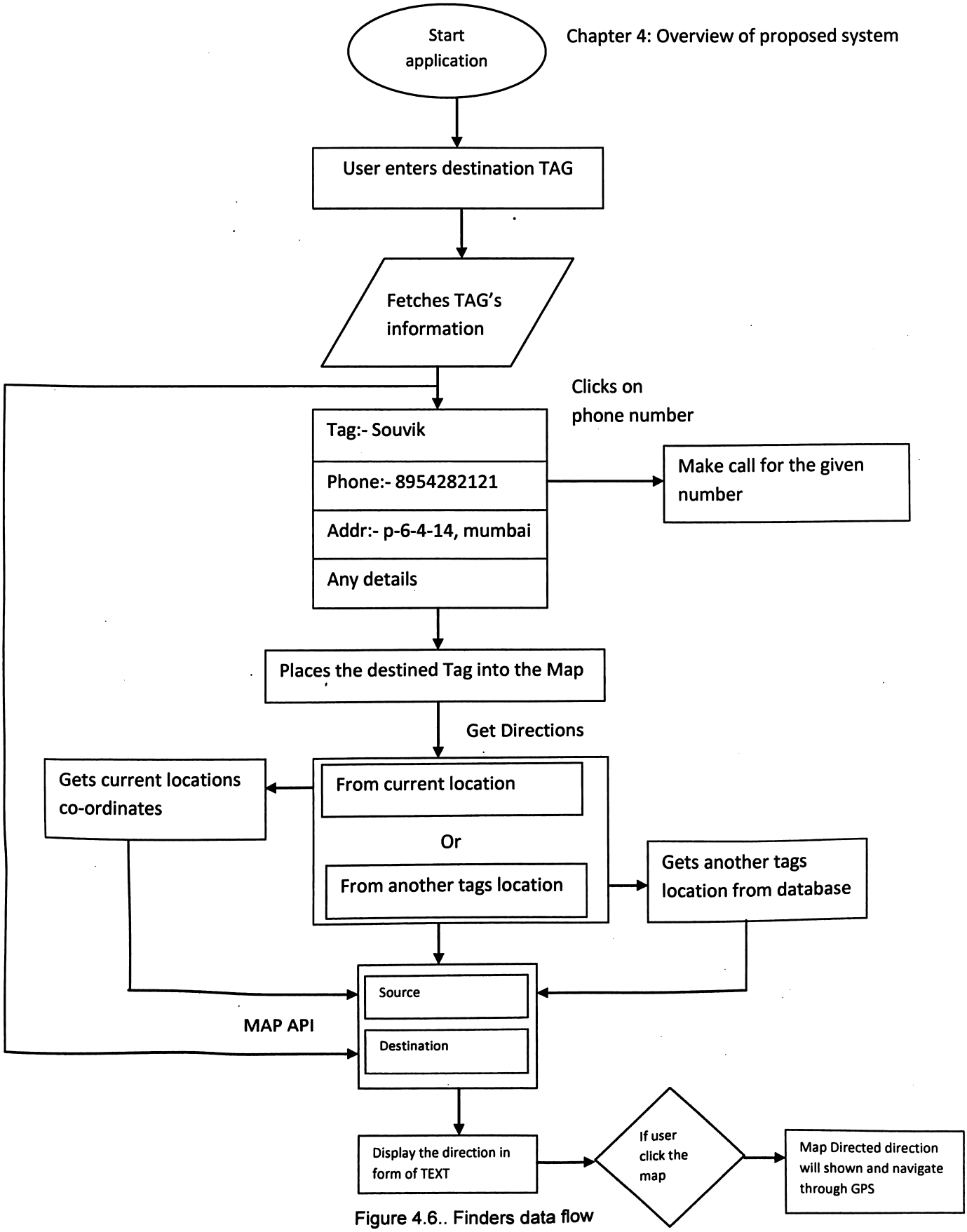


Figure 4.6.. Finders data flow

Chapter 5

Prototype Implementation

- Design Analysis
- User Interface Design
- Implementation

5. Prototype Implementation

5.1. Design Analysis:-

The design analysis proposed here aims at an abstraction model of the Fast map tag. Included are the main functionalities and internal and external dependencies as shown in figure. The user should have Wi-Fi or 3G or 2G connection to run this application [21]. The user can save the direction or maps once it is used i.e. the user does not need data connection every time he uses the map. Once the application is installed the user can downloads the map for future use [22].

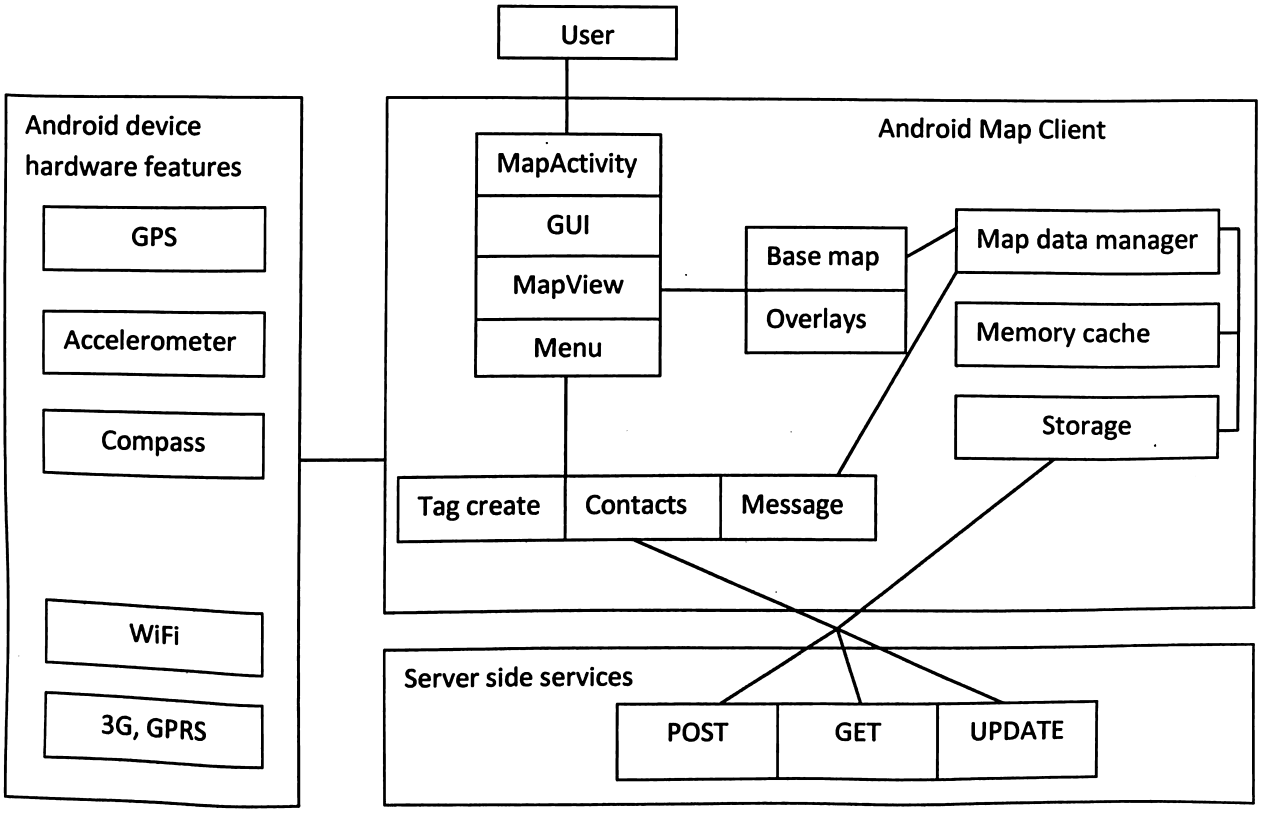


Figure 5.1.Abstraction of Fast map Tag

MapActivity

MapActivity is the main instance that holds the map client application together. It extends the Android base class Activity and manages initialisation of the MapView and data management on start-up and collects instance states for reuse on possible re-start when being shut down. It defines UI layout and controls the menu system. The map activity is shown by fragment command, which shows the map in front of the application to show the necessary details:- The fragment code is as follows:-

```
map = ((SupportMapFragment) getSupportFragmentManager().findFragmentById(R.id.map))
                            .getMap();
                if (map != null)
                      map.setMyLocationEnabled(true);
               //map.setOnMarkerDragListener((OnMarkerDragListener) this);
                       map.setMapType(GoogleMap.MAP_TYPE_HYBRID);
```

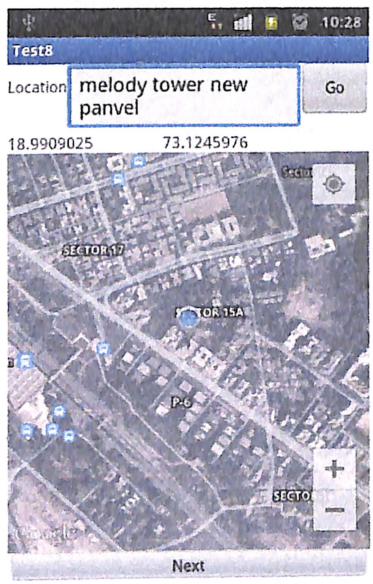


Figure 5.2.Map Activity

MapView

The MapView is the map window, visualising map data on screen. It compiles the map data layers into the final map image and updates it following the command of user input and data updates, such as scrolling or zooming [23]. MapView extends the Android base class View, thatrepresents graphic objects on screen. MapView is built up by three parts:

- Base map – Underlying map covering the whole map image.
- Overlays – Layers of interactive map data objects. Tapping an overlay object could for example open a pop-up window showing attribute data or invoke a new Activity forperforming work related to that object.
- Optional view-objects shown on top of the spatial graphics, for example scale bar, zoom controls and indicators.

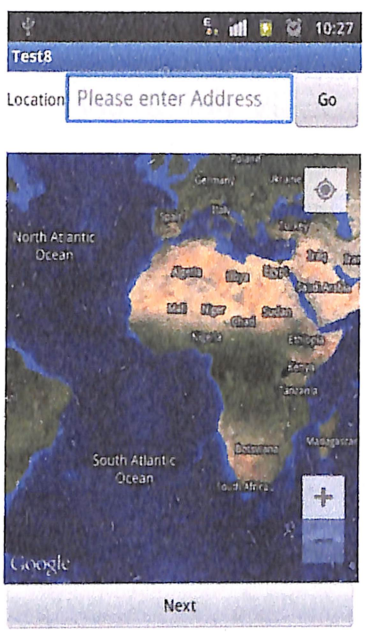


Figure 5.3.Map View

Menu

Menus are an integrated part of the Android application structure. Standard Android devices have a button for reaching the options menu of the current running Activity. It is easiest implemented through the Menu interface in the Android.view package. Through menu only one can create tag, got for the existing phonebook contacts, message etc.

Data management

The Android platform includes a SQLite relational database. It would probably provide the best way to store data on the device. The alternative is to read and write data directly to the file system, which does not provide the convenient

query tools of the database approach. The database storage would serve as a middle tier between objects in memory and data downloaded from servers in various formats [24]. Once server data is parsed, it is stored in a common data model for generic access. The common data model is not further investigated but should consist of a number of common geometries (at minimum point, line and polygon), a generic object model and attribute tables.

5.2. User Interface design

The User Interface is build using Views and ViewGroup objects. Android UI is responsible for the Layout manager and the widget organizer. View objects are the basic units of user interface expression on the Android platform. The View class serves as the base for subclasses called "widgets," which offer fully implemented User Interface (UI) objects, like text fields and buttons [25]. The ViewGroup class serves as the base for subclasses called "layouts, " which offer different kinds of layout architecture, like linear, tabular and relative. A View object is a data structure whose properties store the layout parameters and content for a specific rectangular area of the screen. A View object handles its own measurement, layout, drawing, focus change, scrolling, and key/gesture interactions for the rectangular area of the screen in which it resides. As an object in the user interface, a View is also a point of interaction for the user and the receiver of the interaction events.

The Android UI toolkit offers several layout managers that are easy to use to implement an user interface. This application employs Linear Layout, Frame Layout and Relative Layout. Linear Layout wraps the TextView, EditText, ListView and Buttons; MapView exploits Frame Layout and Relative Layout is used in Trip Logger Activity for displaying the various locations in the trip. Eventually the event is dequeued—first in, first out—and dispatched to an appropriate event handler. The handler responds to the event by notifying the Model that there has been a change in state. The Model takes the appropriate action [18].

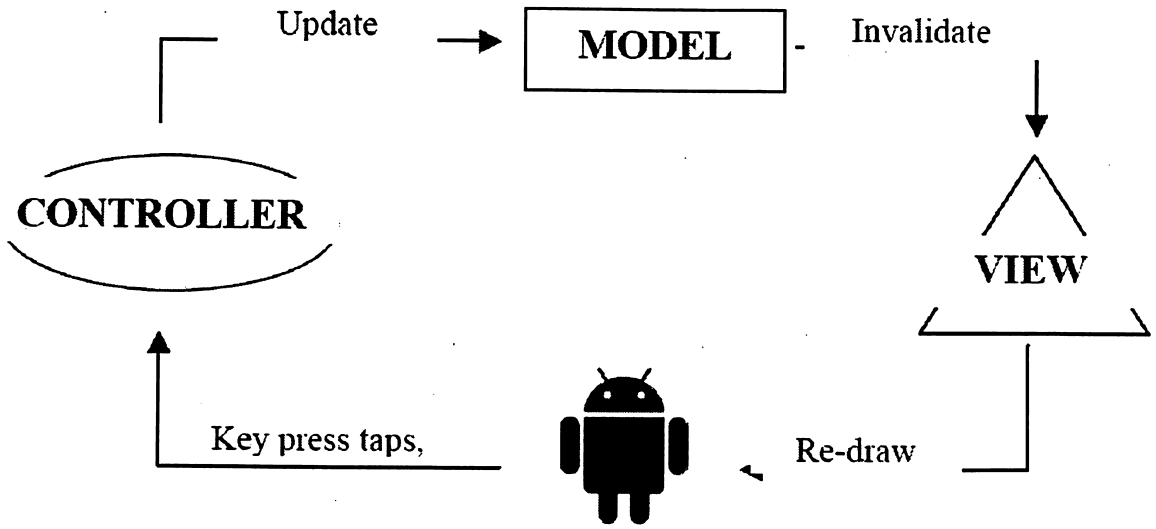


Figure 5.4. Model-view-controller concept

The application is constructed with different views that the user navigates through [16]. The home screen lets the user see an overview of all the available pages the user can navigate to, see figure

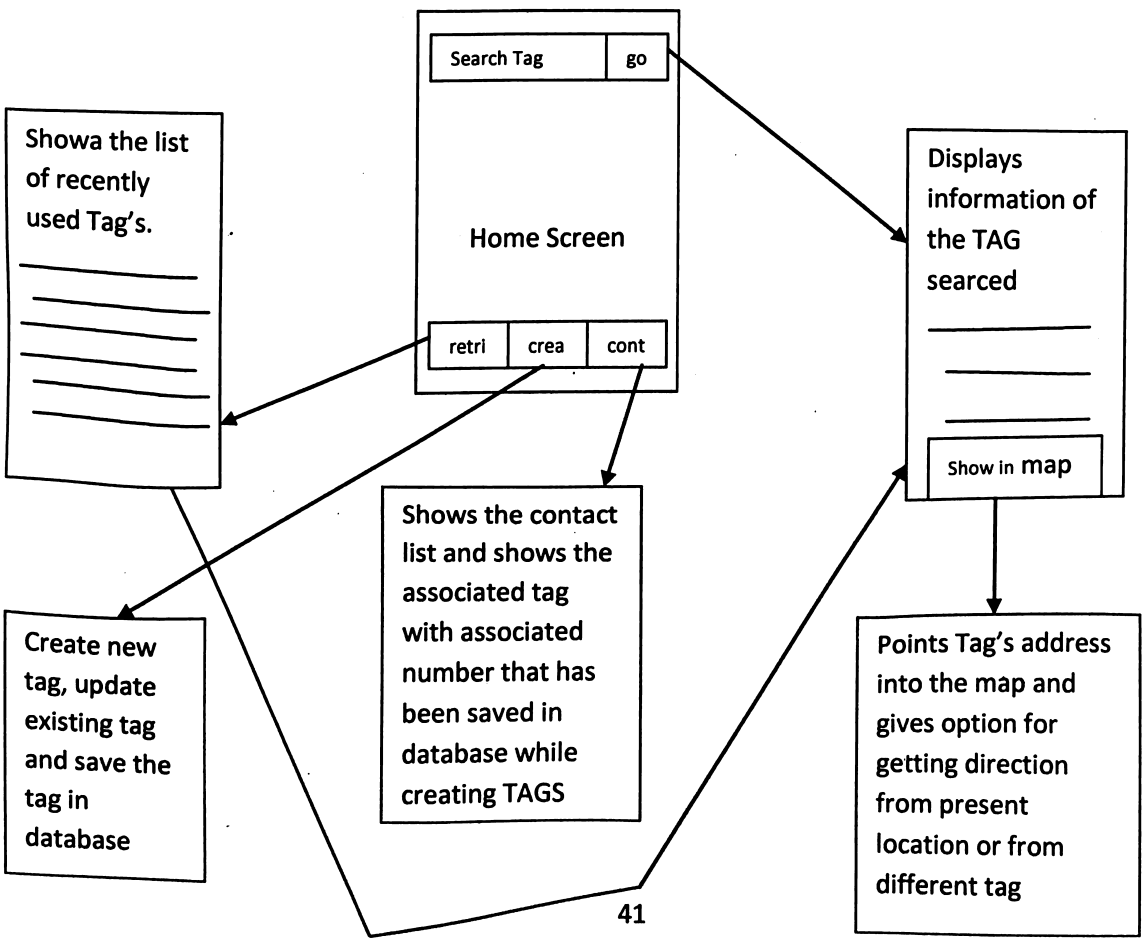


Figure 5.5. User interface design

5.3. Implementation:-

The initial activity gets loaded when the application starts. This is designed to display the tag details which are retrieved by the user from the database. This activity also has a function of showing recently viewed Tags and phone contacts through the user gets the other information. The activity performs in two ways:-

1. Tag creation
2. Getting Directions

5.3.1.Tag creation:-

Tag create can be done on the home screen by clicking on tag create button.

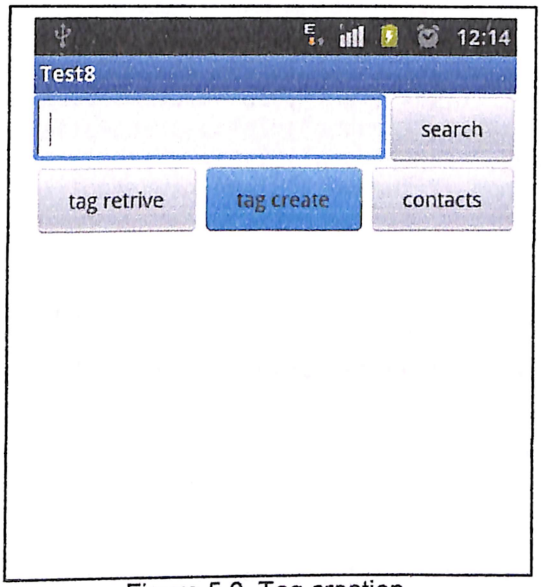


Figure 5.6. Tag creation

The user has an option of creating a tag by using phone number or by using the address. The tag created by addresses and phone number are updated by JSON Parser that are that are updated in database by POST. Asynchronous task is used in background for posting the data. The background task is shown below:-

```
/**
     * Background Async Task to Create new product
     * */
      class CreateNewProduct extends AsyncTask<String, String, String> {

        /**
         * Creating product
         * */
        protected String doInBackground(String... args) {
            String tag = edit_tag.getText().toString();
            String lat = edit_lat.getText().toString();
            String lng = edit_lng.getText().toString();
            String address = edit_address.getText().toString();
            String phonenumber = edit_phonenumber.getText().toString();
            String comments = edit_comments.getText().toString();


            // Building Parameters
            List<NameValuePair> params = new ArrayList<NameValuePair>();
            params.add(new BasicNameValuePair("tag", tag));
            params.add(new BasicNameValuePair("lat", lat));
            params.add(new BasicNameValuePair("lng", lng));
            params.add(new BasicNameValuePair("address", address));
            params.add(new BasicNameValuePair("phonenumber", phonenumber));
            params.add(new BasicNameValuePair("comments", comments));



            // getting JSON Object
            // Note that create product url accepts POST method
            JSONObject json = jsonParser.makeHttpRequest(url_create_product,
                    "POST", params);

            // check log cat fro response
            Log.d("Create Response", json.toString());

            // check for success tag
            try {
                int success = json.getInt(TAG_SUCCESS);
                //int success1 = 1;

                if (success == 1) {
                    // successfully created product

                    showAlert();
                            // closing this screen

                } else {

                    if
                    (success==2)
```

```
                {
                showAlert2();
                }
            else
            {
                showAlert3();
            }


            }

            // failed to create product

        } catch (JSONException e) {
            e.printStackTrace();
        }

        return null;
    }


     /**
     * After completing background task Dismiss the progress dialog
     * **/
    protected void onPostExecute(String file_url) {
        // dismiss the dialog once done


    }
```

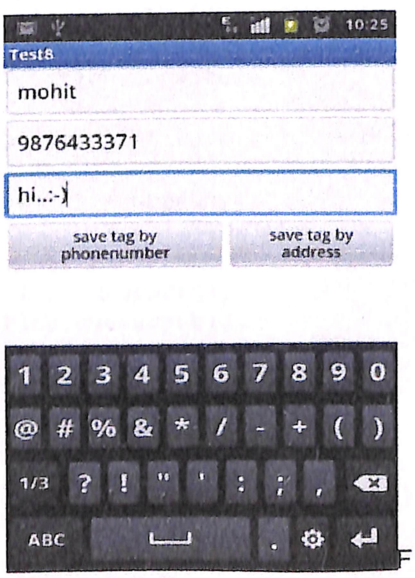


Figure 5.7. Tag creation choice

1:- The user can save the tag by number or by address. If saved by phone number the application will get the contact details by phone contact in the phone and save the information in the server. The Tag saved by address is saved after getting the address in the map.

2:- Here in this activity the user write the address in the map and checks whether the address defined by the user is correctly pointed or the map or not. If the address is correctly pointed the user proceeds enter necessary details otherwise user can drag the address marker to its appropriate location and proceeds further.

The drag class is used in system to drag the marker and the address is located via Geocoding. The drag marker class and geocoding class is given below:-

Drag marker class:-

```
map.setOnMarkerDragListener(new OnMarkerDragListener() {
        public void onMarkerDragStart(Marker marker) {
        }

        public void onMarkerDrag(Marker marker) {
        }

        public void onMarkerDragEnd(Marker marker) {
         LatLng dragPosition = marker.getPosition();
         double dragLat = dragPosition.latitude;
         double dragLng = dragPosition.longitude;

         marker.setTitle(dragPosition.latitude + "," + dragPosition.longitude);

         Log.i("info", dragLat + " " + dragLng);
         double q= dragLat;
         double e= dragLng;
         String w=java.lang.String.valueOf(q);
         String r=java.lang.String.valueOf(e);
         tv.setText(w);
         tv2.setText(r);
        }

       });
```

Geocoding Class:-

```
public void geoLocate() throws IOException
	{
		String loc=location.getText().toString();
		SharedPreferences prefs = getSharedPreferences(
				"com.hk.sqlitedemo", Context.MODE_PRIVATE);
		prefs.edit().putString("name3",location.getText().toString()).commit();

		if(loc.length()>0 )
		{
			Geocoder gc=new Geocoder(this, Locale.getDefault());
			List<Address> list=gc.getFromLocationName(loc,5);

			if
			( list.size()>0 )
			{
				Address add=list.get(0);
				//int index =13;
				String locality = add.getLocality();
				Log.i("abc", locality);
				//String country = add.getCountryName();
				lat = add.getLatitude();
				lng = add.getLongitude();
				double lat =add.getLatitude();
			double lng = add.getLongitude();
			gotoLocation(lat,lng,DEFAULTZOOM);
			}
```

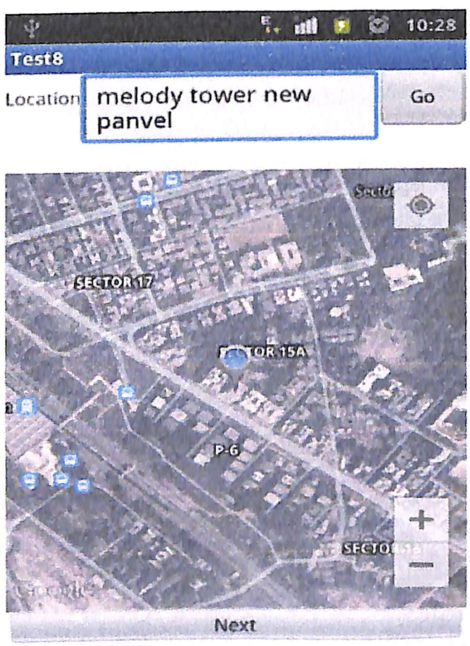


Figure 5.8. Checking Address in the map

After this the user enters the necessary details like phone number, email, landmarks etc and save the data in the server database.

The user can update the Tag details by whenever the user wanted.

5.3.2.Getting Directions:-

From main activity only the tag's direction can be obtained. The implementation is as follows:-

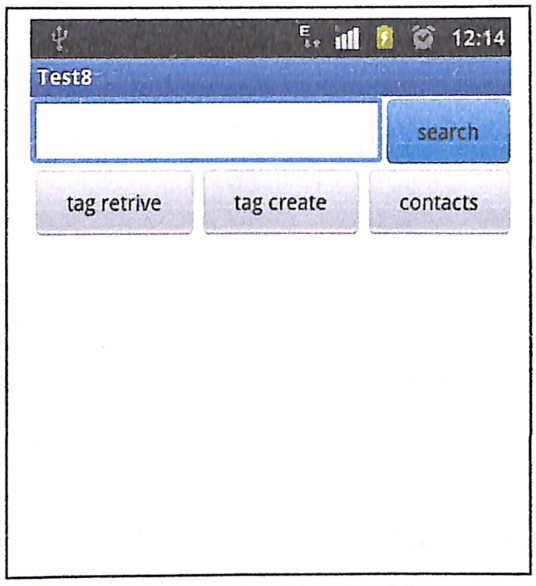


Figure 5.9. Getting Direction View

1:- setOnClickListener(): This listener is activated when the user clicks on any of the button on the activity. This method redirects to intent to call the associated activity.

2:- clicking on search for the user defined Tag will retrieve the Tags information and shows to the user.

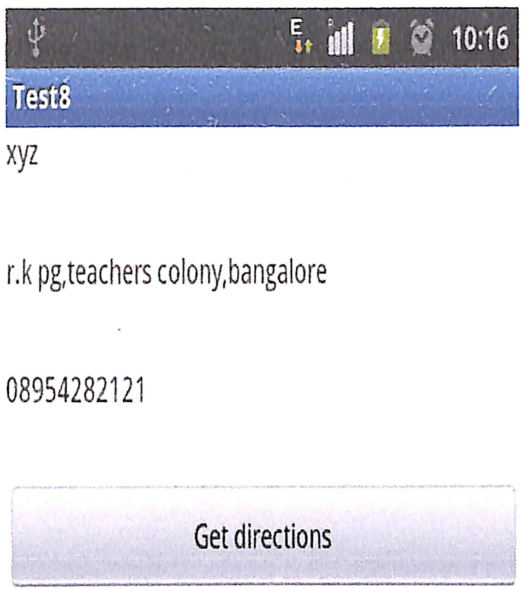


Figure 5.10. Tag details

As soon as user gets the direction the tag information is stored in recently viewed tags. By using the recently viewed tags the user doesn't have to use the internet again to retrieve the same tag. The list of recently viewed tag is present of Tag retrieval button on the main activity.

3:- On click get direction will point the tags address into Map used. Here the marker drag gable option is set false, as Tag position is constant for the given address. The marker in the map of the retrieved tag is given by class place marker in the map. The class code for placing the marker is as follows:-

```
Marker kiel = mMap.addMarker(new MarkerOptions()
                .position(tag)
                .title(tag2)
                .snippet(address)

.icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_AZURE))
                    );

            // Move the camera instantly to hamburg with a zoom of 15.
            mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(tag,16));

            // Zoom in, animating the camera.
            mMap.animateCamera(CameraUpdateFactory.zoomTo(10), 2000, null);
        }
```

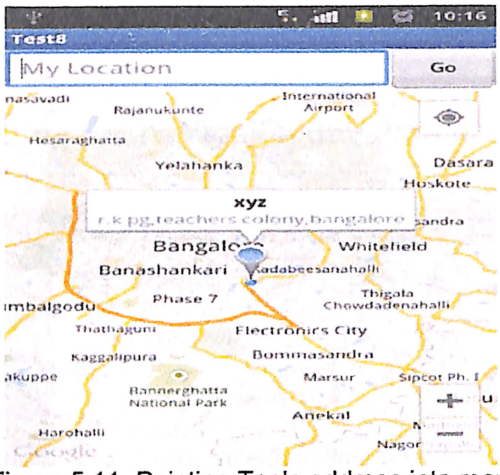


Figure 5.11. Pointing Tag's address into map

4:-If entry in My location is null it will fetch direction from current location to destined tag which is pointed in the Map. Otherwise one can get direction from different tag's address to destined address. After getting the source and destination the application will prompt for getting direction from internet or from maps. As per users choice the direction will be produced.

For getting the directions through Google navigation URI command has been used to get interaction with default Google navigation application.

The URI code is as follows:-

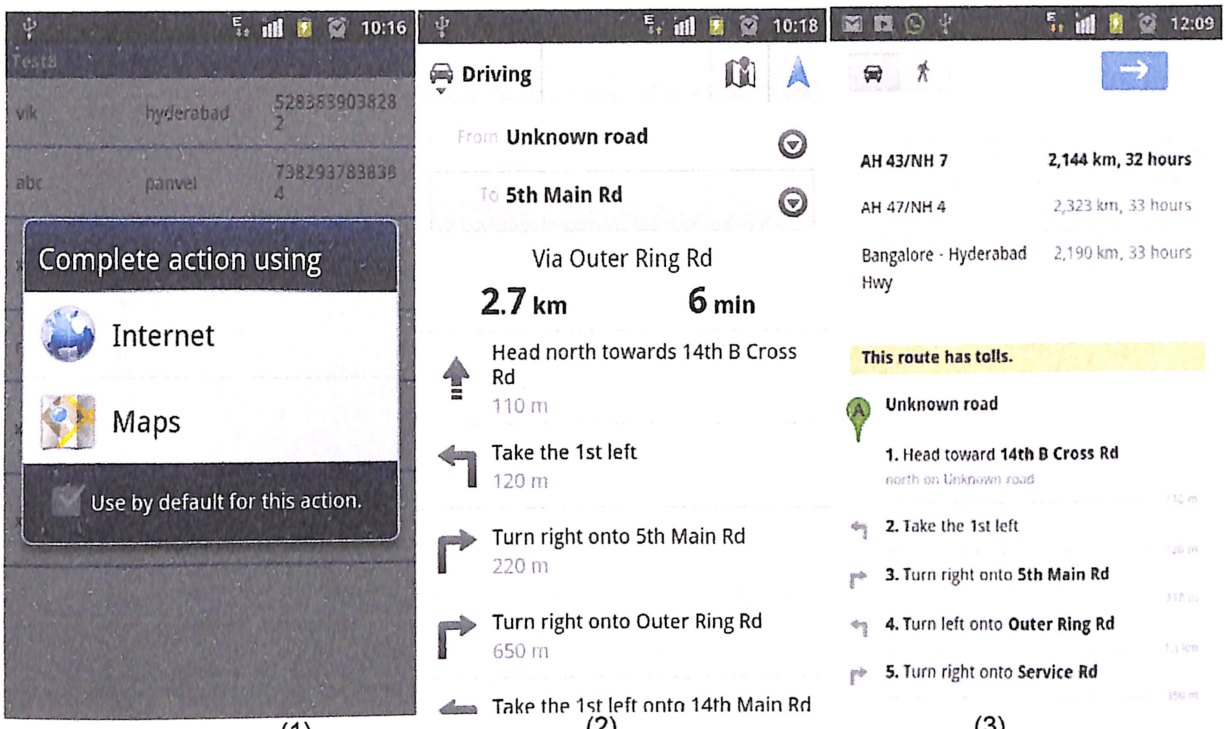
```
Uri geoUri =
Uri.parse("http://maps.google.com/maps?saddr="+currentLattitude+","+currentLongitude+
"&daddr="+targetLat+","+targetLang);


                    startActivity(mapIntent);
                    finish();
```

The current location can be obtained by location manger to get the present location of the mobile. The class used for getting present location is as follows:-

```
{
                        LocationManager locationManager = (LocationManager)
getSystemService(Context.LOCATION_SERVICE);
                        Criteria criteria = new Criteria();
                        provider = locationManager.getBestProvider(criteria,
true);
                        Location location =
locationManager.getLastKnownLocation(provider);
                        if(location!=null){
                         double   a = location.getLatitude();
```

```
        double b = location.getLongitude();
    }
```

The snapshot for the getting the navigation direction is given below:-



(1) (2) (3)

Figure 5.12. Directions options:- (1):- Choosing the method to get directions
(2):- Directions given by maps
(3):- Directions given by internet

By on click on navigation icon one can point to point navigation with the help of GPS or seeing in the map one can get the direction.

Chapter 6
Application

6. Application

There are several use cases and application that can be used by this application. The important use cases that are essential are described below:-

- **Find me now**:-

Findees in locations that they are unfamiliar with, can provide directions to potential Finders through a 1-click-process on their mobile phone Application, that creates a temporary tag and sends it to the Finders in one of the following methods:

(i) With 1-click the Fast-MapTag Application can generate a temporary map-tag that is a simple English or vernacular phrase (few words) and which is associated with the precise geo-location of the Findee's current location. This phrase can be verbally or otherwise communicated to the Finder (e.g. via email or SMS).

(ii) By providing the Finder's mobile number or Finder's email address, the Findeee can, with a 1-click process the Fast-MapTag Application can automatically send a temporary map-tag to the Finder

(iii) By providing the Finder's Fast-MapTag username, the Findeee can, with a 1-click process create and send a maptag automatically to the mobile Application of the Finder.

Using any of these or some combination of these methods, the Findee can provide precise location and hence directions, to the Finder, even without having any knowledge of the location or nearby landmarks.

- **Call a cab:-**

Findees in locations can call a cab, and provide directions to cab driver through a 1-click-process on their vehicle GPS system, that creates a temporary tag and sends it to the Finders in one of the following methods:

1:- The user calls a cab operator and asks for a cab service.

2:- The cab service operator asks for the present location tag or tag which is already created. The operator sends this tags detail to the cab driver nearest to the tags location.

3:- The cab driver gets the direction to the customers location through his GPS embedded in the cab.

- **Direction to any events:-**

In India daily thousands of events keep going on like auction, sale, fuctions etc. The event organiser can create a temporary tag for their event which not only shows the direction to reach the event but also shows the necessary details about the events like timing, programmes etc. The event tag has to be present with their advertising pamphlet so that people can know about the event tag.

- **No long address has to type while driving:**

While driving it is very hard to type the full address into the map and get the direction. With the help of fast map tag the user can easily type a 3-4 alphabet tag into the mobile or vehicles GPS and reach the destined direction.

- **Phone contact Synchronization:-**

Once the application is installed the app will synchronize with phone contact and shows the contact list in the application. The contact list in app will show the names of other contacts that are using the same application. So even by help of phone contact also user can get the directions to person listed in phone contacts.

- **Commercialised Tags:-**

Through this application one can create a tag for the commercial purposes also. The commercial shop can create their tag and stores necessary

information like phone number and shop timing. User can directly reach to the Shops tag without asking anyone. The user can be fully dependent on mobile application as sometimes human tends to misguides the user.

- **Delivery Services:-**

For delivery services such as pizza delivery, couriers etc., the method provides for an easier way to create automated training and help applications, cutting down the time of the trainers and supervisors as well as improving overall customer satisfaction.

Chapter 7
Results and Discussion

7.Results and discussions

By performing various surveys and market research it has been found that for country like India where all locations are not mapped in the map. This type of application will be much more helpful for the user. It can be helpful in commercial way or it can be helpful in general way. The advantages will for both sides i.e. for findee and finder.

Findee's advantages:-

- Can easily communicate their location with simple natural language words.
- Can use maptag in their email signature for easy communication of physical addresses
- Reduction in time for Findee in helping Finders with last minute direction requests and/or delayed meetings
- Reduction of effort in providing directions repeatedly to all Finders seeking the Findee by referring them to the online Fast-MapTag.

Finder's advantages:-

- Convenience: Simple 1-click process from entering address to locating directions
- Most time-efficient - removes the hassle of long process in opening mapping applications and/or enabling GPS to find out directions for a destination
- Augments automated mapping services with currently practiced landmark driven navigation process followed conventionally in countries like India.
- Removes the need of internet to find navigations to the destinations as basic directions can also be obtained via an SMS service.

Chapter 8
Conclusion and Future Work

8. Conclusion and Future work

Fast map tag is much easier and convenient application to get the destination's direction especially in places like India, where maximum locations are not mapped in existing maps. This application helps in reaching the destination address most accurately other than existing navigation applications. Moreover it can be used in both commercial and general ways and giving advantages to both sides. The commercial advantages are like advertising, event scheduling, and courier services etc. The major advantage of this application is one can get to the findees address by using findess mobile number and can also get the necessary landmark details of the destination.

Future work

1. The existing system should be interfaced with Facebook so that tag related to the person can be known by everyone.
2. The system should fetch tag from Google earth so that universally tag can be known to the other users. For this the intelligent system has to be made so that it can fetch the tag.
3. Creating a batch process in the default mobile phone contact so that it will be easy for the user to directly use the navigation application.

References

[1]. Meier, R. *Professional Android 2 Application Development.* Indianapolis, IN. Wiley Publishing, Inc

[2]. Bishop, T "*Google Play replaces Android Market, cosolidates Google's media marketplaces*". Retrieved March 14,2014,http://www.geekwire.com/2012/google-playreplaces-android-market-consolidates-googles-media-marketplaces

[3]. "*what is Android?*". Retrieved March 10, 2014 from http://developer.android.com/guide/basics/what-is-android.html

[4]. Siddartha Sreenivasa Reddy, *TRIP TRACKER APPLICATION ON ANDROID*, Master thesis, May 2011.

[5]. Israfil Coskun and Osman Celik. "*Challenges of Mobile Application Development*", Retrieved Feb 2014. http://developer.smartface.biz/challenges-of-mobile-application-developementn74.html.

[6]." *3D navigation world*" Retrieved January 2014,www.wonobo.com/india/

[7]. G. Inc."Android Developer Guide". [Online]. Retrived March 11, 2014, http://developer.android.com/guide/basics/what-is-android.html

[8]. A. D. Team ,YouTube presentation. [Online].Retrived December 28, 2013 http://www.youtube.com/watch?v=Mm6Ju0xhUW8

[9]. S. Hashimi, S. Komatineni, and D. MacLean, "*Pro Android 2," in Pro Android 2. New York,* US: Apress, 2014, ch. 1, pp. 10-11.

[10]. Eclipse Foundation, Java development tools. Retrieved March 25, 2014 http://www.eclipse.org/jdt/,2009. online:

[11]. Steele, J. *The Android Developer's Cookbook. Boston, MA. Pearson Education, Inc. The Apache Software Foundation. (n.d.).* "Chapter 2. Connection management". Retrieved April 1, 2014

http://hc.apache.org/httpcomponents-client-ga/tutorial/html/connmgmt.html

[12]. Android Developers. *Google.* Tools. Retrieved January 2014http://developer.android.com/guide/developing/tools/index.html

[13]. Aman Singhal, *"Place Me: Location Based Mobile App for Android Platform,"* November 22, 2010

[14]. Android Developers. SQLiteOpenHelper, Retrived January 2014,.http://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html

[15]. Android Developers. Android Location Package, Retrieved April 2, 2014.http://developer.android.com/reference/android/location/package-summary.html

[16]. Sergey Ivanenko,Agatha Wong, *Working with Geocoding APIs[pdf]*,ESRI development summit, 2008

[17]. Njunjic, Ivan, "*Development Techniques for Android Platform Mobile Device Application". Master's Theses and Doctoral Dissertations.* Paper 394.

[19]. Rick Rogers and John Lombardo. *Android Application Development,* 2009. Retrived from http://androidapps.org.ua/i_sect13_d1e11121.html
Android GUI Architecture

[20]. ListView Backgrounds: *An Optimization. (n.d.).* Retrieved February 10, 2014 fromhttp://developer.android.com/resources/articles/listview-backgrounds.html

[21]. Wikipedia. Dalvik virtual machine. http://en.wikipedia.org/wiki/Dalvik_virtual_machine, 2009. online: June 2009.

[22]. Vogel, L. (2008). "*Apache HttpClient - Tutorial*". Retrieved March 21, 2014 from http://www.vogella.de/articles/ApacheHttpClient/article.html

[23]. Vogel, L. (2010a). "Android HTTP Access - Tutorial". Retrieved March 14, 2014 from http://www.vogella.de/articles/AndroidNetworking/article.html