CHAPTER 1

INTRODUCTION

1.1 ABSTRACT

The Internet-based instant messaging services (e.g. *MSN* messenger, GTalk) have become the most popular communication choice among millions of people worldwide for communicating with family, friends and business associates. Due to the huge number of mobile subscribers, there is a real demand to provide such a service to mobile clients. Despite technology's significant advantages, the primary disadvantage is that only users who are on the same network can chat with each other on that particular network. For instance, a Yahoo! user can only chat with another Yahoo! user but not to the AOL user. Thus, we propose a mobile instant messaging system, IIMS, which integrates all popular networks like AOL, GTalk and Jabber designed for mobile environments. This instant messaging system is expected to provide to its members an improved, single contact point, efficient, portable, easy-to-use, inter-network connectivity within a single application.

1.2 INTRODUCTION & MOTIVATION

The combination of presence awareness and real-time messaging offered by instant messaging systems has proved to be the source of a killer application in the Internet, gathering hundreds of millions of users. Although originally considered a teenager toy, the benefits of instant messaging have been realized by both academic and corporate organizations in the last few years.

Concurrently with the instant messaging revolution of the Internet, the introduction of text messaging has launched a similar phenomenon in mobile networks. Over time, mobile messaging has proved one of the most important sources of revenue for providers of mobile services and new messaging services, such as multimedia messaging and mobile email, have been introduced in recent years.

Instant messaging services have enjoyed a constant growth ever since their introduction. Real-time messages and presence information are the pieces of technology that makes instant messaging different from previous communication services. However, the success of instant messaging is not based on technical differences only; also the methods and concepts used in instant messaging clients, such as popup windows and buddy lists, have contributed to the birth of a completely new type of communication.

1.3 PROBLEM STATEMENT

The already existing applications are dedicated to particular network and allow users of that network to communicate with the users of the same network. Thus there exists no interoperability among various chat networks. For instance, a Gmail user can only chat with another Gmail user but not with the AOL user. These applications if deployed on mobile phones, requires the subscriber to open multiple applications to chat on different networks, thus consuming the memory and battery power of the mobile phone.

This project delivers a single, easy-to-use, portable and efficient instant messaging system to be used on mobile phones. The system provides a single application with inter-network connectivity allowing the users log into multiple accounts to chat with their buddies regardless of which network they belong. It allows the users to view and manage their contacts (buddies). Also, the application allows the user to insert smiley's portraying their personality, thoughts and emotions and manage user groups.

1.4 SCOPE

- IIMS is a multi-protocol IM which will integrate all popular networks like Jabber, GTalk and AOL to be deployed on mobile phone.
- The system will be a single point of contact allowing the users to connect to different chat networks in one place. It will allow the users to chat with their buddies regardless on which network they belong to.
- It provides a rich and powerful UI allowing the users to access most of the handy features in a single tap or two.
- The system will provide support for multiple account login to its users.
- The system will allow the users to reflect their personality, thoughts and emotions via the use of emoticons.
- It will also allow the user to change their presence and status settings.
- The system will also provide the facility of maintaining the buddy list like adding / deleting a user from his contacts.

1.5 ORGANIZATION OF THE REPORT:

Chapter 2: Review of Literature: This chapter consists of the domain explanation, current technology and methodology available. It also describes the technology and methodology used in the project and the overview of the project.

Chapter 3: Analysis and Design: This chapter determines the functional, non-functional, hardware, software requirements as well as describes the architecture of the system with different kinds of diagrams like use-case diagram, class diagram, state chart diagram, etc.

Chapter 4: Implementation and Results: This chapter provides information about the implementation details of the project which includes task sheets and Gantt chart.

Chapter 5: Testing: This chapter deals with different tests conducted while testing the application in a real time environment.

Chapter 6: Conclusion and Future Work: This chapter determines the new functionalities that can be added to enhance the functionalities of the application.

CHAPTER 2

REVIEW OF LITERATURE

2.1 DOMAIN EXPLANATION

Instant Messaging (IM) systems such as ICQs, AIM and MSN Messenger have become extremely popular during the last few years. All these systems are designed for desktop PCs with the assumption that a user only uses one computer or at least only one computer at a time. With the rapid advancement of wireless network technologies, Mobile Instant Messaging has become a major application in the telecommunications industry.

2.2 CURRENT TECHNOLOGY & METHODOLOGY USED

Instant Messaging applications are tools that allow users to exchange messages with remote users, similarly to what provided by e-mail services, but in an instantaneous fashion. The number of users of instant messaging software is quickly growing as it not only helps to communicate better, but also provides a cheapest way to communicate on long distance using free voice/video conferencing or to share files and directories. In this section, we introduce and discuss the protocols of some of the most popular leading brands among instant messaging services.

The main issue in instant messaging (IM) is that it does not exist a precise standardization of the architecture and protocols, as they are most often proprietary and the owners do not generally allow exchanging instant messages with users of rival IM applications, nor they disclose contents of the source code. All the information and software based on such systems, unless provided by the owner, is due to reverse engineering work. The IETF working group IMPP is working to develop architecture for instant messaging and presence service in order to provide a specification on how authentication, message integrity, encryption and access control must be integrated.

2.2.1 INSTANT MESSAGING AND PRESENCE

Instant messaging is a type of communication service providing users with two elements; presence information and real-time messaging [2]. As it is an essential part of instant messaging it is necessary to provide a definition for presence as well. **Presence** is a means for finding, retrieving, and subscribing to changes in the presence information (e.g. “online” or “offline”) of other users. [3].

2.2.2 PRESENCE SERVICE

Figure 2.1 displays an overview of a presence service. A presence service has two distinct types of clients: ‘presentities’ and ‘watchers’. Presentities provide presence information to the presence service, while watchers request presence information about presentities from the presence service. Naturally, the same application can act both as presentity and as a watcher.

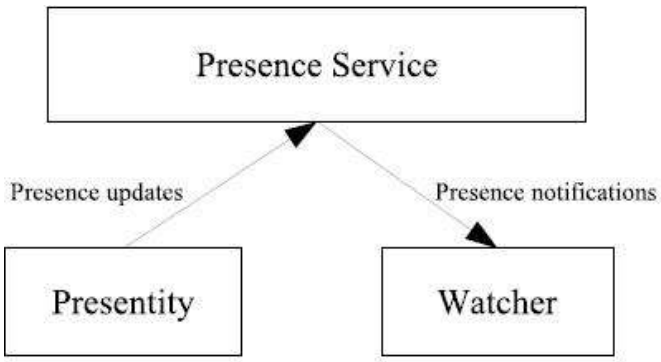


Fig 2.1:- Overview of presence service

The watchers are classified as well; there are ‘subscribers’, ‘fetchers’ and ‘pollers’. A subscriber is a watcher that has subscribed to the presence information of presentity. The presence service keeps track of the subscriptions and sends a notification to the subscriber whenever the presence information of the subscribed presentity changes. A fetcher requests the presence service for presence information about presentity. The presence service does not send notifications to fetchers, presence information is only sent upon the request of the fetcher. A poller is a special kind of a fetcher that polls the presence service for presence information about presentity on a regular basis.

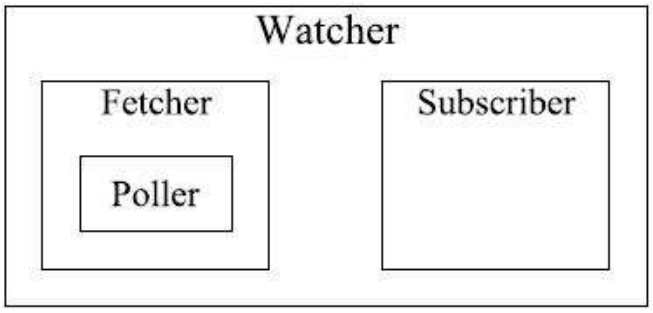


Fig 2.2:- Different kinds of watcher

2.2.3 INSTANT MESSAGING SERVICE

Equally to the presence service, the instant message service also has two kinds of clients: ‘senders’ and ‘instant inboxes’ (see Figure 2.3). Senders are the source of instant messages to be delivered by the instant message service. An instant inbox is a container for instant messages that are to be read by the owner of the inbox. The instant message service accepts instant messages from senders and attempts to deliver them to the instant inboxes, to which they are addressed.

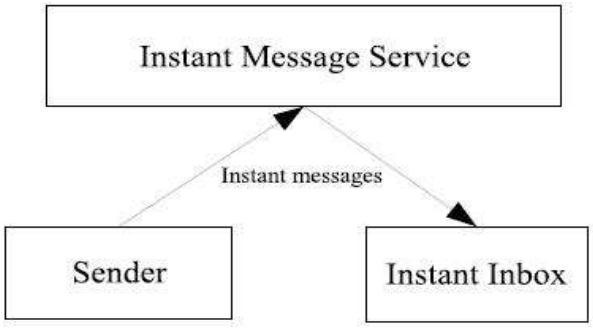


Fig 2.3:- Overview of Instant Message service.

2.2.4 STANDARDS AND PROTOCOLS

Ever since it became evident that users of the proprietary solutions would not be able to communicate with each other, IETF has been striving to produce a standardized solution for instant messaging. This section presents the results of the standardization efforts.

2.2.4.1 IMPP

IETF originally chartered IMPP (Instant Messaging and Presence Protocol) in order to define protocols and data formats necessary to build an internet-scaled instant messaging system. The working group managed to produce a model for presence and instant messaging in RFC 2778 [4] and requirements for an instant messaging protocol in RFC 2779 [3]. However, as stated in Section 2.2, the working group failed to achieve a common consensus for an instant messaging protocol. This resulted in the launch of several new working groups specifying protocols based on IMPP. It was decided that although the IMPP working group was not to specify any instant messaging protocol, it would carry on with its work. It would focus on producing standards for enabling interoperability between instant messaging systems. The working group has since created RFCs containing:

1. a common extensible instant message format (message/cpim)
2. a common extensible presence information format (application/pidf+xml)
3. a common profile for instant messaging (CPIM)
4. a common profile for presence (CPP)

CPIM [5] and CPP [6] specify semantics and data formats for common instant messaging and presence services. The goal of the profiles is to facilitate the creation of gateways between instant messaging systems for interoperability. CPIM uses the message/cpim MIME (Multipurpose Internet Mail Extension) type [7] as data format for instant messages while PIDF (Presence Information Data Format) [8] is used for formatting presence information in CPP.

In order for an instant messaging system to be IMPP compliant, it must conform to the CPIM and CPP profiles and their data formats as well as meet the requirements of RFC 2778 and RFC 2779.

2.2.4.2 SIMPLE

The IETF SIMPLE (SIP for Instant Messaging and Presence Leveraging Extensions) working group was one of the three working groups formed in 2000 when the IMPP working group failed to agree on a common protocol for instant messaging. Of the three, the SIMPLE solution is the only one still going strong; the other two have more or less failed. As the name indicates, SIMPLE is based on SIP (Session Initiation Protocol) [12]. The primary work of the SIMPLE working group is to generate an IMPP-compliant proposed standard SIP extension for instant messaging. SIMPLE defines the presence protocol as an instantiation of the general event notification framework for SIP [11]. For sending instant messaging SIMPLE provides two modes: a pager mode [10] and a session mode [9]. When using the pager mode instant messages are sent as SIP messages. In session mode SIP is used to initiate a session, in which the instant messages then are sent. The progress of the SIMPLE working group has been quite slow, mostly due to the complexity of the SIP protocol. Overall, the SIMPLE specifications currently consist of close to 20 Internet Drafts. Only a few Requests for Comments has been produced so far, including pager mode messaging specified in RFC 3428. Despite the slow-moving standardization process, the SIMPLE standard has gained support from several major companies including Microsoft, IBM and Yahoo.

2.3 TECHNOLOGY AND METHODOLOGY USED IN PROJECT

2.3.1 MOBILE INSTANT MESSAGING SYSTEM

Similar to instant messaging in the Internet, mobile messaging also took off in the last decade of the 20th century. Mobile IM and presence detection technology is revolutionizing the way in which we communicate by a factor equal to or greater than e-mail. Most communications today are time-delayed: You send an e-mail and wait for a reply, leave a phone message and wait for a return call, or submit a document for review and wait indefinitely for feedback. Mobile IM's presence concept means quicker responses in most cases because the sender knows at the time of sending a message that the other person is available online, thus reducing telephone and email by allowing simulated face-to-face communications.

Due to the huge number of mobile subscribers, there is a real demand to provide such a service to mobile clients. In order to do so, there is a need to keep track on both the presence and the locations of the participating clients. That is, each subscriber must be always aware about the where about of all the other members in its group, and who are the active members at any given moment. A mobile instant messaging service is expected to provide to its members an improved, easy-to-use inter-group connectivity, and to support advanced services such as a conference call, a multicast message, and a group search, in an efficient manner in near future.

As the word 'mobile' has several distinct interpretations depending on the context, attempting to provide an unambiguous definition of 'mobile instant messaging' is in place.

MobileIN.com provides a proper definition stating that **"Mobile Instant Messaging is the ability to engage in IM from a mobile handset via various bearer technologies, which may include SMS, WAP or GPRS" [1].**

Although implied by the examples of bearers in the previous definition, it should be stressed that only devices connected through wireless bearers can participate in mobile instant messaging. A device must be able to be both mobile and connected to a network simultaneously in order to engage in mobile instant messaging. A device connected to a wire line network that first is disconnected, then moved and finally reconnected does not fit this requirement. Furthermore, a handset is required to participate in mobile instant messaging. Therefore, using instant messaging from a laptop connected through a wireless connection is not considered mobile instant messaging as the laptop does not have the limitations of a typical handset.

2.3.2 AOL/Oscar

2.3.2.1 Overview

Oscar [Oscar] (Open System for Communication in Realtime) is the official IM protocol developed by AOL. Oscar is a closed protocol and all the knowledge available about it comes from reverse engineering. It is TCP-based and binary. Since Oscar is not an open protocol, non-proprietary clients do not support a lot of the features it offers, while it is easier to provide all the functionalities included in TOC [Proto].

2.3.2.2 Protocol Operation

The architecture of the Oscar protocol is illustrated in Figure 3. The server is distributed and the two main components are the *Authorizer*, which validates username and password and the *BOS* (Basic Oscar Service). Before connections are made to any of the BOS or special-purpose servers it is necessary to be authorized by the *Authorization Server* (login.oscar.aol.com). It will reply sending a cookie that automatically authorizes the user to connect to any of the BOS or special-purpose (for example Advertisement, Chat, etc) servers.

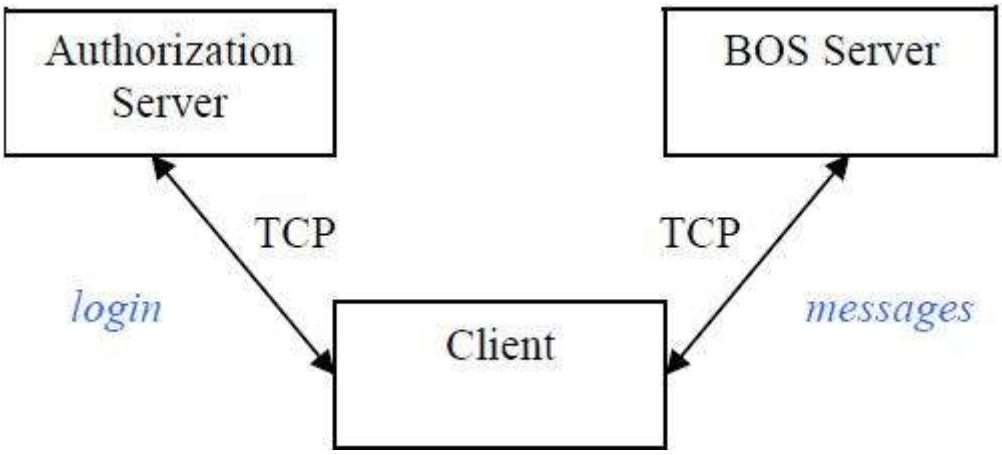


Fig 2.4:- Working of OSCAR Protocol.

The normal steps taken to create an average AIM session are as follows [Oscar]:

1. Connect to Authorizer and retrieve Cookie.
2. Connect to the Authorizer-recommended BOS server and initiate BOS service
3. (Optional) Connect to Advertisements server and retrieve first block of ads
   (repeat at regular interval)
4. (Optional) Connect to any other non-BOS services that may be available
   (AFAIK, none at this point)

The last three steps can actually follow any order, but authorization must always be the first.

2.3.3 Jabber

Jabber is the most widespread open source platform, using an XML encoded protocol, especially tailored to provide instant messaging (IM) and presence services over the Internet; however, Jabber is not designed just for this purpose, but several are the applications that may benefit and use the Jabber protocol suite. The protocol is totally free from legacy rights; both on the server and on the client side, which means that anyone can design its own Jabber client and even that any organization can freely implement an internal jabber server. Many are the advantages that come out from this approach:

1. The fact that the protocol is open lead to a better understanding of it, as everyone can learn from the work previously done and make available its code to other developers for the same purposes.

2. XML allows easy extensibility to the main features of the protocol. The Jabber Software Foundation accounts for the common extensions.

3. Decentralized approach. Since any organization can have its Jabber server, the resulting architecture is more scalable as lighter load is posed on the single servers, compared to a centralized approach.

Although the protocol itself does not provide means to achieve interoperability with other IM protocols, this is possible by means of server-side gateways, which take care of the communication between users in the Jabber space and users in the space of other (possibly) proprietary protocols. Jabber-related activities are ongoing in the Extensible Messaging and Presence Protocol (XMPP) IETF working group [XMPP]; the working group has been chartered to discuss extensions to the XMPP protocol, which is the core of the Jabber platform, especially to be compliant to the requirements posed by [RFC2779], RFC from the IETF Instant Messaging and Presence Protocol (IMPP) working group, chartered to define a standard protocol for providing instant messaging and presence services. A good document describing Jabber main features is [SA01].

2.3.3.1 Protocol Operation

Jabber is essentially a client-server architecture, where users register to a server to have access to the Jabber system, and use that server as intermediary when exchanging messages with other users, also if they are registered with another server. User names in Jabber are in e-mail like format, such as userA@serverA.com.

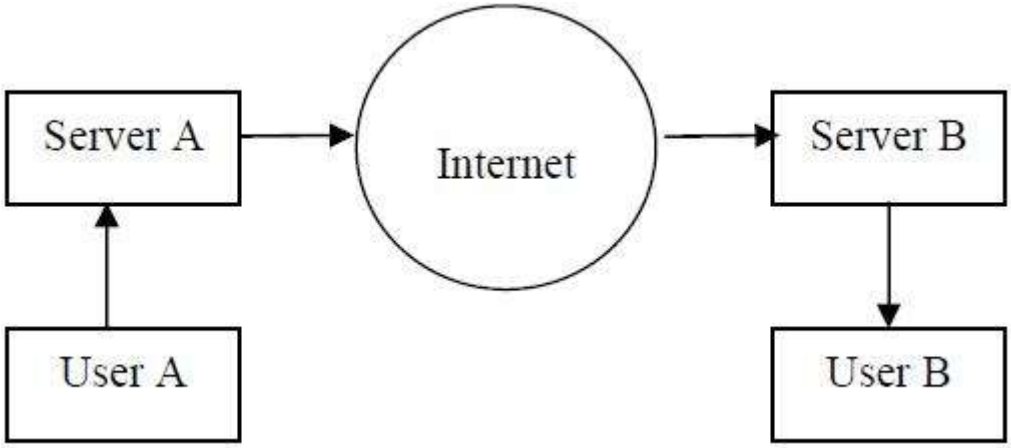


Fig:- Simplified Jabber Network Architecture.

The Jabber protocol basically foresees the following kinds of messages:

1. <message/> used to actually carry the messages exchanged.
2. <presence/> used for the service of presence
3. <iq/> (info/query) used for messages of other kind, such as for authorization purposes.

The exchange of message in the Jabber protocol is stream-based, and a Jabber session is indeed identified by two XML streams flowing in the two client-server directions. The protocol runs over TCP, using the well known, at protocol level, port 5222. A typical message exchange in a Jabber session, could be the following:

1. A Jabber client sends a message to a Jabber server to open a connection and initiate an XML stream.

2. The server sends its reply and opens an XML stream for the server to client direction.

2.3.4 XMPP

The Extensible Messaging and Presence Protocol (XMPP) is beyond doubt the strongest challenger to the SIMPLE standard in the Internet. Like SIMPLE, XMPP is also administered by an IETF working group, i.e. the XMPP working group. XMPP has been formed from the basis of the Jabber protocol. The Jabber protocol is the result of a project started by Jeremie Miller in 1998 [13]. The goal of the project was to produce an interoperable and open instant messaging protocol, an alternative to the proprietary solutions. The first public release of the protocol took place in May 2000. In June 2000 project members sent an Internet Draft of the Jabber protocol to the IMPP working group (see Section 2.5.1) as an instant messaging protocol proposal. However, the organization of the Jabber project was not mature enough at the time and the Internet Draft was left to expire. In 2001 the Jabber Software Foundation was formed to organize the projects and commercial bodies involved in the Jabber community. Following the reorganization, a new Internet Draft was submitted to the IETF in February 2002, eventually leading to the birth of the XMPP working group in October 2002. XMPP is in essence the core of the XML (Extensible Markup Language) based Jabber protocol. XMPP has been made IMPP compliant by the working group. The Jabber Software Foundation will continue to work on the parts of the Jabber protocol that are not part of XMPP or IMPP, exploring features such as: multi-user chat, calendaring and whiteboarding.

Compared to SIMPLE, the XMPP solution is more mature. Recently, three of four Internet Drafts have been approved as Proposed Standards and the protocol has already been widely deployed with tens of thousands of active servers and millions of users. Although it has not acquired the support of as many major companies as SIMPLE, XMPP is also embraced by big enterprises, including Hewlett-Packard and Intel. An in-depth comparison of SIMPLE and XMPP from the viewpoint of XMPP can be found in [14].

CHAPTER 3

ANALYSIS AND DESIGN

3.1 REQUIREMENT ANALYSIS

3.1.1 FUNCTIONAL REQUIREMENTS

3.1.1.1 SERVER

The server holds all the information for the buddy lists and the profiles of all the clients. The server allows multiple clients to connect to the network and have a group chat.

3.1.1.2 CLIENT

The client can connect to the server to have a group chat as well as private chats with other clients. They can have buddy lists and profiles for each individual user. There is a complex GUI for the client that entails every aspect it is supposed to do

3.1.1.2.1 Description and Priority

- Allows for an unlimited amount of private, one-on-one chats between any two users.
- Priority = HIGH

3.1.1.2.2 Stimulus/Response Sequences

- Stimulus: User double clicks on a name in the buddy list.
- Response: System presents new message dialog.
- Stimulus: User selects to send a new message from the menu.
- Response: System presents new message dialog with a username field.

3.1.1.2.3 Functional Requirements

- REQ-1: System queries server to ensure desired contact is online
- REQ-2: System queries server to verify send permissions
- REQ-3: System queries server to determine IP address.
- REQ-4: System attempts to contact client at appropriate IP address.
  On error: System attempts to retry several times
  On further error: System reports that client is unreachable and notifies server.

3.1.1.3 CONNECTIONS TO OTHER MESSENGING SYSTEMS

3.1.1.3.1 Description and Priority

- Allows users to connect to third-party networks.
- PRIORITY = HIGH / MEDIUM (High for at least one network, medium for all others)

3.1.1.3.2 Stimulus/Response Sequences

- Stimulus: User attempts to connect to third-party networks.
- Response: System authenticates and connects to third-party network.

3.1.1.3.3 Functional Requirements

- REQ-1: System must support a number of third party networks.

3.1.1.4 LOGGING

3.1.1.4.1 Description and Priority

- Allows for client side and server side logging of chats and private messages.
- PRIORITY = MEDIUM

3.1.1.4.2 Stimulus/Response Sequences

- Stimulus: User enables logging
- Response: System logs text of all future conversations

- Stimulus: Administrator enables logging of a particular chat
- Response: System logs chat to specified log file

3.1.1.4.3 Functional Requirements

- REQ-1: System transparently writes all text to a file on the system, whether server-side or client-side.

3.1.1.5 BUDDY LIST / PROFILE SUPPORT

3.1.1.5.1 Description and Priority

- Allows for server-side storage of Buddy Lists and user created Profiles.
- PRIORITY = HIGH

3.1.1.5.2 Stimulus/Response Sequences

- Stimulus: User creates a buddy list.
- Response: Server Responds by connecting to third party APIs.

- Stimulus: User updates buddy list.
- Response: System updates the stored buddy list.

- Stimulus: Admin updates a user's buddy list
- Response: System updates the stored buddy list.

- Stimulus: User creates a profile
- Response: System stores profile on server

- Stimulus: User uploads a picture for identification
- Response: System checks size of picture then accepts or denies it.
- Stimulus: Admin edits a profile
- Response: System re-writes the profile file.

3.1.1.5.3 Functional Requirements

- REQ-1: System must support writing of files to the local hard drive
- REQ-2: System may support a lightweight database backend.
- REQ-3: System must transfer buddy lists to clients upon connection

3.1.2 NON FUNCTIONAL REQUIREMENTS

3.1.2.1 Operating platform

- Android OS

3.1.2.2 Reliability

The server applies all translation algorithms as well as establishing connections which means that as long as the client is connected to the server the entire application works.

3.1.2.3 Portability

The users can launch the application where ever internet connection is available since it's a mobile device

3.1.2.4 Performance

The performance of the system depends on the device it is being used. A 3G enabled device may have high performance. The internet connection with high speed will have high performance.

3.1.2.5 Authentication

The user is authenticated on application start up. The user and password are checked by the server. There should be no duplicate user names when the account is created. Once the user is connected to the server the contacts are loaded into the user's device. The Auto login feature will enable users to sign into different account upon application start-up.

3.1.2.6 Scalability

As the users increase the system could be implemented on a cloud service. The project may not require large server space since it is related more to academic purpose and not commercial.

3.1.3 Constraints

3.1.3.1 Message length

Messages may be a maximum length of 500 characters.

3.1.3.2 Credentials

User names and passwords must be a maximum length of 25 characters. Usernames must be a minimum of five characters, and must not be duplicated in the login database. Passwords must be a length of at least four characters, and may be duplicated in the Login database. All passwords must be hidden fields.

3.1.4 HARDWARE & SOFTWARE REQUIREMENTS

3.1.4.1 HARDWARE REQUIREMENTS

Hardware Requirement	**Development**	**Use**
Android Mobile Phone	Android OS 2.3 or higher	Deployment Platform
Mobile Phone Memory	1.5 Mb free space	Application Size: 840kb (before installation)
GPRS	LAN/WAN/WIFI	Internet access for connecting to the server

3.1.4.2 SOFTWARE REQUIREMENTS

Software Requirement	**Development**	**Use**
Windows Operating System	Windows Vista/ 7	For development of application
Jdk	1.6 and higher version	For developing code.
Eclipse IDE	Java EE bundle	Environment setup
ADT Plugin		To be used with eclipse
Android SDK	r12 or higher	For development of application
Android SDK manager	To download Android OS platforms	API for android devices
AVD manager	Creating a emulator	For running the application

3.2 PROJECT DESIGN

3.2.1 xADL Diagram:-

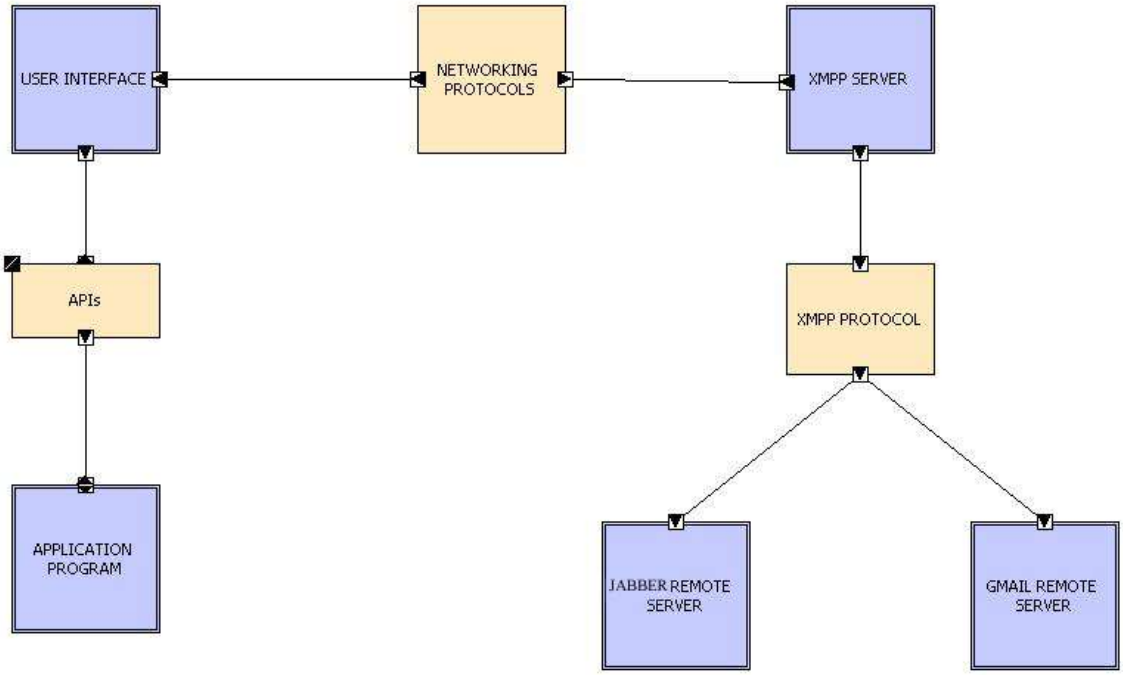


Fig 3.1:- xADL Architecture of the System

3.2.2 USE CASE DIAGRAM:-

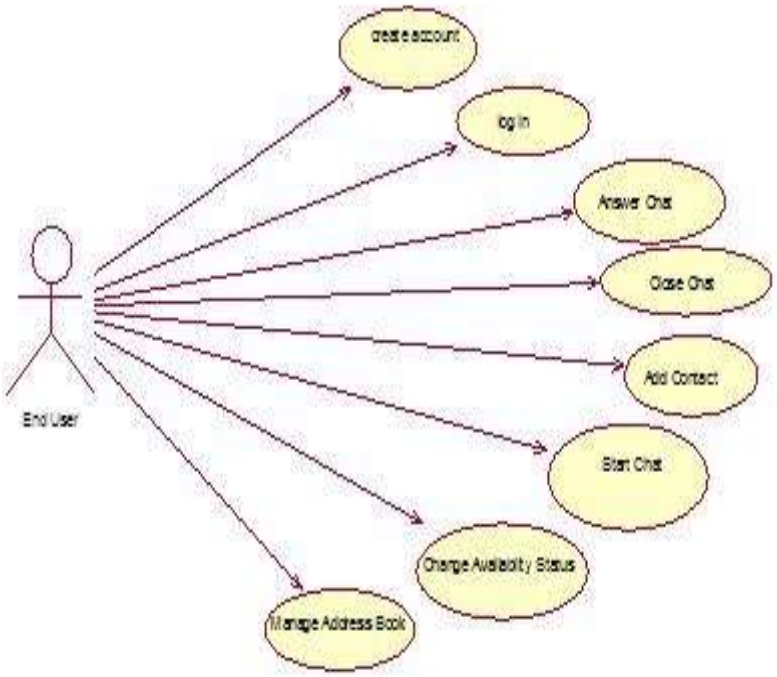


Fig 3.2:- Use Case Diagram for the overall system.

3.2.3 ACTIVITY DIAGRAM:-

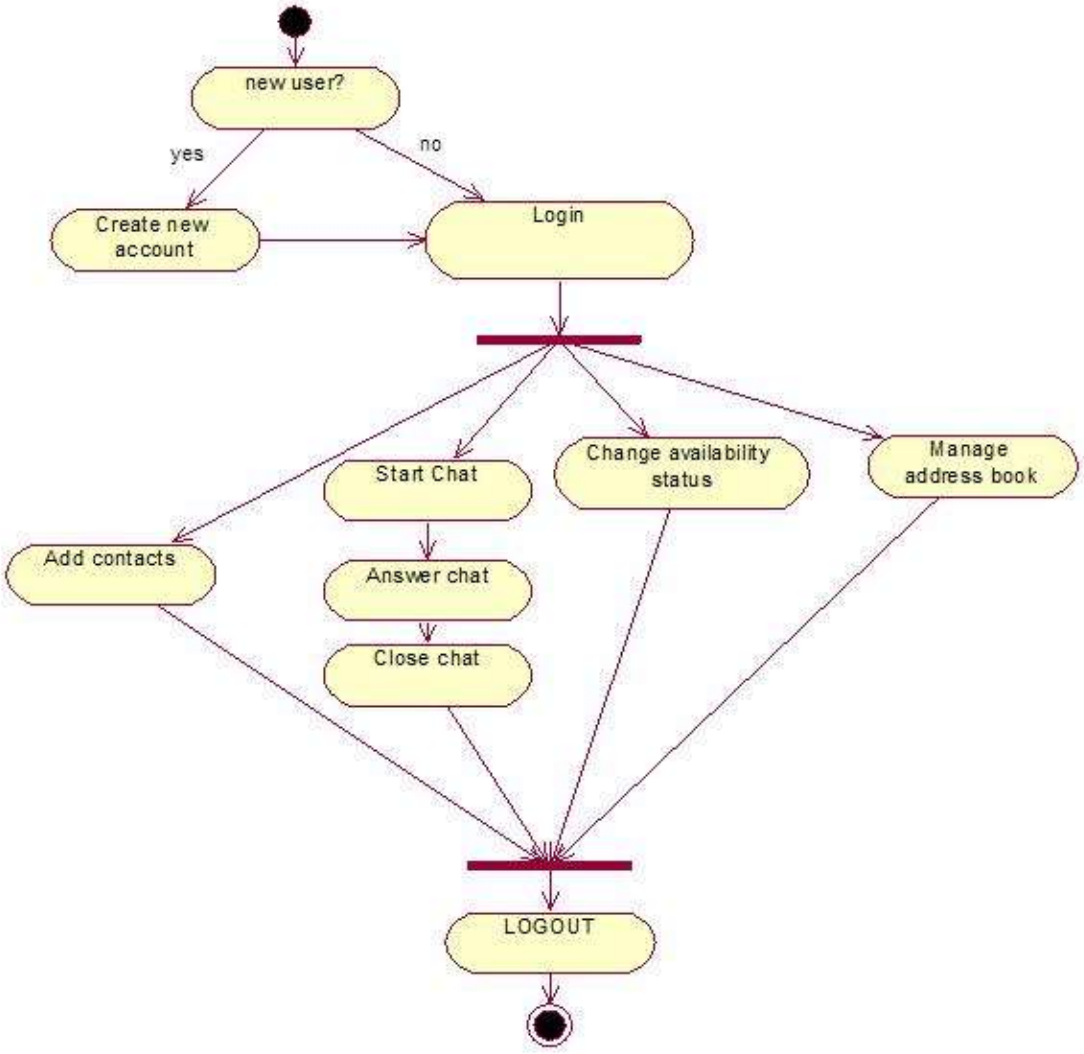


Fig 3.3:- Activity diagram

3.2.4 STATE CHART DIAGRAM:-

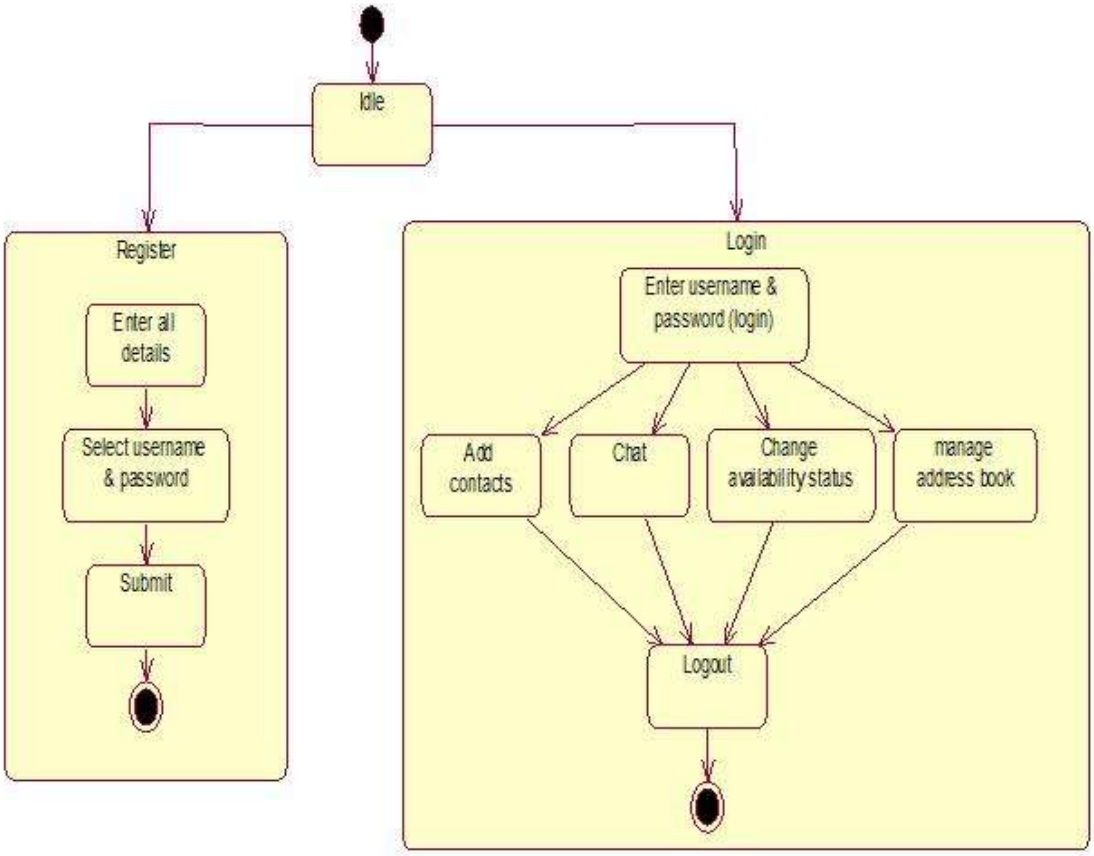


Fig 3.4:- State Chart Diagram

3.2.5 SEQUENCE DIAGRAM:-

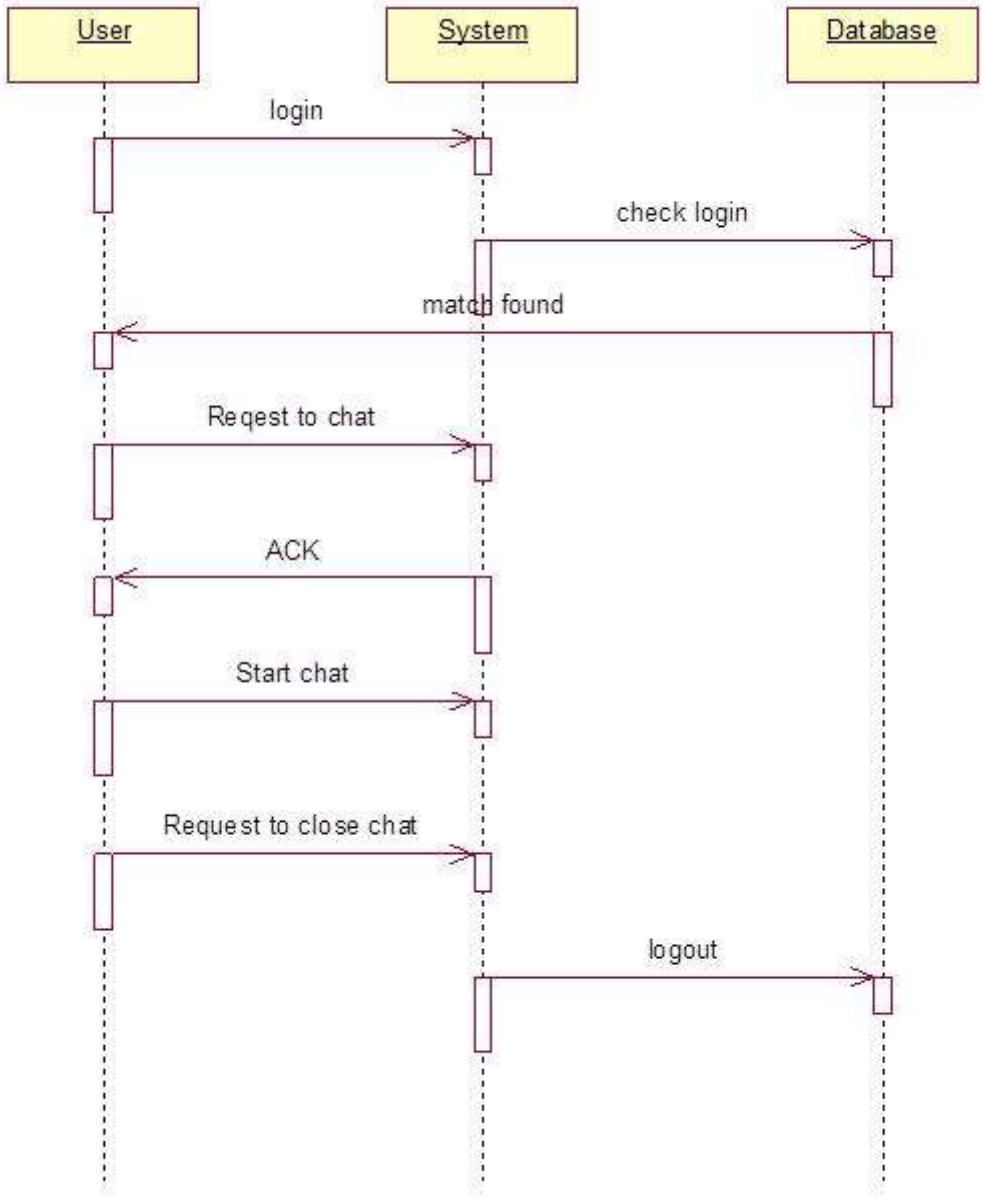


Fig 3.5:- Sequence diagram

3.2.6 COLLABORATION DIAGRAM:-

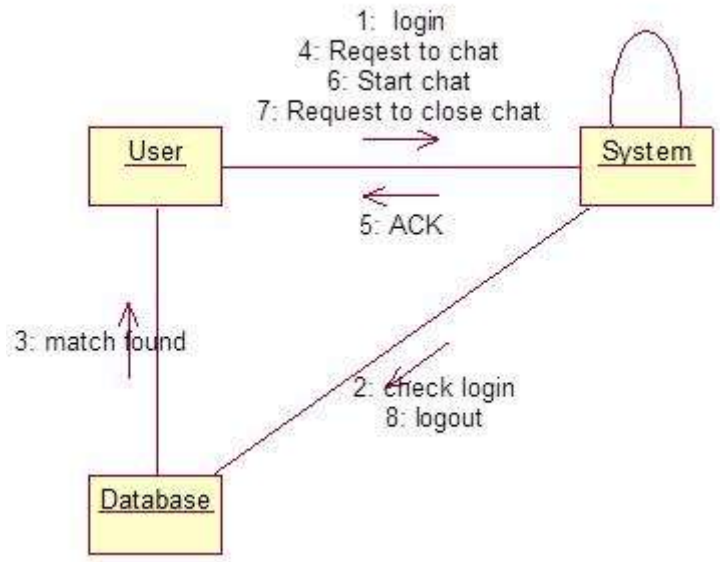


Fig 3.6:- Collaboration Diagram

CHAPTER 4

IMPLEMENTATION AND RESULTS

4.1 IMPLEMENTATION DETAILS

Implementation is the realization of an application, or execution of a plan, idea, model, design, specification, standard, algorithm, or policy. In computer science, an implementation is a realization of a technical specification or algorithm as a program, software component, or other computer system. Many implementations may exist for a given specification or standard. In implementation of a software project the design model prepared is converted into a robust, reusable code using appropriate coding language. The implementation process runs along with the unit testing for checking the correctness of each coding unite. Different modules of the software are implemented separately and after appropriate testing these are integrated with each other logically to form the entire product.

4.1.1 LOGIN

- The user must enter the account username, which is his JID (jabbered). The jabber id could be a Gmail username as well.
  << private void checkUsername (String username){}>>
- The user must enter the account password.
  << private void checkPassword(String password){}>>
- If the user is using a proxy server, he must enter the IP address of the server.
- The IIMS Application uses xmppConnectionAdapter to create a socket connection to XMPP server. This is a default connection to a Jabber/gmail server.
- The jabber/gmail server authenticates the username and password.
  - If username or password is incorrect the user gets notified by the message “bad login and username”.
  - If the remote server is down, the error message is “remote server error.

4.1.2 IIMS CONTACTS

- Once the user is authenticated.
  - The buddy list is retrieved from the existing accounts database from the Xmpp server.
  - If the buddy/contact list is empty the user can add contacts to the buddy list using the add contact option in the menu.
- Buddy list is the dialog showing the buddies associated to the connected accounts. Additionally the status of active connections is displayed using the presence service.
  <<public class PresenceAdapter >>.
  This class imports a smack library from the jive software.
  << import org.jivesoftware.smack.packet.Presence; >>
- The user can scroll the contact list using the andrioid.app interface.
- The contact can be added to specific groups. When adding a contact not already existing in the buddy list the contact name, alias and group is to be specified.
- Upon the next login the users will be added to the specific groups.

4.1.3 ADD CONTACT

If the user is added via the IIMS application.

- Contacts are added using the AddContact.java file. The user must first click on the menu button on the bottom and select addcontact.
- On click (onClick(view v)) the add contact layout appears. The email id is entered in the contact text box. The email id is matched to check whether the email id is valid.
  - If the contact is already added then a prompt message appears “contact already exists”.
  - If the email has error the prompt “error login”.
  - Else the contact is added in the buddy list.

If the user is added from other source other than the IIMS application.

- If a user wants to add a contact using other service except iims application and if the requested user is using the iims application during the request, a subscription notification is displayed on the android device.
- The iims user when taps on the notification, the user is directed to the iims application where he can accept or deny the subscription request.

4.1.4 MESSAGING

When using the messaging feature the application requires user to have a Gmail account. While connecting the application uses Google talk server. Google talk server provides the service choice. The Google Talk network supports open interoperability with hundreds of other communications service providers through a process known as federation. This means that a user on one service can communicate with users on another service without needing to sign up for, or sign in with, each service.

The Google talk server support open federation with any service provider that supports the industry standard XMPP protocol.

- To message a contact tap on the active contact. The android text box appears where the user can type in the message.
- The message window consist of the following:
  - The receiver of the message.
  - The type of message.
  - The body of the message.
  - The sender of message.
  - The time stamp.

- After writing the body of the message the user can send the message using the send button.

  - If the receiver is available the message will be sent to the user.
  - If the receiver is unavailable the following cases should be considered

    - If(gmail user)

      The message will be stored on to the gmail xmpp server. When the receiver is online the user will receive the message.

    - If(jabber user)

      The message will be stored on to the jabber xmpp server. When the receiver is online the user will receive the message

    - If(aim user)

      The message will not be received till the user is online. The error message “recipient-unavailable” is prompted on the chat window.

4.1.5 OTR MESSAGING

Off-the-record Messaging, commonly referred to as OTR, is a cryptographic protocol that provides strong encryption for instant messaging conversation. OTR uses a combination of the AES symmetric-key algorithm, the Diffie-Hellman key exchange, and the SHA-1 hash function.

- The iims application provides a support to have private conversations over instant messaging by using off-the-record (OTR) Messaging feature.
- While chatting both the user can start an OTR session by using the OTR actions from the chat menu.
- When the OTR session is activated from both the users the text gets encrypted on the communication channel.

4.1.6 IIMS PERMISSIONS

Here is a list of different permissions used by IIMS.

- android.permission.internet: to access the internet connection.
- android.permission.vibrate : to set vibrate on IIMS notifications
- android.permission.write_external_storage : to stock the avatar data on the sd card
- android.permission.access_network_state : to check the connectivity status
- com.iims.project.iims.IIMS_SERVICE : Custom permission which will be used to allow other applications to use the iims service

4.1.7 THREADS

The events from the user interface are dispatched from the main thread. To make it fell efficient there are threads maintained for each connection. The idea is to do background job while being interactive to user. The communication basic is show below.

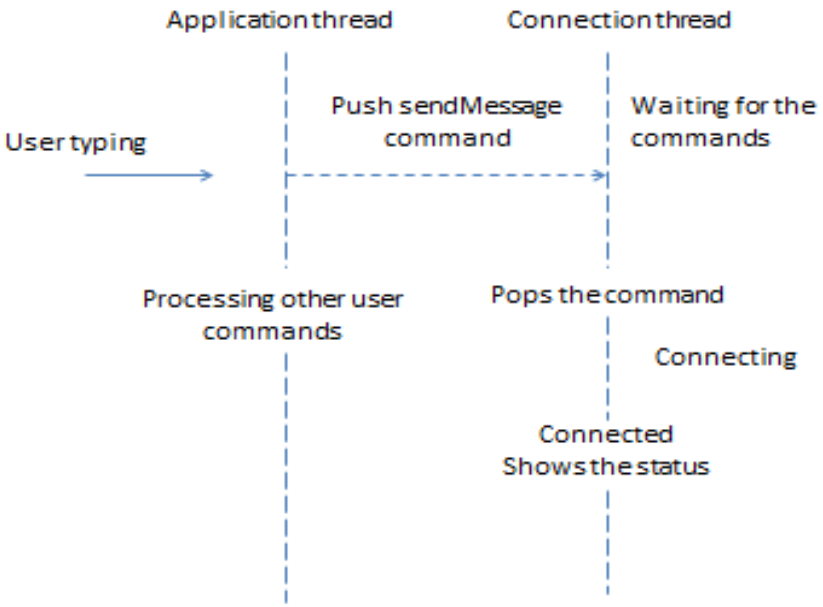
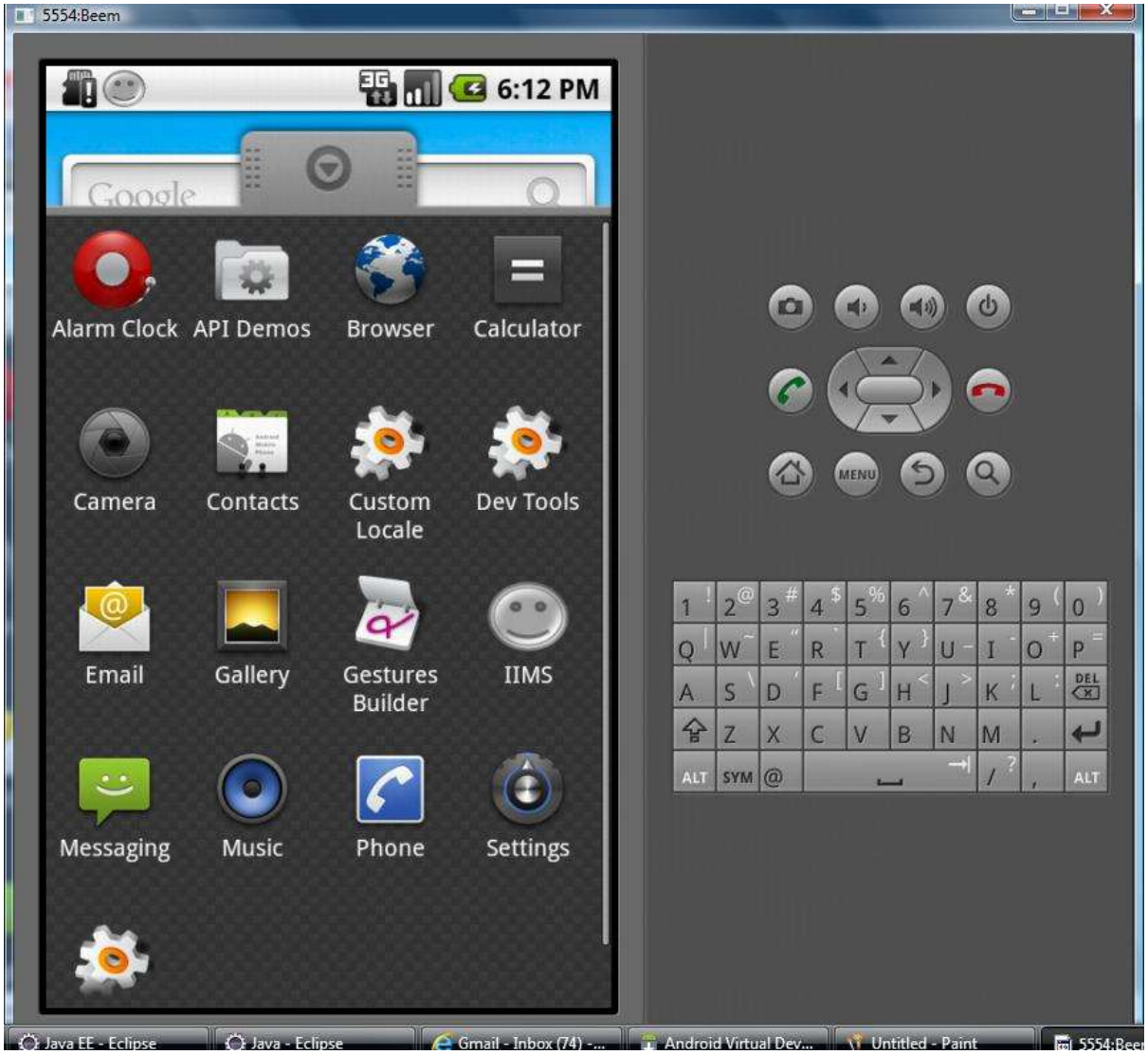


Fig 4.1: - Threads for connection

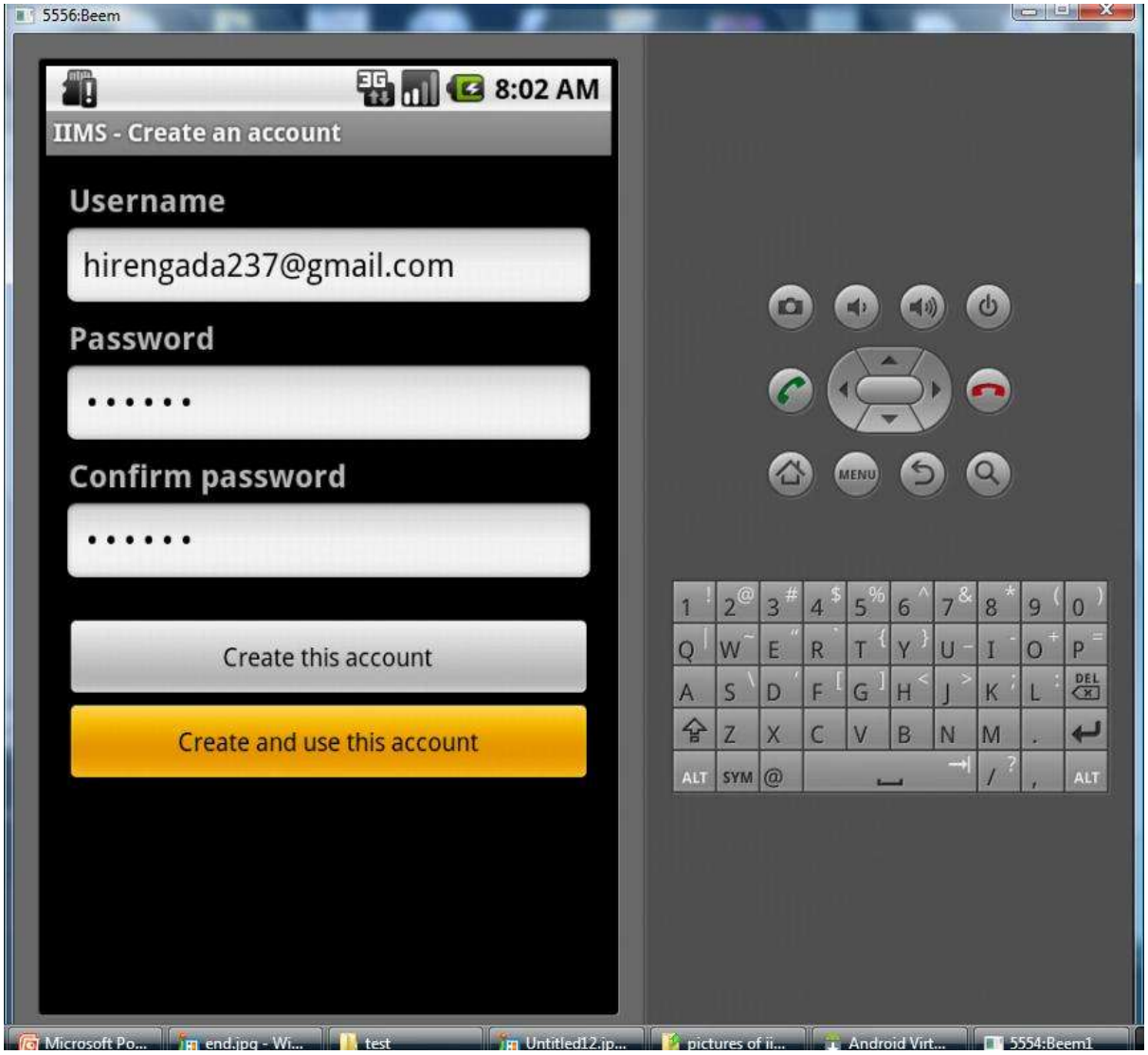
4.1.8 GANTT CHART

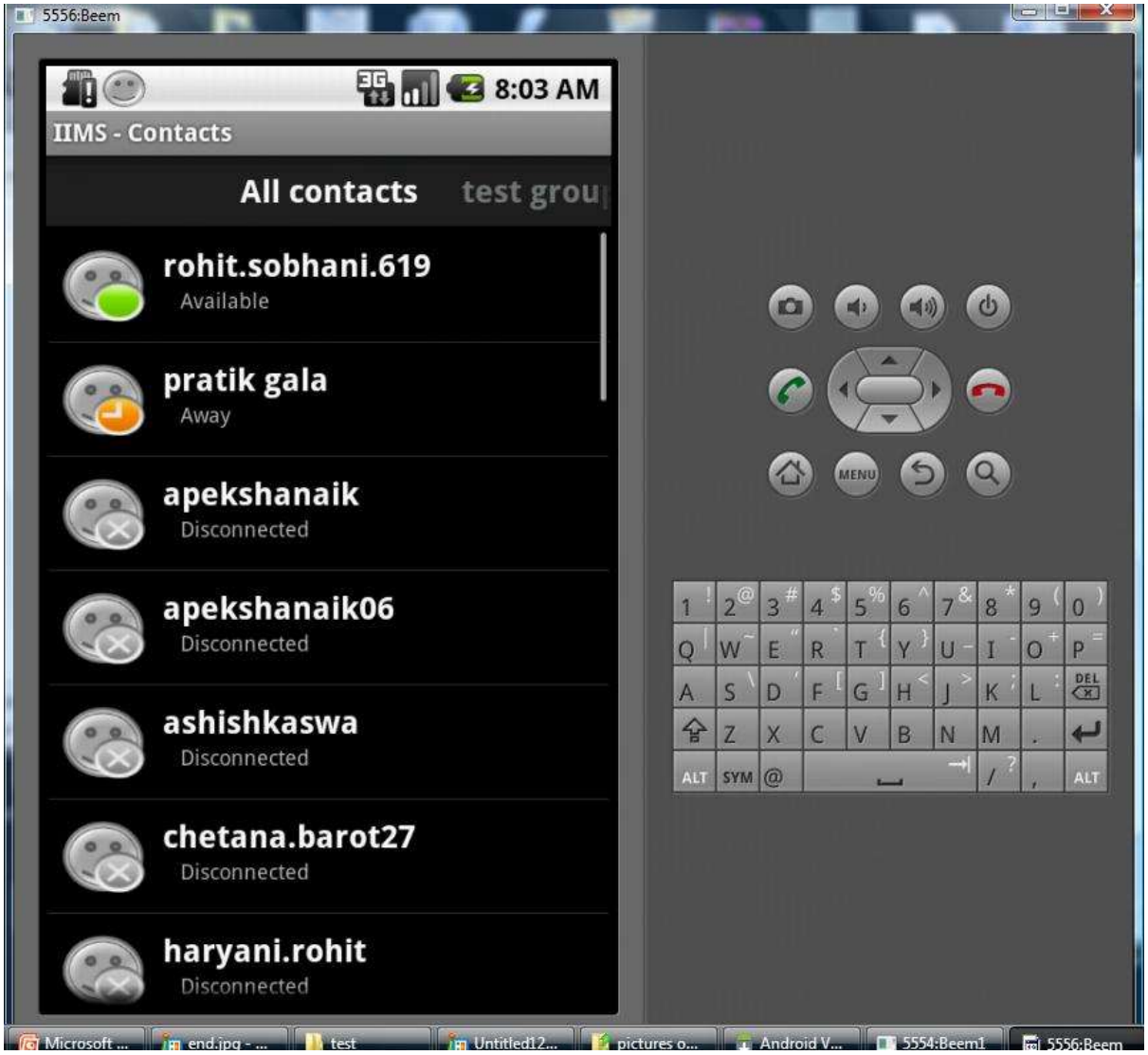
Number	Task	Start	End
	My Project	8/8/11	12/4/12
1	**INCEPTION PHASE**	**8/8/11**	**29/9/11**
1.1	ESTABLISH PROJECT STATEMENT	8/8/11	8/8/11
1.2	REVIEW OF LITERATURE	10/8/11	17/8/11
1.3	STUDY ON INSTANT MESSAGING	18/8/11	8/9/11
1.4	STUDY ON EXISTING SYSTEM	9/9/11	29/9/11
2	**ELABORATION PHASE**	**30/9/11**	**20/10/11**
2.1	SOFTWARE REQUIREMENTS	30/9/11	4/10/11
2.2	HARDWARE REQUIREMENTS	4/10/11	5/10/11
2.3	GATHERING FUNCTIONAL REQUIREMENTS	7/10/11	12/10/11
2.4	GATHERING NON FUNCTIONAL REQUIREMENTS	15/10/11	19/10/11
2.5	ELABORATION PHASE COMPLETED	20/10/11	20/10/11
3	**DESIGN PHASE**	**21/10/11**	**4/11/11**
3.1	SYSTEM DESIGN	21/10/11	25/10/11
3.2	UML MODELING	1/11/11	4/11/11
3.2.1	DEVELOPING DATA FLOW DIAGRAM	1/11/11	2/11/11
3.2.2	DEVELOPING SEQUENCE DIAGRAM	3/11/11	3/11/11
3.2.3	DEVELOPING COLLABORATION DIAGRAM	3/11/11	3/11/11
3.2.4	DEVELOPING ACTIVITY DIAGRAM	4/11/11	4/11/11
3.3	DESIGN PHASE COMPLETED	4/11/11	4/11/11
4	IMPLEMENTATION PHASE	16/1/12	10/3/12
4.1	CLIENT AUTHENTICATION	16/1/12	30/1/12
4.2	CREATING CHAT BOX	31/1/12	13/2/12
4.3	INTEGRATION OF NEW CHAT NETWORKS	16/2/12	3/3/12
4.4	MAKING GUI USER FRIENDLY	4/3/12	10/3/12
5	TESTING	11/3/12	20/3/12
5.1	CHANGES INCORPORATED BASED ON TESTING	11/3/12	20/3/12
6	DOCUMENTATION PHASE	22/3/12	5/4/12
6.1	WRITING REPORTS	22/3/12	10/4/12
6.2	CREATING PRESENTATION	10/4/12	11/4/12
6.3	DOCUMENTATION COMPLETED	12/4/12	12/4/12

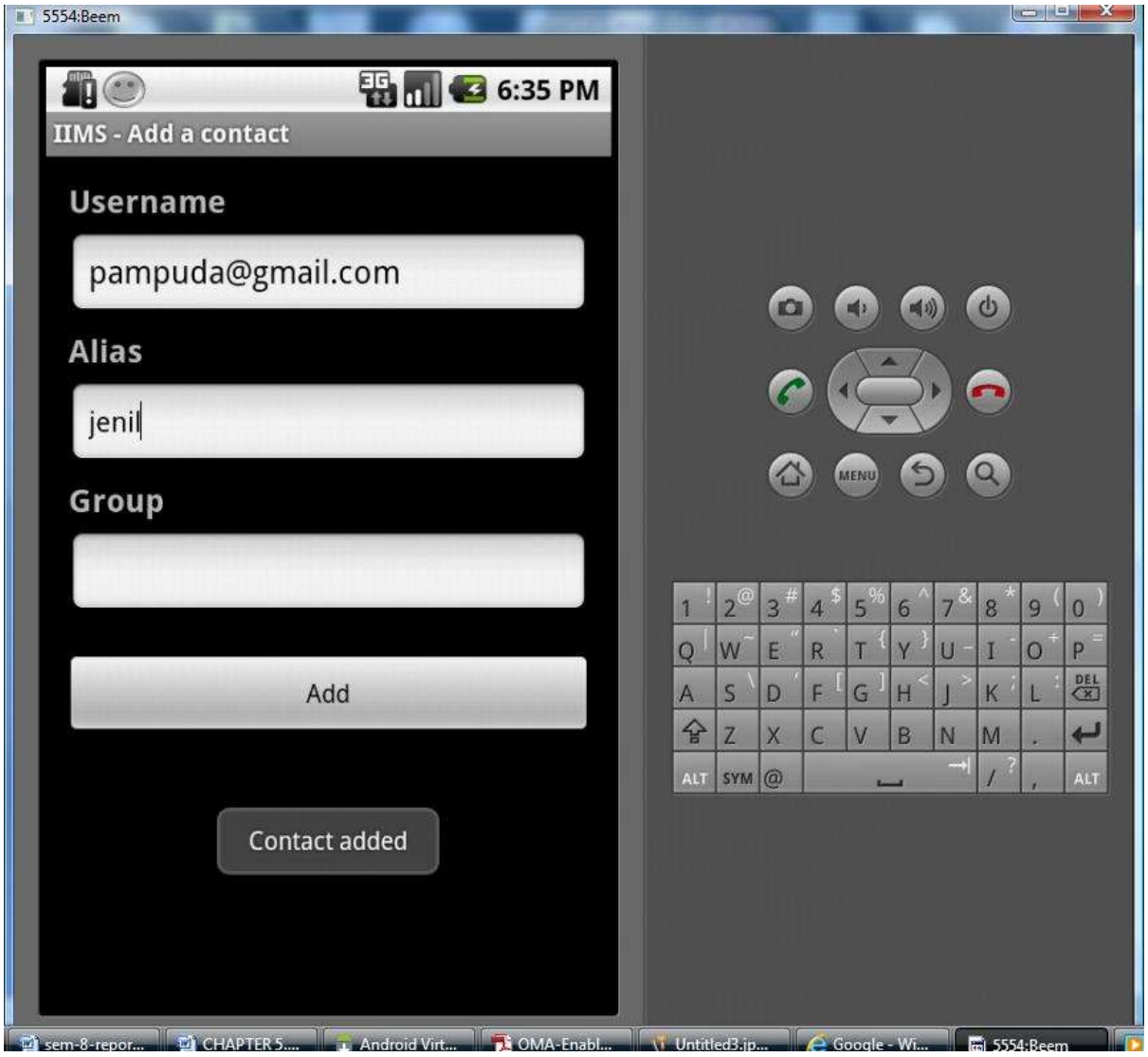
4.2 RESULTS:-

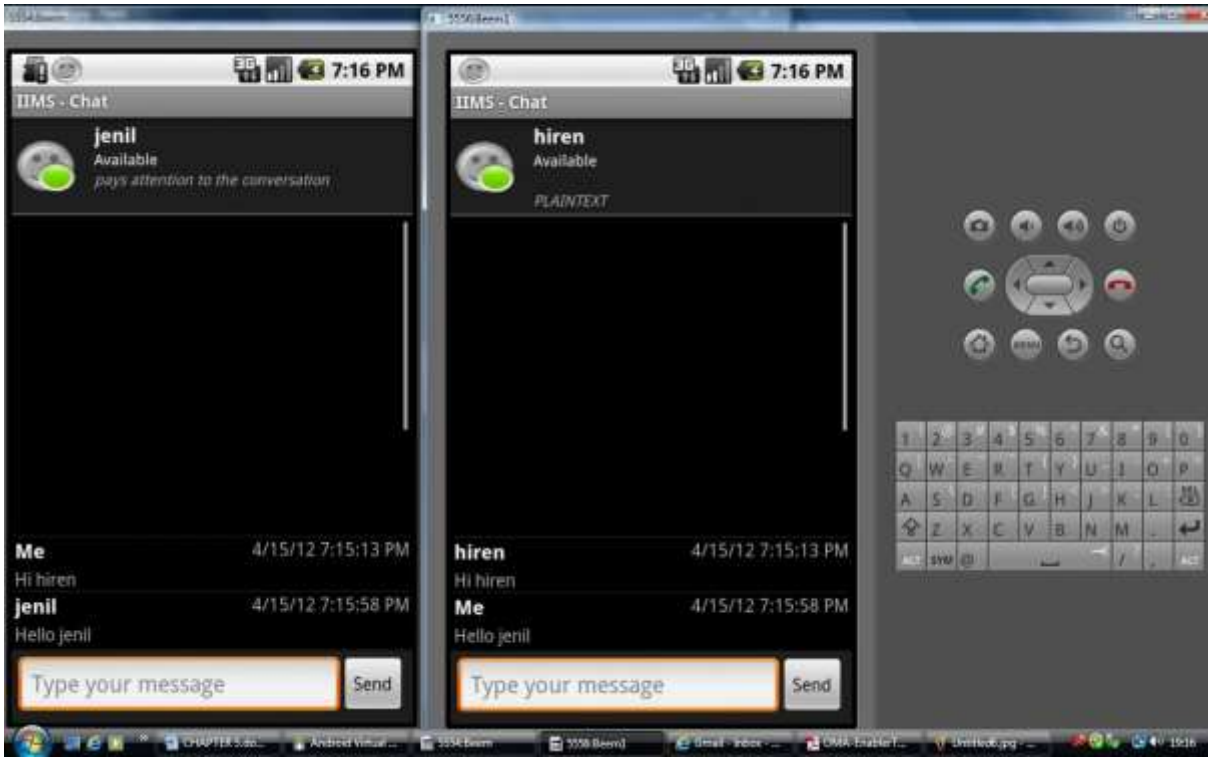

5554:Beem
6:12 PM
Google
Alarm Clock
API Demos
Browser
Calculator
Camera
Contacts
Custom Locale
Dev Tools
Email
Gallery
Gestures Builder
IIMS
Messaging
Music
Phone
Settings
MENU
Java EE - Eclipse
Java - Eclipse
Gmail - Inbox (74) - ...
Android Virtual Dev...
Untitled - Paint

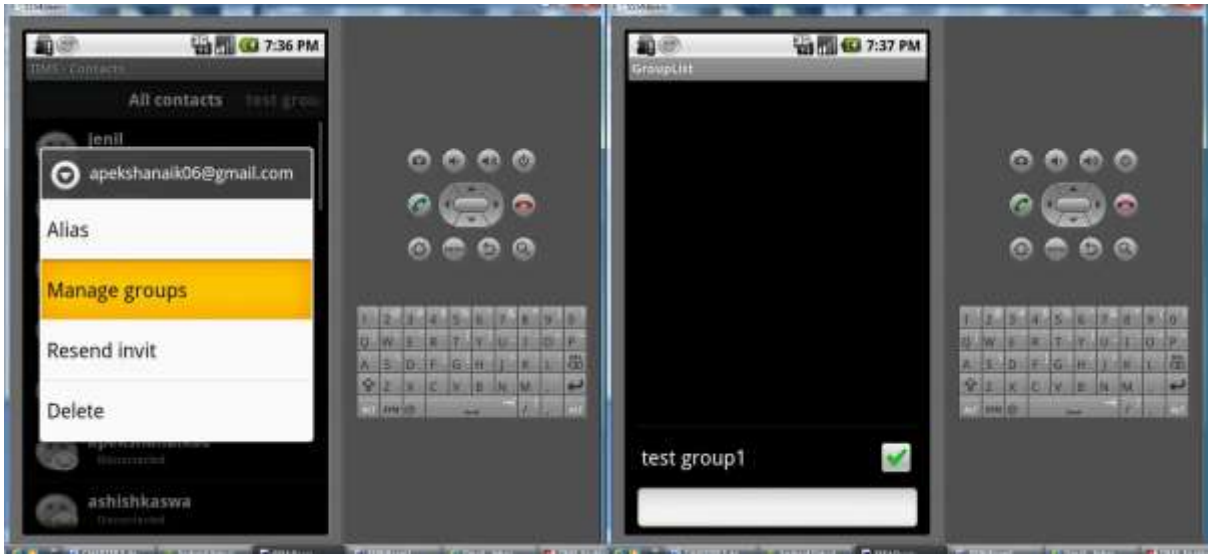

5554:Beem
1:54 PM
Account wizard
IIMS
Welcome on IIMS(Integrated Instant Messaging System).
You have not configured an XMPP (Jabber) account yet.
Choose one of the following options :
I already have an account I want to use
I want to register for a new account
Continue
MENU
Java - Eclipse
Gmail: Email from G...
5554:Beem

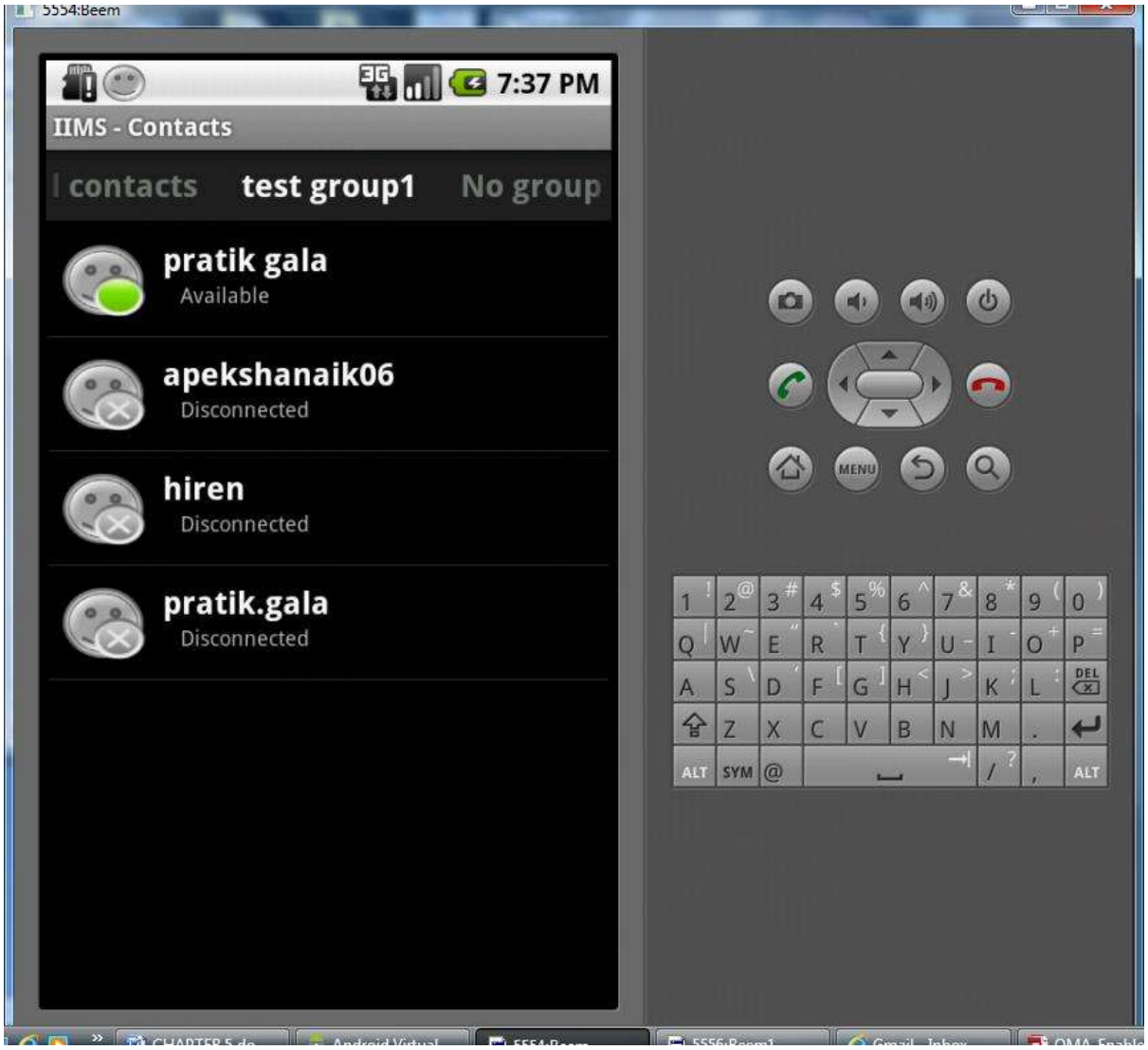

5556:Beem
8:02 AM
IIMS - Create an account
Username
hirengada237@gmail.com
Password
Confirm password
Create this account
Create and use this account
MENU
Microsoft Po...
end.jpg - Wi...
test
Untitled12.jp...
pictures of ii...
Android Virt...
5554:Beem1

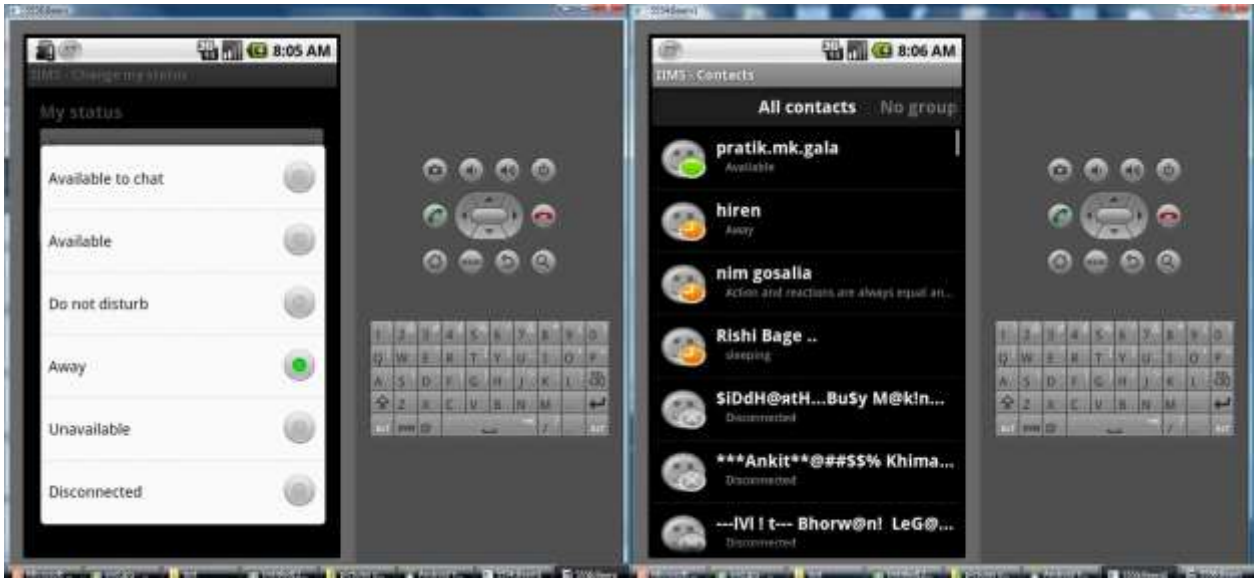

5556:Beem
8:03 AM
IIMS - Contacts
All contacts
test grou
rohit.sobhani.619
Available
pratik gala
Away
apekshanaik
Disconnected
apekshanaik06
Disconnected
ashishkaswa
Disconnected
chetana.barot27
Disconnected
haryani.rohit
Disconnected
MENU
Microsoft ...
end.jpg - ...
test
Untitled12...
pictures o...
Android V...
5554:Beem1
5556:Beem

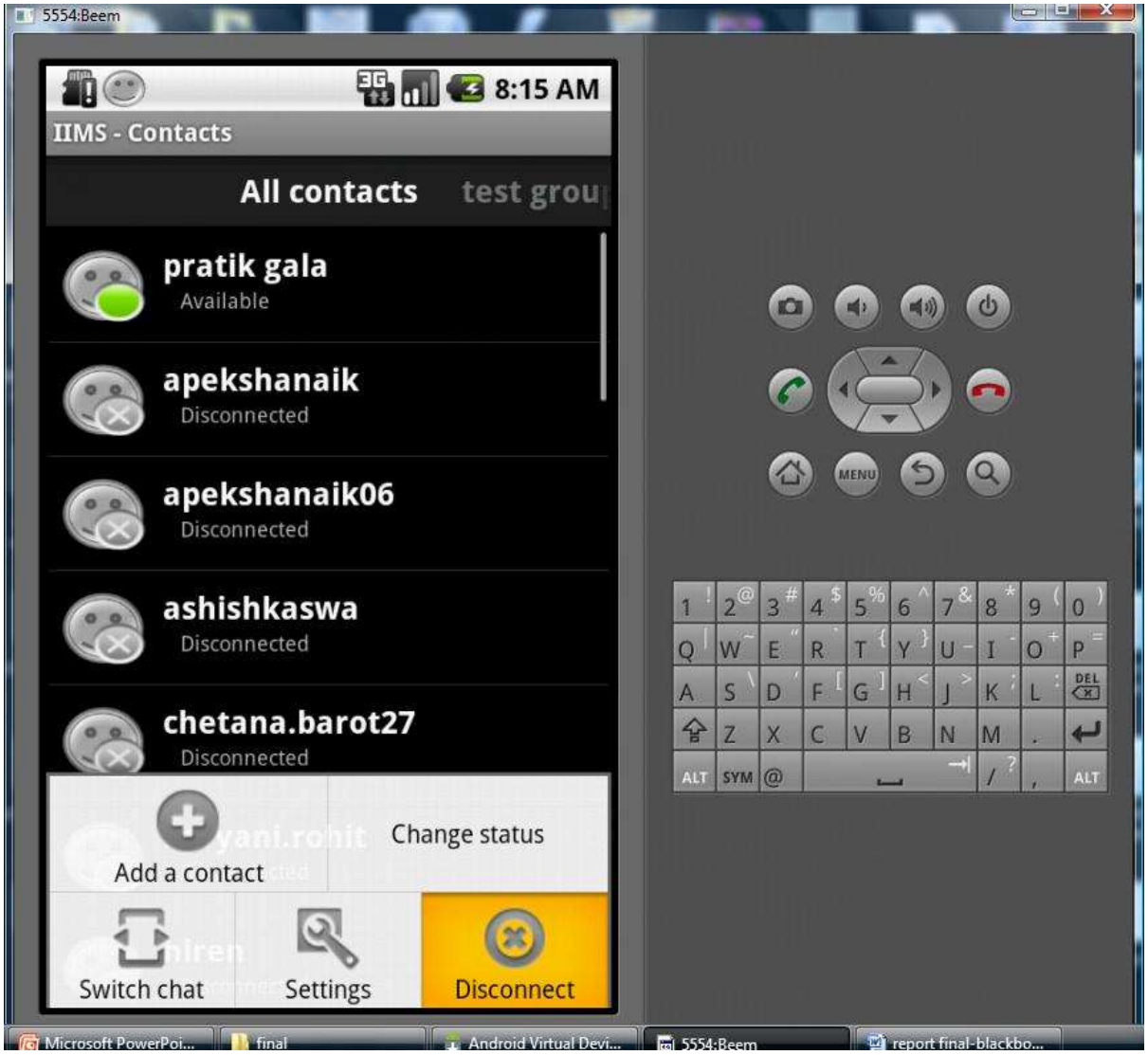

5554:Beem
6:35 PM
IIMS - Add a contact
Username
pampuda@gmail.com
Alias
jenil
Group
Add
Contact added
MENU
sem-8-repor...
CHAPTER 5....
Android Virt...
OMA-Enabl...
Untitled3.jp...
Google - Wi...
5554:Beem

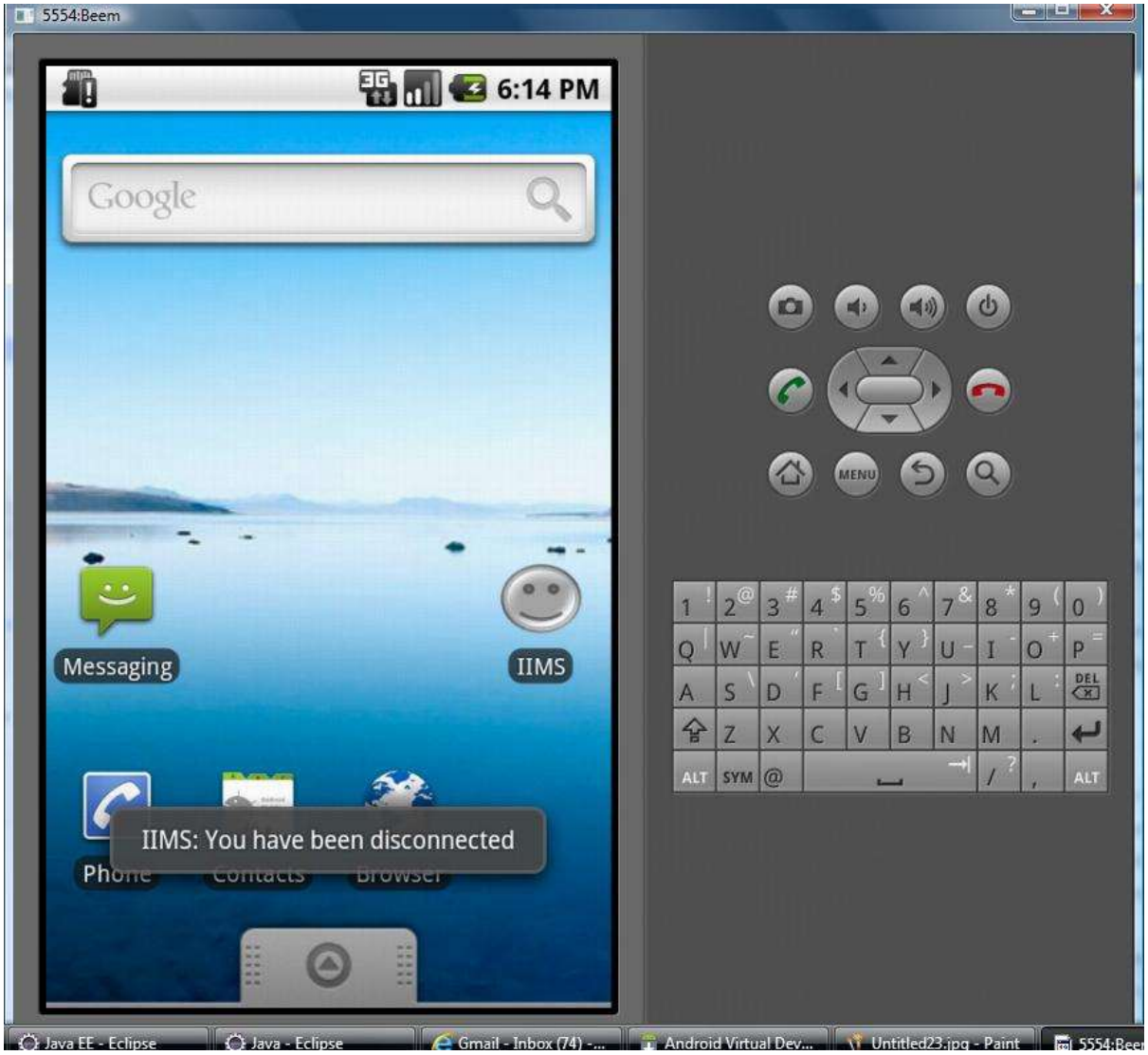

7:16 PM
IIMS - Chat
jenil
Available
pays attention to the conversation
Me
4/15/12 7:15:13 PM
Hi hiren
jenil
4/15/12 7:15:58 PM
Hello jenil
Type your message
Send
7:16 PM
IIMS - Chat
hiren
Available
PLAINTEXT
hiren
4/15/12 7:15:13 PM
Hi hiren
Me
4/15/12 7:15:58 PM
Hello jenil
Type your message
Send


7:36 PM
All contacts
jenil
apekshanaik06@gmail.com
Alias
Manage groups
Resend invit
Delete
ashishkaswa
7:37 PM
GroupList
test group1


5554:Beem
7:37 PM
IIMS - Contacts
contacts
test group1
No group
pratik gala
Available
apekshanaik06
Disconnected
hiren
Disconnected
pratik.gala
Disconnected
MENU


8:05 AM
My status
Available to chat
Available
Do not disturb
Away
Unavailable
Disconnected
8:06 AM
IIMS - Contacts
All contacts
No group
pratik.mk.gala
hiren
nim gosalia
Rishi Bage ..
$iDdH@¤tH...BuSy M@k!n...
***Ankit**@##$$% Khima...
---|V| ! t--- Bhorw@n! LeG@...


5554:Beem
8:15 AM
IIMS - Contacts
All contacts
test grou
pratik gala
Available
apekshanaik
Disconnected
apekshanaik06
Disconnected
ashishkaswa
Disconnected
chetana.barot27
Disconnected
Add a contact
Change status
Switch chat
Settings
Disconnect
MENU
Microsoft PowerPoi...
final
Android Virtual Devi...
5554:Beem
report final-blackbo...


5554:Beem
6:14 PM
Google
Messaging
IIMS
IIMS: You have been disconnected
Phone
Contacts
Browser
MENU
Java EE - Eclipse
Java - Eclipse
Gmail - Inbox (74) - ...
Android Virtual Dev...
Untitled23.jpg - Paint

CHAPTER 5

TESTING

5.1 INTRODUCTION

5.1.1 PURPOSE

The objective of this document is to determine the list of Test Cases that needs to be executed when assuring the system's quality. Each test case is described below in accordance with Test Case Specification.

5.1.2 BACKGROUND

The system to be developed is an integrated instant messaging system for Android platform. According to Software Requirement Specification the system is to support the use of Jabber, Gmail, and AOL/AIM instant messaging service in a single integrated application. This instant messenger is intended for Android software market and is positioned as a standalone product.

5.1.3 SCOPE

Test Cases document contains the list of Use Cases only without their detailed description.

5.2 TEST CASES

Test Case Id: - 1

Features to be tested: - Login Functionality

Steps: - Enter user id and password.

Input Specifications: - 1. Incorrect user id and password

2. Incorrect user id but correct password

3. Correct user id but incorrect password

4. Correct user id and password

Expected Results: - 1**.** Error

2. Error

3. Error

4. Login Successful

Actual Results: - 1. Error during authentication, bad login or password (Refer fig: 5.1)

2. Error during authentication, bad login or password (Refer fig: 5.1)

3. Error during authentication, bad login or password (Refer fig: 5.1)

4. Error during authentication, bad login or password (Refer fig: 5.2)

Comments: - Tested and Passed.

Fig 5.1:- Error during authentication

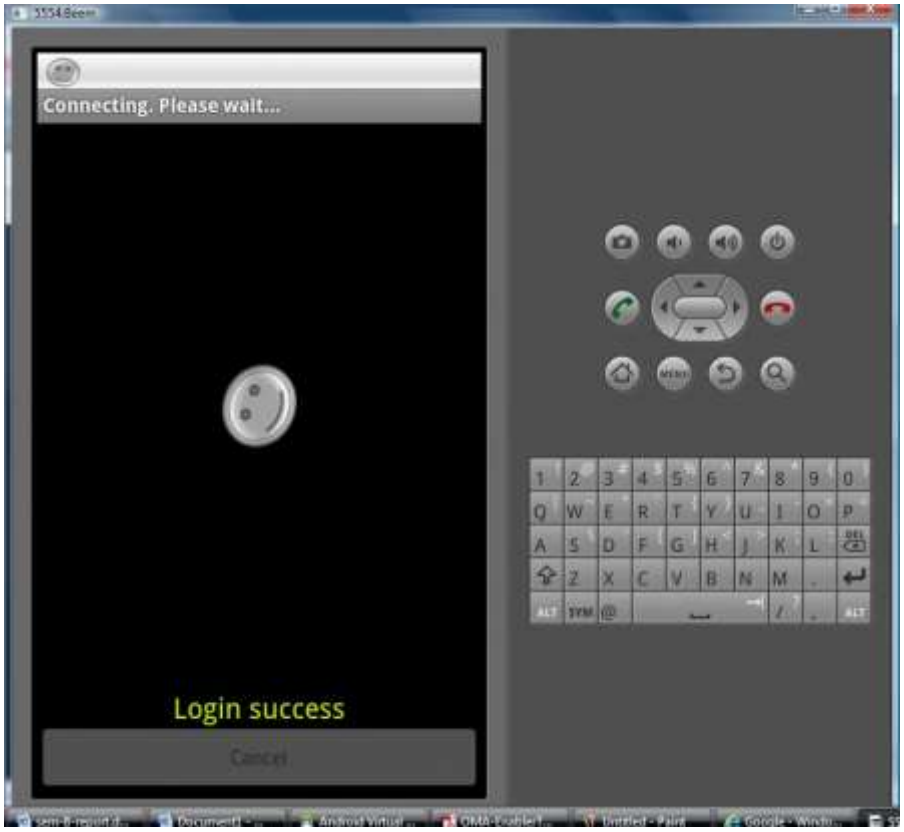


Fig 5.2:- Login Successful

Test Case Id: - 2

Features to be tested: - Add a Contact Functionality

Steps: - Enter user id, alias name(optional) and group(optional).

Input Specifications: - 1. Already existing User

2. Non-existing User

3. Deleting a User

Expected Results: - 1**.** Contact cannot be added as it is already existing.

2. Contact Successfully added

3. Contact Successfully deleted

Actual Results: - 1. Contact already exists (Refer fig: 5.3)

2. Contact added (Refer fig: 5.4)

3. Contact deleted (Refer fig: 5.5)

Comments: - Tested and Passed.

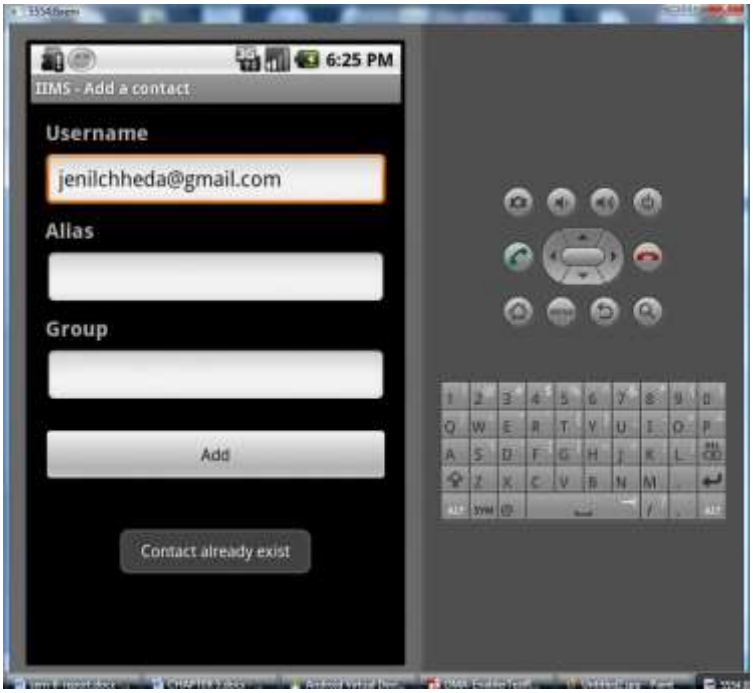


Fig 5.3 Contact Already Exist

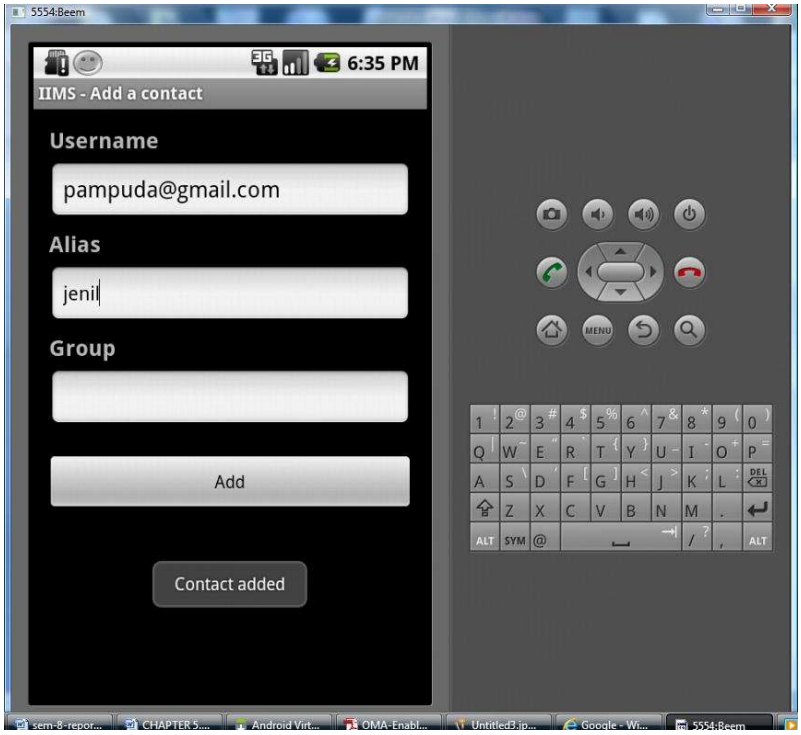


Fig 5.4:- Contact Added

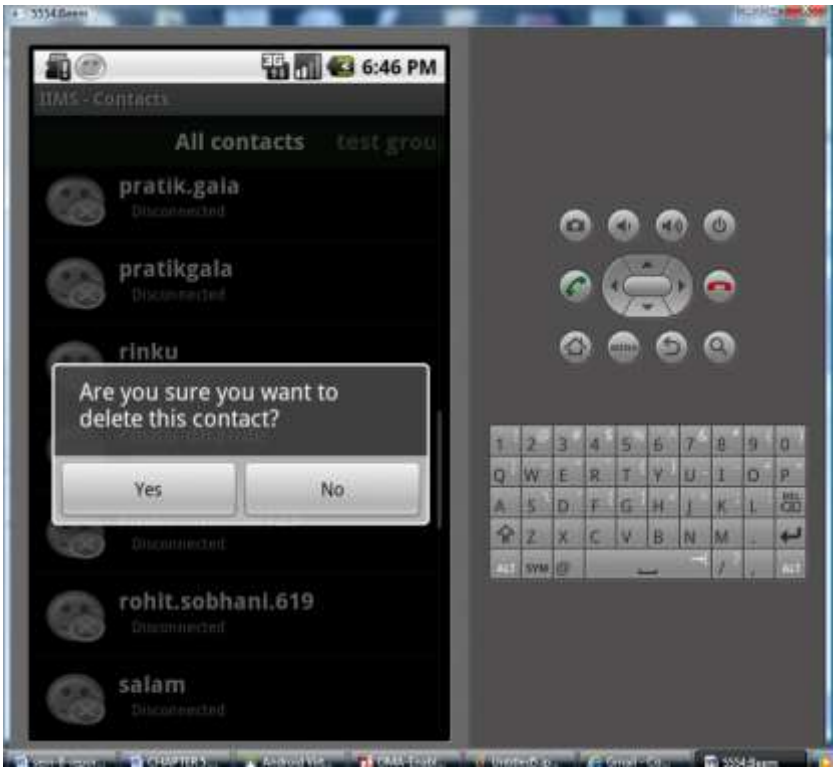


Fig 5.5:- Contact Deleted.

Test Case Id: - 3

Features to be tested: - Chat Functionality

Steps: - N/A

Input Specifications: - Select a user to chat with and enter your messages.

Expected Results: - 1. Message sent by one client is delivered to other client

2. Message sent by one client to other client that is logged out is delivered to the client when it logs in

Actual Results: - 1. Message is delivered to the client (Refer fig: 5.6)

2. Message is delivered as a notification to the client when it logs in (Refer fig: 5.7 & 5.8)

Comments: - Tested and Passed.

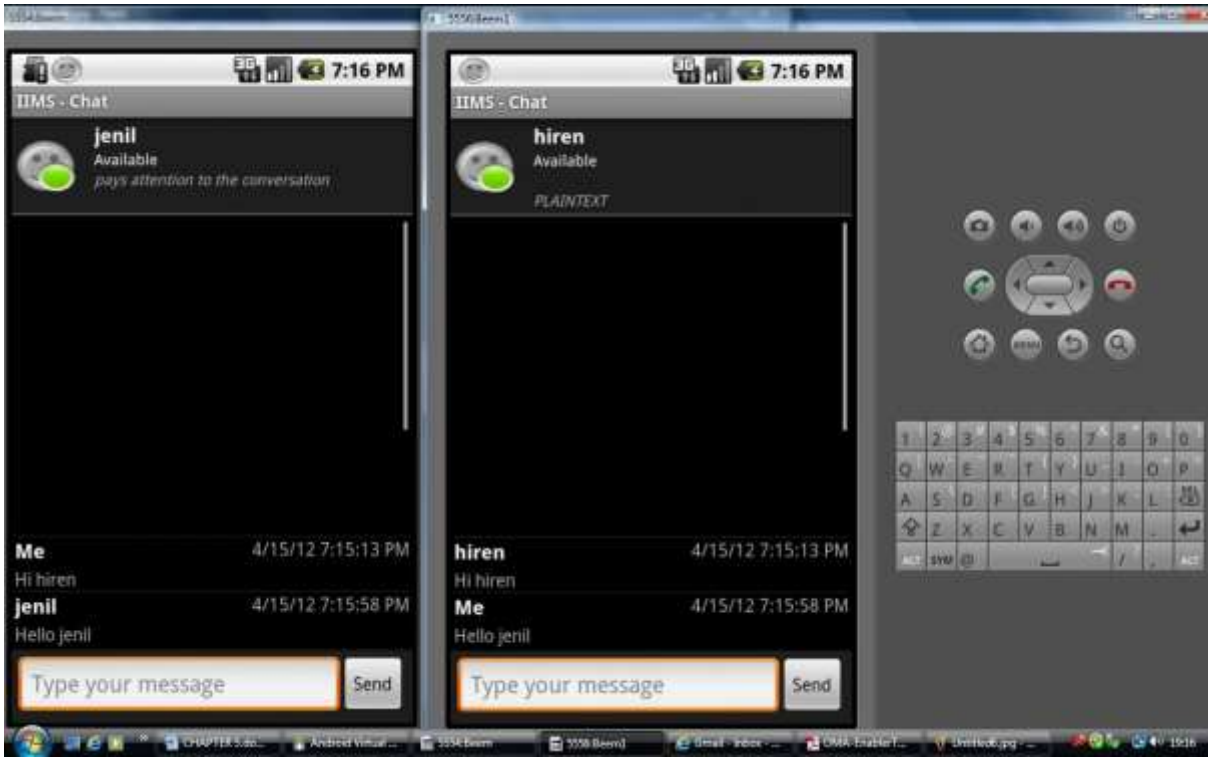


Fig 5.6:- Message delivery from one client to other client.

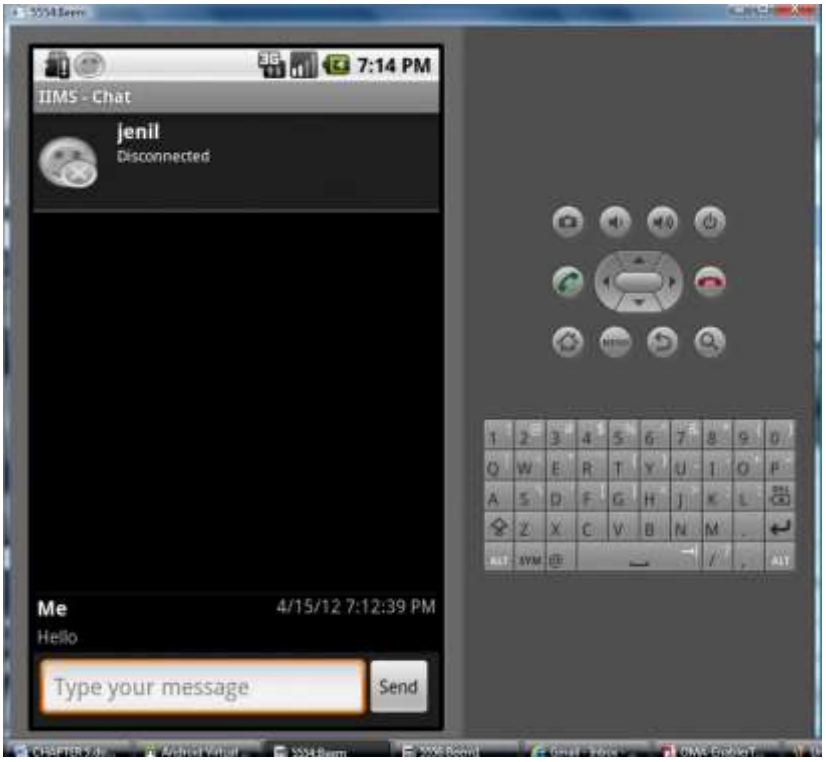


Fig 5.7:- Message sent to a logged off client

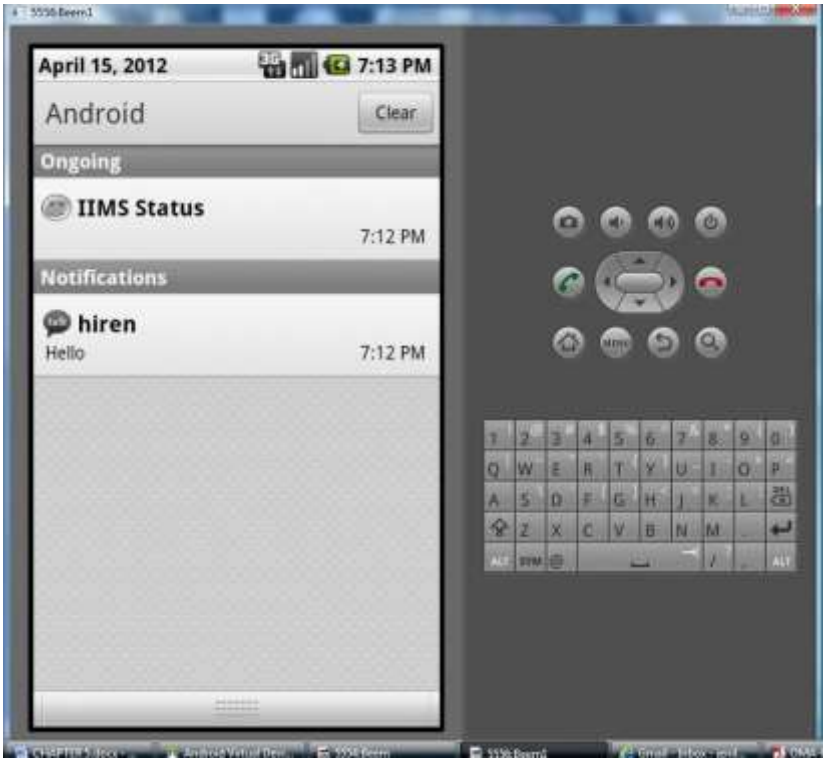


Fig 5.8:- Message received by the client as a notification.

Test Case Id: - 4

Features to be tested: - Add/Delete a user to/from a Group

Steps: - Create a group.

Input Specifications: - Select the user, click on manage user, manage group.

Add the user to the group or delete the user from an existing group.

Expected Results: - 1. User is added to the group

2. User is deleted from the group

Actual Results: - 1. User gets added (Refer fig: 5.9, 5.10 & 5.11)

2. User is deleted (Refer fig: 5.12)

Comments: - Tested and Passed.

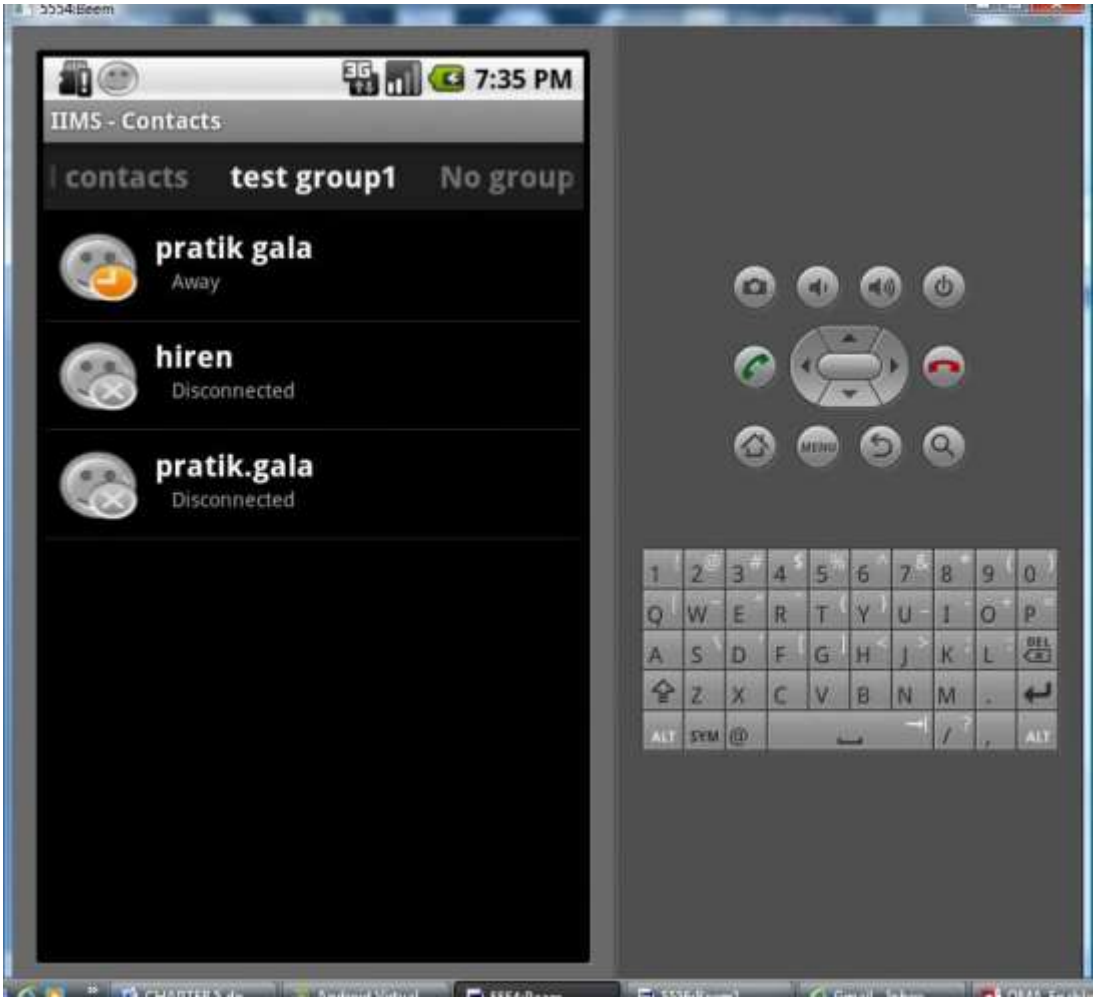


Fig 5.9:- Already created group (test group1).

Fig 5.10:- Adding a user to the group.

Fig 5.11:- User added to the group

Fig 5.12: - User deleted from the group.

CHAPTER 6

CONCLUSION AND FURTHER WORK

6.1 CONCLUSION

The project delivered allows the users of Jabber/Gmail/AOL to interoperate and communicate with each other without any barriers. The users can log on to any network and chat with other users belonging to other network. Also the system allows the users to reflect their personality, thoughts, and emotions with the use of emoticons in their messages. The system allows the user to maintain multiple accounts and buddy list.

However, due to advancements made in the technology, various new features can be added to the existing system which will enhance the functionality of the system. Features like online file sharing, email, voice over IP, conferencing, location tracker to track the location of the available user via GPS, etc can be incorporated into the project to make the system more useful and widely acceptable in corporate and non-corporate areas.