

# Instant Messenger for Integrated Messaging System



By

NC Taha Iqbal

NC Sarah Azhar

GC Muhammad Saad Munir

Submitted to the Faculty of Computer Science, Military College of Signals  
National University of Sciences and Technology, Rawalpindi in partial fulfillment for  
the requirements of a B.E Degree in Computer Software Engineering

August 2009

## **ABSTRACT**

# **Instant Messenger for Integrated Messaging System**

This product is intended to be delivered to Corp of Signals. The system currently deployed by the Signals Centre is Integrated Messaging System (IMS) that interconnects the Signals centre all over Pakistan. The current system uses electronic mailing technology to send and receive messages and files. Major drawbacks of the existing system (IMS) are that only text messages can be handled, no instant messaging, no voice chat availability and no facility for online file sharing. The new system should interconnect the Sigs. Centre all over Pakistan incorporating the functionalities of text chat, file transfer, voice over IP and offline message.

# DEDICATION

In the name of Allah, the Most Merciful, the Most Beneficent

To our parents, without whose unflinching support and unstinting cooperation, a  
work of this magnitude would not have been possible

## **ACKNOWLEDGMENTS**

We are eternally grateful to Almighty Allah for bestowing us with the strength and resolve to undertake and complete the project.

We gratefully recognize the continuous supervision and motivation provided to us by our Project Supervisor, Col Naveed Sarfraz Khattak. We are highly gratified to our co-supervisor Mr. Nabeel Younis and Mr. Bashir Bilal for his continuous and valuable suggestions, guidance, and commitment towards provision of undue support throughout our thesis work. We are also grateful to our external supervisor Brig Ashraf for his constant guidance throughout the project. We are highly thankful to all of our professors whom had been guiding and supporting us throughout our course and research work. Their knowledge, guidance and training enabled us to carry out this research work.

We would like to offer our admiration to all our classmates, and our seniors who had been supporting, helping and encouraging us throughout our thesis project. We are also indebted to the MCS system administration for their help and support.

We deeply treasure the unparalleled support and tolerance that we received from our friends for their useful suggestions that helped us in completion of this project.

## **TABLE OF CONTENTS**

**LIST OF**

**TABLES.....viii**

**LIST OF**

**FIGURES.....ix**

**1.**

**Introduction.....**

1

1.1

Preface.....1

1.2 Project

Vision.....2

1.3 Proposed

Solution.....2

1.4 Aim of the

Project.....3

1.5 Organization of Project

Report.....3

**2.**

**Literature**

**Review.....5**

2.1

Client

Server

Architecture.....5

2.1.1			
Description.....			5
2.1.2	Comparison	to	client-queue-client
architecture.....			7
2.1.3			
Advantages.....			8
2.1.4			
Disadvantages.....			9
2.2	Voice	over	Internet
Protocol.....			9
2.2.1			Circuit
Switching.....			11
2.2.2			Packet
Switching.....			13
2.2.3	Advantages	of	using
VOIP.....			14
2.2.4	Disadvantages	of	using
VOIP.....			16
2.2.5			
Codecs.....			18
2.3			G.7-11
Protocol.....			20

2.3.1		A-Law	
Algorithm.....			21

2.3.2	μ-Law	Algorithm.....	
.....			21

**3. SYSTEM**

<b>ANALYSIS.....</b>			<b>23</b>
----------------------	--	--	-----------

3.1			
Introduction.....			23

3.2		Project	
Scope.....			23

3.3		Requirement	
Specification.....			24

3.3.1	External	Interface	
Requirements.....			24

3.3.1.1		User	
Interface.....			24

3.3.1.2		Software	
Interfaces.....			25

3.3.2	Major	Functional	Requirements
.....			25

3.3.3	Major	Non-Functional	
Requirements.....			27
3.4	Use	Case	
Diagram.....			28
3.5		Sequence	
Diagram.....			30
3.6			
Conclusion.....			32
<b>4.</b>		<b>SYSTEM</b>	
<b>DESIGN.....</b>			<b>33</b>
4.1			
Introduction.....			33
4.2		Architectural	
Diagram.....			33
4.3	High	Level	
Diagram.....			35
4.4	Low	Level	
Diagram.....			36
4.4.1		Overall	
Client.....			36



4.4.2		Overall
Server.....		37
4.4.3	Voice	Chat
Module.....		38
4.4.4	File	Transfer
Client.....		39
4.4.5	File	Transfer
Server.....		40
4.5		Class
Diagram.....		41
4.6	Data	Flow
Diagram.....		42
4.7		
Conclusion.....		48
<b>5.</b>		
<b>IMPLEMENTATION.....</b>		<b>49</b>
5.1		
Introduction.....		49
5.2		Implementation
Language.....		49

5.3	Distribution of Classes with respect to	
Modules.....		50
5.3.1		IMIMS
Client.....		50
5.3.1.1		
Login.....		51
5.3.1.2		Main
Menu.....		51
5.3.1.3		Text
Chat.....		51
5.3.1.4		Voice
Chat.....		52
5.3.1.5		File
Transfer.....		52
5.3.1.6		Offline
Messaging.....		52
5.3.1.7		A-Law
Encoder.....		53
5.3.1.8		A-Law
Decoder.....		53
5.3.2		IMIMS
Server.....		53

5.3.2.1	
Form1.....	53
5.4	
Conclusion.....	54
<b>6.</b>	
<b>TESTING.....</b>	<b>5</b>
5	
6.1	
Introduction.....	55
6.2	Testing
Process.....	55
6.2.1	Client
Testing.....	56
6.2.1.1	Text
Messaging.....	56
6.2.1.2	File
Transfer.....	57
6.2.1.3	Offline
Messaging.....	58
6.2.1.4	Voice
Chat.....	59

6.2.1.5		Online
Clients.....		59
6.2.2		Server
Testing.....		60
6.2.3	Static	Analysis of
Code.....		61
6.2.3.1	Control	Flow
Analysis.....		61
6.2.3.2		Data
Analysis.....		61
6.2.3.3		Interface
Analysis.....		62
6.4		
Conclusion.....		62

**APPENDIX A (User Manual)**

**APPENDIX B (Hardware & Software Requirements)**

**APPENDIX C (References)**

## LIST OF TABLES

6-1 Test case for Text	
Messaging.....	56
6-2 Test case for File	
Transfer.....	57
6-3 Test case for Offline	
Messaging.....	58
6-4 Test case for Text	
Messaging.....	59
6-5 Test case for Text	
Messaging.....	59
6-6 Test case for	
Server.....	60

## LIST OF FIGURES

3-1	Usecase	Diagram	of
Administrator.....			28
3-2	Usecase	Diagram	of
User.....			29
3-3	Sequence	Diagram	of
Administrator.....			30
3-4	Sequence	Diagram	of
User.....			31
4-1			Architectural
Diagram.....			34
4-2		High	Level
Design.....			35
4-3		Overall	Client
Design.....			36

4-4	Overall	Server
Design.....		37
4-5	Voice Chat	Module
Design.....		38
4-6	File Transfer	Client
Design.....		39
4-7	File Transfer	Server
Design.....		40
4-8	Class Diagram	of
Client.....		41
4-9	Class Diagram	of
Server.....		41
4-10	DFD	for
Login.....		43
4-11	DFD for	Sending
File.....		44
4-12	DFD for	Receiving
File.....		45
4-13	DFD for	Sending
Voice.....		46
4-14	DFD for	Receiving
Voice.....		47

A-1		
Login.....		64
A-2		Main
Menu.....		65
A-3		Changing
Password.....		67
A-4		Sending
Message.....		68
A-5		Sending
File.....		69
A-6		Voice
Call.....		70
A-7	Sending	Offline
Messages.....		71
A-8	Receiving	Offline
Messages.....		72
A-9		
Server.....		73
A-10	Adding	User
Account.....		75
A-11		History
Table.....		76



# ***Chapter 1***

## **Introduction**

### **1.1 Preface**

IMS (Integrated Messaging System) is fundamentally a computer network with secure E-mail facility. Signals Research & Development Estb has devised this system to replace the AMSS (Automatic Messaging Switching System). The system was installed at 9 Signal Centers (Sigcens) as pilot project in May 2005 after rigorous test and trials. This system is installed in 35 Sigcens (Total 62 Sigcens) all over Pakistan. ADN (Army Data Network) is being used as the backbone media network. The modes used by the IMS workstations are Leased Line (LL), Dialup PC to Mail Server (through ADN), Dialup PC to PC (through PASCOM/DEFCON) and Off-line.

Every Signal center is envisioned to house a LAN. This LAN will comprise of IMS Workstations, IMS workstations and computers of supervisory staff to enter data in "Local Query Server (LQS) regarding different messages and perform routine management tasks. These LANs all over Pakistan is

connected to a centrally located IMS Central Mail Server (CMS). This LAN is connected with the ADN through LL connection using leased line modems or directly on Ethernet depending on the location of the nearest ADN switch. It facilitates exchange of messages among signal centers throughout Pakistan. IMS/IMS workstations can also be connected through dialup facility using ADN and one to one without involving ADN. IMS keep record/track of all the signals/messages being exchanged between the signal centers. For this purpose a LQS will be implemented. The system can also be connected on HF//VHF radio if it has data communication capability. Major drawbacks of the existing system (IMS) are only text messages can be handled, no instant messaging, no voice chat availabilitys, no facility for online file sharing and no video conferencing facility.

## **1.2 Project Vision**

The present IMS clearly lacks in the above incorporating facilities. Therefore the basic idea behind the project is to overcome the above shortcomings possessed by the system.

## **1.3 Proposed Solution**

The solution proposed for the following above shortcomings possessed by the system is to inter-connect the Sigcens all over Pakistan through Voice over IP, to display the online users, to provide text chatting facility, to deliver offline messages according to the list mentioned by the

user, to make the users capable of voice chatting and to offer file sharing facility to the users.

## **1.4 Aim of the Project**

The aim of the project is to incorporate the missing functionalities in existing Integrated Messaging System (IMS), required by the users.

This could be achieved by developing an Instant Messenger for Integrated Messaging System. This messenger will include a Text Chatting module which will facilitate online sending of messages. It will also indicate the user about all the online clients. File sharing module will facilitate in sending and receiving files from any online client. The file sharing will be done over File Transfer Protocol. Voice chatting module will facilitate the users with voice chatting to communicate important messages through voice over IP. Offline messaging module will facilitate the users to send messages to offline users which they will received at log-in.

## **1.5 Organization of Project Report**

The project report has been drafted carefully deciding the sequence to be followed. After the introduction section, the report incorporates the Literature Review chapter summarizing the text studied before and during

the project's execution. Subsequently, the System Analysis chapter comes which includes the major interface, functional and non-functional requirements of the system. Next is the System Design chapter comprising of the architectural diagram, high and low level design, data flow diagram, class diagram and interaction diagram. Following this the report includes the implementation chapter identifying and elucidating the classes which are implemented. Then is the testing chapter incorporating the testing process employed to test the system and the results that were obtained. The next chapter then discusses the work that can be done in future to further enhance the system and ultimately this chapter wraps the report.

## ***Chapter 2***

# **Literature Review**

## **2.1 Client Server Architecture**

Client-server computing or networking is a [distributed application](#) architecture that partitions tasks or workloads between service providers ([servers](#)) and service requesters, called [clients](#). Often clients and servers operate over a [computer network](#) on separate hardware. A server machine is a high-performance host that is running one or more server programs which share its resources with clients. A client does not share any of its resources, but requests a server's content or service function. Clients therefore initiate communication sessions with servers which await (listen to) incoming requests.

### **2.1.1 Description**

Client-server describes the relationship between two computer programs in which one program, the client program, makes a service request to another, the server program. Standard networked functions such as email exchange, web access and database access, are based on the client-server model. For example, a [web browser](#) is a client program at the user computer that may access information at any web server in the world. To check bank account from computer, a web browser client program in computer forwards request to a web server program at the bank. That program may in turn forward the request to its own database client program that sends a request to a database server at another bank computer to retrieve account balance. The balance is returned to the bank database client, which in turn serves it back to the web browser client in personal computer, which displays the information for you.

The client-server model has become one of the central ideas of [network computing](#). Many business applications being written today use the client-server model. So do the Internet's main application protocols, such as [HTTP](#), [SMTP](#), [Telnet](#), [DNS](#). In marketing, the term has been used to distinguish distributed computing by smaller dispersed computers from the "monolithic" centralized computing of mainframe computers. But this distinction has largely disappeared as [mainframes](#) and their applications have also turned to the client-server model and become part of network computing.

Each [instance](#) of the client software can send data [requests](#) to one or more connected servers. In turn, the servers can accept these requests,

process them, and return the requested information to the client. Although this concept can be applied for a variety of reasons to many different kinds of [applications](#), the architecture remains fundamentally the same.

The most basic type of client-server [architecture](#) employs only two types of hosts: clients and servers. This type of architecture is sometimes referred to as two-tier. It allows devices to share files and resources. The two tier architecture means that the client acts as one tier and application in combination with server acts as another tier.

The interaction between client and server is often described using [sequence diagrams](#). Sequence diagrams are standardized in the [Unified Modeling Language](#). Specific types of clients include [web browsers](#), [email clients](#), and [online chat](#) clients. Specific types of servers include [web servers](#), [ftp servers](#), [application servers](#), [database servers](#), [name servers](#), [mail servers](#), [file servers](#), [print servers](#), and [terminal servers](#). Most [web services](#) are also types of servers.

## **2.1.2 Comparison to Client-Queue-Client**

### **Architecture**

While classic client-server architecture requires one of the communication endpoints to act as a server, which is much harder to implement, [Client-Queue-Client](#) allows all endpoints to be simple clients, while the server consists of some external software, which also acts as

passive queue (one software instance passes its query to another instance to queue, e.g. database, and then this other instance pulls it from database, makes a response, passes it to database etc.). This architecture allows greatly simplified software implementation. Peer-to-peer architecture was originally based on the Client-Queue-Client concept.

### **2.1.3 Advantages**

In most cases, client-server architecture enables the roles and responsibilities of a computing system to be distributed among several independent computers that are known to each other only through a network. This creates an additional advantage to this architecture: greater ease of maintenance. For example, it is possible to replace, repair, upgrade, or even relocate a server while its clients remain both unaware and unaffected by that change.

All the data is stored on the servers, which generally have far greater security controls than most clients. Servers can better control access and resources, to guarantee that only those clients with the appropriate permissions may access and change data.

Since data storage is centralized, updates to that data are far easier to administer than what would be possible under a P2P paradigm. Under a P2P



architecture, data updates may need to be distributed and applied to each peer in the network, which is both time-consuming and error-prone, as there can be thousands or even millions of peers.

Many mature client-server technologies are already available which were designed to ensure security, friendliness of the user interface, and ease of use. It functions with multiple different clients of different capabilities.

### **2.1.4 Disadvantages**

Traffic congestion on the network has been an issue since the inception of the client-server paradigm. As the number of simultaneous client requests to a given server increases, the server can become overloaded. Contrast that to a P2P network, where its aggregated bandwidth actually increases as nodes are added, since the P2P network's overall bandwidth can be roughly computed as the sum of the bandwidths of every node in that network.

The client-server paradigm lacks the robustness of a good P2P network. Under client-server, should a critical server fail, clients' requests cannot be fulfilled. In P2P networks, resources are usually distributed among many nodes. Even if one or more nodes depart and abandon a downloading file, for example, the remaining nodes should still have the data needed to complete the download.

## 2.2 Voice over Internet Protocol

Voice over Internet Protocol (VoIP) is a general term for a family of transmission technologies for delivery of [voice communications](#) over [IP](#) networks such as the [Internet](#) or other [packet-switched networks](#). Other terms frequently encountered and synonymous with VoIP are IP telephony, Internet telephony, voice over broadband (VoBB), [broadband](#) telephony, and broadband phone.

Internet telephony refers to communications services — voice, facsimile, and/or voice-messaging applications — that are transported via the Internet, rather than the [public switched telephone network](#) (PSTN). The basic steps involved in originating an Internet telephone call are conversion of the analog voice signal to digital format and compression/translation of the signal into [Internet protocol](#) (IP) packets for transmission over the Internet; the process is reversed at the receiving end.

VoIP systems employ session control protocols to control the set-up and tear-down of calls as well as [audio codecs](#) which encode speech allowing transmission over an IP network as [digital audio](#) via an [audio stream](#). Codec use is varied between different implementations of VoIP (and often a range of codecs are used); some implementations rely on [narrowband](#) and [compressed speech](#), while others support [high fidelity stereo](#) codecs.

Most VoIP companies provide the features that normal phone companies charge extra for when they are added to service plan. VoIP

includes [caller ID](#), call waiting, call transfer, repeat dial, return call and three-way calling.

There are also advanced call-filtering options available from some carriers. These features use caller ID information to allow you make a choice about how calls from a particular number are handled. You can forward the call to a particular number, send the call directly to voice mail, give the caller a busy signal, play a "not-in-service" message or send the caller to a funny rejection hotline.

With many VoIP services, you can also check voice mail via the Web or attach messages to an e-mail that is sent to computer or handheld. Not all VoIP services offer all of the features above. Prices and services vary so if you're interested, it's best to do a little shopping.

Now that we've looked at VoIP in a general sense, let's look more closely at the components that make the system work. To understand how VoIP really works and why it's an improvement over the traditional phone system, it helps to first understand how a traditional phone system works.

### **2.2.1 Circuit Switching**

Existing phone systems are driven by a very reliable but somewhat inefficient method for connecting calls called circuit switching. Circuit switching is a very basic concept that has been used by [telephone networks](#) for more than 100 years. When a call is made between two parties, the connection is maintained for the duration of the call. Because you're

connecting two points in both directions, the connection is called a circuit. This is the foundation of the Public Switched Telephone Network (PSTN).

Here's how a typical telephone call works. You pick up the receiver and listen for a dial tone. This lets you know that you have a connection to the local office of telephone carrier. You dial the number of the party you wish to talk to. The call is routed through the switch at local carrier to the party you are calling. A connection is made between telephone and the other party's line using several interconnected switches along the way. The phone at the other end rings, and someone answers the call. The connection opens the circuit. You talk for a period of time and then hang up the receiver. When you hang up, the circuit is closed, freeing line and all the lines in between.

Let's say you talk for 10 minutes. During this time, the circuit is continuously open between the two phones. In the early phone system, up until 1960 or so, every call had to have a dedicated wire stretching from one end of the call to the other for the duration of the call. So if you were in New York and you wanted to call Los Angeles, the switches between New York and Los Angeles would connect pieces of copper wire all the way across the United States. You would use all those pieces of wire just for call for the full 10 minutes. You paid a lot for the call, because you actually owned a 3,000-mile-long copper wire for 10 minutes.

Telephone conversations over today's traditional phone network are somewhat more efficient and they cost a lot less. Voice is digitized, and voice along with thousands of others can be combined onto a single [fiber optic](#)

cable for much of the journey (there's still a dedicated piece of copper wire going into house, though). These calls are transmitted at a fixed rate of 64 kilobits per second (Kbps) in each direction, for a total transmission rate of 128 Kbps. Since there are 8 kilobits (Kb) in a kilobyte (KB), this translates to a transmission of 16 KB each second the circuit is open, and 960 KB every minute it's open. In a 10-minute conversation, the total transmission is 9,600 KB, which is roughly equal to 10 megabytes (check out [How Bits and Bytes Work](#) to learn about these conversions). If you look at a typical phone conversation, much of this transmitted data is wasted.

## 2.2.2 Packet Switching

A packet-switched phone network is the alternative to circuit switching. It works like this: While you're talking, the other party is listening, which means that only half of the connection is in use at any given time. Based on that, we can surmise that we could cut the file in half, down to about 4.7 MB, for efficiency. Plus, a significant amount of the time in most conversations is dead air -- for seconds at a time, neither party is talking. If we could remove these silent intervals, the file would be even smaller. Then, instead of sending a continuous stream of bytes (both silent and noisy), what if we sent just the packets of noisy bytes when you created them?

Data networks do not use circuit switching. Internet connection would be a lot slower if it maintained a constant connection to the [Web page](#) you were viewing at any given time. Instead, data networks simply send and

retrieve data as you need it. And, instead of routing the data over a dedicated line, the data packets flow through a chaotic network along thousands of possible paths. This is called packet switching.

While circuit switching keeps the connection open and constant, packet switching opens a brief connection -- just long enough to send a small chunk of data, called a [packet](#), from one system to another. It works like this that the sending computer chops data into small packets, with an address on each one telling the network devices where to send them. Inside of each packet is a payload. The payload is a piece of the e-mail, a music file or whatever type of file is being transmitted inside the packet. The sending computer sends the packet to a nearby router and forgets about it. The nearby router sends the packet to another router that is closer to the recipient computer. That router sends the packet along to another, even closer router, and so on. When the receiving computer finally gets the packets (which may have all taken completely different paths to get there), it uses instructions contained within the packets to reassemble the data into its original state.

Packet switching is very efficient. It lets the network route the packets along the least congested and cheapest lines. It also frees up the two computers communicating with each other so that they can accept information from other computers, as well.

### **2.2.3 Advantages of Using VoIP**

VoIP technology uses the Internet's packet-switching capabilities to provide phone service. VoIP has several advantages over circuit switching. For example, packet switching allows several telephone calls to occupy the amount of space occupied by only one in a circuit-switched network. Using PSTN, that 10-minute phone call we talked about earlier consumed 10 full minutes of transmission time at a cost of 128 Kbps. With VoIP, that same call may have occupied only 3.5 minutes of transmission time at a cost of 64 Kbps, leaving another 64 Kbps free for that 3.5 minutes, plus an additional 128 Kbps for the remaining 6.5 minutes. Based on this simple estimate, another three or four calls could easily fit into the space used by a single call under the conventional system. And this example doesn't even factor in the use of [data compression](#), which further reduces the size of each call.

Let's say that you and friend both have service through a VoIP provider. You both have analog phones hooked up to the service-provided ATAs. Let's take another look at that typical telephone call, but this time using VoIP over a packet-switched network. You pick up the receiver, which sends a signal to the ATA. The ATA receives the signal and sends a dial tone. This lets you know that you have a connection to the Internet. You dial the phone number of the party you wish to talk to. The tones are converted by the ATA into digital data and temporarily stored. The phone number data is sent in the form of a request to VoIP company's call processor. The call processor checks it to ensure that it's in a valid format. The call processor determines to whom to map the phone number. In mapping, the phone

number is translated to an [IP address](#) (more on this later). The soft switch connects the two devices on either end of the call. On the other end, a signal is sent to friend's ATA, telling it to ask the connected phone to ring. Once friend picks up the phone, a session is established between computer and friend's computer. This means that each system knows to expect packets of data from the other system. In the middle, the normal [Internet infrastructure](#) handles the call as if it were [e-mail](#) or a Web page. Each system must use the same protocol to communicate. The systems implement two channels, one for each direction, as part of the session. You talk for a period of time. During the conversation, system and friend's system transmit packets back and forth when there is data to be sent. The ATAs at each end translate these packets as they are received and convert them to the analog audio signal that you hear. ATA also keeps the circuit open between itself and analog phone while it forwards packets to and from the IP host at the other end. You finish talking and hang up the receiver. When you hang up, the circuit is closed between phone and the ATA. The ATA sends a signal to the soft switch connecting the call, terminating the session.

Probably one of the most compelling advantages of packet switching is that data networks already understand the technology. By migrating to this technology, telephone networks immediately gain the ability to communicate the way computers do.

## **2.2.4 Disadvantages of Using VoIP**



The current Public Switched Telephone Network is a robust and fairly bulletproof system for delivering phone calls. Phones just work, and we've all come to depend on that. On the other hand, computers, e-mail and other related devices are still kind of flaky. Let's face it -- few people really panic when their e-mail goes down for 30 minutes. It's expected from time to time. On the other hand, a half hour of no dial tone can easily send people into a panic. So what the PSTN may lack in efficiency it more than makes up for in reliability. But the network that makes up the Internet is far more complex and therefore functions within a far greater margin of error. What this all adds up to be one of the major flaws in VoIP: reliability. First of all, VoIP is dependent on wall power. Current phone runs on phantom power that is provided over the line from the central office. Even if power goes out, phone (unless it is a [cordless](#)) still works. With VoIP, no power means no phone. A stable power source must be created for VoIP. Another consideration is that many other systems in home may be integrated into the phone line. [Digital video recorders](#), digital subscription TV services and [home security systems](#) all use a standard phone line to do their thing. There's currently no way to integrate these products with VoIP. The related industries are going to have to get together to make this work. Emergency 911 calls also become a challenge with VoIP. As stated before, VoIP uses IP-addressed phone numbers, not NANP phone numbers. There's no way to associate a geographic location with an IP address. So if the caller can't tell the 911 operator where he is located, then there's no way to know which call center to route the emergency call to and which EMS should respond. To fix this,

perhaps geographical information could somehow be integrated into the packets. Because VoIP uses an Internet connection, it's susceptible to all the hiccups normally associated with home broadband services. All of these factors affect call quality e.g. Latency, Jitter, Packet loss. Phone conversations can become distorted, garbled or lost because of transmission errors. Some kind of stability in Internet data transfer needs to be guaranteed before VoIP could truly replace traditional phones. VoIP is susceptible to worms, [viruses](#) and hacking, although this is very rare and VoIP developers are working on VoIP encryption to counter this. Another issue associated with VoIP is having a phone system dependant on individual [PCs](#) of varying specifications and power. A call can be affected by processor drain. Let's say you are chatting away on softphone, and you decide to open a program that saps processor. Quality loss will become immediately evident. In a worst case scenario, system could crash in the middle of an important call. In VoIP, all phone calls are subject to the limitations of normal computer issues.

One of the hurdles that was overcome some time ago was the conversion of the analog audio signal phone receives into packets of data. How it is that analog audio is turned into packets for VoIP transmission? The answer is codecs.

## **2.2.5 Codecs**

A codec, which stands for coder-decoder, converts an audio signal into compressed digital form for transmission and then back into an uncompressed audio signal for replay. It's the essence of VoIP.

Codecs accomplish the conversion by sampling the audio signal several thousand times per second. For instance, a [G.711 codec](#) samples the audio at 64,000 times a second. It converts each tiny sample into digitized data and compresses it for transmission. When the 64,000 samples are reassembled, the pieces of audio missing between each sample are so small that to the human ear, it sounds like one continuous second of audio signal. There are different sampling rates in VoIP depending on the codec being used e.g. 64,000 times per second, 32,000 times per second, 8,000 times per second. A [G.729A codec](#) has a sampling rate of 8,000 times per second and is the most commonly used codec in VoIP.

Codecs use advanced algorithms to help sample, sort, compress and packetize audio data. The CS-ACELP algorithm (CS-ACELP = conjugate-structure algebraic-code-excited linear prediction) is one of the most prevalent algorithms in VoIP. CS-ACELP organizes and streamlines the available bandwidth. Annex B is an aspect of CS-ACELP that creates the transmission rule, which basically states "if no one is talking, don't send any data." The efficiency created by this rule is one of the greatest ways in which packet switching is superior to circuit switching. It's Annex B in the CS-ACELP algorithm that's responsible for that aspect of the VoIP call.

The codec works with the algorithm to convert and sort everything out, but it's not any good without knowing where to send the data. In VoIP, that task is handled by soft switches.

E.164 is the name given to the standard for the [North American Numbering Plan](#) (NANP). This is the numbering system that phone networks use to know where to route a call based on the dialed numbers. A phone number is like an address e.g. (313) 555-1212. 313 = State, 555=City and 1212 = Street address

The switches use "313" to route the phone call to the area code's region. The "555" prefix sends the call to a central office, and the network routes the call using the last four digits, which are associated with a specific location. Based on that system, no matter where you're in the world, the number combination "(313) 555" always puts you in the same central office, which has a switch that knows which phone is associated with "1212."

The challenge with VoIP is that IP-based networks don't read phone numbers based on NANP. They look for IP addresses, which look like 192.158.10.7.

[IP addresses](#) correspond to a particular device on the network like a computer, a router, a switch, a gateway or a telephone. However, IP addresses are not always static. They're assigned by a DHCP server on the network and change with each new connection. VoIP's challenge is translating NANP phone numbers to IP addresses and then finding out the

current IP address of the requested number. This mapping process is handled by a central call processor running a soft switch.

The central call processor is hardware that runs a specialized database/mapping program called a soft switch. Think of the user and the phone or computer as one package -- man and machine. That package is called the endpoint. The soft switch connects endpoints.

Soft switches know where the network's endpoint is, what phone number is associated with that endpoint and the endpoint's current IP address.

## **2.3 G.7-11 Protocol**

G.711 is the international standard for encoding telephone audio on an 64 kbps channel. It is a pulse code modulation (PCM) scheme operating at a 8 kHz sample rate, with 8 bits per sample. According to the Nyquist theorem, which states that a signal must be sampled at twice its highest frequency component, G.711 can encode frequencies between 0 and 4 kHz. Telcos can select between two different variants of G.711: A-law and mu-law. A-law is the standard for international circuits.

Each of these encoding schemes is designed in a roughly logarithmic fashion. Lower signal values are encoded using more bits; higher signal values require fewer bits. This ensures that low amplitude signals will be well represented, while maintaining enough range to encode high amplitudes.

The actual encoding doesn't use logarithmic functions, however. The input range is broken into segments, each segment using a different interval between decision values. Most segments contain 16 intervals, and the interval size doubles from segment to segment.

### 2.3.1 A-Law Algorithm

An A-law algorithm is a standard [companding](#) algorithm, used in [European digital communications](#) systems to optimize, i.e., modify, the [dynamic range](#) of an [analog signal](#) for digitizing. A-law encoding effectively reduces the dynamic range of the signal, thereby increasing the [coding](#) efficiency and resulting in a signal-to-[distortion](#) ratio that is superior to that obtained by linear encoding for a given number of bits.

### 2.3.2 $\mu$ -Law Algorithm

The  $\mu$ -law algorithm is a [companding](#) algorithm, primarily used in the [digital telecommunication](#) systems of [North America](#) and [Japan](#). Companding algorithms reduce the [dynamic range](#) of an audio [signal](#). In analog systems, this can increase the [signal-to-noise ratio](#) (SNR) achieved during transmission, and in the digital domain, it can reduce the quantization error (hence increasing signal to quantization noise ratio). These SNR increases can be traded instead for reduced bandwidth for equivalent SNR.

The  [\$\mu\$ -law algorithm](#) provides a slightly larger dynamic range than the A-law at the cost of worse proportional distortion for small signals. By

convention, A-law is used for an international connection if at least one country uses it.

## ***Chapter 3***

# **System Analysis**

## **3.1 Introduction**

This chapter covers the system analysis phase of the project. In this phase, first of all scope of the project is presented as it's clear definition and understanding is needed for the absolute comprehension of the system's requirements' specification phase, including major functional and non-functional requirements, is described. The requirements' specification phase is then followed by usecase diagram and domain model of the system for the better understanding of the system analysis phase of the project.

## **3.2 Project Scope**

This product is intended to be delivered to Signals Corp. The Integrated Messaging System (IMS) currently deployed by the Siggens, interconnect the Siggens all over Pakistan. The current system uses electronic mailing technology to send and receive messages and files. The new system should interconnect the Siggens all over Pakistan incorporating the functionalities of text chat, file transfer, offline messaging and voice over IP.

## **3.3 Requirements Specification**

The requirements specifications of the system include its external interface requirements including user and software interface requirements. It also involves the analysis of the major functional requirements including the main functionalities that the system is expected to deliver. Analysis of main



non-functional requirements is also important as it tends to give an idea about the characteristics of the system such as accuracy, scalability etc.

### **3.3.1 External Interface Requirements**

Requirements which include the interaction of the system with the external requirement e.g. user interface through which the user interacts with the system and software interfaces means the software tools which are employed in the system.

#### **3.3.1.1 User Interfaces**

The GUI of this application will be made in a way that the user will be presented with a screen to login at first. After the authentication of the user, the user will be presented with a screen showing the list of online users. When a user wishes to chat with a certain user he will need to click on the user's name to know the list of available options. Available options will be including written chat & voice chat. After choosing a certain operation, a separate window will be opened for the conversation whether it is text or voice.

SIP .Net will be used for the implementation of the voice conversation. SIP is the most suited protocol for the implementation of VOIP under the given circumstances. This API provides the following features for establishing a voice conversation e.g. authentication, supports the SIP

methods e.g. REGISTER, INVITE, ACK etc, creating and parsing SDP messages, show online users, instant Messaging.

### **3.3.1.2 Software Interfaces**

The project employs Windows XP and Vista as the operating system and software tool such as SQL 2005 and DirectX is required to run the system.

### **3.3.2 Major Functional Requirements**

Functional requirements means the core functionalities that the system is meant to deliver e.g. the Users of IMIMS should be able to see other users of IMIMS that are online simultaneously, the Sigsens all over Pakistan should be capable of interacting with each other through instant text messages, the Users of IMIMS should be able to send offline messages according to the priority list mentioned by the same user, the Sigs. Centre should be able to communicate with each other via voice chatting.

The Users of IMIMS should be able to see other users of IMIMS that are online simultaneously. A user will be able to see the other online users working at different workstation situated at several locations (Sigsens). It is a necessity to display the users the information of other users online so that they can know that with what users they can communicate at that time. No dependencies with other requirements.

The Siccens all over Pakistan should be capable of interacting with each other through instant text messages. The users of the IMIMS (people sitting at workstations at different Siccens) will be made capable of interacting with each other via text messages which can be sent and received in real-time. The existing IMS system does not facilitate the users with the functionality of having an interaction via text messages instantaneously. The users wanted to have a way of interacting in real-time so that they can know if the specific mail or file has been received at the other end and get an acknowledgement. The users will be able to send instantaneous text messages depending on whether the target user is logged on to the system or not. The user should be able to send instantaneous text messages only if the other user is logged on.

The Users of IMIMS should be able to send offline messages according to the list mentioned by the same user. A user will be able to send offline messages to another user who is not online at that time, according to a priority list mentioned by the sending user, so that if the required user doesn't come online, then the message is delivered to the user next in the priority list. A message which needs to be responded in certain period of time can be catered through this functionality. No dependencies with other requirements.

The Siccens should be able to communicate with each other via voice chatting. The users of the IMIMS will be able of having a voice conversation using voice over IP. Any computer equipped with a sound card installed and

having a microphone should be able to have a voice chat with any other user of the IMIMS with same specs of his system. The user will only be able to have a voice conversation with the other user of the system if they are appearing online to each other. The most important technical issues involved with this requirement are the voice quality of calls which depends on network congestion.

### **3.3.3 Major Non-Functional Requirements**

The major non-functional requirements that the system provides are security, reliability and compatibility.

The system should be secure in a sense that the information should be received by the intended user only. The system should be reliable in a sense that the system should provide the users with the required functionality round the clock. The system will be made compatible with all systems with Microsoft Windows installed.

## **3.4 Usecase Diagram**

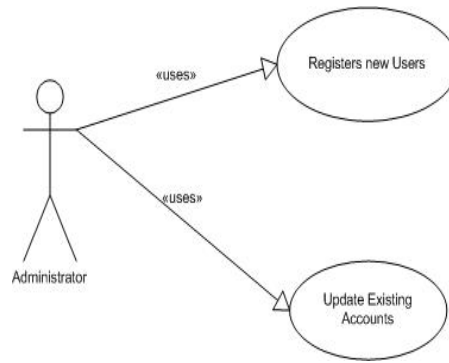


Figure 3-1: Usecase Diagram of Administrator

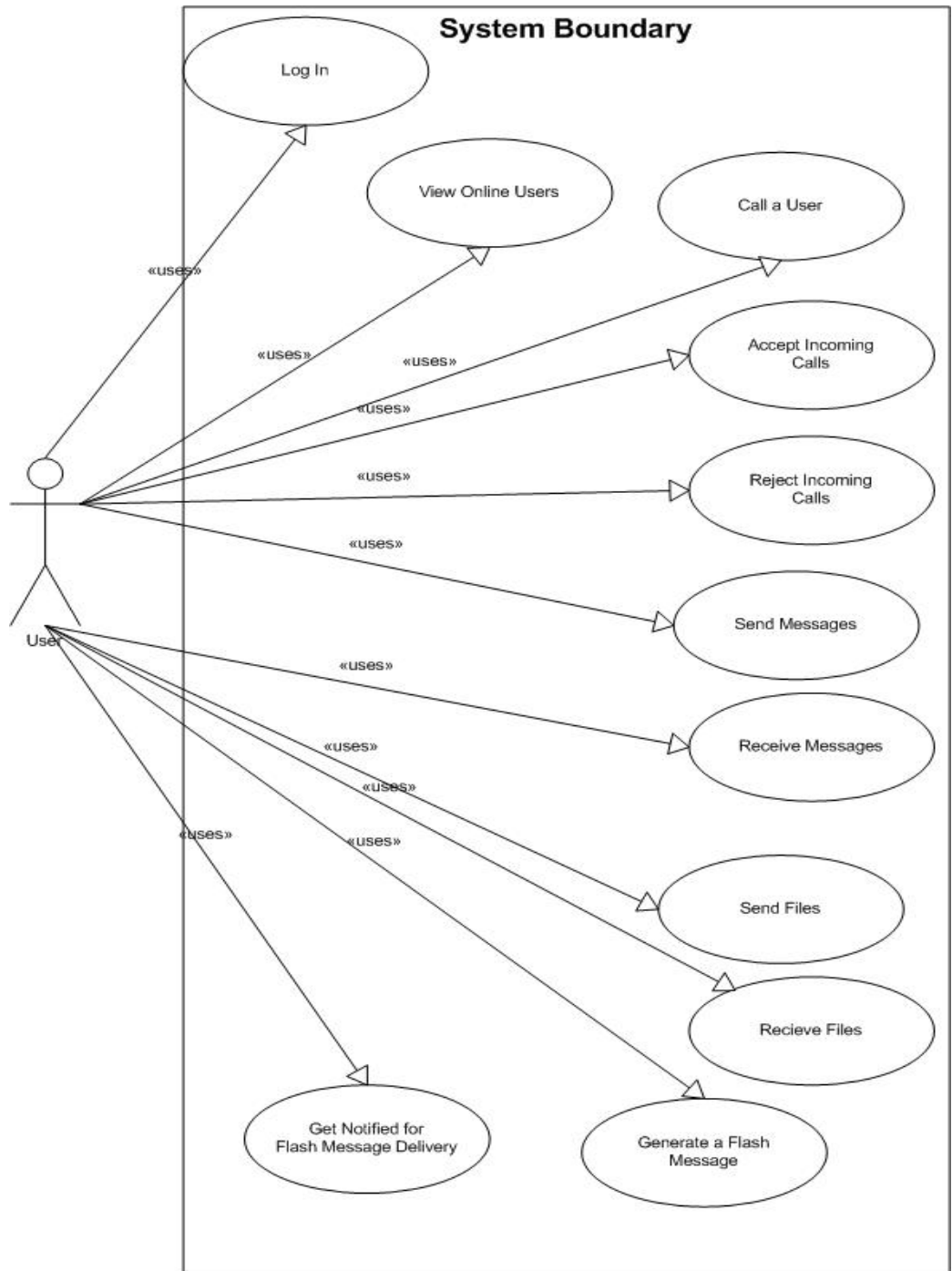


Figure 3-2: Usecase Diagram of User

The use-case diagram created above shows the possible use-cases for the administrator. The administrator can register a new user if anyone wants

to register with this system. The administrator will also be able to update the existing accounts. This use-case diagram shows all the possible use-cases for any user of the system.

### 3.5 Sequence Diagram

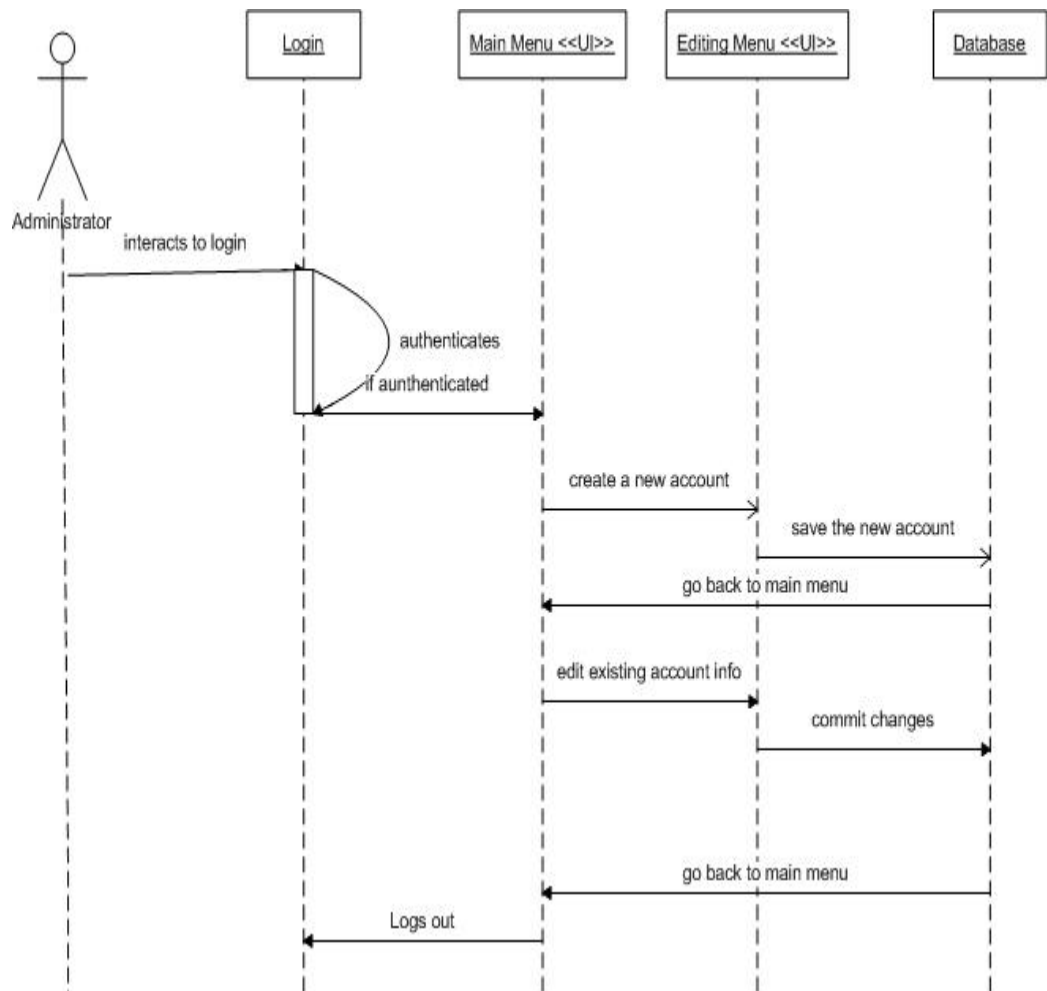


Figure 3-3: Sequence Diagram of Administrator

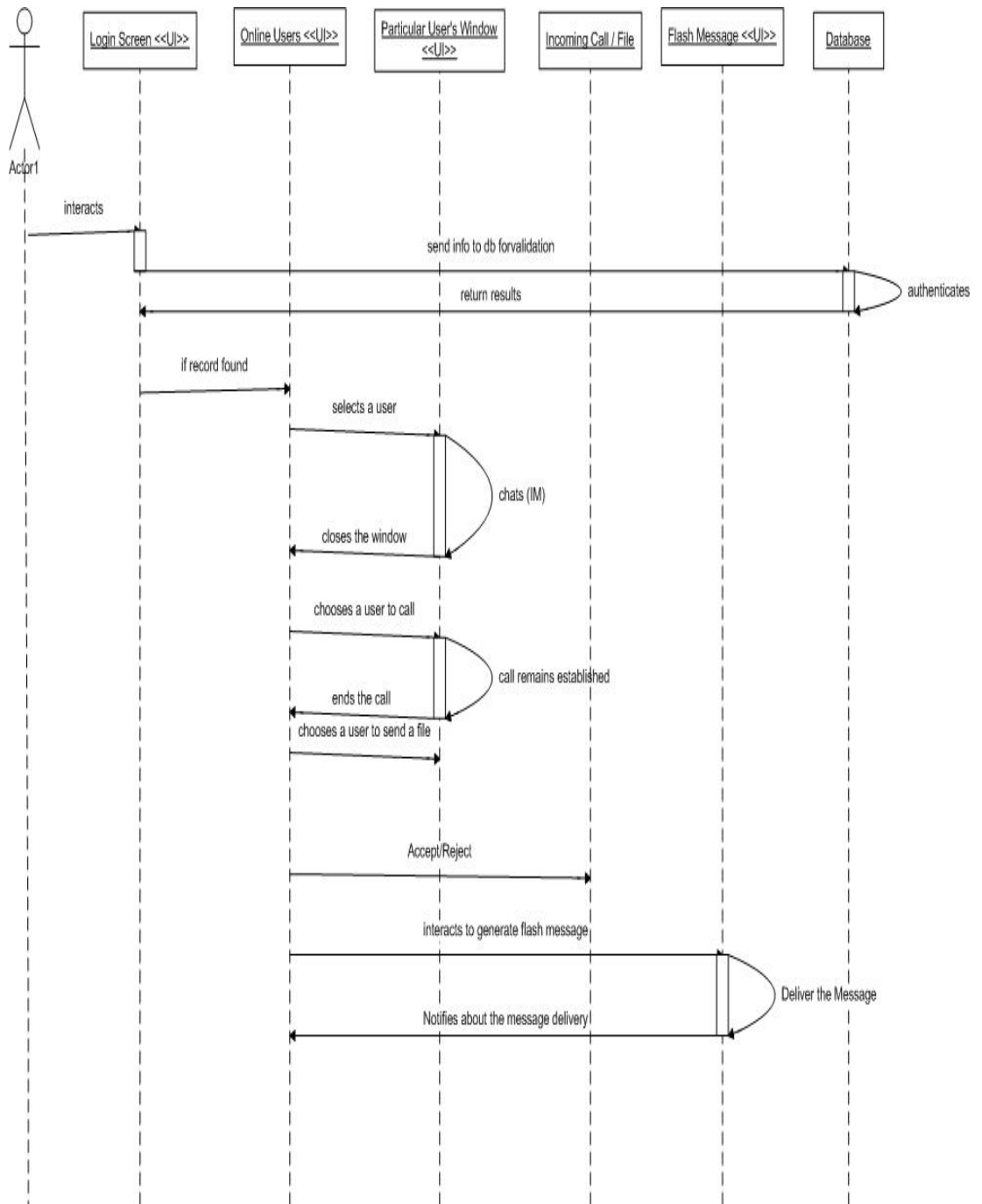


Figure 3-4: Sequence Diagram of User

The sequence diagram for the administrator shows that the administrator first interacts with the login screen to prove himself as an administrator. Then after the authentication, he will get the main menu



where he can choose either of the use-cases specified above. The sequence diagram for the user shows that first the user will be authenticated by the system from the database. After authentication, the user gets a screen where he sees his online and offline contacts. If he chooses to interact with a particular user of the system he will get a new window to chat or call or send a file.

### **3.6 Conclusion**

The system analysis of the project has been covered in this chapter. The scope of the project has been revised for the clear understanding of the requirements, key functional and non-functional requirements have been enumerated, the usecase diagram showing the major actors and their actions have been included. The sequence diagram identifying the sequence of actions taken has been incorporated as well. This chapter has been written comprehensively so that the fore coming design chapter becomes easy to comprehend.

## ***Chapter 4***

# System Design

## 4.1 Introduction

System design is a very important phase in the software development process. The succeeding implementation phase is performed taking into consideration the design constraints. This chapter begins by presenting the high level design of the project showing the main modules of the system without including much detail. Next the low level design is incorporated elucidating the modules identified in the high level design. It is then followed by the data flow diagrams of the project. Class diagram is also included focusing on the implemented classes, their attribute and their relationships with each other.

## 4.2 Architectural Diagram

The system is mainly composed of four modules. They are Instant Messaging (Text Chatting), Voice Chat Module, File Transfer Module and Offline Messaging.

The diagram presented here shows two nodes using this system. Both of them are having the above stated modules. These two nodes need a mediator (The Server) to make the connection get established between them to complete any of the desired tasks.

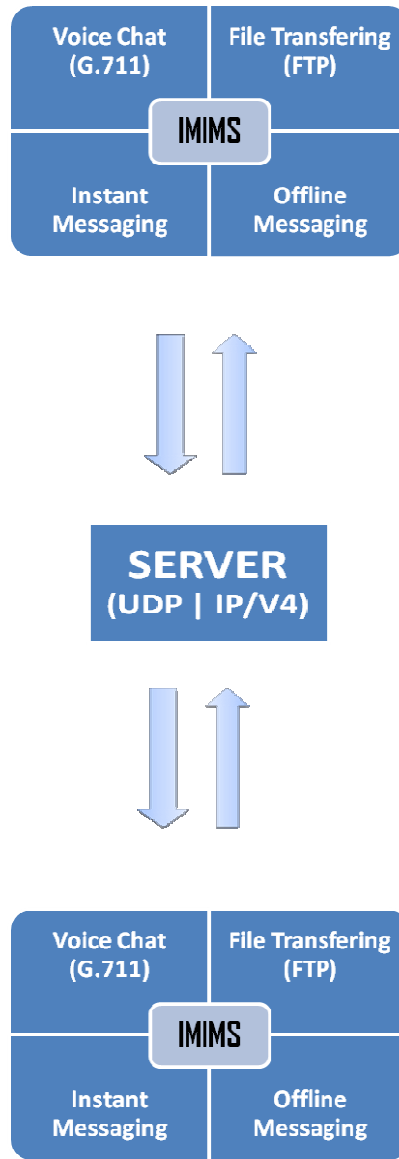


Figure 4-1: Architectural Diagram

### 4.3 High Level Diagram

The High level design of the project is shown in Figure 4-2. It is built using black box approach, focusing on the main modules of the system and not considering their inner details. The Figure 4-2 identified the four

fundamental modules of the project as instant messaging (Text Chatting), voice chat, file sharing and offline messaging modules.

Figure 4-2: High Level Design

## **4.4 Low Level Diagram**

Figure 4-3 to Figure 4-7 illustrates the detailed low level design of the project. Here each module is explained in more details.

### **4.4.1 Overall Client**

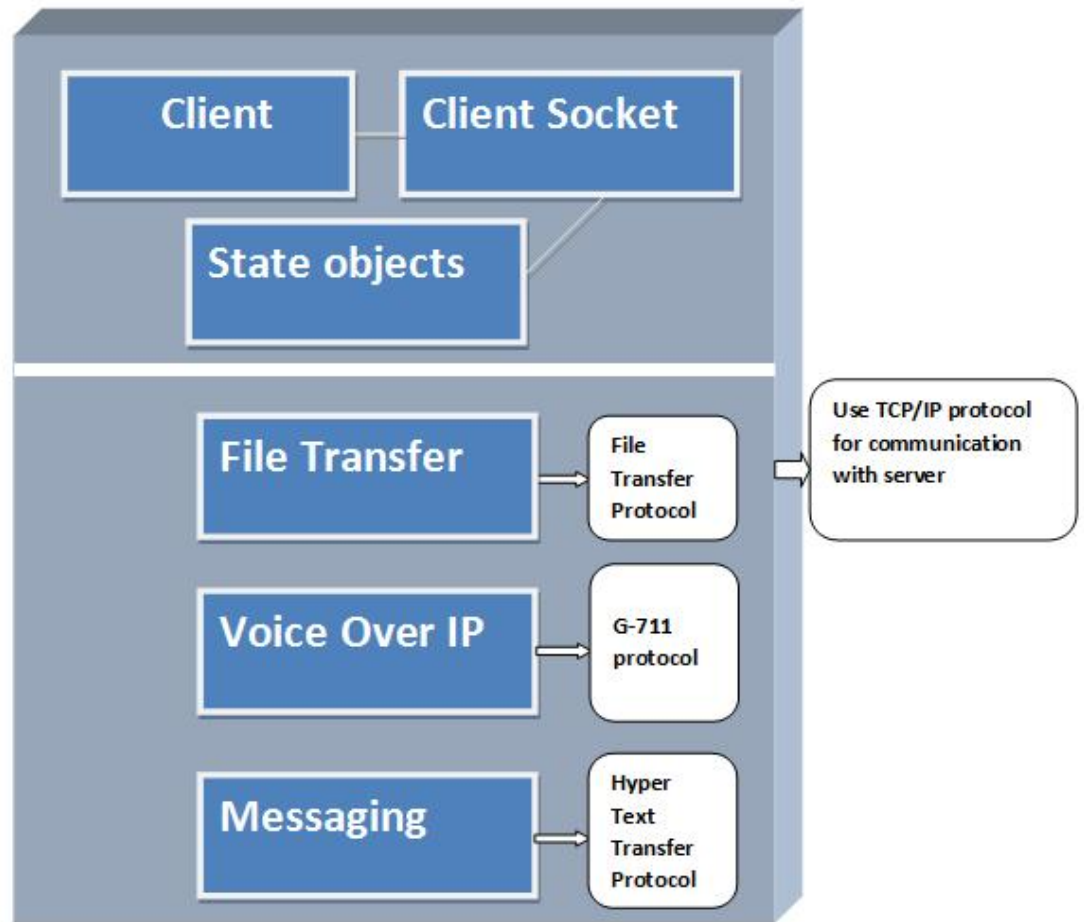


Figure 4-3: Overall Client Design

The Figure 4-3 explains the overall client design in detail. It is using TCP/IP protocol for communication with the server. File transferring is done by File Transfer Protocol (FTP), voice over IP by G-711 Protocol and messaging by Hyper Text Transfer Protocol.

#### 4.4.2 Overall Server

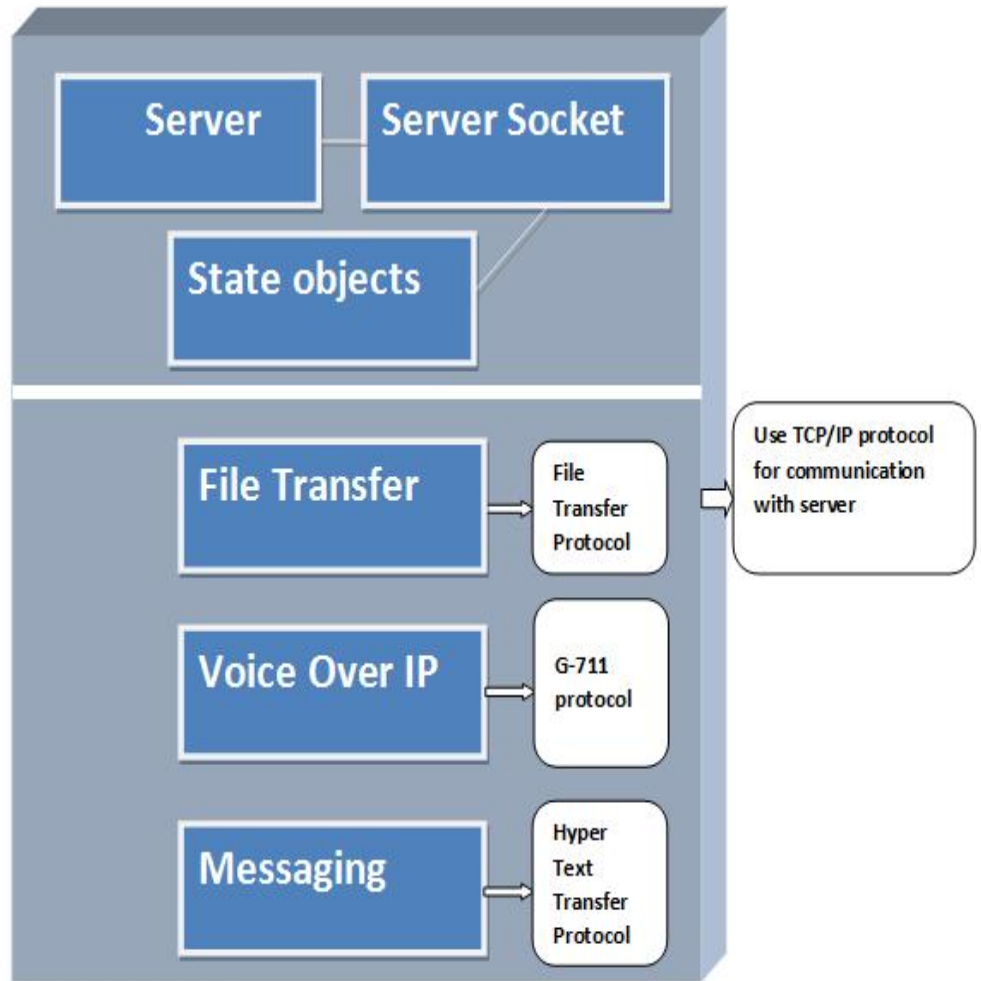


Figure 4-4: Overall Server Design

The Figure 4-4 explains the overall server design in detail. It is also using TCP/IP protocol for communication with the client.

### 4.4.3 Voice Chat Module

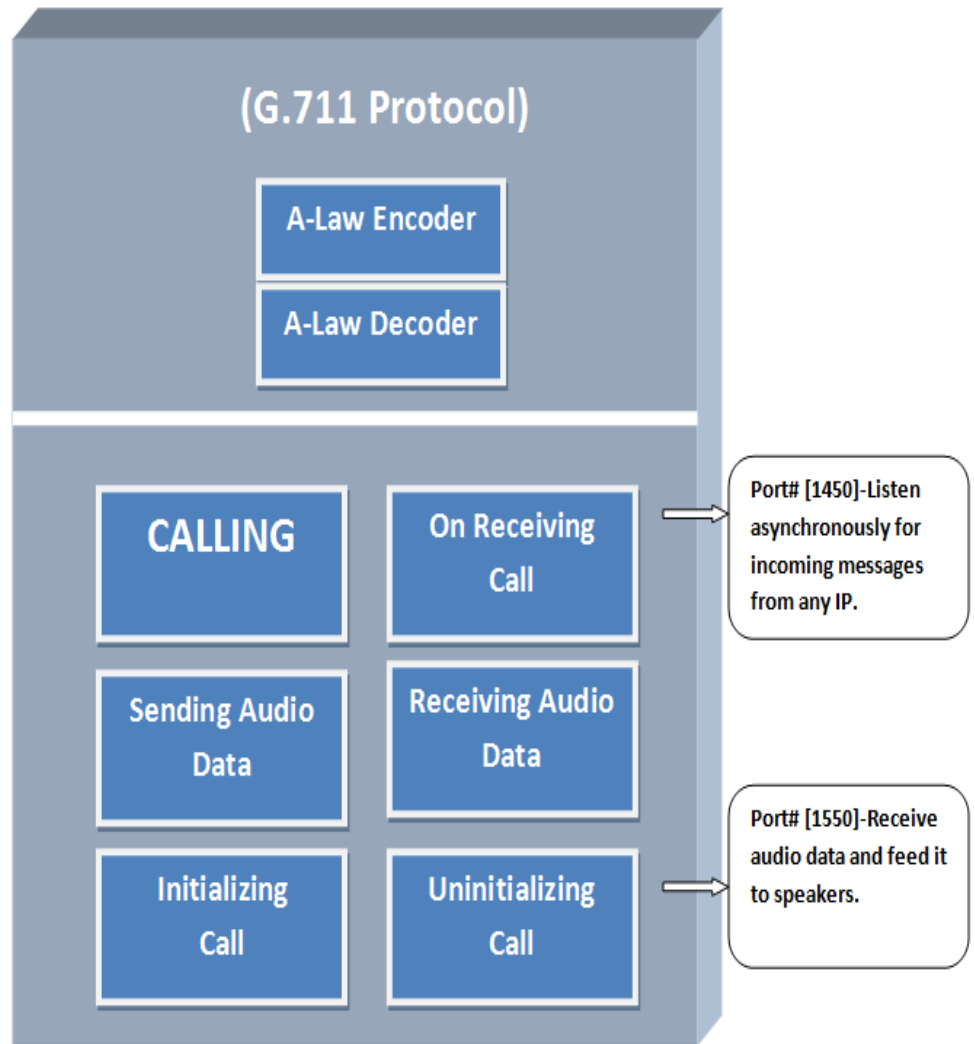


Figure 4-5: Voice Chat Module Design

The Figure 4-5 explains the voice chat module in detail. A-law encoding and decoding is used in G-711 Protocol. Listening asynchronously for incoming messages from any IP is done on Port# 1450. Port #1550 receives audio data and feed it to speakers.

#### 4.4.4 File Transfer Client

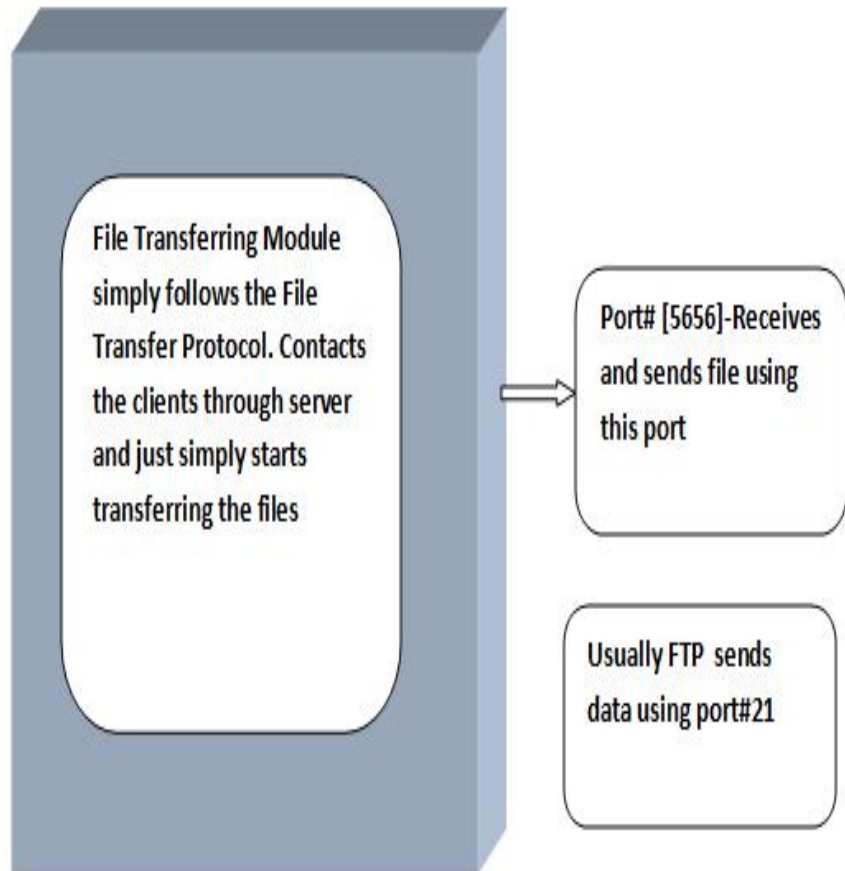


Figure 4-6: File Transfer Client Design

The Figure 4-6 explains the FTP client in detail. It simply follows File Transfer Protocol, contact the clients through server and stars transferring files. Receiving is done on Port# 5656 and data send using Port# 21.

#### **4.4.5 File Transfer Server**



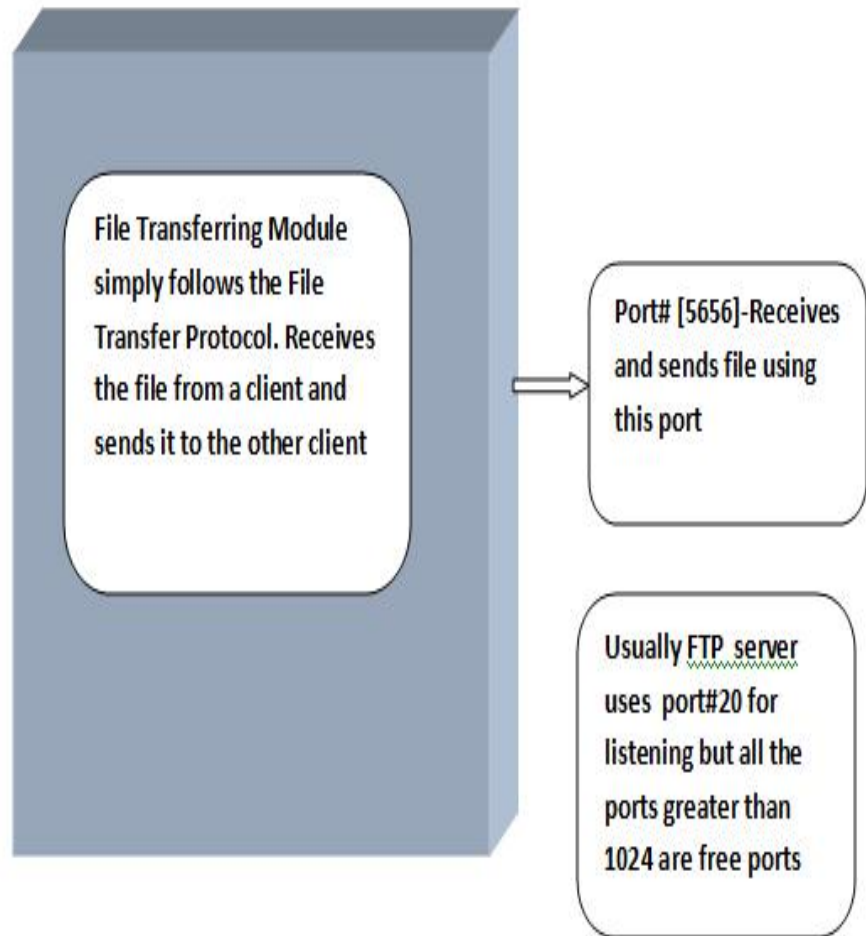


Figure 4-7: File Transfer Server Design

The Figure 4-7 explains the FTP server in detail. It simply follows File Transfer Protocol, receives the file from client and send it to another client. Receiving is done on Port# 5656 and listening using Port# 20.

## 4.5 Class Diagram

Class diagrams of Server and Client are illustrated in Figure 4-8.

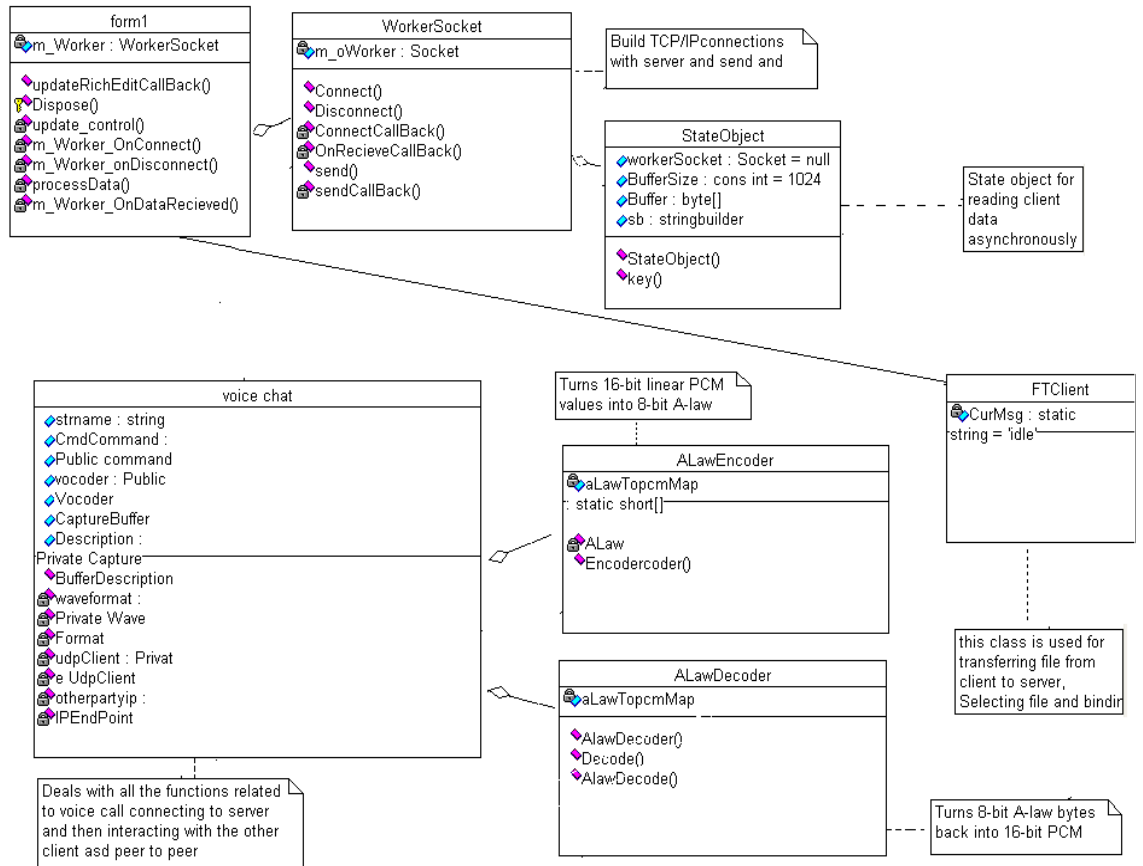


Figure 4-8: Class Diagram of Client

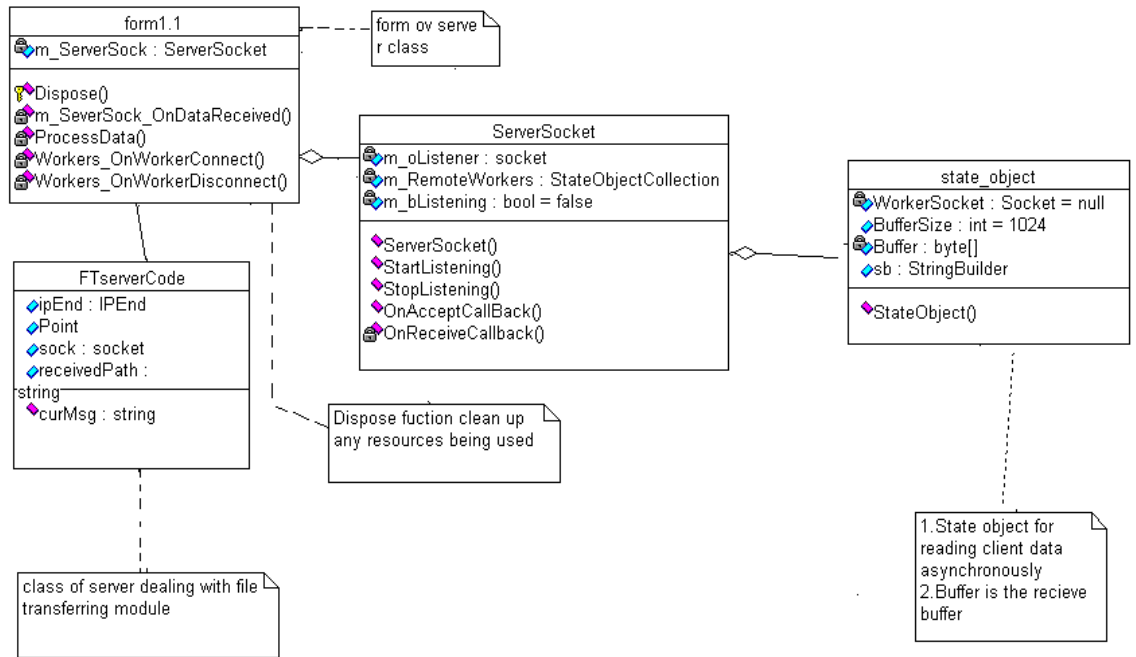


Figure 4-9: Class Diagram of Server

## 4.6 Data Flow Diagram

Data flow diagrams for login, sending/receiving file and sending/receiving voice is illustrated in following figures.

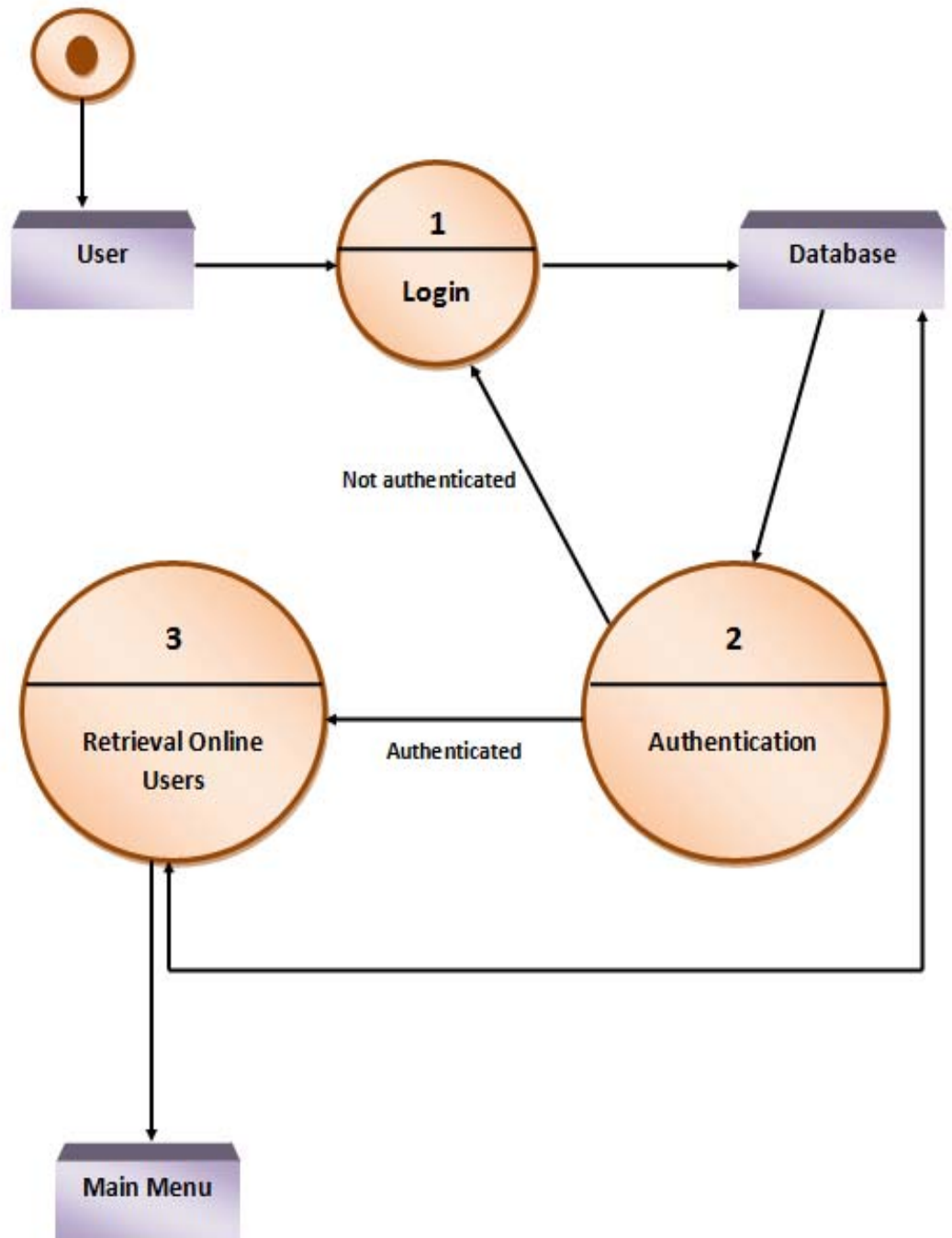


Figure 4-10: DFD for Login

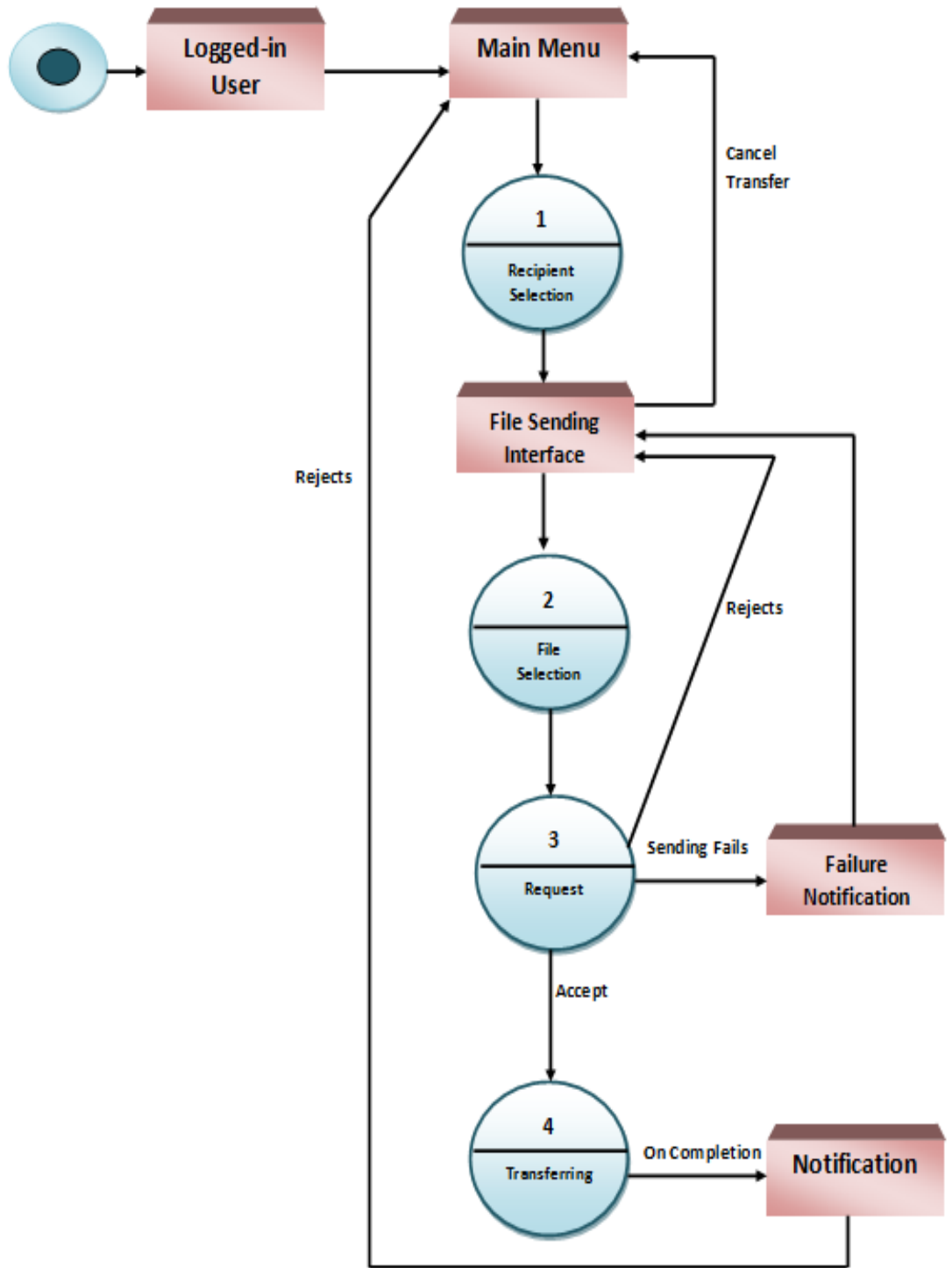


Figure 4-11: DFD for Sending File

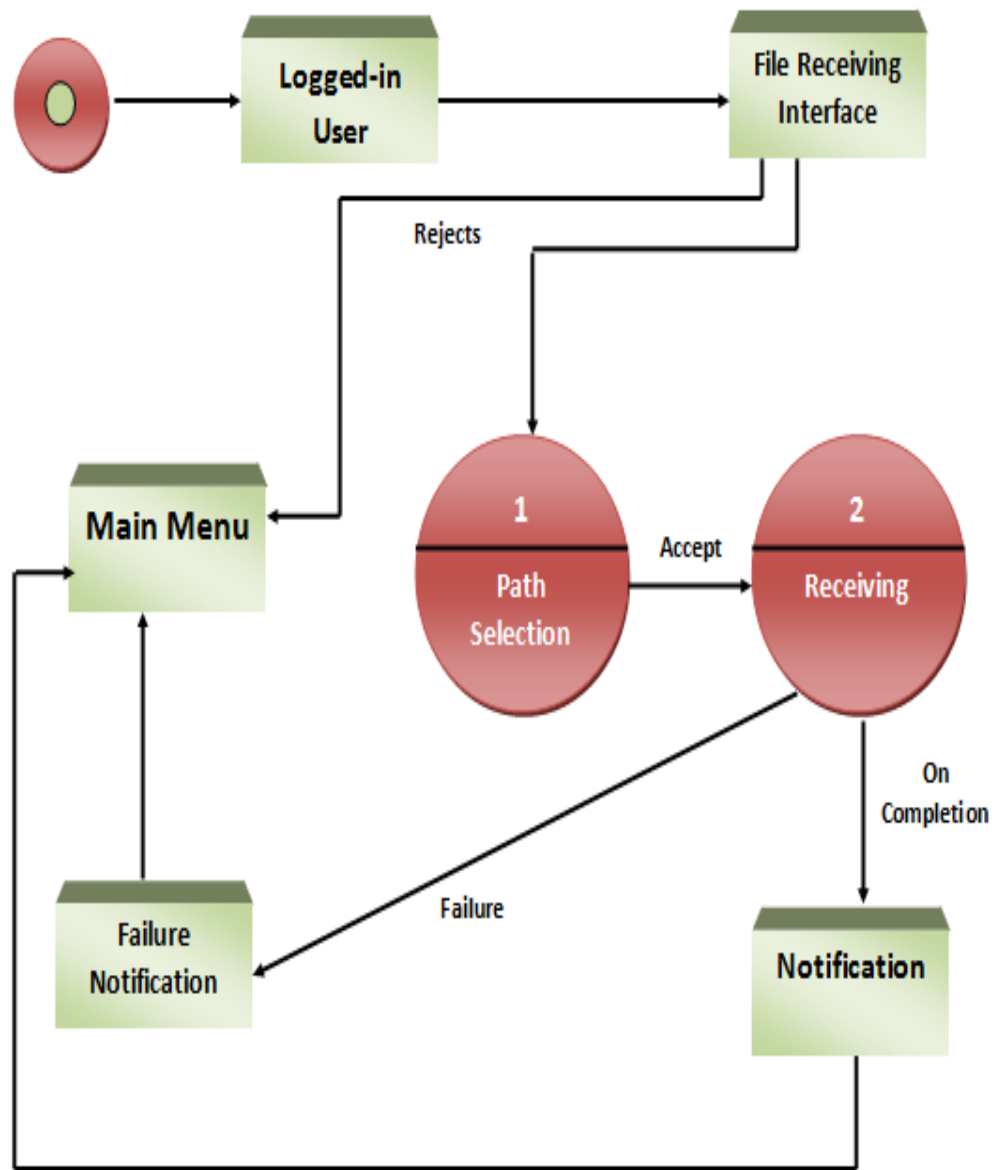


Figure 4-12: DFD for Receiving File

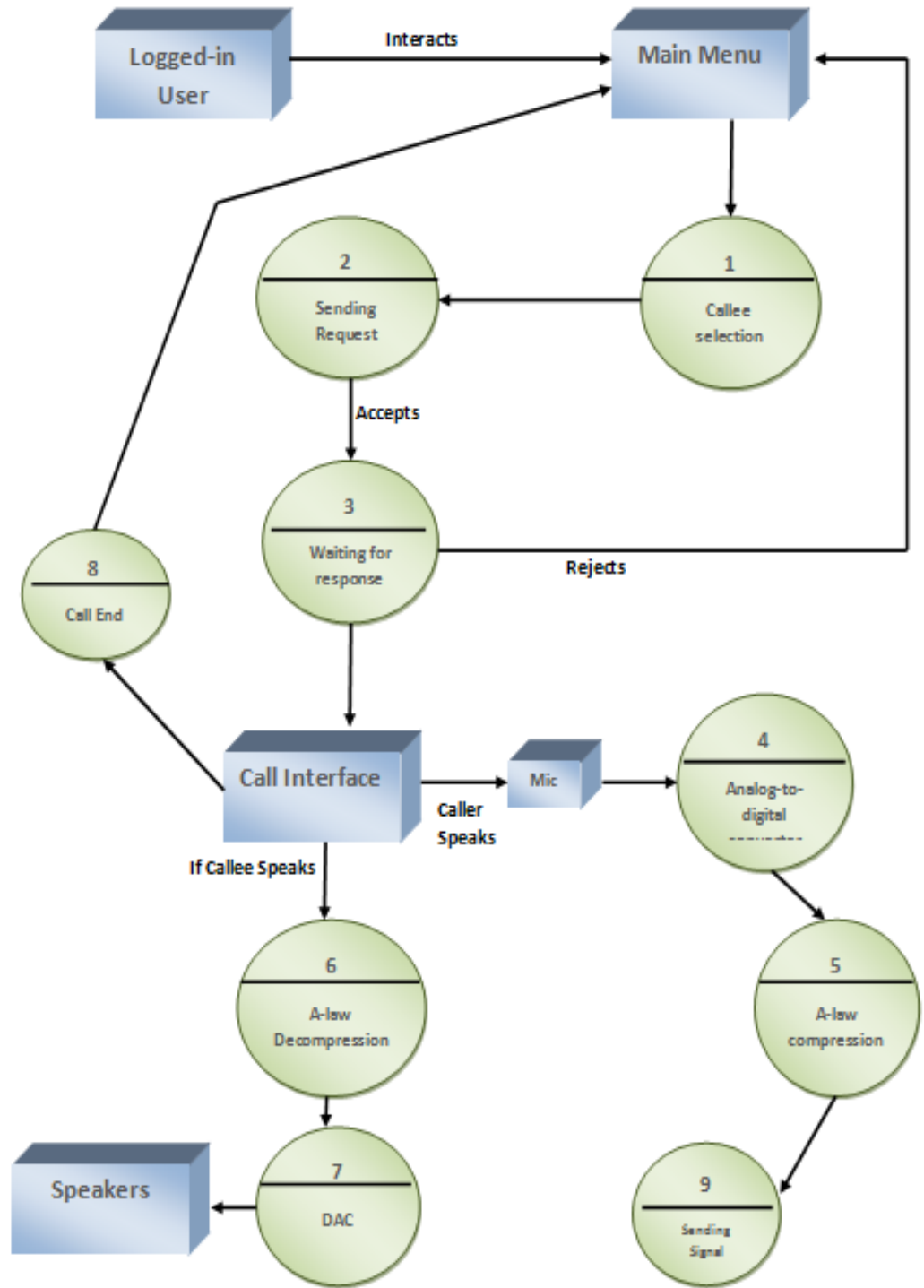


Figure 4-13: DFD for Sending Voice

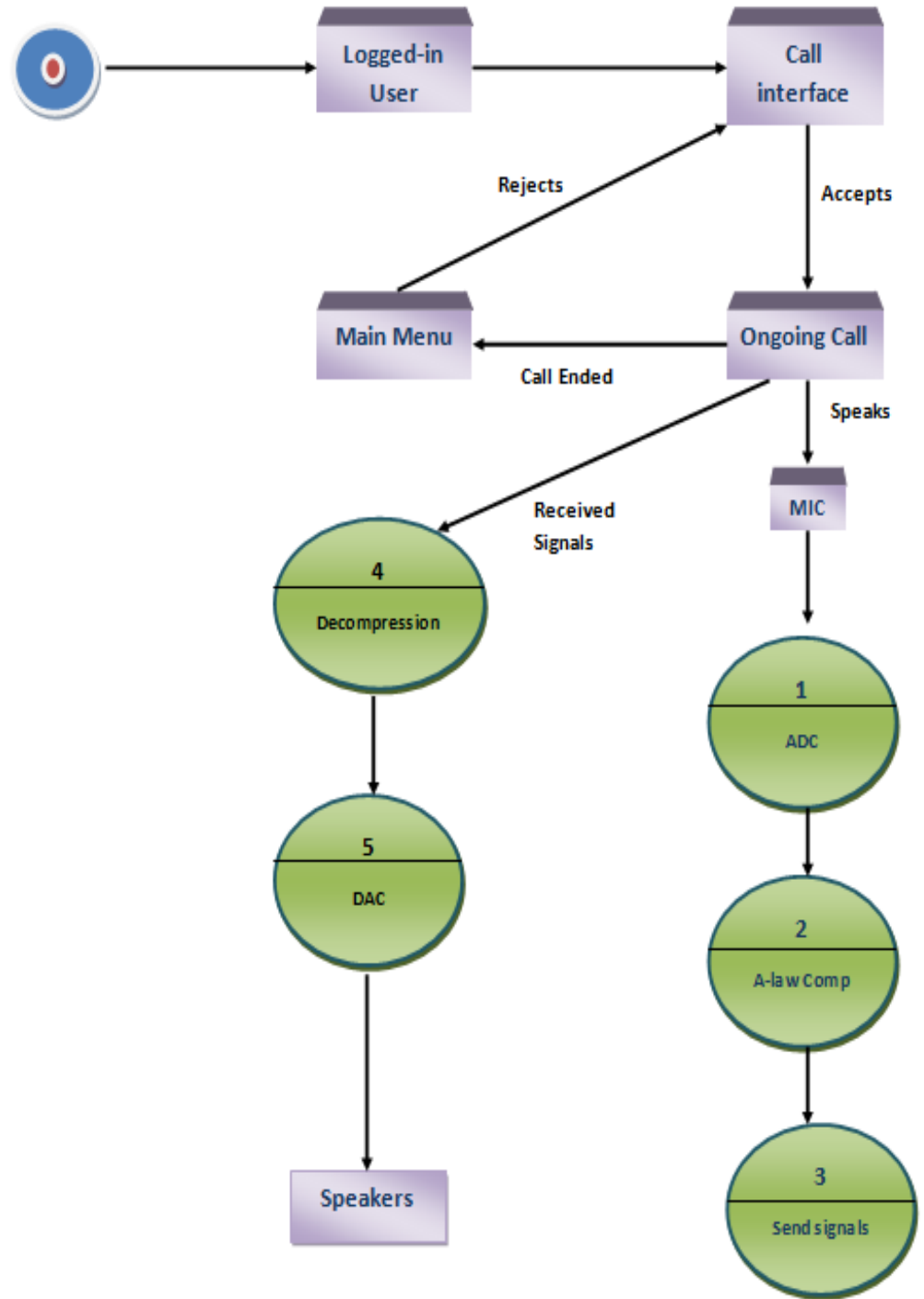


Figure 4-14: DFD for Receiving Voice

## 4.7 Conclusion



This chapter presented the architecture of Instant Messenger for Integrated Messaging System (IMIMS). It has incorporated the high level design, low level design, data flow diagram and class diagram of the system. Four main modules have been identified which are instant messaging (Text Chatting), offline messaging, voice chat and file transfer.

## ***Chapter 5***

# Implementation

## 5.1 Introduction

This chapter presents the implementation details of the project. The coding is done in C#. As illustrated in the prior chapter, there are four modules of the system, Instant messaging (Text Chatting), Offline messaging, Voice chat and File transfer. As the system is distributed so the classes which are implemented are disseminated among these modules. The implementation chapter is hence structured in the same way i.e. mentioning each module and then elucidating the classes which are needed on that module.

## 5.2 Implementation Language

The implementation language which has been chosen for the project is C#. It has been preferred over other languages because of several reasons e.g. it is a platform independent language which enhances the system's portability. This factor is significant as the system built is of distributed nature. Also it supports object oriented language which makes it simple to visualize the objects in real life. It also supports an easy integration with database. Database is build using SQL language in Microsoft SQL Server 2005.

## 5.3 Distribution of classes with respect to Modules

Considering in account the fact that the system is distributed, the implemented classes will be explained with respect to their placement in the system. From the high level design in the preceding chapter, the two identified sub-systems are IMIMS Client and IMIMS Server. The four identified modules of IMIMS Client are Voice Chat, Text Chat, Offline Messaging and File Transfer.

Each module of the system contains a certain set of implemented C# classes and their interaction carry out the operation for which they are destined for i.e. text chat for the user using IMIMS.

### **5.3.1 IMIMS Client**

This sub-system holds a set of classes which collectively form the client side functionalities including user interface. It means that it is through these classes that the user interacts with the system without indulging in the internal details of the system. The classes present on this system can mainly be separated into eight sets.

#### **5.3.1.1 Login**

Login class (Form1.cs) provides the implementation for the login form. This class deals both with the user interface and the user data for the login

form. The functionalities provided by this class are authentication of the user login into the system, receive offline messages as soon as the user logs in to the system.

### **5.3.1.2 Main Menu**

Main Menu class (Main\_Menu.cs) provides the implementation for the Main Menu form. This class deals both with the user interface and the user data for the main menu form. This class creates UDP socket for receiving offline messages and opens up the socket to listen at port# 7000. It also creates two TCP sockets listening at port # 5656 and port # 9050 for file transfer and instant text respectively. The major functionalities of this class are connecting to the other client for text chatting, sending text messages in chatting, receiving offline messages in main menu, receiving file, disconnecting from the server and receiving voice call invitations.

### **5.3.1.3 Text Chat**

Text Chat class (TextMessaging.cs) provides the implementation for the Text Chat form. This class deals both with the user interface and the user data for the Text Chat form. This class receives the TCP socket as an argument from the main menu class whenever a user receives a request for text chat or clicks the text chat button from the main menu, driving the control to text chat form. The major functionalities of this class are sending text data, receiving text data.

### **5.3.1.4 Voice Chat**

Voice Chat class (VoiceChat.cs) provides the implementation for the Voice Chat form. This class deals both with the user interface and the user data for the Voice Chat form. It creates two UDP sockets, one socket for receiving and sending the voice data and other for sending and receiving acknowledgement. Once the user accepts the invitation for the voice call or clicks the voice chat button on the main menu, the control is driven to voice chat class. This class uses the objects of A-law encoder and A-law decoder class to encode the voice data before sending the UDP datagram on the network and decodes voice data after receiving the UDP datagram from the network.

### **5.3.1.5 File Transfer**

File Transfer class (FileTransfer.cs) provides the implementation for the File Transfer form. This class deals both with the user interface and the user data for the File Transfer form. This class creates a TCP socket which is used to send a file. The other client receives the file using the TCP socket listening at port # 5656.

### **5.3.1.6 Offline Messaging**

Offline Messaging class (OfflineForm.cs) provides the implementation for the Offline Messaging form. This class deals both with the user interface

and the user data for the Offline Messaging form. The only functionality of this form is to send the offline messages to the clients added by the user.

### **5.3.1.7 A-Law Encoder**

This class is an implementation of G.7-11 audio protocol. It turns the 16-bit linear PCM values into 8-bit A-law bytes.

### **5.3.1.8 A-Law Decoder**

This class supports the implementation of G.7-11 audio protocol. It turns the 8-bit A-law bytes back into 16-bit linear PCM values.

## **5.3.2 IMIMS Server**

This sub-system deal with the server side of the system consisting of classes mentioned below along with their details.

### **5.3.2.1 Form1**

It creates a TCP socket listening at port # 11000 to accept request for connections from clients. Also creates a UDP socket and opens it for listening at port # 5000 for user authentication. It receives username and password from the client and connects it the server to the database to verify the user referring to the credentials table maintained in the database. It also creates a UDP socket opened for listening at port # 1300 to receive offline messages

from the client and connects to the database to save offline messages in Offline Message table in IMIMS database.

## **5.4 Conclusion**

This chapter incorporated the details of the classes implemented. . The classes have been distributed among the two basic sub- systems which are IMIMS Server and IMIMS Client. C# has been used as the programming language for the project due to its object-oriented and platform independent nature. Also the database has been deployed in Microsoft SQL Server 2005.

## ***Chapter 6***

# Testing

## 6.1 Introduction

Testing is a very important phase in the software development process. Once the coding process is completed, then the software goes under the testing process which involves checking the codes for errors and bugs. It involves any activity aimed at evaluating an attribute or capability of a program or a system and determining that it meets its required results\*. This chapter involves all the testing techniques which have been employed in the project and conclusions which have been deduced on the basis of the results of the testing procedures. Test cases for different units and components have been drafted illustrating their expected behaviors on the success and failure of each test. The output of each test is then compared with the one documented in the test case to make sure that the system behaves in the same way in which it is meant to behave.

## 6.2 Testing Process

The testing process has been carried out throughout the development process as an iterative approach has been used in the project for development. Each phase of development was visited several times making sure that the testing process goes in parallel with the development process.



The testing was basically done at three levels, Unit testing, Integration and System testing.

## 6.2.1 Client Testing

The test cases for different components of the system are elucidated and shown under the following headings.

### 6.2.1.1 Text Messaging

Identity	TextMessaging.cs	
Category	Component testing	
Description	This class accepts a socket created in the MainMenu.cs. Once the system accepts the connection it creates a socket randomly on any port and this socket is passed to this class. After the creation of this socket, it handles the text chatting between the two client machines	
Set up	Dependable classes, MainMenu.cs	
Expected Results	Success	The two clients are able to have text chat successfully

	Failure	Exceptions are thrown.
--	---------	------------------------

Table 6-1: Test case for Text Messaging

### 6.2.1.2 File Transfer

Identity		FileTransfer.cs
Category		Component testing
Description		This class is responsible for sending a file over a tcp stream. The client starts listening for the files at port 5656 on first run. The listening is on the MainMenu.cs
Set up		Dependable classes, MainMenu.cs
Expected Results	Success	The two clients are able to send and receive file successfully
	Failure	Exceptions are thrown.

Table 6-2: Test case for File Transfer

### 6.2.1.3 Offline Messaging

Identity		OfflineForm.cs
Category		Component testing
Description		This class sends messages to the server and the server stores these messages and displays instantly to the user if he is online and if he is offline, it is displayed on the next sign-in
Set up		Dependable classes, MainMenu.cs (for the listening of Flash Messages), Form1.cs (for Offline Messaging listening)
Expected Results	Success	The client is able to send a message to the server and the server stores them successfully and the message is displayed to the user on successful sign-in
	Failure	Exceptions are thrown.

Table 6-3: Test case for Offline Messaging

#### 6.2.1.4 Voice Chat

Identity		VoiceChat.cs
Category		Component testing
Description		This class is responsible for sending and receiving voice data
Set up		Dependable classes, ALawDecoder.cs, ALawEncoder.cs, MuLawDecoder.cs, MuLawEncoder.cs
Expected Results	Success	The two clients are able to have voice chat
	Failure	Exceptions are thrown.

Table 6-4: Test case for Text Messaging

### 6.2.1.5 Online Clients

Identity		MainMenu.cs
Category		Component testing
Description		This class is responsible for overall interaction with all the online users and it maintains a list of online users.
Set up		Dependable classes, ServerSocket.cs, WorkerSocket.cs
Expected	Success	The list of online clients is accurate

Results	Failure	Exceptions are thrown.
---------	---------	------------------------

Table 6-5: Test case for Text Messaging

## 6.2.2 Server Testing

Identity		Form1.cs
Category		Component testing
Description		This class is responsible for overall interaction with all clients to update online clients with them.
Set up		Dependable classes, Database_IMIMS.cs, ServerSocket.cs,
Expected Results	Success	The list of online clients with the server and client is accurate
	Failure	Exceptions are thrown.

Table 6-6: Test case for Server

## 6.2.3 Static Analysis of Code

Besides testing the code dynamically, static analysis of the code has been done as well to find defects, if any, in the blocks of code due to which it

does not implement the exact requirement or to determine the ways by which the code can be optimized to make it full proof.

The code has been statically analyzed in many ways which are briefly illustrated under following headings.

### **6.2.3.1 Control Flow Analysis**

Control flow analysis has been carried out for the verification and validation of control blocks in the source code, for instance, the 'for', 'while' loops and the 'if' condition blocks. It has been observed that no unnecessary code has been included and all these blocks are optimized.

### **6.2.3.2 Data Analysis**

Data analysis has been done to find and remove improper initializations, unnecessary assignments and those variables that are declared but never used. All such unnecessary lines have been eliminated thus giving a refine code.

### **6.2.3.3 Interface Analysis**

Interface analysis has also been done to insure consistency of interface, class, procedure declaration, definition and their use. It has also

been observed through test that all the method declared in the interface is correctly implemented in the classes and that there are no redundant methods.

## **6.4 Conclusion**

This chapter illustrated the testing process of the system that has been carried out and the corresponding results obtained. The testing of a system has been done in complete detail using test cases. Using these test cases, results have been authenticated. Static inspection of the code has been carried out as well so that it becomes optimized and does not become redundant. All the test results were very successful proving that system delivers all its functionalities in an efficient way.

# **APPENDIX A**

## **User Manual**

## **User Manual**

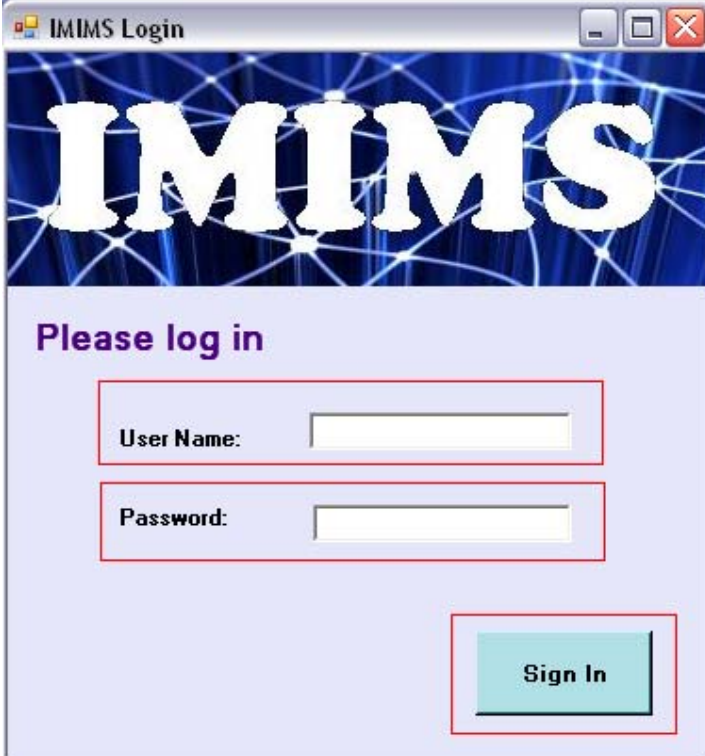
This application software has two categories of users.

1. The user that uses the functionality of the software as a client.
2. The user that mostly manages the database and accounts as server.



## 1. Client Side

Once the program is opened, the user will be ready to begin using the messenger. The initial interface of the application will look like as follows:



1

2

3

Figure A-1: Login

Each of the demarcations will have a specific purpose.

1- Place where users should write the email address (for example, XYZ@signals.com).

2- Here the user should write the password of the email address. No changes should be made as regards the password use for the email.

3- After completing the previous steps, users can sign in.

After signing in, for the first time the users will see the interface of their instant messenger with the contacts they have.



Figure A-2: Main Menu

Each of the demarcations will have a specific purpose.

**1-** Change Password button. This button will open change password form where user can change its password.

**2-** List of Online users. This list will show the online users and it is continuously updated by the server.

**3-** Status bar. This will show whether user is connected with the server or not connected.

**4-** Sign Out button. This button will sign out the user from the messenger.

**5-** Send Message button. This button will open a send message form where user can chat with other users.

**6-** Send File button. This button will open a send file form where user can send a file to a user.

**7-** Call button. This button will open a voice chat form where user can voice chat with other user.

**8-** Offline Messages button. This button will open an offline messages form where user can send messages to the users that are offline.

## **1.1 Changing Password**

After clicking the change password button on the main menu a Change Password form will be open this looks like:



Figure A-3: Changing Password

Each of the demarcations will have a specific purpose.

- 1- Old Password field. Place where user writes the current set password.
- 2- New Password field. Place where user writes the new password that he wishes to set.
- 3- Confirm Password field. User confirms the new password by writing it again here.
- 4- Change Password button. This button changes the password of the user.

## 1.2 Sending Message

For chatting with other users, select any user from the online user list and then click on the Send a Message button on the main menu. It will open a Text Messaging form which looks like:

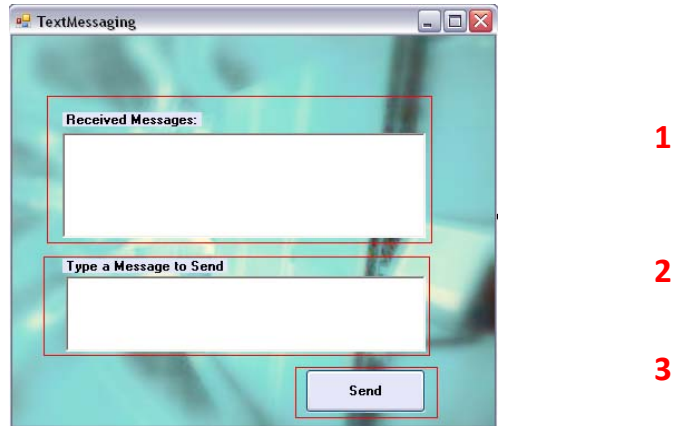


Figure A-4: Sending Message

Each of the demarcations will have a specific purpose.

**1-** Received Messages filed. Here messages received by the other user appear including the messages sent by you.

**2-** Type a Message field. To send a message to the other user, type a message over here.

**3-** Send button. To send a message written in the above field 2, click the send button and the message will be delivered to the other user.

## 1.3 Sending File

To send a file to one user at a time, click on the Send File button on the main menu. This will open a File Transfer form which looks like:

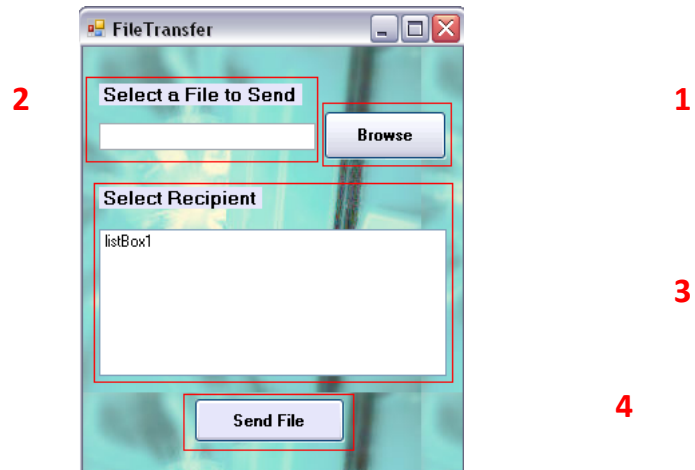


Figure A-5: Sending File

Each of the demarcations will have a specific purpose.

**1-** Browse Button. To send a file click on the browse button to browse for the file in the memory.

**2-** Select a File to Send field. This field will show the complete URL of the sending file.

**3-** Select Recipients filed. This field will show online users. Select one user to whom you want to send a file.

**4-** Send File button. Click this button to send a file to the desired user.

## 1.4 Voice Call

For having voice chat with other users, click on the Call button on the main menu. This will open a Voice Chat form which looks like:

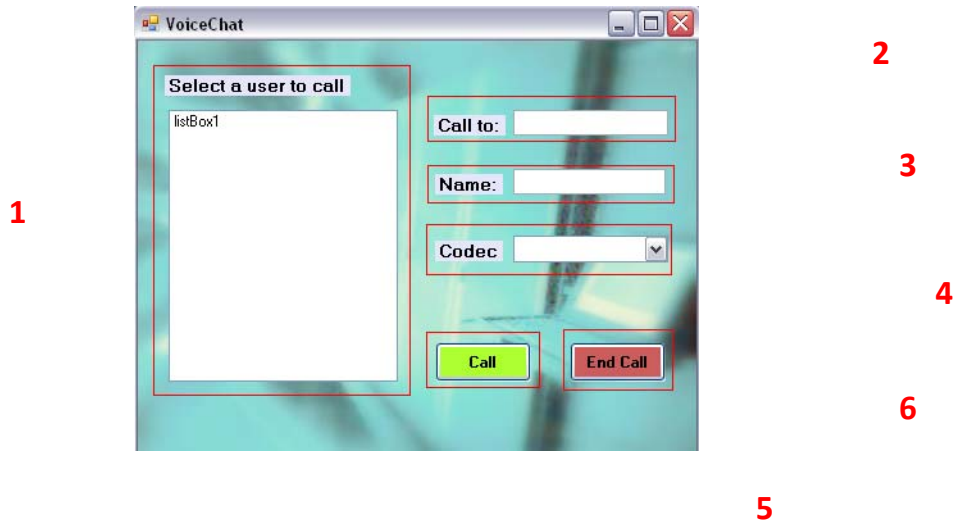


Figure A-6: Voice Call

Each of the demarcations will have a specific purpose.

**1-** Select a user to call field. This field will show list of online users. Select any one user.

**2-** Call to field. This field will indicate the user you have selected for calling.

**3-** Name field. Write your name on this field.

**4-** Codec selection. Select any one from the three choices such as A-law,  $\mu$ -law or none.

**5-** Call button. This button will send your request for voice chat to other user.

**6-** End Call button. This button will terminate the voice chat.

## 1.5 Sending Offline Messages

To send offline messages to offline users or Flash messages to online users, click on the offline messages button on the main menu. This will open an Offline Messages form which looks like:

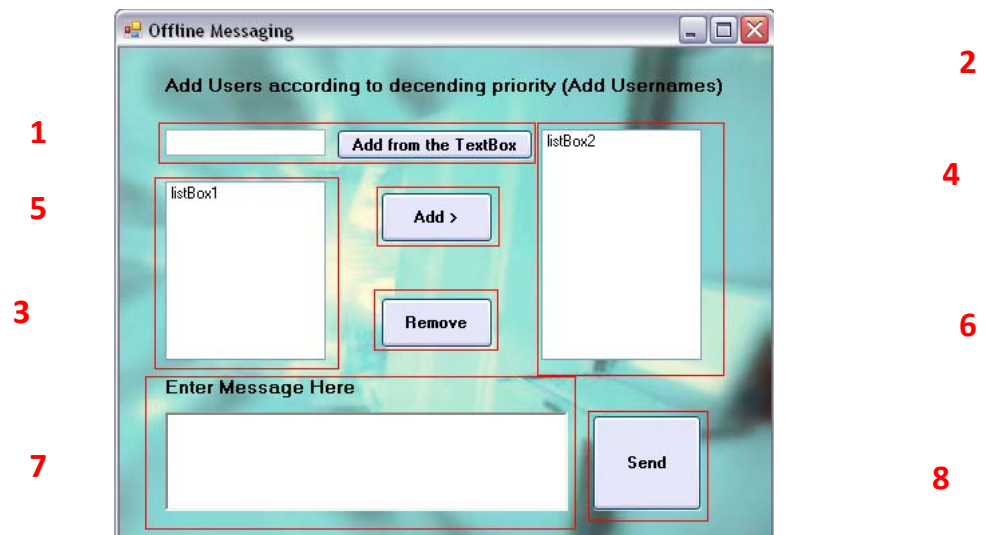


Figure A-7: Sending Offline Messages

Each of the demarcations will have a specific purpose.

1- Add name field. Add an email id in field to whom you wish to send an offline message.

2- Add from text box button. Clicking this button will add the id from field 1 to the field 4.



**3-** Online users filed. From here you can select any online user to send a message. This message will be a Flash message. Just select any user in this field.

**4-** Selected users filed. Here list of users that are selected for sending message are listed.

**5-** Add Button. Clicking this button will add the selected user from the field 3 to field 4.

**6-** Remove button. Clicking this button will remove the selected id from field 4.

**7-** Enter Message field. Write the message the message here to send to the users listed in filed 4.

**8-** Send button. Clicking this button will send offline messages to offline users and flash messages to online users listed in the field 4.

## **1.6 Receiving Offline Messages**

When the user will login, any offline messages sent to him by the other users will pop up on an offline messages form. This will looks like:



Figure A-8: Receiving Offline Messages

Each of the demarcations will have a specific purpose.

1- Offline messages list box. This list box will show offline messages.

2- Delete Messages button. Clicking this button will delete the selected offline message from the list box.

## 2. Server Side

Once the server is open, the IMIMS Server form will appear which looks like:

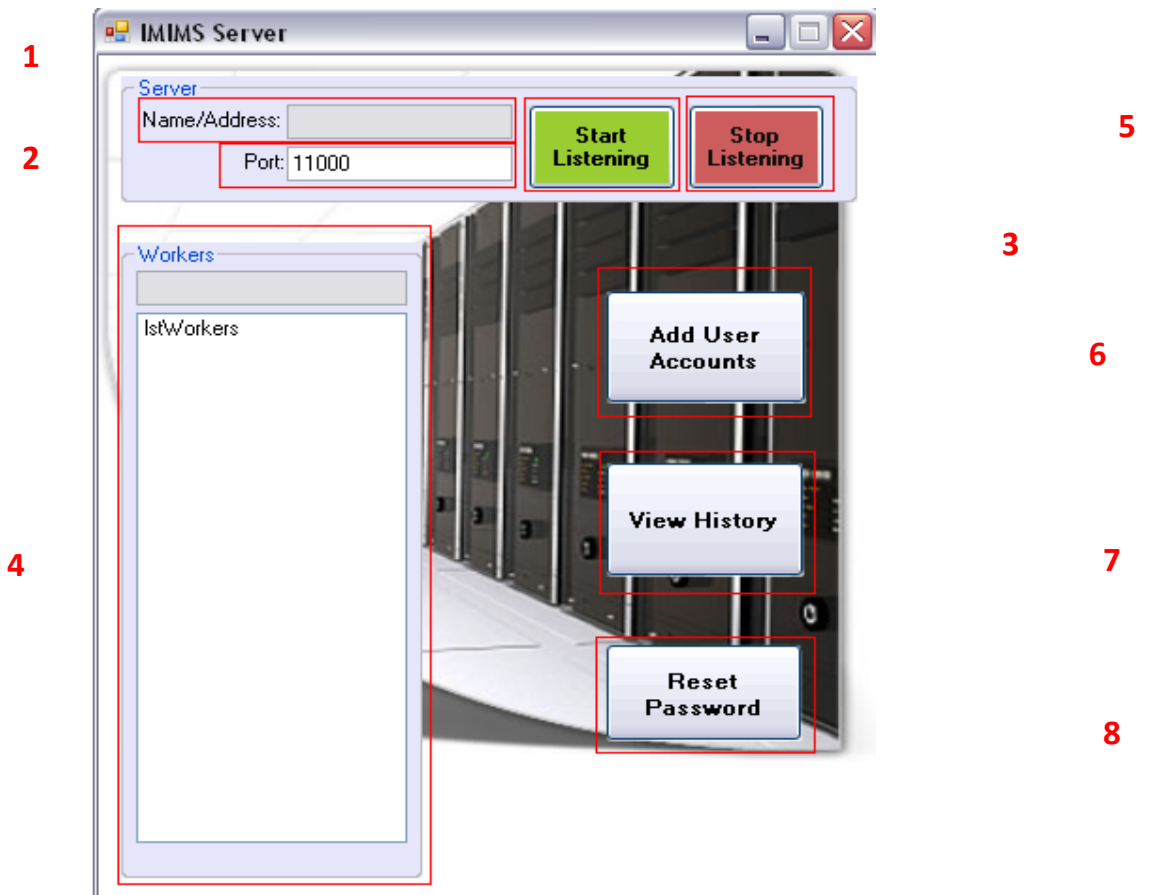


Figure A-9: Server

Each of the demarcations will have a specific purpose.

**1-** Name field. This field will show the name of computer on which server is running.

**2-** Port field. Add the port number on which server will listen all the requests.

**3-** Start Listening button. By clicking this button server will open its port for listening requests.

**4-** Workers list. This list will show the workers that are logged in.

5- Stop Listening button. By clicking this button server will stop listening to requests.

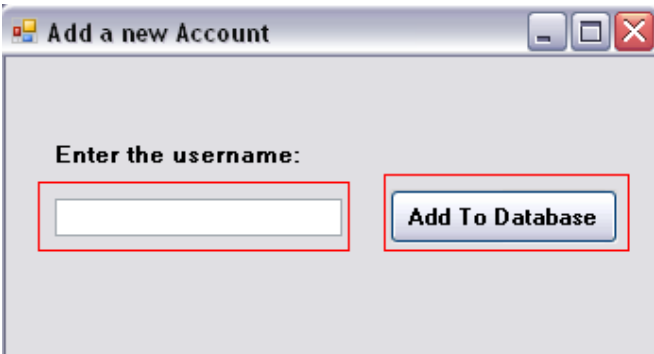
6- Add User Account button. Clicking this button will open an Add a New Account form where new users can be added.

7-View History button. Clicking this button will open a History form where table will demonstrate the history of the users.

8-Reset Password button. Clicking this button will open a Reset Password form where user's password could be reset to default password 123456 without knowing the users password.

## 2.1 Adding User Accounts

To add a user account, click on the Add User Accounts button on the IMIMS Server form. This will open an Add a New Account form which looks like:



1

2

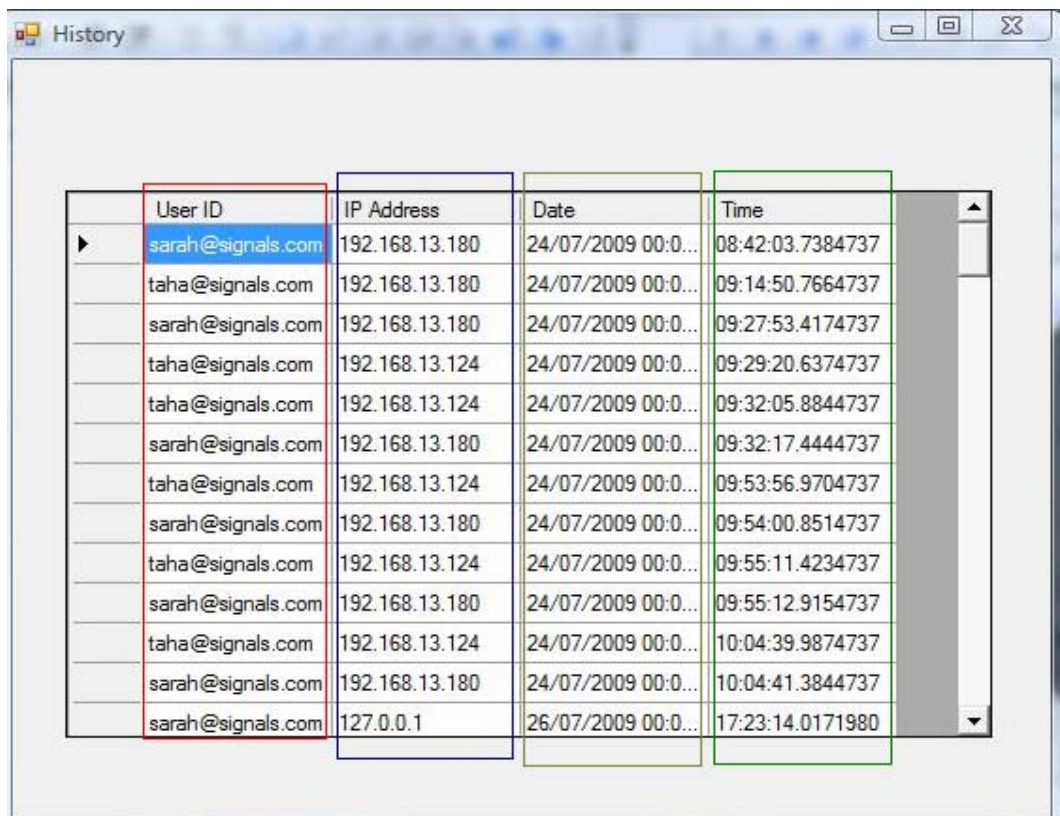
Figure A-10: Adding User Account

Each of the demarcations will have a specific purpose.

- 1- Enter Username field. Enter the username in this field which is to be added.
- 2- Add to Database button. Clicking this button will create a new account in the database of the username entered in the previous field.

## 2.2 Viewing History

To view history of the users logged in, click on the View History button on the IMIMS Server form. This will open a history table which looks like:



User ID	IP Address	Date	Time
sarah@signals.com	192.168.13.180	24/07/2009 00:0...	08:42:03.7384737
taha@signals.com	192.168.13.180	24/07/2009 00:0...	09:14:50.7664737
sarah@signals.com	192.168.13.180	24/07/2009 00:0...	09:27:53.4174737
taha@signals.com	192.168.13.124	24/07/2009 00:0...	09:29:20.6374737
taha@signals.com	192.168.13.124	24/07/2009 00:0...	09:32:05.8844737
sarah@signals.com	192.168.13.180	24/07/2009 00:0...	09:32:17.4444737
taha@signals.com	192.168.13.124	24/07/2009 00:0...	09:53:56.9704737
sarah@signals.com	192.168.13.180	24/07/2009 00:0...	09:54:00.8514737
taha@signals.com	192.168.13.124	24/07/2009 00:0...	09:55:11.4234737
sarah@signals.com	192.168.13.180	24/07/2009 00:0...	09:55:12.9154737
taha@signals.com	192.168.13.124	24/07/2009 00:0...	10:04:39.9874737
sarah@signals.com	192.168.13.180	24/07/2009 00:0...	10:04:41.3844737
sarah@signals.com	127.0.0.1	26/07/2009 00:0...	17:23:14.0171980

Figure A-11: History Table

Each of the demarcations will have a specific purpose.

**1-** User ID column. Number of users logged in will be shown in this column.

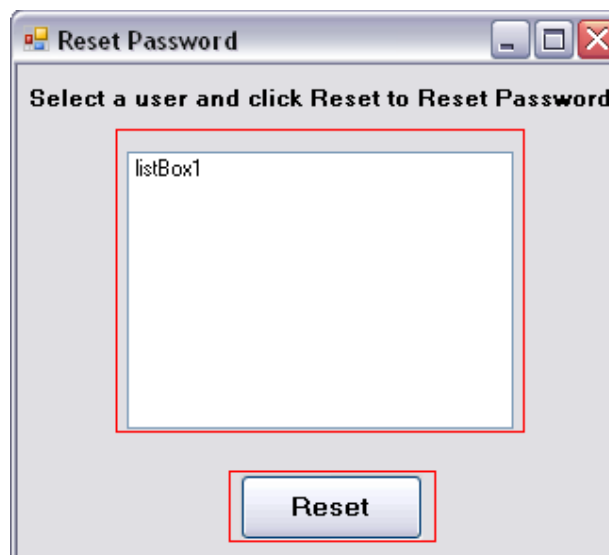
**2-** IP Address column. IP addresses of the users logged in will be shown in this column.

**3-** Date column. Date on which user logged in will be shown in this column.

**4-** Time column. Time at which user logged in will be shown in this column.

## 2.3 Resetting Password

To reset password of any user, click on the Reset Password button on the IMIMS Server form. This will open a Reset Password form which looks like:



The screenshot shows a Windows-style dialog box titled "Reset Password". The dialog contains the text "Select a user and click Reset to Reset Password" at the top. Below this text is a large, empty list box labeled "listBox1". At the bottom of the dialog is a button labeled "Reset". A red rectangular box highlights the list box, and another red rectangular box highlights the "Reset" button.

**1**

**2**

Figure A-12: Resetting Password

Each of the demarcations will have a specific purpose.

**1-** Registered Users list. Select a user from this list whose password needs to reset.

**2-** Reset button. Clicking this button will set the selected users password to default 123456.

## **APPENDIX B**

### **Hardware and Software Requirements**

# Hardware and Software Requirements

## Hardware Requirements:

- 1.0 GHz Processor or More
- 256 MB of RAM or More
- Microphone and speakers (Voice Chatting)

## Software Requirements:

- **Platform:**  
.Net framework 2.0 or higher
- **Operating System:**  
Windows ® Vista/XP/98/2000/NT 4.0
- **Other:**



DirectX 9.0 or higher

## **APPENDIX C**

### **References**

## References

[1] <http://en.wikipedia.org/wiki/Client-server>

[2] <http://communication.howstuffworks.com/ip-telephony.htm>

[3] [http://www.lincoln.edu/math/rmyrick/ComputerNetworks/  
InetReference/127.htm](http://www.lincoln.edu/math/rmyrick/ComputerNetworks/InetReference/127.htm)

[4] [http://en.wikipedia.org/wiki/A-law\\_algorithm](http://en.wikipedia.org/wiki/A-law_algorithm)

[5] [http://en.wikipedia.org/wiki/%CE%9C-law\\_algorithm](http://en.wikipedia.org/wiki/%CE%9C-law_algorithm)