# UNIVERSITY OF CINCINNATI

**Date:** 02/03/2005

**I,** Ainars Klavins,

**hereby submit this work as part of the requirements for the degree of:**

Master of Science

**in:**

Computer Science

**It is entitled:**

Instant Messaging Tool for Collaboration in a Peer-to-Peer Network:
MyBook Instant Messenger

**This work and its defense approved by:**

**Chair:** Kenneth Berman, PhD

Fred Annexstein, PhD

Jerry Paul, PhD

# Instant Messaging Tool for Collaboration in a Peer-to-Peer Network: MyBook Instant Messenger

A thesis submitted to the
Division of Research and Advanced Studies
of University of Cincinnati

In partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE
In
COMPUTER SCIENCE

from the Department of Electrical and Computer Engineering and
Computer Science
Of the College of Engineering

February, 2005

by

Ainars Klavins

B.S, University of Latvia, Riga, Latvia, 2002

Committee Chair: Kenneth Berman, PhD
Fred Annexstein, PhD
Jerry Paul, PhD

**ABSTRACT**

The object of this thesis is to develop a collaborative tool for interactive communication and file sharing among peers in a peer-to-peer (p2p) network where collaborations are established and maintained using the MyBook Desktop System application. The collaborative tool we develop, the *MyBook Instant Messenger*, allows groups of peers to collaborate on topics of common interest and organize and compile the information they have contributed in a visually effective organizational structure called a MyBook. MyBook Instant Messenger is different from existing instant messengers in the following ways. It allows for the creation of topic-focused communities. Peers who do not know each other, but have a common interest, can find each other by searching for a MyBook on the topic of interest and then interacting with other peers in the collaboration associated with the MyBook using the MyBook Instant Messenger. The features for MyBook Instant Messenger are multi & private messaging, file sending, file logging, various text formatting, and creation of new communities of common interest topics. User authentication is performed using salted one-way MD5 hashing algorithm.

Experiments and testing for MyBook Instant Messenger were performed on test data; such as text messages and file transfer testing. Test results were mostly obtained from benchmarking as well as human factor testing.

Every task module is coded and subsequently all the modules are put together to form the final MyBook Instant Messenger application. The software package is developed using Microsoft C#.NET language and Java.

Index Terms: Collaboration among peers, Instant messaging, MyBook software

**ACKNOWLEDGEMENTS**

I would like to thank Dr. Ken Berman and Dr. Fred Annexstein for providing and inspiring research environment and for their academic and personal support. I also would like to thank my committee member Dr. Jerry Paul for his constructive comments on my thesis as well as for being helpful and understanding with all the TA work I have done, while being under his supervision. Thanks to Dr. Jerry Paul and his wife Ruta for helping me with many daily routine things, while being at University of Cincinnati in Cincinnati. This thesis would not have been written without the support and encouragement of my wife Julija and rest of my family in Latvia, as well as class- and lab mates. This medium is not sufficient to show my appreciation and I will thank them more appropriately in person.

**TABLE OF CONTENTS**

## LIST OF FIGURES

# CHAPTER 1

## 1 Introduction

### 1.1 Motivation

The object of this thesis is to develop a collaborative tool for collaboration using the MyBook Desktop System application in topic based peer-to-peer network. It would allow peers to collaborate among other peers on a topic of common interest. Peers, who would have common interests in MyBook books system, could find each other using collaborative tool - MyBook Instant Messenger, which is implemented in MyBook Desktop System. The features for MyBook Instant Messenger are multi & private messaging, file sending, file logging, various text formatting, and creation of new communities of common interest topics. Future plans for MyBook Instant Messenger would be offline messages and ability to transfer replicated files, which are located in MyBook Desktop System's peer to peer network, simply by clicking on file link in MyBook book's file. User authentication is performed using salted one-way MD5 hashing algorithm.

Experiments and testing for MyBook Instant Messenger were performed on test data, mostly obtained from benchmarking as well as human factor testing.

Every task module is coded and subsequently all the modules are put together to form the final MyBook Instant Messenger application. The software package is developed using Microsoft C#.NET language and Java.

## 1.2 Organization

The thesis is organized as follows:

Chapter 2 contains a brief overview of different collaborative projects developed throughout the world. Collaboration among people, computing power, sharing knowledge and information is essential part for success in the domains of business, sciences, engineering and various kinds of databases.

Chapter 3 takes a closer look at the features of the MyBook Instant Messenger. It also outlines approaches chosen by us for its design.

Chapter 4 describes in detail algorithms, functions and procedures of MyBook Instant Messenger that we have developed. It also looks in depth into all of the MyBook Instant Messenger's features, such as user authentication procedure, connection properties between MyBook Instant Messenger's client and servers, message exchange, file transfer protocol between users and lots more.

Chapter 5 mentions the conclusions of the completed research and outlines possible future extension of this work.

# CHAPTER 2

## 2 Background

### 2.1 Collaboration among peers

Collaboration among users now is the central technology that is facilitating and enabling this new reorganization of communication among peers. It's a computing technology that permits geographically dispersed teams to develop, edit, and use common databases, or "repositories" of information. These repositories can contain financial data, text, memos, documents, financial information, and even digital images, video and audio information. We can say that collaboration fundamentally facilitates cooperation and coordination between team members by allowing common information to be easily stored, shared, and communicated. Not so long time ago, brute force could be counted on to solve many business communication problems. You could gather all the parties in the conference room, talk out the issues and then they could return to their offices to solve less urgent problems. Unfortunately, the world has become more complex and demanding of our time [7]. Also, it's not so easy anymore to gather all the players and resources in one place to focus on the issues. Fortunately, technology has brought refinements and finesse to the communication process, allowing us to better schedule our most valuable resource – time. The traditional communication mediums, such as paper mail, e-mail, phone-mail, faxes, phones, and face-to-face conferencing have different attributes and characteristics. When rated on speed, accuracy, privacy, ease of access, ability to handle large volumes of data, cost, and support for collaboration, each scores differently. While face-to-face conferencing can achieve most communication objectives, it is just not practical, as practices grow and geography separates people and businesses.

E-mail is a simple example of two-way communication of information. The software resides on networked computers, which also handle other computer tasks such as word processing, spreadsheets, and audit applications. But does e-mail easily handle the real time information exchange and sharing of large volumes of data that are only temporarily needed?

Some well-known collaboration products are Lotus Notes by Lotus Development, TeamLinks by DEC, Cooperation by NCR, and Windows for Workgroups by Microsoft, Groove by Groove Net (Fig 2.1).

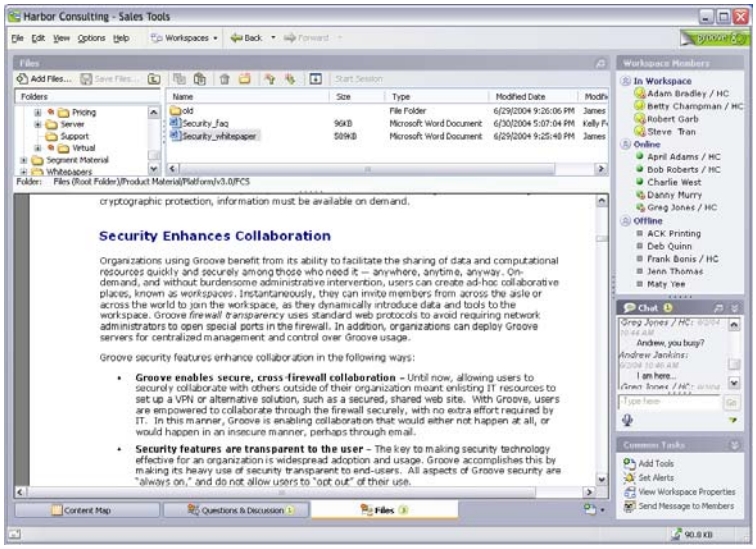


Fig. 2.1. Groove.Net workspace

All enterprise level collaborative products are based on the networked client/server model of computing. In this model some central computers serve other client computers at people's desks by providing LAN functions, such as routing e-mail, holding central data bases, and maintaining security. LAN servers are then connected via a firm's wide-area network (WAN).

Collaboration used to be simple in the early days: stock a meeting room with soda, coffee, and a whiteboard, add workers, and shake until done. No longer. Today collaboration involves not only co-workers or classmates but also members of dispersed "virtual" work teams: salespeople, distributors, retailers, suppliers, customers, professors, students and who knows whom else.

While definitions and groupings vary [7], it's generally agreed that collaboration technology includes groupware, instant messaging (IM), Web conferencing, unified messaging and communications, e-mail, and calendaring. That is, it runs the range from the non-substitutable (E-mail) to the nearly indefinable (suites of products that somehow make new ways of working together possible).

Collaboration technologies vary. IM runs like a mini-Web browser on a computer screen and lets you instantly send short text messages and data files to others equipped with compatible IM software. E-mail, of course, lets anyone send a message to anyone else with an E-mail address. Unified messaging allows users to receive multiple types of messages—including E-mail, voice mail, and faxes—from a single, universal inbox, and to send and receive messages from various devices, including office and mobile telephones, computers, and PDAs. Web conferencing lets team members, working from various locations, share documents and slides in a virtual work space accessed from a PC (via the Internet, of course), while simultaneously talking via traditional teleconference or Internet-enabled phone service (aka Voice over Internet Protocol or VoIP).

## 2.2 Collaboration - The Way We Work Now

Taken to its logical conclusion, collaborative technology could transform, not only nearly any kind of business computing application, but also ultimately the way people work.

Microsoft is adding collaborative technology to its operating systems and applications. IBM continues to drive its Lotus subsidiary into new realms of collaborative computing, expanding the Domino and Notes products to facilitate, among other things, broad shared access to data and applications. Instant messaging has exploded in popularity, and Web conferencing continues to gain ground.

With sales of collaborative-computing products reaching $3.6 billion year 2003, it's clearly a mainstream application. E-mail, the foundational collaborative application, is, by most accounts, the current "killer app"[6]. Yet collaboration technology stands at a transition point. Instant Messaging, which started out as a consumer product offered by AOL, Yahoo, and Microsoft, has found its way onto corporate desktops. Top IT vendors have noticed, and they've decided, that if E-mail and IM are good, product suites that combine them with other collaborative tools are vastly better. Yahoo was the first to offer Yahoo Messenger for corporate companies.

Microsoft has begun shipping Office Live Communications Server 2003, software that lets companies run their own enterprise IM networks, log those messages, and determine whether a user is online and available for communication via Office applications. OpenScape from Siemens combines voice, E-mail, IM, and collaboration features. Apple Computer has added video to IM with its iChat AV software and iSight digital camera.

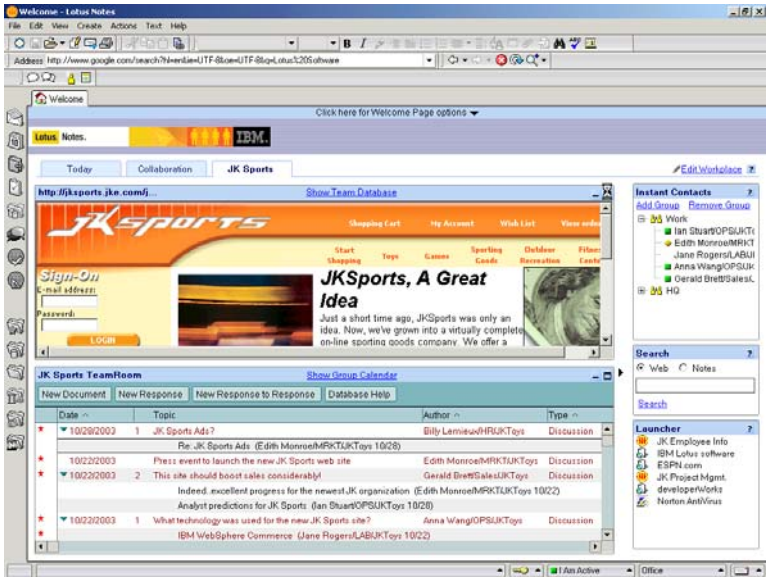


Fig. 2.2. Lotus Notes workspace

IBM, as mentioned earlier, continues to build out its Lotus Notes (Fig. 2.2.) and related products such as QuickPlace, a more function-rich "team environment." And Oracle is marketing its Collaboration Suite both on its technological capabilities and as a lower-cost option to Microsoft's dominant E-mail products.

A large group of lesser-known vendors is also offering collaboration products. Some, including Hummingbird and Open Text, have found success selling to departments of large corporations and smaller companies. Larger competitors have acquired others, including Gauss and iManage and Groove.Net. WebEx, PlaceWare, and others have made Web conferencing a $500 million business.

## 2.3 Collaboration by using Peer-to-peer applications

The P2P space can be classified into three categories based on the anatomy of the network and application [6]. These are collaborative computing, instant messaging and affinity groups. Collaborative computing is also referred to as distributed computing and its biggest and most successful example is the Seti@Home project. Another is Fight AIDS at Home from Entropia. We all know about instant messaging and regularly use MSN, ICQ and Yahoo instant messengers. The hottest P2P applications in use today are the affinity groups like (erstwhile) Napster, Kaaza (Fig. 2.3.) and Gnutella file-sharing networks. They are extensively used for music, movies and software sharing. Then there are the less well-known workspace-sharing applications, like Groove.NET, which also fall in this category.

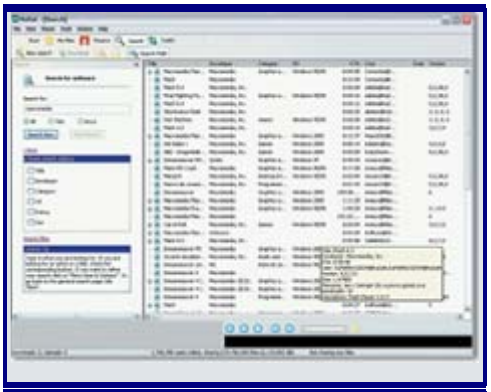

Fig. 2.3. Kazaa is one of the hottest P2P apps in use today

While on one side, P2P technology puts unused and idle resources to constructive use, problems abound about the usage of such a network to trade software and multimedia content. In a corporate environment, the problem with implementing it is the lack of central control. Plus, there can also be security issues when individual desktops directly connect to each other.

## 2.4 Instant Messaging as a collaboration tool

Instant messaging is another widely used P2P application—MSN, AOL, ICQ, Yahoo and others — that we are using everyday. These messengers work in coordination with a central server. The server is, however, used to perform the task of authenticating a client to the network, storing buddy lists and making the first contact between two clients. Once a connection is established, two client applications talk to each other directly, though you also have the option of using the server to send messages. Instant messengers come with a lot of features that we overlook. We think of them as only text-messaging windows.

Take MSN Messenger. With it you can do message conferencing, application sharing, white boarding, transfer of files. It even allows you to make phone calls. Application sharing allows both interacting parties to work on an application together. White-boarding works like a virtual white board on which the interacting parties can write or draw.

There are also private messengers. They consist of a server and a client part that a corporate can buy and install for its network. This can then be used for similar functions as public messengers within the organization. These private messengers offer better security and encryption features, apart from local manageability.

# CHAPTER 3

## 3 Motivation

### 3.1 Mybook Instant Messenger – collaboration tool for MyBook project

Let's start description of MyBook Instant Messenger with main Authorization window (Fig. 3.1.).


Fig. 3.1. Auhorization Window for MyBook Instant Messenger

It allows non-registered user to register in a MyBook Instant Messenger system and then he/she can log on to MyBook Instant Messenger. User can join chat and use instant messenger to collaborate with other MyBook application users. If user is already registered in a system, he/she can use MyBook Instant Messenger immediately by entering user name and password and if needed – community and its password. Authorization window contains New

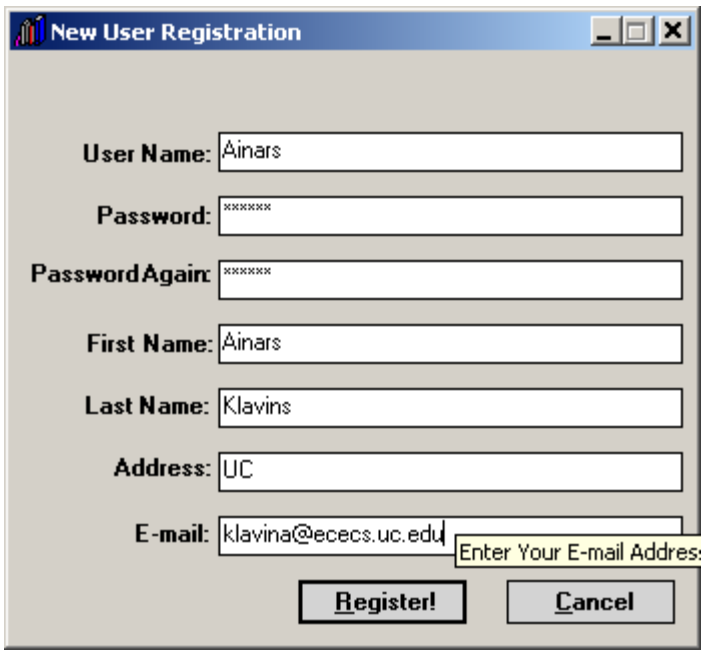


Fig. 3.2. New user registration window for MyBook Instant Messenger

Community button. We will talk about it later. Fig. 3.2 shows us New User Registration Window. User has to enter user name, password, first name, last name, address and e-mail to be able to get access to MyBook instant messaging resources.

Figure. 3.3 shows us MyBook Instant Messenger main window. After user has successfully logged on system, he/she joins other users, who use MyBook Instant Messenger at the same time. Right window contains list of users who have joined particular community. Left upper window is Multi Chat window, where all the users can exchange with information and see everything, what other users say. Left bottom window is text box, where user enters his text. Between Multi Chat window and text window there is formatting bar. It lets user to choose different font style, font size, font color and bold, italic or underlined fonts.

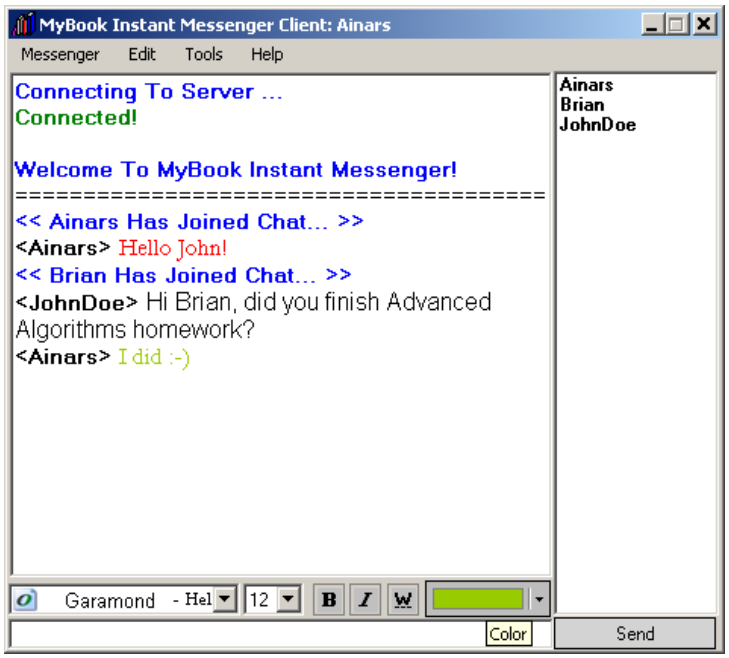


Fig. 3.3. Main window for MyBook Instant Messenger

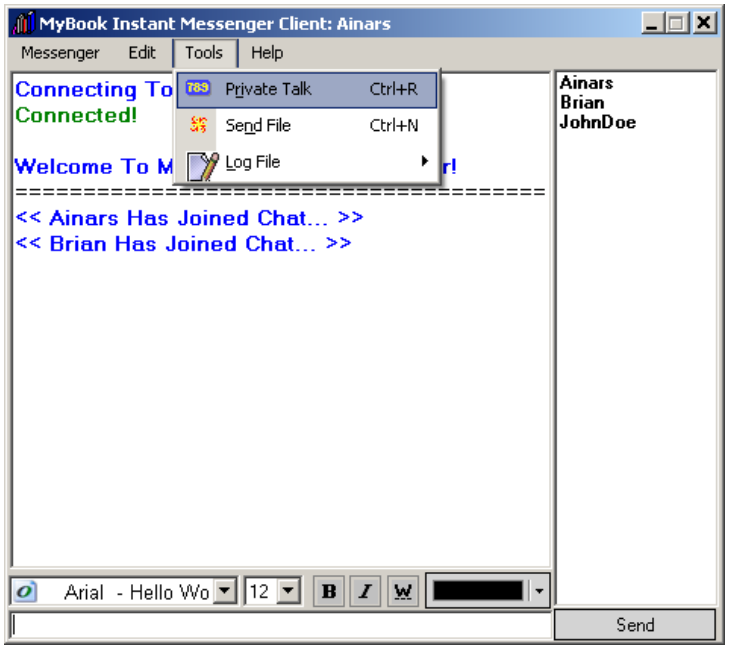


Fig. 3.4. MyBook Instant Messenger features – private talk, file sending and file logging possibilities

MyBook Instant Messenger main window has Tools menu (Fig. 3.4) with 3 available features – Private Talk, Send File and

File Logging. Those features allow MyBook IM users exchange with files, talk privately just with one person on the other end and save log file for later use, in case information was important and user will need it again some day later.

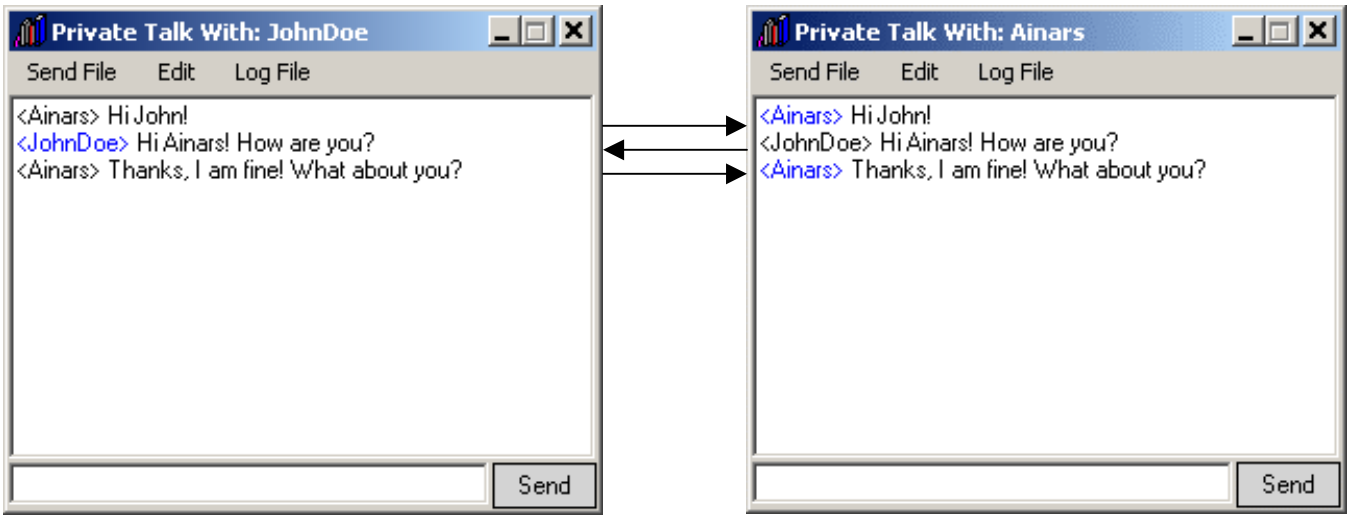


Fig. 3.5. MyBook Instant Messenger Private Talk windows – users can exchange information in private and secure way.

Private Talk feature (Fig. 3.5.) allows users to exchange with information in a private and secure manner. User can double click on other user, he/she wants to talk to, choose user from context menu (right mouse click on user and choose entry – Private Talk) or go to Tools/Private Talk. Opens new Private Talk window and two users can exchange information privately.

Private Talk window contains couple useful features, which MyBook Instant Messenger users can use, for example, file sending to each other as well as file logging possibilities.

Additional feature is http links recognition. If text starts with “http://…”, MyBook Instant Messenger application recognizes it as a valid URL. It becomes blue underlined text with possibility to click on link. Internet browser will be triggered and clicked link will be open in it.

Every user can talk privately to as many users as there are online at the present moment. For each user we can create log file, so it will be easier to follow conversation with every user.

## 3.2 Work sequence of File Send feature

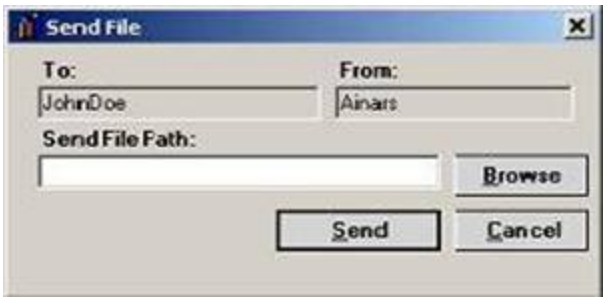


Fig. 3.6 User-sender clicks *Browse* button, which opens *Open File Dialog* window

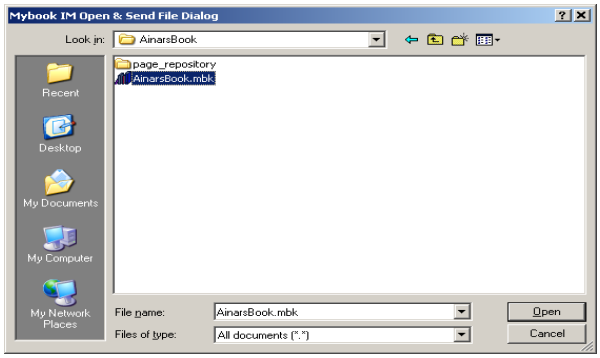


Fig. 3.7. User-sender chooses file and clicks *Open.*

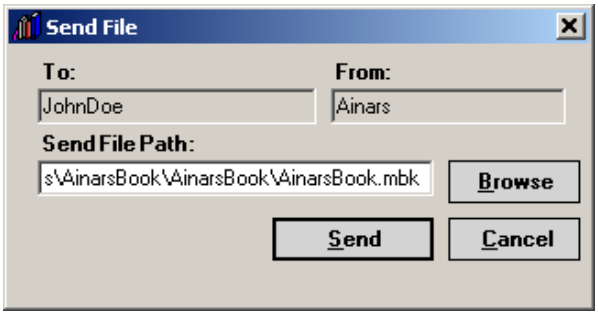


Fig. 3.8. User-sender cliks Send to notify receiver about sent file, he has to receive

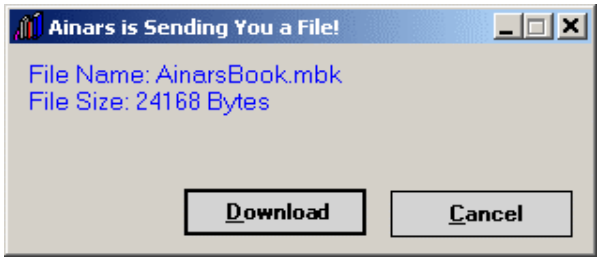


Fig. 3.9. User-receiver, clicks *Download*, *Save File Dialog* window opens

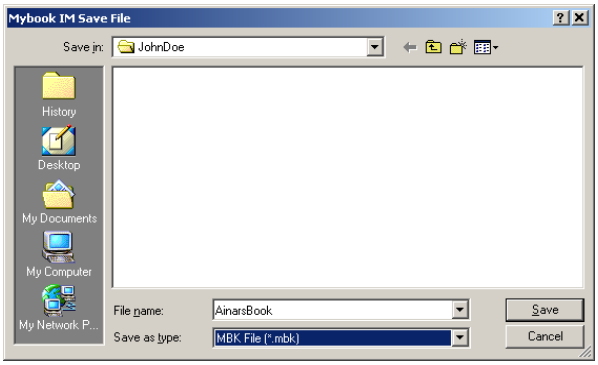


Fig. 3.10. User-receiver chooses location and clicks Save to save file

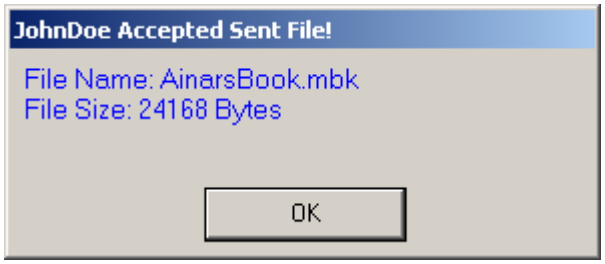


Fig. 3.11. User-sender receives approval, that User-receiver wants the file, and clicks *Ok.*

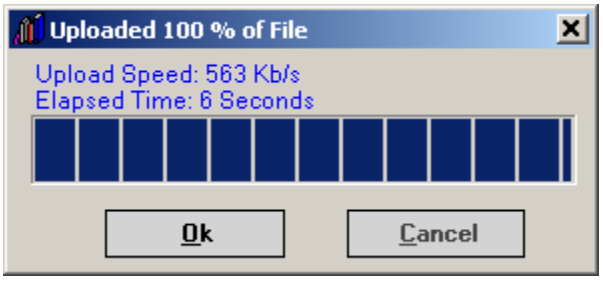


Fig. 3.12. User-sender gets progress bar, which shows progress of uploaded file. File sending is done.

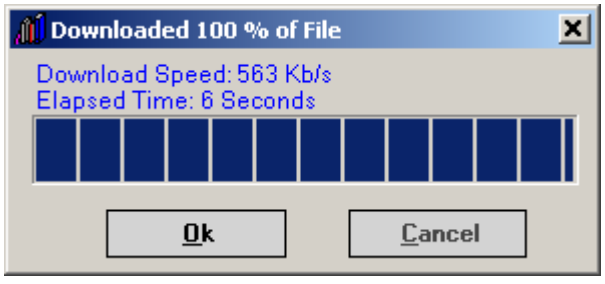


Fig. 3.13. User-receiver gets progress bar, which shows progress of downloaded file. File sending is done.

As we talked before, Private Talk window has log file feature (Fig. 3.14.), which is useful, if user wants to save his private conversation for future reference. *Log File* menu contains three submenu records: *Save Log File*, *Open Log File* and *Clear Log File*. Save Log File saves text, which is located on *Private Talk* window, into a separate for each user [UserName] _Log.txt file. For example, User name we are talking to is JohnDoe, so log file's name will be JohnDoe_Log.txt.

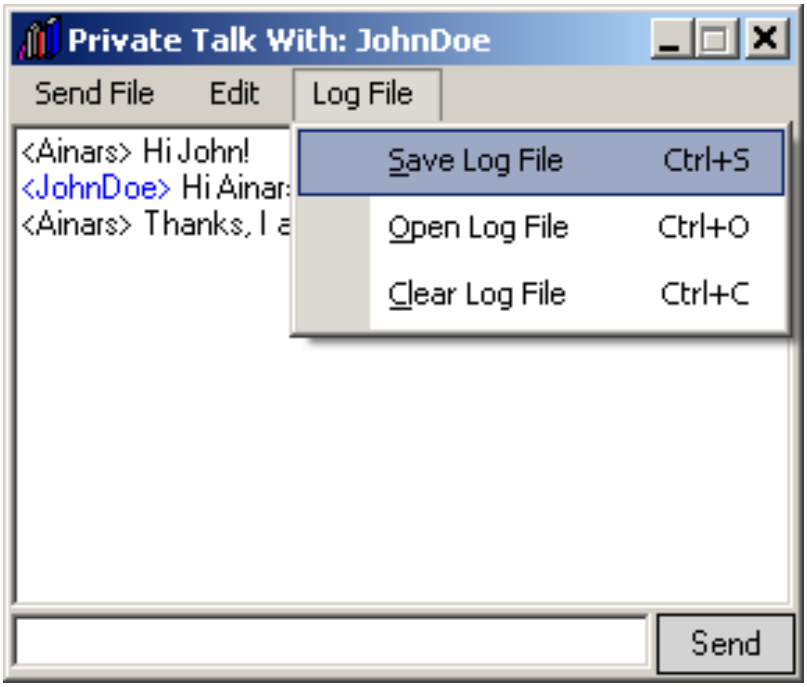


Fig. 3.14. Private Talk window with Send File and Log File features.

*Open Log File* opens log file, if it exists or if it doesn't exist, it creates new, empty log text file and opens it.

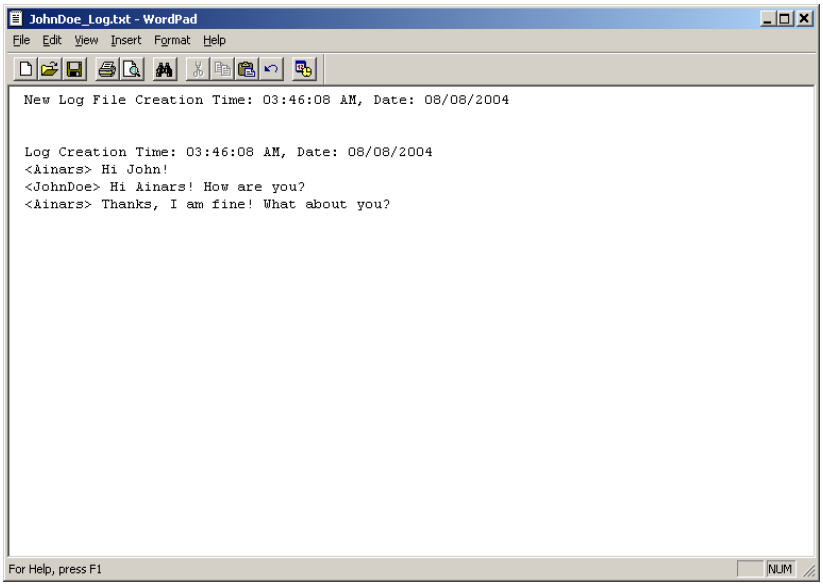


Fig. 3.15. Log file. Available for multichat and for private chat.

*Clear Log File* clears existing log file, so it is empty.

Multi Chat log file is only one and is called "MyLog.txt". Every conversation is saved there, because it doesn't require any separate text file.

Figure 3.15. shows saved log file, which is opened in Microsoft Wordpad application. First line shows, when log text file was created, date and time, then it shows each log file's saved contents.

*New Community Registration* window (Fig. 3.16.) helps MyBook application users to create new communities. Idea behind it is following: MyBook application user decides to create a new community with specific topic of interest. User fills out *New Community Registration* form and waits for MyBook Instant Messenger administrator response. If administrator approves new community, it will be added and be chosen in Authorization window communities list (Fig. 3.17). Later on all the MyBook application users who are interested in taking part of newly created community, can contact owner (creator of new community) and get permission to join that particular community. So, in such way, all the users with common interest may join same community channel, discuss topic of interest and exchange with information As we all know, sharing the knowledge among group of users is very important, because people teach each other and learn at the


Fig. 3.16. New Community Registration window for MyBook Instant Messenger.

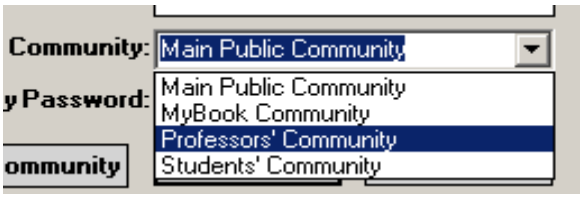


Fig. 3.17. List of available communities, user can choose, on MyBook IM Authorization window.

same time so it is good for everybody. Sharing with knowledge and information is on of the most important aims and motivations in developing MyBook Instant Messenger for MyBook application in topic based peer-2-peer network.

To be able to connect to some particular community, user will need password for it. This extra security is needed to prevent unwanted users to join every community and try to create chaos there. So only serious users who really are interested in joining some community will spend his/her time and efforts to join it.

Additional MyBook Instant Messenger features include web links recognition, blinking Private Talk windows (with additional beeeping sound), if they are minimized or inactive and somebody has written a new message to you.

If user has too long nick name and it doesn't fit in users' list widow, tooltip shows up and shows full nick name for that particular user, while moving mouse cursor over that user.

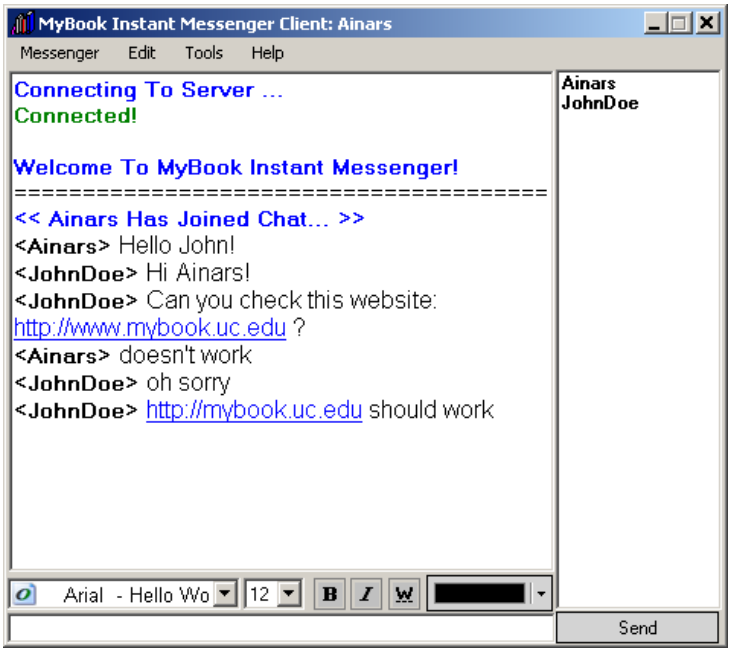


Fig. 3.18. MyBook Instant Messenger recognizes weblinks

## 3.3 Different approach for MyBook Instant Messenger

MyBook Instant Messenger is different from existing instant messengers in the following ways. The most significant difference is that it allows for the creation of topic-focused communities. For instance, we take an ICQ instant messenger as a comparison basis, which is one of the first instant messengers for public use. Users who wanted to join ICQ network, were assigned a unique user ID, which consisted of many digits. In such way, it was a small drawback for this application, because it is lot easier to remember user name consisting of name and not composition of figures. Latest version allows you to use your email as ID or same old ICQ number as well.

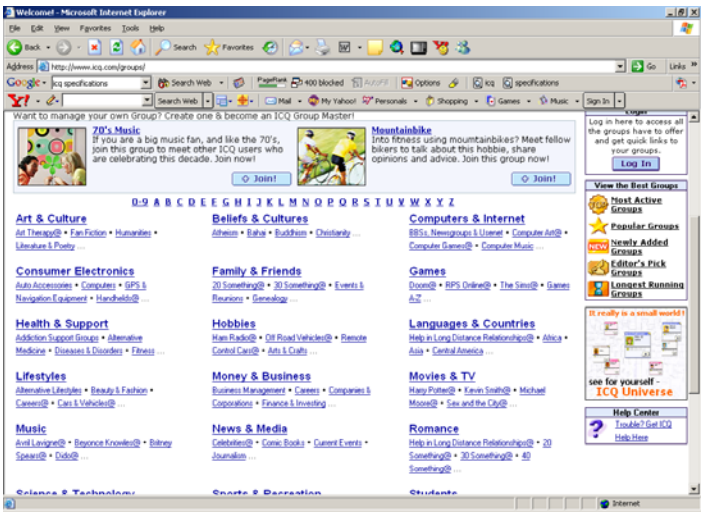


Fig. 3.19 ICQ predetermined communities.

*New Community Registration* window (Fig. 3.16.) helps MyBook application users to create new communities. Idea behind it is as follows: MyBook application user decides to create a new community with specific topic of interest. User fills out *New Community Registration* form and waits for MyBook Instant Messenger administrator response. If administrator approves new community, it will be added and chosen in Authorization window communities list (Fig. 3.17). Later on, all the MyBook application users who are interested in taking part of newly created community, can contact owner (creator of new community) and get permission to join that particular community. So, in such way, all the

users with common interest may join the same community channel, discuss topic of common interest and exchange with information.

To compare MyBook Instant Messenger to previously mentioned ICQ instant messenger, we go one step further and allow people to create their own communities with very narrow topics of interest, so every person would find something valuable, which interests only him/her. Otherwise user has right to create his/her own community and be the owner of it. ICQ has community groups, but they consist of very wide topics of interest, for example - computer music, Sex and City episodes, Linux etc. Additionally, users themselves do not create topics of interest. ICQ adds most interesting and most popular topics of interest to capture people's interest using those widely popular topics.

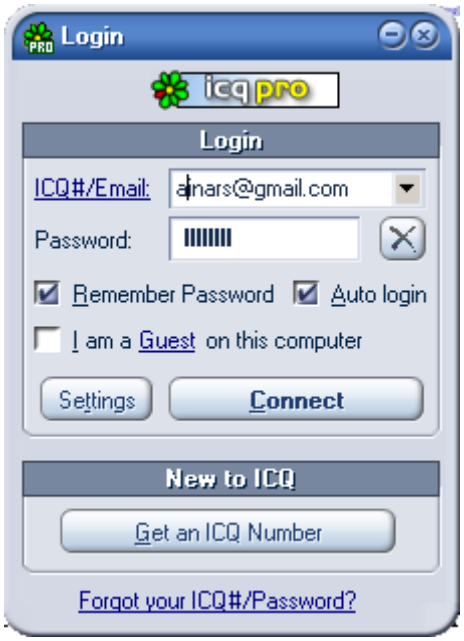


Fig. 3.20 ICQ instant messenger

MyBook Instant Messenger on the other hand allows user to create its own topic of interest, which could be very, very narrow, for example – "Discussion of 1st chapter of Sequential and Parallel Algorithms textbook by Ken Berman and Jerry Paul". In such case main user auditorium would consist of students, who take Advanced Algorithms course at the university. And it wouldn't consist of one university only, because everywhere, where this book would be used as a textbook for particular Advanced Algorithms course, students would be interested in sharing and receiving knowledge form other students.

As we all know, sharing the knowledge among the group of users of common interest is very important, because people teach each other and learn at the same time so it is good for everybody. Everybody wins. Sharing knowledge and information is on of the most important aims and motivations in developing MyBook Instant Messenger for MyBook application in topic based peer-2-peer network.

To be able to connect to some particular community, MyBook instant messenger user will need password for it. This extra security feature is added to be able to prevent unwanted users to join every community and to create chaos there. In such case only serious users who are really interested in joining some community will spend his/her time and efforts to join it.

# CHAPTER 4

## 4 MyBook Instant Messenger tehnical overview

### 4.1 Verification of Login & Passwords using one-way MD5 hashing algorithm

"An algorithm created in 1991 by Professor Ronald Rivest that is used to create digital signatures. It is intended for use with 32 bit machines and is safer than the MD4 algorithm, which has been broken. MD5 is a one-way hash function, meaning that it takes a message and converts it into a fixed string of digits, also called a message digest.

When using a one-way hash, one can compare a calculated message digest against the message digest that is decrypted with a public key to verify that the message hasn't been tampered with. This comparison is called a "hashcheck"." [7]

We go further by "salting" the password fields before hashing them. This is done by adding a second piece of information to the hash that is non-changing and unique for every user – in our case - the username.

We will explain how authorization of MyBook Instant Messenger is technically executed (Figure 4.1.). Using one-way salted MD5 hashing algorithm encrypts Username and password. Username "Ainars" will look something like "@$##@k" and password "ABC123" – "#f5%*)$$". Server has record in its database about username "Ainars" and its encrypted username: "@$##@k" and password "ABC123" - "#f5%*)$$". If sent username and password match, SQL server grants permission to use MyBook Instant Messenger resources, otherwise user doesn't have granted access. Passwords may not be equal because of salted hashing property – we add username (which is unique) to password and only then encrypt it.

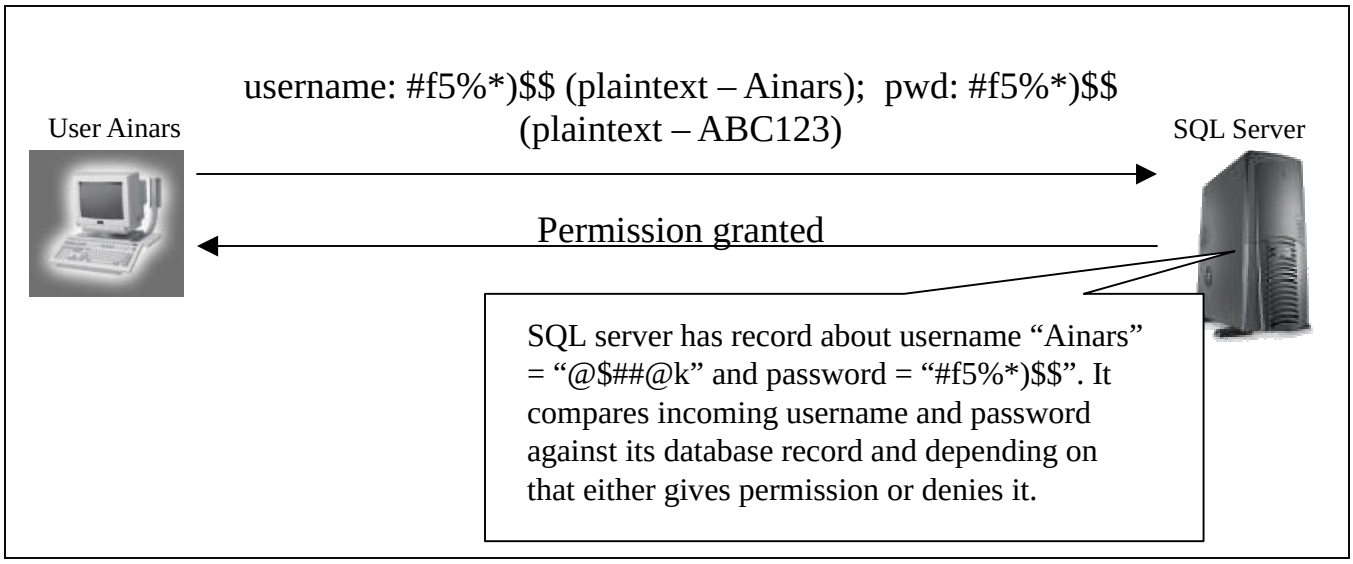


Fig. 4.1 User login name and password authorization scheme.

## 4.2 Multi Chat message exchange between MyBook Instant Messenger client and server

First of all we'll look into structure of message exchange between MyBook Instant Messenger client and server. When client sends message to server, it is formed

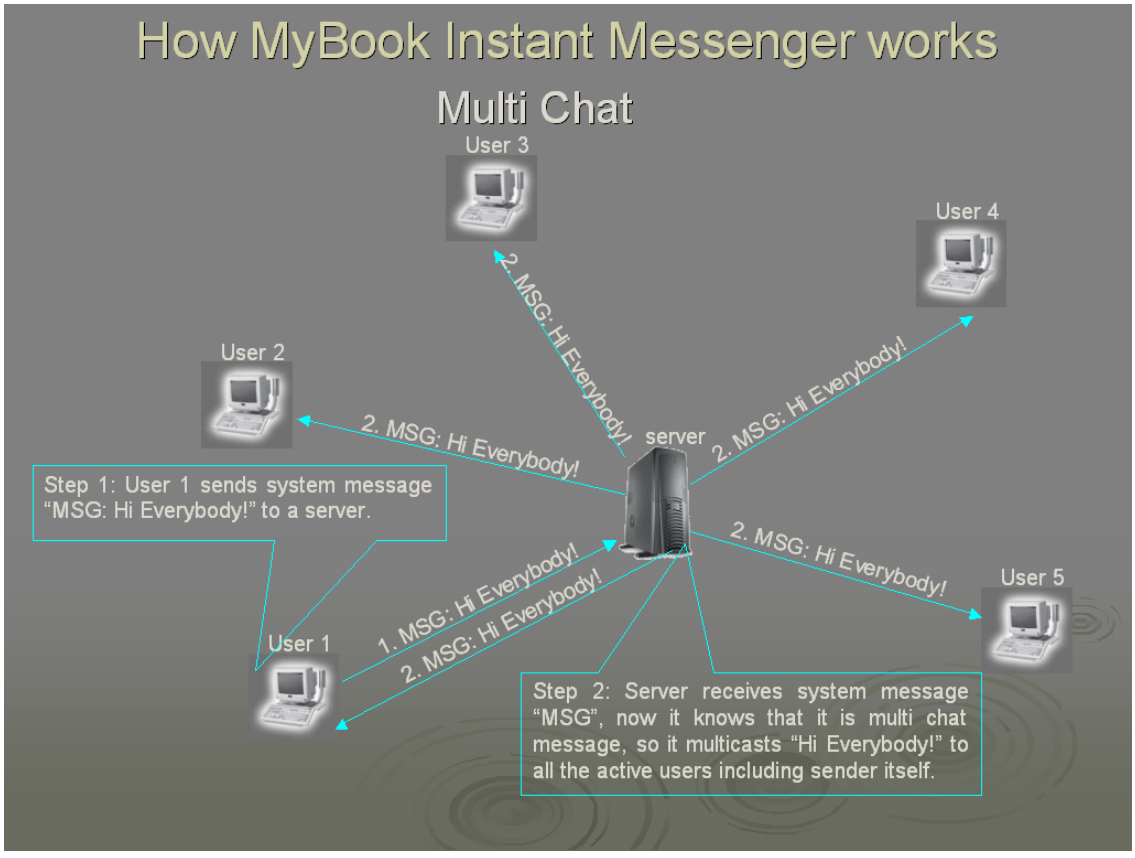


Fig. 4.2 MyBook IM Multi Chat operation scheme.

something like this: “*command[0] + ":" + client.NickName + ":" + command[2]*”. It might look

different, depending on what kind of message we want to send to the server. The most important part of that message is first part of it. In our case - *command[0]*. It is very first part, and server has to know what kind of message user is sending to it.

When we look deeper into Multi Chat message exchange (Figure 4.2), we can see how it works. Let’s say, User 1 (in Figure 4.2) sends message “Hi Everybody!” to all the users, who are online. MyBook Instant Messenger server side receives message in a format, which look like this: “MSG: Hi Everybody!”. Server first sees first part of message “MSG”, until the seperator symbol – colon. Server knows that it has to send a message to everybody, even sender himself,  and does so. It sends second part of message after the seperator colon – “Hi Everybody!”.

## 4.3 Private Talk message exchange between MyBook Instant Messenger client and server

Now we look deeper into Private Talk message exchange (Figure 4.3.), we can see how it works. It is similar to Multi Chat message exchange with some changes. Let's say, User Ainars (in Figure 4.3.) sends message "Hi John!" to a user John. MyBook Instant Messenger server side receives message in a format, which look like this: "PRIVMSG: John: Hi John!". Server first sees first part of message "PRIVMSG", until the seperator symbol – colon. Server knows that it has to send a private message, then it looks further and knows that message is meant for user John. It sends the last part of the message after the seperator colon – "Hi John!".

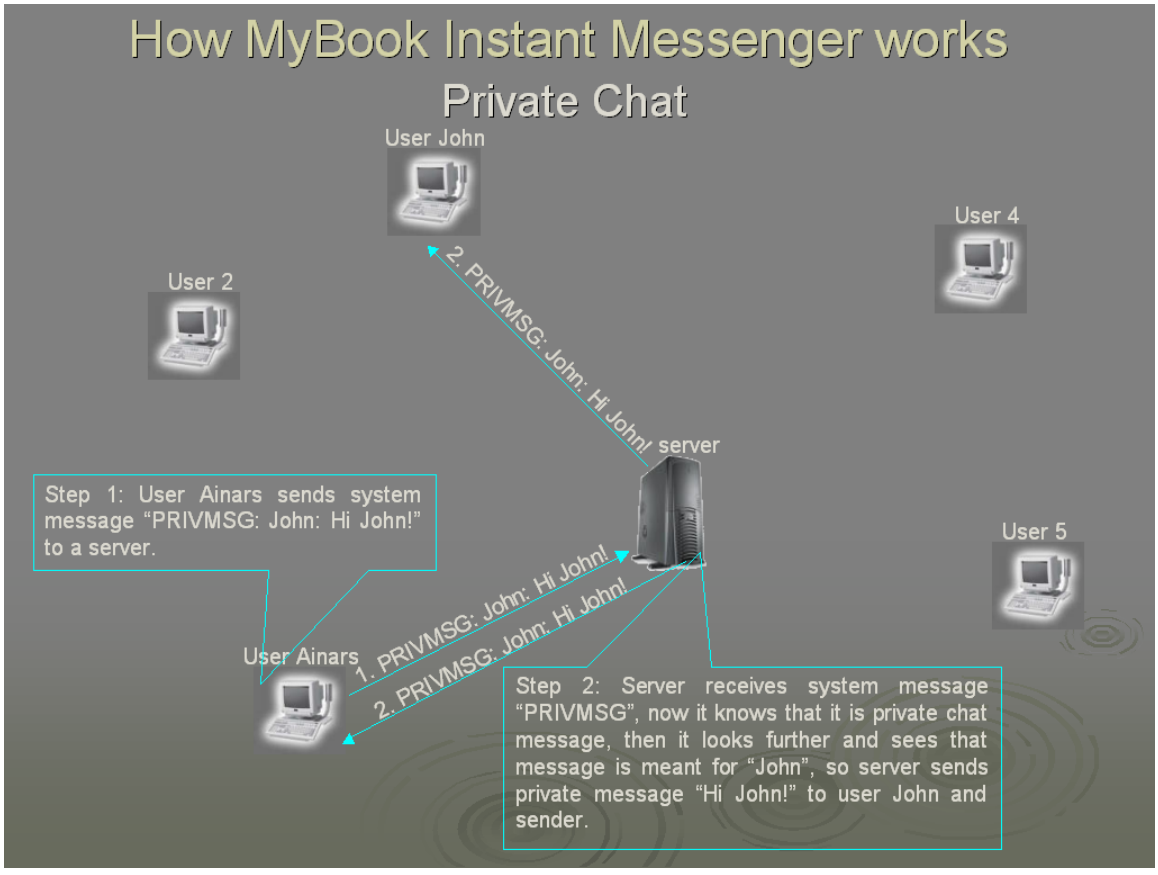


Fig. 4.3. MyBook IM Private Talk operation scheme.

## 4.4 File Sending scheme among MyBook Instant Messenger clients and server

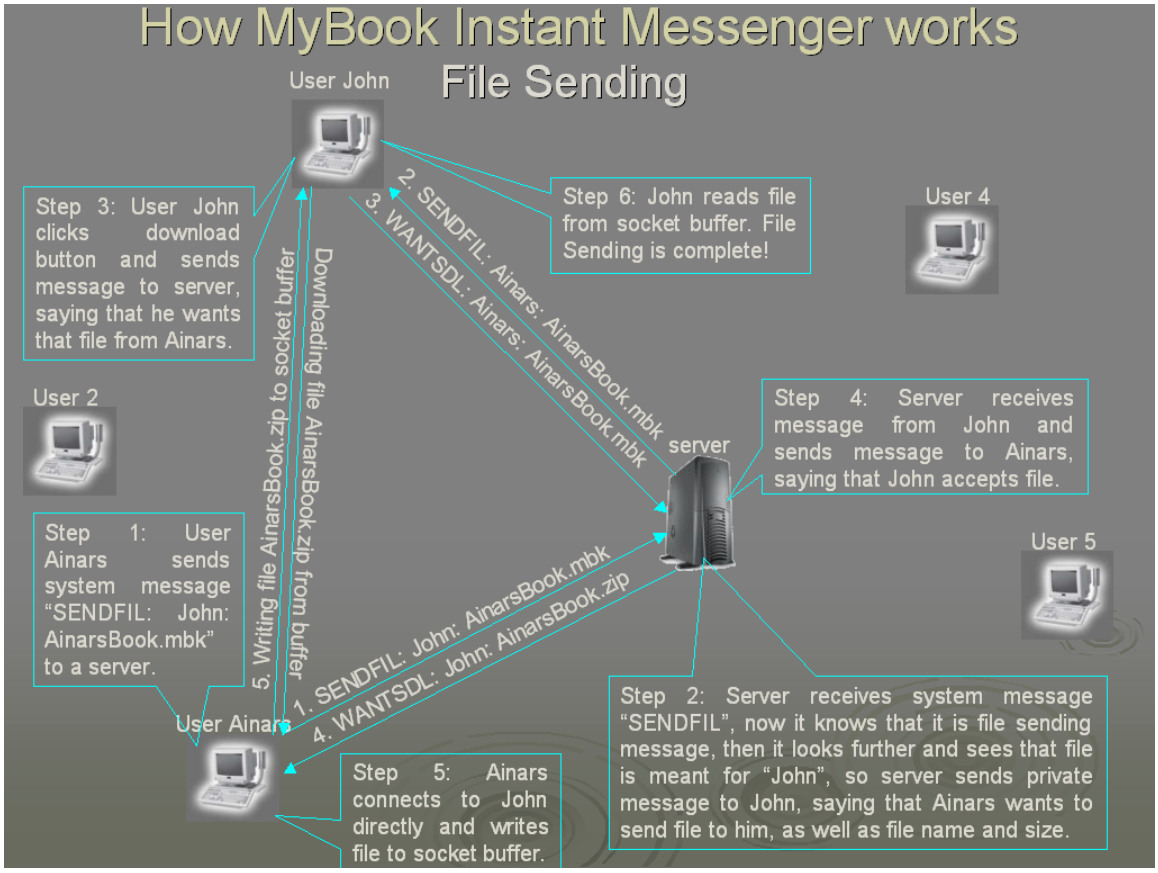


Fig. 4.4. MyBook IM File Sending operation scheme.

File Sending is very important function for MyBook Instant Messenger. It allows users to exchange not only with textual information, we've discussed previously – Multi Chat and Privat Talk, but also exchange with different types of files. It is very essential in a virtual world of collaboration. File Sendig (Figure 4.4.) has six stages, which have to complete to be able to finish file sending from one user to an other. First step involves user-sender (in our example it'll be *User Ainars*), who sends formatted message (i.e. "*SENDFIL: John: AinarsBook.mbk*") along with additional information to user-receiver (*User John*). Second step: MyBook Instant Messenger server receives this message, it checks for first part of message and knows that *User Ainars* sends file. It checks second

part of message, now knows – this file is meant for *User John*, checks last part of message and finds out the name of the file – *AinarsBook.mbk*. Now it forwards this message to *User John*. Step three: *User John* approves, that he wants this file and sends message "*WANTSDL: Ainars: AinarsBook.mbk*" back to *User Ainars* through server. Server now checks message and knows, that *User Ainars* has to receive it. Step four involves server forwarding message "*WANTSDL: John: AinarsBook.mbk*" along with additional information (IP address) to *User Ainars*, so he knows, that *User John* wants to download file *AinarsBook.mbk*. Step five: *User Ainars* connects to *User John's* computer directly, using IP address and pre-defined port number and writes file to the socket buffer. Step six: *User John* reads file from the socket buffer and saves it onto his computer's hard drive. Now file sending is complete.

If for any reason, file transfer is disconnected by any of users, network failure, power failure or any other reason, file transfer will be stopped and both users will be informed about file sending failure. By having this information they can transfer needed file(-s) to each other at any other time.

## 4.5 File Sending scheme among MyBook Instant Messenger clients and server (using DHQ[1] tunneling technique)

DHQ tunneling technique is used to get around firewalls and send files through them, using tunneling technique. Files are sent through server and not directly as in Figure 22.

File Sendig using tunneling technique (Figure 4.5.) has six stages, which have to complete to be able to finish file sending using DHQ from one user to an other. First step

[1] DHQ – Distributed Hash Queues by Chad Yoshikawa. (http://www.ececs.uc.edu/~yoshikco/dhq/distributed_hash_queues.htm)

involves user-sender (in our example it'll be *User Ainars*), who sends formatted message (i.e. "*SENDFIL: John: AinarsBook.mbk*") along with additional information to user-receiver (*User John*). Second step: MyBook Instant Messenger server receives this message, it checks for first part of message and knows that *User Ainars* sends file. It checks second part of message, now knows – this file is meant for *User John*, checks last part of message and finds out the name of the file – *AinarsBook.mbk*. Now it forwards this message to *User John*. Step three: *User John* approves, that he wants file and sends message "*WANTSDL: Ainars: AinarsBook.mbk*" back to *User Ainars* through server. Server now checks message and knows, that *User Ainars* has to receive it. Step four involves server forwarding message "*WANTSDL: John: AinarsBook.mbk*" along with additional information (IP address) to *User Ainars*, so he knows, that *User John* wants to download file *AinarsBook.mbk*. Step five: *User Ainars* connects to *User John's* computer directly, using IP address and pre-defined port number and tries to write file to the socket buffer.. but it cannot complete this step, because *User John* is behind the firewall. This is where Distributed Has Queues come into play. Now *User Ainars* sends link *http://localhost:8081/Ainars/AinarsBook.mbk* to a *User John*. Step six: *User John* then clicks on a link and downloads file from *User Ainars's* file repository using DHQ tunneling technique through DHQ server. Now file sending using DHQ is complete.

If for any reason, file transfer is disconnected by any of users, network failure, power failure or any other reason, file transfer will be stopped and both users will be informed about file sending failure. By having this information they can transfer needed file(-s) to each other at any other time.

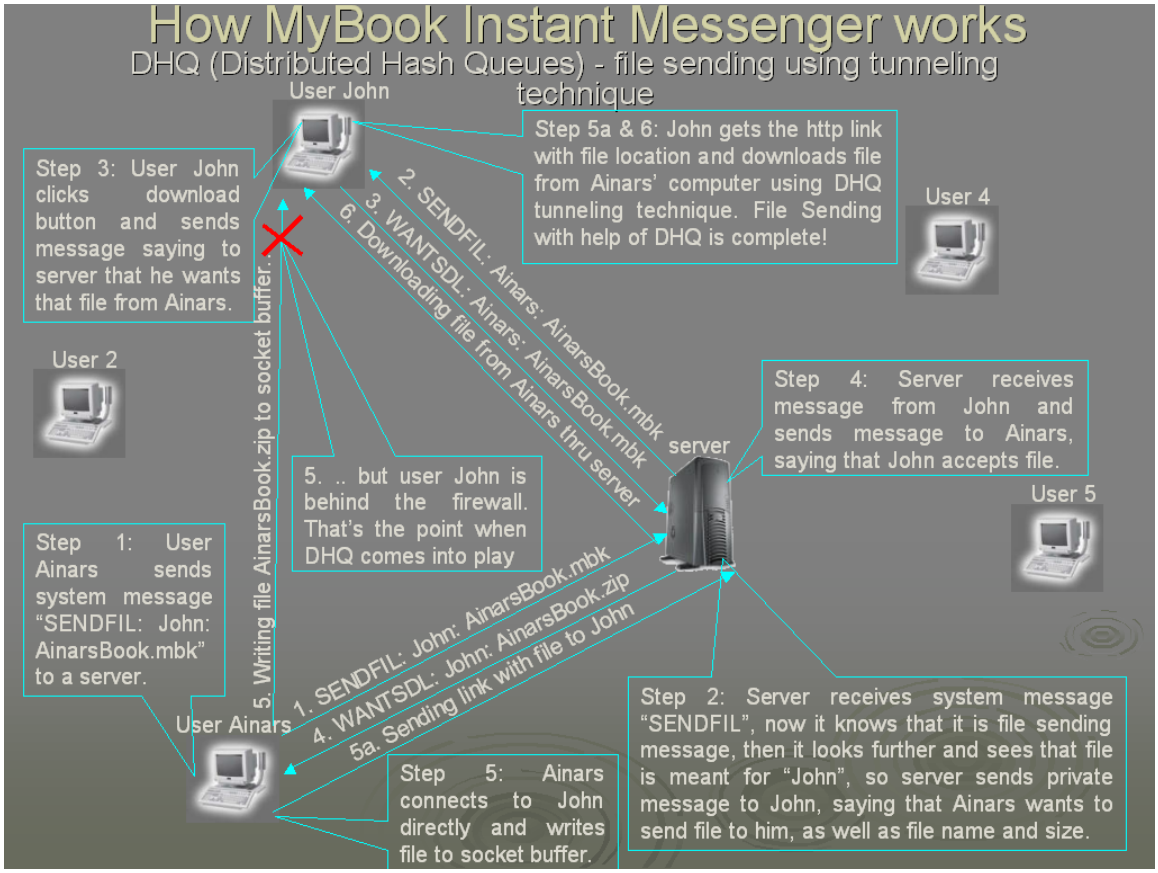


Fig. 4.5. MyBook IM File Sending (using DHQ) operation scheme.

## 4.6 Ability to create new communities and connect to them

One of important aspects for MyBook Instant Messenger is creation of new communities for different interest groups. Idea behind it is as follows - MyBook application user decides to create a new community with specific topic of interest. User fills out *New Community Registration* form and waits for MyBook Instant Messenger administrator response for approval, as we've discussed previously. Then all the users with common interest may join the same community channel, discuss topic of interest and exchange with information. Figure 4.6. shows simple sample of how communities model is working – two users, who join the same community, can talk to each other either in Multi Chat room or Private Talk window. Because every community has its own port assigned to it, no other users from other communities can talk to users from different

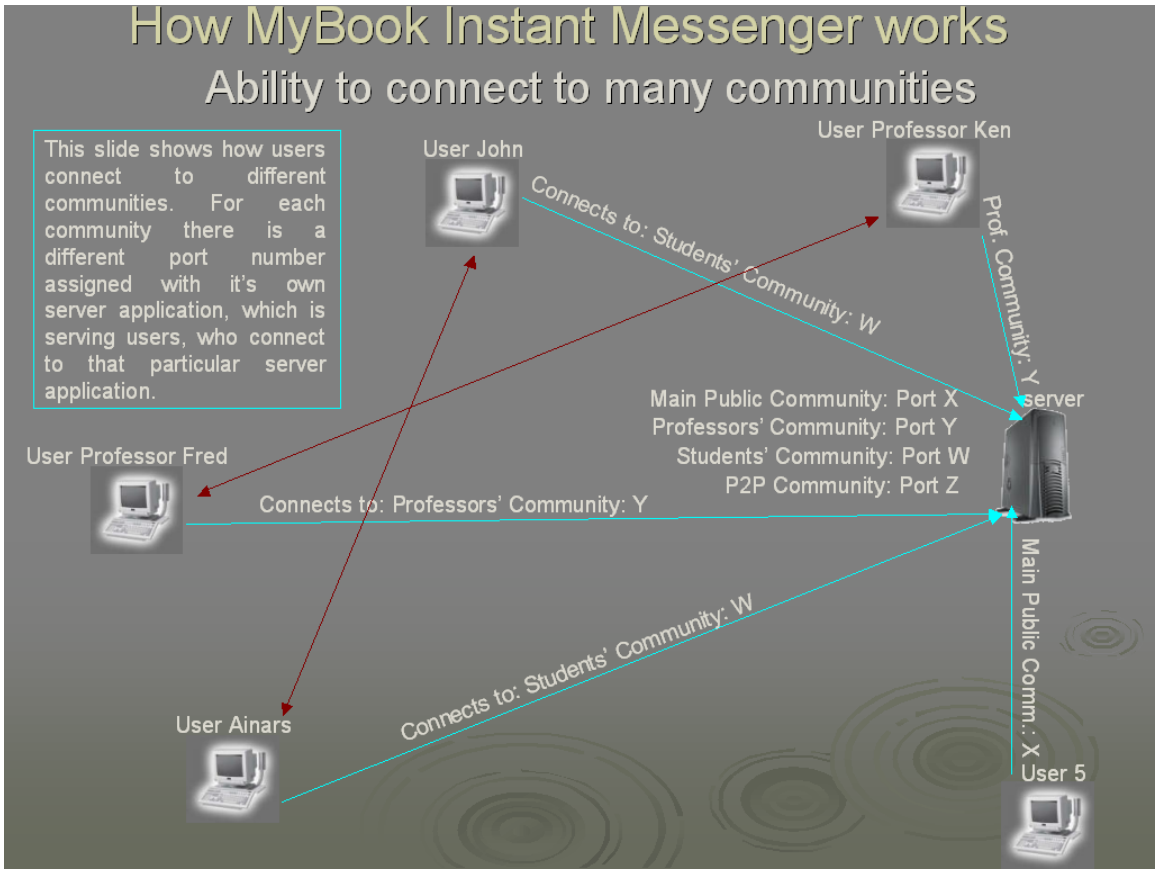


Fig. 4.6. MyBook IM Communities' operation scheme.

community. For example, User *Ainars* and User *John* connect to Students' Community, which has port number *W*[2], users *Ken* and *Fred* connect to Professors' Community. Users *Ainars* and *John* can't meet users *Ken* and *Fred* in any way, because their communities are different and each community has different port number assigned, so it means, every community has its own server, where users log onto.

Only exception, where users don't need any permission to join community, is Main Public Community, where everyone can log on and share information and knowledge, simply using their regular user name and password.

[2] X, Y, W and Z ports (Figure 23) are variables from 1-65536

## 4.7 Microsoft SQL Server deployment for user registration and authorization

### 4.7.1 Registration

MyBook Instant Mesenger uses Microsoft SQL Server to complete users registration and authorization. In order to access MyBook Instant Messenger resources, user has to go through registration process which is user friendly for end user, although if we look deeper into registration process, it isn't that simple. User first has to fill out New User Registration form. Programming code behind it checks if all the required fields are filled out, chosen password is not too similar to newly chosen user name, given email is valid (*somebody@somecompany.com* and not *somebody.some,com* or any other text), etc. When user clicks *Register!* button, user name, password and other sensitive information are encrypted with one-way "salted" MD5 algorithm (we will discuss it more in next topic), before sending it over the internet to Microsoft SQL server. When that information reaches SQL server, it saves it in MyBook Instant Messenger database account.

Now we will take a deeper look into, how encrypting and storing of this important MyBook Instant Messenger information is working. The following C#.NET code provides a sample for creating a user account. User enters his/her username and password and stores these values into a UserAccount database table. However, instead of storing the password as-entered by the user, it first encrypts it using the MD5 code and then saves to the database this encrypted version.

```
//Create connection string connect to SQL server //and save all data in database
string strDSN = "Data Source=1111.22.333.44;initial catalog=Authortize;user
iD=User;password=Password;";
//Create a command query
string strSQL = "INSERT INTO MybookIMUsers (User_Name, Passw, First_Name,
Last_Name, Address, Email) VALUES
```

```
('"+UserNameTextBox.Text+"',@Password,"'+FirstNameTextBox.Text+"',
"'+LastNameTextBox.Text+"', "'+AddressTextbox.Text+"', "'+EmailTextBox.Text+"')";
//Create Objects of SQLConnection and SQLCommand
SqlConnection myConn = new SqlConnection(strDSN);
SqlCommand myCmd = new SqlCommand( strSQL, myConn );
//The array of bytes that will contain the encrypted value of strPlainText
byte[] hashedDataBytes;
//The encoder class used to convert strPlainText to an array of bytes
UTF8Encoding encoder = new UTF8Encoding(false);
//Create an instance of the MD5CryptoServiceProvider class
MD5CryptoServiceProvider md5Hasher = new MD5CryptoServiceProvider();
//Call ComputeHash, passing in the plain-text string as an array of bytes
//The return value is the encrypted value, as an array of bytes
hashedDataBytes = md5Hasher.ComputeHash(encoder.GetBytes(PasswTextBox.Text +
UserNameTextBox.Text));
//Create parameters here
SqlParameter paramPwd;
paramPwd = new SqlParameter("@Password", SqlDbType.Binary, 16);
paramPwd.Value = hashedDataBytes;
myCmd.Parameters.Add(paramPwd);
//Insert the records into the database
myConn.Open();
myCmd.ExecuteNonQuery();
myConn.Close();
```

Figure 4.7 shows the values in the UserAuthorization table after some users have been created. Note that the password contains a 16-element binary array, representing the encrypted password. If hacker was able to break into the

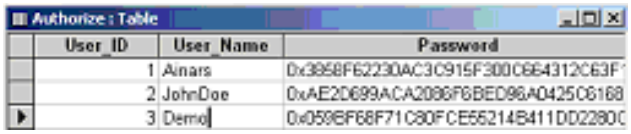


Fig. 4.7. MyBook IM UserAuthorization table with encrypted 16-element binary array passwords.

UserAuthorization table, he couldn’t figure out the plain-text password of any of the users.

### 4.7.2 Authorization

MyBook Instant Messenger user authorization is designed and implemented in a way that we create user accounts by creating a database table named UserAuthorization, with fields like User_Name and Password. This table contains a row for each user; when a users wish to log on, they submit their username and password, which would be used to query the UserAuthorization to see if there was a row whose User_Name and Password fields are matching user supplied username and password. This approach is quite simple and typical, but one of its downsides is that each user's password is stored in an unencrypted form in the database. That means that if hacker compromises our MyBook Instant Messenger database he’ll have access to the passwords for every user. That’s why we’ll use MD5 algorithm to encrypt our passwords so that even the database administrator won’t be able to determine a user's password.

## 4.8 A Brief Summary for MD5 Encryption

There are two general classes of encryption: one-way encryption and two-way encryption. Two-way encryption is the most common form of encryption. It takes a plain-text input and encrypts it into some kind of encrypted text. Later on, this encrypted text will be decrypted, which will result in the plain text, that was originally encrypted. Two-way encryption is useful for private communications. For example, let’s say you wanted to send an online shop website your credit card number to make a purchase. Of course, you wouldn't want to have your credit card numbers sent over the Internet in plain-text, because hacker monitoring the Internet might see your credit card information fly by.

Rather, you would want to send your credit card information in an encrypted message form. When this encrypted message reaches online shop web-server, it will be decrypted, which will result in the real credit card numbers.

One-way encryption, on the other hand, only allows for a plain-text input to be encrypted. It means, there is no way to decrypt the encrypted data. At first it may seem that such an encryption scheme is not needed - after all, why would you only want to be able to encrypt data and not decrypt it? A practical example of this is storing encrypted passwords on a database server, which is what we will be using for MyBook Instant Messenger as well. It means, when user creates a new account, he or she will supply their password. Rather than storing this password to the database as plain text, this password will be encrypted using a one-way encrypting algorithm and its encrypted form will be saved to the UserAuthorization database. That way, if hacker gains access to the database he will not see any of the passwords in plain-text.

MD5 encryption is an example of a one-way encryption algorithm; specifically, MD5 encryption maps a plain-text string of an arbitrary length to a small encrypted string of a fixed length. Two important properties of the MD5 algorithm are that two plain-text inputs cannot map to the same encrypted form, and that any given input always maps to the same encrypted value. The former property means that no two plain-text inputs will have the same encrypted value; the latter property means that if you wish to encrypt a particular plain-text input that it will always result in the same encrypted output.

### 4.8.1 MD5CyptoServiceProvider class

The MD5CyptoServiceProvider class in the System.Security.Cryptography namespace of the .NET Framework provides a class for performing one-way, MD5 encryption. This is the class that we'll use to provide encryption in our MyBook Instant

Messenger database. Before we examine how to implement encrypted passwords, let's first look at the functionality of the MD5CyptoServiceProvider class. The main method of this class is the ComputeHash method, which takes as input an array of bytes (the plain-text string to encrypt) and returns an array of bytes, which is the encrypted value. In other words, we'll want to encrypt a string, meaning that we must convert our string to an array of bytes in order to use the ComputeHash method. This conversion can be accomplished by using the UTF8Encoding encoding class, as shown in the following example:

```
//The string we wish to encrypt
string strPlainText = "Encrypt me!";
//The array of bytes that will contain the encrypted value of strPlainText
byte[] hashedDataBytes;
//The encoder class used to convert strPlainText to an array of bytes
UTF8Encoding encoder = new UTF8Encoding(false);
//Create an instance of the MD5CryptoServiceProvider class
MD5CryptoServiceProvider md5Hasher = new MD5CryptoServiceProvider();
hashedDataBytes = md5Hasher.ComputeHash(encoder.GetBytes())
//Call ComputeHash, passing in the plain-text string as an array of bytes
//The return value is the encrypted value, as an array of bytes
hashedDataBytes = md5Hasher.ComputeHash(encoder.GetBytes(strPlainText));
```

The ComputeHash method deals with arrays of bytes, not strings. Hence, to encrypt a plain-text string we must convert it to an array of bytes. This is accomplished by using the UTF8Encoding encoding class's GetBytes method (see the last line of the above code example). The return result of the ComputeHash method is the encrypted data as an array of bytes. (For all practical purposes, the encrypted array has exactly 16 elements.)

### 4.8.2 Implementation of MD5 encryption algorithm for MyBook IM application

Now that we've discussed the motivation behind using encrypted passwords and looked at the MD5 encryption algorithm, let’s implement encrypted passwords for MyBook IM application using MD5.

#### 4.8.2.1 Using MD5 To Store Encrypted Passwords

Previously we talked about using a database table UserAuthorization, with fields User_Name and Password. If password is saved as plain-text, both the User_Name and Password fields are of type “varchar”. However, we are planning to use encrypted passwords, so we will change the type of the Password field from a varchar to a binary type of length 16. We need this change, because the encrypted version of the user's password will be a 16-element array of bytes.

#### 4.8.2.2 Using MD5 To Authenticate the User

Since we are storing the passwords on a SQL server in encrypted form, and since, by the properteis of a one-way encryption algorithm, it is impossible to retrace from the encrypted form to the plain-text form, you may be wondering how in the world we'll authenticate a user. That is, when a user wants to login and supplies his or her username and password, how will we know if he or she provided the correct password?

Now we look at one of the properties of MD5 algorithm - that for any plain-text input the encrypted version of the input will be the same, always. For example, if we use MD5 to generate an encrypted form of the plain-text string "Password", the encrypted version of this will be the same today and forever more. Therefore, to authenticate a user we can

simply take the password they provide, encrypt it using MD5, and then see if that encrypted form exists in the database (along with their username). The following C#.NET code performs this check, logging in a user:

```
//check if user's name and/or password are valid

public bool Login1 (string uid, string @Password)
//Create a connection string
SqlConnection myConn = new SqlConnection ("Data Source=111.22.333.44;initial
catalog=Authortize;user id=User;password=Password;");
SqlCommand myComm = new SqlCommand("SELECT * FROM MybookIMUsers
WHERE User_Name = '" + uid + "' AND passw = @Password", myConn);
//The array of bytes that will contain the encrypted value of strPlainText
byte[] hashedDataBytes;
//The encoder class used to convert strPlainText to an array of bytes
UTF8Encoding encoder = new UTF8Encoding(false);
//Create an instance of the MD5CryptoServiceProvider class
MD5CryptoServiceProvider md5Hasher = new MD5CryptoServiceProvider();
//Call ComputeHash, passing in the plain-text string as an array of bytes
//The return value is the encrypted value, as an array of bytes
hashedDataBytes = md5Hasher.ComputeHash(encoder.GetBytes(Passwrd.Text +
UserName.Text));
//create parameters
SqlParameter paramPwd;
paramPwd = new SqlParameter("@Password", SqlDbType.Binary, 16);
paramPwd.Value = hashedDataBytes;
myComm.Parameters.Add(paramPwd);
//connect to SQL server and execute login and password check
myConn.Open();
SqlDataReader myReader = myComm.ExecuteReader();
Result = myReader.Read();
myReader.Close();
myConn.Close();
return Result;
```

#### 4.8.2.3 Limitations of Storing Encrypted Passwords in the Database

Before making any project and choosing, weather or not use encrypted passwords in our project, there are some limitations to be aware of. First of all - since the passwords are encrypted, there is no way to determine what a user's password is! At the beginning this we could think that this is exactly what we are running for – but, by encrypting passwords in the first place, it means that we cannot provide users with a "Click here to have your password e-mailed to you" feature. Now, if user forgets his password he or she will have to have his/her password reset to some random password, and then be e-mailed that new, random password. In other words, we simply can't e-mail the user his/her forgotten password, just because there is no way to determine what, exactly, his/her password is.

Second of all, converting from a plain-text password system to an encrypted system is possible, but can be a bit difficult. We need to create a new table with the Password field which would be of type binary and of length 16. Next step, we have to use a .NET program to read the contents of the existing user database, and for each record, add it to the new table making sure to encrypt the user's password using MD5. We must be careful not to delete the old user's account information until we have copied over their information fully.

#### 4.8.2.4 MD5 implementation for MyBook Instant Messenger authorization

User authorization for MyBook Instant Messenger uses Microsoft SQL server to be able to authorize user. The MD5 hashing technique discussed previously is sensible to dictionary attacks. Let's assume that a hacker has managed to break into the MyBook Instant Messenger user database. Even if there are 1 million user accounts, and let's say,

passwords are protected by the same MD5 hash, we talked about previously. Now hacker wants to obtain a valid password for any arbitrary MyBook Instant Messenger user (not necessarily a specific user). The hacker can create a dictionary of common password entries, for which he can compute their hashed values using the MD5 algorithm. Next step, hacker starts going through the dictionary of common passwords in their hashed form, and check to see if any of the 1 million MyBook Instant Mesenger user accounts have a matching encrypted password field.

Just because only the password field is hashed, the hacker at the same time simultaneously check each of the 1 million user's passwords at once. For example, word "ABC123" would be a common password, and that the hashed MD5 version of it would be "#f5%*@F)$$M". The hacker could then run a SQL query like this:

SELECT User_Name FROM Authorize_db WHERE password = “#f5%*@F)$$M”.

If any user has the password "ABC123" the hacker would see user’s user name. So, by only hashing on the password field, hacker could attack 1 million encrypted passwords at once. Because of that weakness of using hashes is well known, we will go one step further and solve this problem by "salting" the password fields before hashing them. This is accomplished by adding a second piece of information to the hash that is non-changing and unique for every user – for example, the user name, or a user ID (which both are unique) in MyBook Instant Messenger database.

In C#.NET, this code might look like:

```
byte[] hashedDataBytes;
UTF8Encoding encoder = new UTF8Encoding(false);
MD5CryptoServiceProvider md5Hasher = new MD5CryptoServiceProvider();
hashedDataBytes = md5Hasher.ComputeHash(encoder.GetBytes(Passwrd.Text));
```

Passwrd.Text is the text value of the user's plain-text password and hashedBytes is the encrypted byte array. (This encrypted byte array would then be stored in the user's password field in the database.) Now we add a “salt” to the value before computing its hashed value. For example, for MyBook Instant Messenger we use the user name, so the last line of code in the above example would be like this:

```
hashedDataBytes = md5Hasher.ComputeHash(encoder.GetBytes(Passwrd.Text +
UserName.Text));
```

UserName.Text is text value of user’s plain-text username. Say that this user, "Ainars" chose password "ABC123". Also, for example, user "JohnDoe" chose the password "ABC123". In previous, “non-salted” MD5 example, both users would have the same value in the encrypted password field. On the other hand, by “salting” the value before hashing, "Ainars"'s password field will contain the hashed version of "ABC123Ainars", while "JohnDoe"'s password field will contain the hashed version of "ABC123JohnDoe" – those passwords will be different and will result in different, non-similar hashed values. Now even if the hacker knows how we “salt” the hashed values, he can attack only one user's password at a time. In other words, let’s say, he wants to check all users to see if any of them has the password "ABC123". Before hacker could issue only one SQL query to check all 1 million MyBook Instant Messenger users, now he must issue a unique SQL query for each of the 1 million MyBook Instant Messenger users.

## 4.9 MyBook Instant Messenger server part

### 4.9.1 MyBook Instant Messenger Server user's interface

MyBook Instant Messenger server is a MyBook Instant Messenger client/server application server's part. User interface for it (Figure 4.9.1) shows what and how many users have connected to MyBook Instant Messenger system. Depending on the amount of created communities, there might be many servers, users could connect to. Any community then would have its own dedicated server application with its own pre-assigned port number. As far as MyBook Instant Messenger system administrator is concerned, he/she must launch as many server applications, as there are communities and assign them their port numbers.


Fig. 4.9 MyBook Instant Messenger Server user's interface.

### 4.9.2 MyBook Instant Messenger Server's functionality

MyBook Instant Messenger server is a MyBook Instant Messenger client/server application server's part. It is main middleman between MyBook Instant Messenger client applications. It passes all the messages from one client to another. Only places,

MyBook Instant Messenger doesn’t take part are user authentication and final part for file transfers, where users are already transmitting and receiving file.

Next exaple shows, how message passing are handled:

```
// Receive procedure
private void Receive(IAsyncResult ar)
        {
        Client client = null;
        try
                {
//user-message sender
client = (Client)ar.AsyncState;
//received bytes from client
int bytesRead = client.Socket.EndReceive(ar);
//if bytes read >0 continue to proceed
if (bytesRead > 0)
                {
string fullCommand = this.ASCII.GetString(client.buffer, 0, bytesRead);
//defining command with splitting character ‘^’
string[] command = fullCommand.Split(new char[] {'^'});
//if first part of “command” = MSG, server will know it is multichat message
if (command[0].ToUpper().Equals("MSG") == true)
        {
//defining and splitting multichat message. It contains 7 ‘commands’
```

```
string message = command[0] + "^" + client.NickName + "^" + command[1] + "^"
+command[2]+"^"+ command[3]+"^" +command[4]+"^" +command[5]+"^"
+command[6]+"^" +command[7];
//now we prepare message to send it further to the client it was meant for
byte[] msg = this.ASCII.GetBytes(message.ToCharArray());
for (int i = 0; i < this.objList.Count; i++)
        {
//Send message to a  user-message receiver
((Client)this.objList[i]).Socket.BeginSend(msg, 0, msg.Length, SocketFlags.None,
this.onSend, (Client)this.objList[i]);
        }
}
```

Most of the message passing are proceeded in such way, except, when we do user authorization – we use SQL server for that, and final part for file sending – when user-sender connects to a user-receiver directly and starts file transfer.

# CHAPTER 5

## 5 Conclusion

### 5.1 Human testing in real settings for MyBook Instant Messenger

Human factor testing was performed to test MyBook Instant Messenger's two features – message sending and file transfer. Three stages of testing were involved. First test was performed within ECECS department's LAN. First thing we checked were round-trip and delay (if too noticeable for human) of the text messages. After that we were testing file transfer speeds. Tests were performed in real settings – using real local area network test bed and my classmates. Results (Figure 5.1) show that within Local Area Network both – file transfers and message passing work excellent, at the best speed provided by the LAN, which averages about 0.6 - 0.7 MB/s and there are no delays at all.

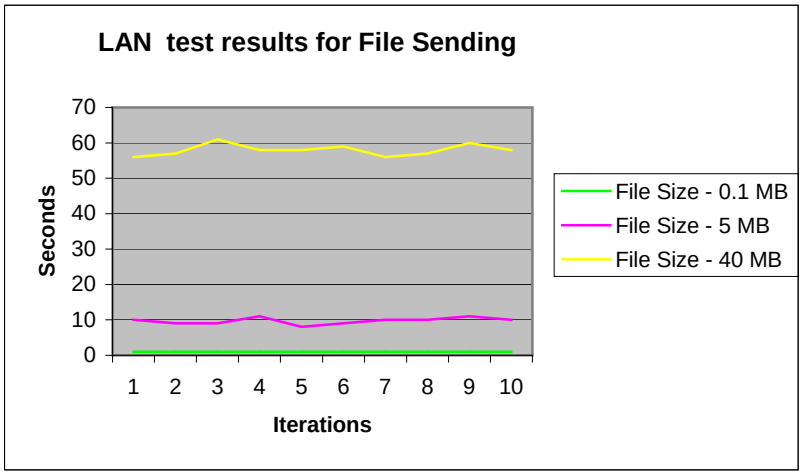


Fig. 5.1 Transfer speeds in LAN for MyBook Instant Messenger File Sending feature.

Second test performed, was between two cities – Cincinnati, OH and Chicago, IL. We chose three test data pieces and repeatedly tested them many times. To compare to LAN test results (Figure 5.1), intercities' results (Figure 5.2), message sending didn't show any difference in its performance, while file transfer speed was smaller (average – 0.5 Mb/s), which is obvious, because of distance, transferred file packets have to travel, which could cause more congestion, different internet network hops, which is causing file transfer speed

decrease.

I myself was being in Cincinnati, OH and my wife Julija in Riga, Latvia, performed third human factor testing. Once again, test results didn't show any unexpected results. Message sending delay was minimal and it didn't impact our real time conversation. To compare to Yahoo Messenger ™ message sending, which has a lot of graphical interface (Emoticons, IM Environments, Audibles) many times had enormous delays, while I was talking to Julija. So because of congested network and graphics take a large chunk of data, which it has to send over the Internet, it sometimes was delayed so much, that it created our real time conversation impossible. So my point here was not to create the same problem, and real time conversation using MyBook Instant Messenger message sending didn't create huge delays.

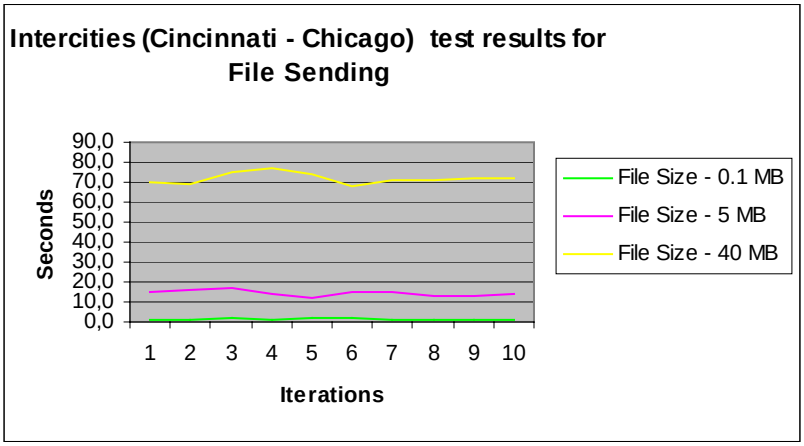


Fig. 5.2 Transfer speeds between Cincinnati and Chicago for MyBook Instant Messenger File Sending feature.

File transfer for MyBook Instant Messenger to Latvia and back to USA was slower to compare to test within LAN results or intercities' test results. It is obvious that file transfer speed will decrease (average 0.4 Mb/s), because we are using TCP transfers and some packets might be delayed or even lost during that long distance over the Internet.

Figure 5.3 shows us summary of third human factor test result. As we can see from all graphs, pattern follows like this: file transfer slows down as soon as number of hops (distance) for sent packages increase.

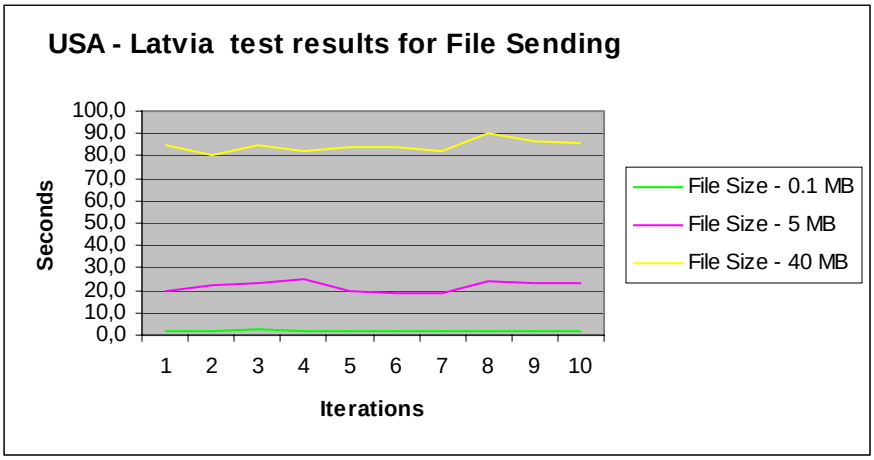


Fig. 5.3 Transfer speeds between USA and Latvia for MyBook Instant Messenger File Sending feature.

Graphs, which show message sending performance, were not included, because they show very similar performance results, so there is no sense to show relevant results.

## 5.2 Conclusion for MyBook Instant Messenger

We now summarize our thesis work by discussing its main points along with its existing imperfections and its potential improvements along with the extensions to this work.

We have presented collaborative tool for MyBook project.

It allows peers to collaborate among other peers on a topic of common interest in peer-to-peer topic based network. Peers, who have common interests in MyBook books system, can find each other using collaborative tool - MyBook Instant Messenger, which is implemented in MyBook Desktop System. The features for MyBook Instant Messenger are multi & private messaging, file sending, file logging, various text formatting, and

creation of new communities of common interest topics. User authentication is performed using salted one-way MD5 hashing algorithm.

Future plans for MyBook Instant Messenger would be implementation of offline messages and ability to transfer replicated files, which are located in MyBook Desktop System's peer to peer network, simply by clicking on file link in MyBook book's file. Experiments and testing for MyBook Instant Messenger were performed on test data, mostly obtained from benchmarking as well as human factor testing.

Every task module is coded and subsequently all the modules are put together to form the final MyBook Instant Messenger application. The MyBook Instant Messenger software package is developed using Microsoft C#.NET language and Java.

**REFERENCES:**

[1] http://mybook.uc.edu

[2] Kenneth A. Berman and Fred .S. Annexstein: Actualizing Context for Personal Knowledge Management, submitted for publication in ACM SIGMOD Record.

[3] K.A. Berman and F.S. Annexstein: An Educational Tool for the 21st Century:Peer-to-peer computing, Ohio Learning Network Conference, Windows on the Future Conference.

[4] F. S. Annexstein, K.A. Berman, K. Henkener, Svetlana Strunjaš: A , First International Workshop on Peer-to-Peer Knowledge Management space-efficient model for sharing personal knowledge objects in peer communities, Boston, MA., August 2004.

[5] http://www.cfo.com/article.cfm/3010931?f=archives&origin=archive

[6] http://www.pcquest.com/content/p2p/102091207.asp

[7] http://www.webopedia.com/TERM/m/md5.html

[8] Kazaa – www.kazaa.com

[9] Groove – www.groove.net

[10] SETI@Home http://setiathome.ssl.berkeley.edu

[11] Fightaids@home – http://www.fightaidsathome.org

# Appendix A

## C#.NET Source Code for Mybook Instant Messenger

The following is C#.NET source code for Mybook Instant Messenger, which I developed for Mybooks Desktop System as a part of collaboration software.

### Client Side:

CClient form:

```
using System;
using System.IO;
using System.Text;
using System.Runtime;
using System.Threading;
using System.Reflection;
using System.Collections;
using System.Net;
using System.Net.Sockets;
using System.Drawing;
using System.Windows.Forms;
using System.ComponentModel;
using System.Diagnostics;
using System.Data;
using System.Text.RegularExpressions ;
using System.Runtime.InteropServices;
using System.Security;
using System.Security.Principal;
using System.Security.Permissions;


namespace MyBookIM
{

	public enum ClientStatus : int
	{
		Disconnected,
		Connected
	}

	public sealed class CClient : Form
	{
		public System.ComponentModel.IContainer components;
		private RichTextBox textBox = null;
		private MainMenu menu = null;
		private ContextMenu popUpMenu = null;
		internal ClientStatus ClientStatus = ClientStatus.Disconnected;
		private ArrayList userList = null;
		private Thread thread = null;
		public Socket socket = null;
		private IPEndPoint endPoint = null;
		private AsyncCallback OnConnect = null;
		public Encoding ASCII = null;
		private byte[] buffer = null;
		private string nickName = null;
		private MyBookMenuItem MenuConnect;
		private MyBookMenuItem MenuAtstarpe;
		private MyBookMenuItem MenuExit;
		public System.Windows.Forms.ListBox listBox;
		private System.Windows.Forms.RichTextBox messageBox;
		private System.Windows.Forms.Button ChatSendButton;
		public MyBookMenuItem MenuEdit;
		public MyBookMenuItem MenuCut;
		public MyBookMenuItem MenuCopy;
		public MyBookMenuItem MenuPaste;
```

```
		public MyBookMenuItem MenuAtstarpe2;
		public MyBookMenuItem MenuSelAll;
		public System.Windows.Forms.ImageList imageList1;
		public MenuExtender.MenuExtender menuExtender1;
		private MyBookIM.MyBookMenuItem MenuTools;
		private MyBookIM.MyBookMenuItem MenuPrivateTalk;
		private MyBookIM.MyBookMenuItem MenuSendFile;
		private MyBookIM.MyBookMenuItem MenuMessenger;
		public MyBookIM.MyBookMenuItem MenuLogFile;
		public MyBookIM.MyBookMenuItem MenuSaveLogFile;
		public MyBookIM.MyBookMenuItem MenuOpenLogFile;
		public MyBookIM.MyBookMenuItem MenuClearLogFile;
		private MyBookIM.MyBookMenuItem MenuHelp;
		private MyBookIM.MyBookMenuItem MenuHelpTopic;
		private MyBookIM.MyBookMenuItem MenuAbout;
		public string usrname;
		public long fileSize;
		public string AuthServerPort;
		public SendFileReceiver SendFileReceiverForm;
		public SendFile SendFileForm;
		public Hashtable userarray;
		public Authorize AuthorizeForm;
		private FontCombo.FontComboBox fontComboBox;
		public ColorButton.ColorButton ButtonColor;
		private System.Windows.Forms.ComboBox comboBoxFontSize;
		private System.Windows.Forms.MainMenu mainPTMenu;
		public System.Windows.Forms.ToolTip toolTipCClient;
		private System.Windows.Forms.FontDialog fontDialog1;
		private System.Windows.Forms.CheckBox m_italic;
		private System.Windows.Forms.CheckBox m_under;
		private System.Windows.Forms.CheckBox m_bold;
		public string File_path;
		public string Receiverusr;
		public string Senderusr;
		public long File_size;
		public string reciever_ipaddress;
		public string userXPxx;
		public string File_Name;
		public string SF_Sender;
		private MyBookIM.MyBookMenuItem MenuHelpSlides;
		public string SF_FileSize;
		public System.Diagnostics.Process javaw_process = new
System.Diagnostics.Process();
		public enum MessageBeepType
		{
			Default = -1,
			Ok = 0x00000000,
			Error = 0x00000010,
			Question = 0x00000020,
			Warning = 0x00000030,
			Information = 0x00000040,
		}

		[DllImport("user32.dll", SetLastError=true)]
		public static extern bool MessageBeep(MessageBeepType type);
		[DllImport("kernel32.dll",CharSet=CharSet.Auto, SetLastError=true)]
		[return:MarshalAs(UnmanagedType.Bool)]
		//Setting Temp Environment Variable
		public static extern bool SetEnvironmentVariable(string lpName, string
lpValue);

		public CClient()
		{
			AuthorizeForm = new Authorize();
			AuthorizeForm.ShowDialog();
			this.userList = new ArrayList();
			this.userarray = new Hashtable();
			this.OnConnect = new AsyncCallback(this.Connecting);
			this.ASCII = Encoding.ASCII;
			this.buffer = new byte[1024];
			this.InitializeComponent();
			this.listBox.MouseDown += new
System.Windows.Forms.MouseEventHandler(this.ButtonDown);
```

```
                        this.listBox.MouseMove += new
System.Windows.Forms.MouseEventHandler(this.listBox_MouseMove);
                        fontComboBox.Populate(true);
                        ButtonColor.Color = Color.Black ;
                        comboBoxFontSize.Text = "12";
                        fontComboBox.Text = "Arial";
                        messageBox.SelectionChanged += new
EventHandler(OnTxtSelectionChanged);

CClient.SetEnvironmentVariableEx("CLASSPATH", "c:\\Program Files\\MyBook Instant
Messenger\\dhq\\jars\\java-getopt-1.0.9.jar;c:\\Program Files\\MyBook Instant
Messenger\\dhq\\jars\\FreePastry-1.3.2.jar;.;c:\\Program Files\\MyBook Instant
Messenger\\dhq\\jars\\log4j-1.2.8.jar;c:\\Program Files\\MyBook Instant
Messenger\\dhq\\jars\\jetty\\lib\\org.mortbay.jetty.jar;c:\\Program Files\\MyBook Instant
Messenger\\dhq\\jars\\jetty\\lib\\javax.servlet.jar;c:\\Program Files\\MyBook Instant
Messenger\\dhq\\jars\\jetty\\ext\\jasper-runtime.jar;c:\\Program Files\\MyBook Instant
Messenger\\dhq\\jars\\jetty\\ext\\jasper-compiler.jar;c:\\Program Files\\MyBook Instant
Messenger\\dhq\\jars\\jetty\\ext\\xercesImpl.jar;c:\\Program Files\\MyBook Instant
Messenger\\dhq\\jars\\DHQ_Binary.jar;");
// Verify that environment variable is set correctly.
Console.WriteLine("The value of CLASSPATH is: " +
Environment.GetEnvironmentVariable("CLASSPATH"));}

                protected override void Dispose(bool disposing)
                {
                        if ((disposing == true) && (this.components != null))
                        {
                                this.components.Dispose();
                        }
                        base.Dispose(disposing);
                }
                protected override void OnClosed(EventArgs e)
                {
                        try
                        {
                        }
                        catch (Exception ex)
                        {
                                MessageBox.Show(ex.Message.ToString());
                        }
                }


public void InitializeComponent() …

public string NickName
                {
                        get
                        {
                                return this.nickName;
                        }
                        set
                        {
                                this.nickName = value;
                        }
                }
                public Socket ClientSocket
                {
                        get
                        {
                                return this.socket;
                        }
                        set
                        {
                                this.socket = value;
                        }
                }
                public IPEndPoint EndPoint
                {
                        get
                        {
                                return this.endPoint;
                        }
                        set
```

```
                {
                        this.endPoint = value;
                }
        }
        private void Connect()
        {
                try
                {


                        this.socket = new Socket(AddressFamily.InterNetwork,
SocketType.Stream, ProtocolType.Tcp);
                        this.socket.BeginConnect(this.endPoint, this.OnConnect,
this.socket);
                }
                catch (Exception)
                {
                        this.userList.Clear();
                        this.listBox.Items.Clear();
                        this.textBox.Text = "";
                        this.ClientStatus = ClientStatus.Disconnected;
                        this.messageBox.Enabled = false;
                        this.MenuTools.Enabled = false;
                        this.MenuExit.Enabled = false;
                        this.MenuEdit.Enabled = false;
                        this.MenuHelp.Enabled = false;
                        this.ChatSendButton.Enabled = false;
                        this.messageBox.Focus();
                        this.ResumeLayout(false);
                        this.messageBox.Enabled = false;
                        this.MenuTools.Enabled = false;
                        this.MenuExit.Enabled = true;
                        this.MenuEdit.Enabled = false;
                        this.MenuHelp.Enabled = false;
                        this.ChatSendButton.Enabled = false;
                        this.comboBoxFontSize.Enabled = false;
                        this.fontComboBox.Enabled = false;
                        this.ButtonColor.Enabled = false;
                        this.m_bold.Enabled = false;
                        this.m_italic.Enabled = false;
                        this.m_under.Enabled = false;
                        this.messageBox.Focus();
                        this.MenuConnect.Text = "Disonnected";
                        this.MenuConnect.Enabled = false;
                        MessageBox.Show("No Connection Could Be Made", "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error);

                }
        }
        private void Connecting(IAsyncResult ar)
        {
                try
                {
                        Socket sock = (Socket)ar.AsyncState;
                        sock.EndConnect(ar);
                        if (sock.Connected == true)
                        {


                                sock.BeginReceive(this.buffer, 0,
this.buffer.Length, SocketFlags.None, new AsyncCallback(this.OnResponse), sock);
                        }
                        else
                        {

                                this.userList.Clear();
                                this.listBox.Items.Clear();
                                this.textBox.Text = "";
                                this.ClientStatus = ClientStatus.Disconnected;
                                this.messageBox.Enabled = false;
                                this.MenuTools.Enabled = false;
                                this.MenuExit.Enabled = false;
                                this.MenuEdit.Enabled = false;
                                this.MenuHelp.Enabled = false;
```

```
                                        this.ChatSendButton.Enabled = false;
                                        this.messageBox.Focus();
                                        this.ResumeLayout(false);
                                        this.messageBox.Enabled = false;
                                        this.MenuTools.Enabled = false;
                                        this.MenuExit.Enabled = true;
                                        this.MenuEdit.Enabled = false;
                                        this.MenuHelp.Enabled = false;
                                        this.ChatSendButton.Enabled = false;
                                        this.comboBoxFontSize.Enabled = false;
                                        this.fontComboBox.Enabled = false;
                                        this.ButtonColor.Enabled = false;
                                        this.m_bold.Enabled = false;
                                        this.m_italic.Enabled = false;
                                        this.m_under.Enabled = false;
                                        this.messageBox.Focus();
                                        this.MenuConnect.Text = "Disconnected";
                                        this.MenuConnect.Enabled = false;
                                        MessageBox.Show("No Connection Could Be Made",
"Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
                                }
                        }
                        catch (Exception)
                        {
                                this.userList.Clear();
                                this.listBox.Items.Clear();
                                this.textBox.Text = "";
                                this.ClientStatus = ClientStatus.Disconnected;
                                this.messageBox.Enabled = false;
                                this.MenuTools.Enabled = false;
                                this.MenuExit.Enabled = false;
                                this.MenuEdit.Enabled = false;
                                this.MenuHelp.Enabled = false;
                                this.ChatSendButton.Enabled = false;
                                this.messageBox.Focus();
                                this.ResumeLayout(false);
                                this.messageBox.Enabled = false;
                                this.MenuTools.Enabled = false;
                                this.MenuExit.Enabled = true;
                                this.MenuEdit.Enabled = false;
                                this.MenuHelp.Enabled = false;
                                this.ChatSendButton.Enabled = false;
                                this.comboBoxFontSize.Enabled = false;
                                this.fontComboBox.Enabled = false;
                                this.ButtonColor.Enabled = false;
                                this.m_bold.Enabled = false;
                                this.m_italic.Enabled = false;
                                this.m_under.Enabled = false;
                                this.messageBox.Focus();
                                this.MenuConnect.Text = "Disconnected";
                                this.MenuConnect.Enabled = false;
                                MessageBox.Show("No Connection Could Be Made", "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                        }
                }
                private void OnResponse(IAsyncResult ar)
                {
                        try
                        {
                                Socket sock = (Socket)ar.AsyncState;
                                int bytesRead = sock.EndReceive(ar);
                                if (bytesRead > 0)
                                {
                                        string res = this.ASCII.GetString(this.buffer, 0,
bytesRead);
                                        if (res.ToUpper().Equals("OK") == true)
                                        {
                                                string reg = "AUTH^" + this.nickName;
                                                this.buffer =
this.ASCII.GetBytes(reg.ToCharArray());
                                                sock.BeginSend(this.buffer, 0,
this.buffer.Length, SocketFlags.None, new AsyncCallback(this.Auth), sock);
                                        }
                                        else
```

```
                                        {
                                        }
                                }
                                else
                                {
                                }
                        }
                        catch (Exception e)
                        {
                                MessageBox.Show(e.Message.ToString());
                        }
                }
                private void Auth(IAsyncResult ar)
                {
                        try
                        {

                                Socket sock = (Socket)ar.AsyncState;
                                sock.EndSend(ar);
                                this.buffer = new byte[1024];
                                sock.BeginReceive(this.buffer, 0, this.buffer.Length,
SocketFlags.None, new AsyncCallback(this.Confirm), sock);
                        }
                        catch (Exception e)
                        {
                                MessageBox.Show(e.Message.ToString());
                        }
                }

                public string[] command;
                private void Confirm(IAsyncResult ar)
                {
                        try
                        {
                                Socket sock = (Socket)ar.AsyncState;
                                int bytesRead = sock.EndReceive(ar);
                                if (bytesRead > 0)
                                {
                                        string cmd = this.ASCII.GetString(this.buffer, 0,
bytesRead);
                                        if (cmd.Equals("001") == true)
                                        {

                                                this.ClientStatus = ClientStatus.Connected;
                                                this.textBox.SelectionColor = Color.Blue;
                                                this.textBox.AppendText("Connecting To
Server ...\r\n");
                                                this.textBox.SelectionColor = Color.Green;
                                                this.textBox.AppendText("Connected!\r\n");
                                                this.textBox.AppendText("\r\n");
                                                this.textBox.SelectionColor = Color.Blue;
                                                this.textBox.AppendText("Welcome To MyBook
Instant Messenger!\r\n");

        this.textBox.AppendText("====================================\r\n");
                                                this.textBox.SelectionColor = Color.Blue;
                                                this.textBox.AppendText("<< "+ this.nickName
+ " Has Joined Chat... >>\r\n");
                                                sock.BeginReceive(this.buffer, 0,
this.buffer.Length, SocketFlags.None, new AsyncCallback(this.ReceiveMode), sock);

                                        }
                                        else if (cmd.Length > 3 && cmd.Substring(0,
3).Equals("001") == true)
                                        {
                                                this.ClientStatus = ClientStatus.Connected;
                                                this.textBox.SelectionColor = Color.Blue;
                                                this.textBox.AppendText("Connecting To
Server ...\r\n");
                                                this.textBox.SelectionColor = Color.Green;
                                                this.textBox.AppendText("Connected!\r\n");
                                                this.textBox.AppendText("\r\n");
                                                this.textBox.SelectionColor = Color.Blue;
```

```
                                        this.textBox.AppendText("Welcome To MyBook
Instant Messenger!\r\n");

        this.textBox.AppendText("====================================\r\n");
                                        this.textBox.SelectionColor = Color.Blue;
                                        this.textBox.AppendText("<<"+ this.nickName
+ " Has Joined Chat....>>\r\n");
                                        string list = cmd.Substring(3);
                                        string[] iList = list.Split(new char[]
{'^'});
                                        if (iList[1].Equals("EMPTY") == true)
                                        {
                                                this.userList.Add(new
Clients(this.nickName));

        this.listBox.Items.Add(this.nickName);
                                        }
                                        else
                                        {
                                                this.userList.Add(new
Clients(this.nickName));
                                                for (int i = 1; i <
iList.GetLength(0); i++)
                                                {
                                                        this.userList.Add(new
Clients(iList[i]));
                                                }
                                                this.userList.Sort();
                                                this.listBox.BeginUpdate();
                                                for (int i = 0; i <
this.userList.Count; i++)
                                                {

        this.listBox.Items.Add(this.userList[i].ToString());
                                                }
                                                this.listBox.EndUpdate();
                                        }
                                        sock.BeginReceive(this.buffer, 0,
this.buffer.Length, SocketFlags.None, new AsyncCallback(this.ReceiveMode), sock);
                                }
                                else if (cmd.Equals("100") == true)
                                {
                                        MessageBox.Show("Nick " + this.nickName + "
Already Exists In The Chat", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
                                        return ;
                                }
                                else
                                {
                                }
                        }
                        else
                        {
                        }
                }
                catch (Exception e)
                {
                        MessageBox.Show(e.Message.ToString());
                }
        }


        public static bool SetEnvironmentVariableEx(string environmentVariable,
string variableValue)
        {
                try
                {
                        // Get the write permission to set the environment
variable.
                        EnvironmentPermission environmentPermission = new
EnvironmentPermission(EnvironmentPermissionAccess.Write,environmentVariable);
                        environmentPermission.Demand();

                        return SetEnvironmentVariable(environmentVariable,
variableValue);
```

```
                        }
                        catch( SecurityException e)
                        {
                                Console.WriteLine("Exception:" + e.Message);
                        }
                        return false;
                }

                public void CClient_Closing(object sender,
System.ComponentModel.CancelEventArgs e)
                {

                        e.Cancel = false;
                        try

                        {
                                Process [] processes = Process.GetProcessesByName
("javaw");
                                if (processes.Length == 1)
                                {

                                        javaw_process.Kill();
                                        System.Environment.Exit(-2147480000);

                                }
                                else
                                {
                                        System.Environment.Exit(-2147480000);
                                }
                        }
                        catch (Exception ex)
                        {
                                MessageBox.Show(ex.Message.ToString());
                        }

                }


                delegate void ShowDelegate();

                private void ReceiveMode(IAsyncResult ar)
                {
                        try
                        {
                                Socket sock = (Socket)ar.AsyncState;
                                int bytesRead = sock.EndReceive(ar);
                                if (bytesRead > 0)
                                {
                                        string fullCommand =
this.ASCII.GetString(this.buffer, 0, bytesRead);
                                        command = fullCommand.Split(new char[] {'^'});

                                        if (command[0].ToUpper().Equals("LIST") == true)
                                        {
                                                if (command[1].ToUpper().Equals("EMPTY") ==
true)
                                                {
                                                        this.userList.Add(new
Clients(this.nickName));

        this.listBox.Items.Add(this.nickName);
                                                }
                                                else
                                                {
                                                        this.userList.Add(new
Clients(this.nickName));
                                                        for (int i = 1; i <
command.GetLength(0); i++)
                                                        {
                                                                this.userList.Add(new
Clients(command[i]));
                                                        }
                                                        this.userList.Sort();
                                                        this.listBox.BeginUpdate();
```

```
                                                for (int i = 0; i <
this.userList.Count; i++)
                                                {

        this.listBox.Items.Add(this.userList[i].ToString());
                                                }
                                                this.listBox.EndUpdate();
                                        }
                                }
                                else if (command[0].ToUpper().Equals("JOINS") ==
true)
                                {

                                        this.userList.Add(new Clients(command[1]));
                                        this.userList.Sort();
                                        this.listBox.Items.Clear();
                                        this.listBox.BeginUpdate();
                                        for (int i = 0; i < this.userList.Count;
i++)
                                        {

        this.listBox.Items.Add(this.userList[i].ToString());
                                        }
                                        this.listBox.EndUpdate();
                                        textBox.AppendText(textBox.SelectedText+
"\r\n");
                                        textBox.SelectionStart = textBox.TextLength
-1;
                                        textBox.SelectionColor = Color.Blue;
                                        textBox.SelectedText = ("<< " + command[1] +
" Has Joined Chat... >>");

                                }
                                else if (command[0].ToUpper().Equals("MSG") == true)
                                {

                                        Color Colorzz =
(Color)TypeDescriptor.GetConverter(typeof(Color)).ConvertFromString(command[5]);
                                        textBox.Focus();
                                        textBox.AppendText("<" + command[1] + "> ");
                                        textBox.AppendText(textBox.SelectedText
+"\r\n");
                                        textBox.SelectionStart = textBox.TextLength
-1;
                                        if (command[6] == "Unchecked" && command[7]
== "Unchecked" && command[8] == "Unchecked")
                                        {
                                                textBox.SelectionFont = new Font
(command[4],Int32.Parse(command[3].ToString()));
                                                textBox.SelectionColor = Colorzz;
                                                textBox.SelectedText = command[2];
                                                if (this.WindowState ==
FormWindowState.Minimized)
                                                {
// each time this is called, the current window flashes once

        clsUser32DLL.FlashWindowOnce(this.Handle.ToInt32(), 1);
                                                        MessageBeep(0);

                                                }
                                        }
                                        else if (command[6] == "Checked" &&
command[7] == "Unchecked" && command[8] == "Unchecked")
                                        {
                                                textBox.SelectionFont = new Font
(command[4],Int32.Parse(command[3].ToString()),System.Drawing.FontStyle.Bold);
                                                textBox.SelectionColor = Colorzz;
                                                textBox.SelectedText = command[2];
                                                if (this.WindowState ==
FormWindowState.Minimized)
                                                {
// each time this is called, the current window flashes once

        clsUser32DLL.FlashWindowOnce(this.Handle.ToInt32(), 1);
```

```
                                                        MessageBeep(0);
                                                }
                                        }
                                        else if (command[6] == "Unchecked" &&
command[7] == "Checked" && command[8] == "Unchecked")
                                        {
                                                textBox.SelectionFont = new Font
(command[4],Int32.Parse(command[3].ToString()),System.Drawing.FontStyle.Italic);
                                                textBox.SelectionColor = Colorzz;
                                                textBox.SelectedText = command[2];
                                                if (this.WindowState ==
FormWindowState.Minimized)
                                                {
// each time this is called, the current window flashes once

        clsUser32DLL.FlashWindowOnce(this.Handle.ToInt32(), 1);
                                                        MessageBeep(0);
                                                }
                                        }
                                        else if (command[6] == "Unchecked" &&
command[7] == "Unchecked" && command[8] == "Checked")
                                        {
                                                textBox.SelectionFont = new Font
(command[4],Int32.Parse(command[3].ToString()),System.Drawing.FontStyle.Underline);
                                                textBox.SelectionColor = Colorzz;
                                                textBox.SelectedText = command[2];
                                                if (this.WindowState ==
FormWindowState.Minimized)
                                                {
// each time this is called, the current window flashes once

        clsUser32DLL.FlashWindowOnce(this.Handle.ToInt32(), 1);
                                                        MessageBeep(0);
                                                }
                                        }
                                        else if (command[6] == "Checked" &&
command[7] == "Unchecked" && command[8] == "Checked")
                                        {
                                                textBox.SelectionFont = new Font
(command[4],Int32.Parse(command[3].ToString()),System.Drawing.FontStyle.Bold |
System.Drawing.FontStyle.Underline);
                                                textBox.SelectionColor = Colorzz;
                                                textBox.SelectedText = command[2];
                                                if (this.WindowState ==
FormWindowState.Minimized)
                                                {
// each time this is called, the current window flashes once

        clsUser32DLL.FlashWindowOnce(this.Handle.ToInt32(), 1);
                                                        MessageBeep(0);
                                                }
                                        }
                                        else if (command[6] == "Checked" &&
command[7] == "Checked" && command[8] == "Unchecked")
                                        {
                                                textBox.SelectionFont = new Font
(command[4],Int32.Parse(command[3].ToString()),System.Drawing.FontStyle.Bold |
System.Drawing.FontStyle.Italic);
                                                textBox.SelectionColor = Colorzz;
                                                textBox.SelectedText = command[2];
                                                if (this.WindowState ==
FormWindowState.Minimized)
                                                {
// each time this is called, the current window flashes once

        clsUser32DLL.FlashWindowOnce(this.Handle.ToInt32(), 1);
                                                        MessageBeep(0);
                                                }
                                        }
                                        else if (command[6] == "Unchecked" &&
command[7] == "Checked" && command[8] == "Checked")
                                        {
```

```
                                                textBox.SelectionFont = new Font
(command[4],Int32.Parse(command[3].ToString()),System.Drawing.FontStyle.Italic |
System.Drawing.FontStyle.Underline);
                                                textBox.SelectionColor = Colorzz;
                                                textBox.SelectedText = command[2];
                                                if (this.WindowState ==
FormWindowState.Minimized)
                                                {
// each time this is called, the current window flashes once

        clsUser32DLL.FlashWindowOnce(this.Handle.ToInt32(), 1);
                                                        MessageBeep(0);
                                                }
                                        }
                                        else if (command[6] == "Checked" &&
command[7] == "Checked" && command[8] == "Checked")
                                        {



                                                textBox.SelectionFont = new Font
(command[4],Int32.Parse(command[3].ToString()),System.Drawing.FontStyle.Bold |
System.Drawing.FontStyle.Italic | System.Drawing.FontStyle.Underline);
                                                textBox.SelectionColor = Colorzz;
                                                textBox.SelectedText = command[2];
                                                if (this.WindowState ==
FormWindowState.Minimized)
                                                {
// each time this is called, the current window flashes once

        clsUser32DLL.FlashWindowOnce(this.Handle.ToInt32(), 1);
                                                        MessageBeep(0);
                                                }
                                        }

                                }
                                else if (command[0].ToUpper().Equals("PRIVMSG") ==
true)
                                {

                                        PTalk pt = new PTalk(this, command[1]);
                                        if (!userarray.ContainsKey(command[1]))

                                        {

        userarray.Add(command[1], pt);

        this.Invoke(new ShowDelegate(pt.Show));

        pt.PrivateTalkReceiveMessageBox.SelectionColor = Color.Blue;

        pt.PrivateTalkReceiveMessageBox.AppendText("<" + command[1] + "> ");

        pt.PrivateTalkReceiveMessageBox.SelectionColor = Color.Black;

        pt.PrivateTalkReceiveMessageBox.AppendText(command[2] + "\r\n");

        pt.Activate();
                                        }

                                        else
                                        {



        ((PTalk)userarray[command[1]]).PrivateTalkReceiveMessageBox.SelectionColor =
Color.Blue;

        ((PTalk)userarray[command[1]]).PrivateTalkReceiveMessageBox.AppendText("<" +
command[1] + "> ");

        ((PTalk)userarray[command[1]]).PrivateTalkReceiveMessageBox.SelectionColor =
Color.Black;
```

```
        ((PTalk)userarray[command[1]]).PrivateTalkReceiveMessageBox.AppendText(command[2]
+ "\r\n");
                                                if
(((PTalk)userarray[command[1]]).WindowState == FormWindowState.Minimized)
                                                {
// each time this is called, the current window flashes once

        clsUser32DLL.FlashWindowOnce(((PTalk)userarray[command[1]]).Handle.ToInt32(), 1);
                                                MessageBeep(0);
                                                }
                                        }


                                }
                                else if (command[0].ToUpper().Equals("SYSMSG") == true)
                                        {

                                PTalk pt = new PTalk(this, command[1]);
                                if (!userarray.ContainsKey(command[1]))

                                        {

        userarray.Add(command[1], pt);

        this.Invoke(new ShowDelegate(pt.Show));

        pt.PrivateTalkReceiveMessageBox.SelectionColor = Color.Black;

        pt.PrivateTalkReceiveMessageBox.AppendText(command[2] + "\r\n");

        pt.Activate();
        pt.PrivateTalkSendMessageBox.Focus();
                                        }

                                        else
                                        {


        ((PTalk)userarray[command[1]]).PrivateTalkReceiveMessageBox.SelectionColor =
Color.Black;

        ((PTalk)userarray[command[1]]).PrivateTalkReceiveMessageBox.AppendText(command[2]
+ "\r\n");
if (((PTalk)userarray[command[1]]).WindowState == FormWindowState.Minimized)
                                                {
// each time this is called, the current window flashes once

        clsUser32DLL.FlashWindowOnce(((PTalk)userarray[command[1]]).Handle.ToInt32(), 1);
                                                MessageBeep(0);

        ((PTalk)userarray[command[1]]).PrivateTalkSendMessageBox.Focus();
                                                }

                                        }


                                }
                                else if (command[0].ToUpper().Equals("SENDFIL") == true)
                                        {
                                this.File_Name = command[2];
                                this.SF_Sender = command[1];
                                this.SF_FileSize = command[3];
                                SendFileReceiver SFR = new SendFileReceiver(this);
                                this.Invoke(new ShowDelegate(SFR.Show));
                                SFR.Text = command[1] + " is Sending You a File!";
                                SFR.labelFileInfo.Text = "File Name: " + command[2] +
"\r\nFile Size: " + command[3] +" Bytes";

                                }
                                else if (command[0].ToUpper().Equals("WANTSDL") == true)
                                        {

                                this.reciever_ipaddress = command[4];
```

```
                                OkSendFile OKSF = new OkSendFile(this);
                                this.Invoke(new ShowDelegate(OKSF.Show));
                                OKSF.InfoLabel.Text = "File Name: "+command[2]+ "\r\nFile
Size: " + long.Parse(command[3])  + " Bytes";
                                OKSF.Text = command[1] + " Accepted Sent File!";

                                        }


                                else if (command[0].ToUpper().Equals("QUITS") == true)
                                        {
                                int index = -1;
                                for (int i = 0; i < this.userList.Count; i++)
                                                {
if (((Clients)this.userList[i]).NickName.ToUpper().Equals(command[1].ToUpper()) == true)
                                                {
                                index = i;
                                break;
                                                }
                                                else
                                                {
                                                        continue;
                                                }
                                        }
                                        if (index != -1)
                                                {
                                        this.userList.RemoveAt(index);
                                        this.listBox.Items.Clear();
                                        this.listBox.BeginUpdate();
                                        for (int i = 0; i < this.userList.Count; i++)
                                                {

        this.listBox.Items.Add(this.userList[i].ToString());
                                                }
                                this.listBox.EndUpdate();
                                textBox.AppendText(textBox.SelectedText+ "\r\n");
                                textBox.SelectionStart = textBox.TextLength -1;
                                textBox.SelectionColor = Color.Blue;
                                textBox.SelectedText = ("<< " + command[1] + " Has Left
Chat... >>");

                                                }
                                                else
                                                {
                                                }
                                        }
                                else if (command[0].ToUpper().Equals("SHUTDOWN") == true)
                                        {
                                MessageBox.Show("Server Has Been Shutdown.", "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                                this.ClientStatus = ClientStatus.Disconnected;
                                this.userList.Clear();
                                this.listBox.Items.Clear();
                                this.textBox.Text = " ";
                                if (this.thread != null && this.thread.ThreadState ==
System.Threading.ThreadState.Running)
                                                {
                                this.thread.Abort();
                                                }
                                        }
                                        else
                                        {
                                        }
                                sock.BeginReceive(this.buffer, 0, this.buffer.Length,
SocketFlags.None, new AsyncCallback(this.ReceiveMode), sock);
                                }
                                else
                                {
                                }
                        }
                        catch (Exception e)
                        {
                                MessageBox.Show(e.Message.ToString());
                        }
```

```
        }

        public class clsUser32DLL

        {

                [DllImport("user32.dll")]

                static extern int FlashWindow(int hWnd, int bInvert);

                public static int FlashWindowOnce(int hWnd, int bInvert)
                {

                // call user32.dll
                return FlashWindow(hWnd, bInvert);

                }

        }

        public void SendData(string message)

        {
                try
                {
                        if (this.ClientStatus == ClientStatus.Connected)
                        {
                        byte[] send = this.ASCII.GetBytes(message.ToCharArray());
                        this.socket.BeginSend(send, 0, send.Length,
SocketFlags.None, new AsyncCallback(this.SendMessage), this.socket);
                        }
                        else
                        {
                                return ;
                        }
                }
                catch (Exception e)
                {
                        MessageBox.Show(e.Message.ToString());
                }
        }
        private void SendMessage(IAsyncResult ar)
        {
                try
                {
                        Socket sock = (Socket)ar.AsyncState;
                        sock.EndSend(ar);
                }
                catch (Exception e)
                {
                        MessageBox.Show(e.Message.ToString());
                }
        }
        private void messageBox_KeyPress(object sender, KeyPressEventArgs e)
        {
                try
                {
                        if ((int)e.KeyChar == 13)
                        {
                        if ((this.ClientStatus == ClientStatus.Connected)&&
(this.messageBox.Text.Trim().Equals("") == false))                              {
                                        e.Handled=true;


                        string Colorz =
TypeDescriptor.GetConverter(typeof(Color)).ConvertToString(ButtonColor.Color);
                        int myFontSize2 =
Int32.Parse(comboBoxFontSize.SelectedItem.ToString());
                        string message = "MSG^" + this.messageBox.Text.Trim()
+"^"+myFontSize2+"^"+fontComboBox.Text+"^"+Colorz +"^"+
this.m_bold.CheckState.ToString()+"^"+ this.m_italic.CheckState.ToString()+"^"+
this.m_under.CheckState.ToString();
                        this.SendData(message);
                        this.messageBox.Text = "";
```

```
                                this.messageBox.Focus();

                                        }
                                        else
                                        {
                                                return ;
                                        }
                                }
                                else
                                {
                                        return ;
                                }
                        }
                        catch (Exception ex)
                        {
                                MessageBox.Show(ex.Message.ToString());
                        }
                }

        private void listBox_MouseMove(object sender, MouseEventArgs e)
        {
        Point MousePositionInClientCoords = new Point(e.X, e.Y);
        int indexUnderTheMouse = this.listBox.IndexFromPoint(MousePositionInClientCoords);
        if (indexUnderTheMouse > -1)
                {
                string s = this.listBox.Items[indexUnderTheMouse].ToString();
                Graphics g = this.listBox.CreateGraphics();
                if (g.MeasureString(s, this.listBox.Font).Width >
this.listBox.ClientRectangle.Width)


                                {

                        this.toolTipCClient.SetToolTip(this.listBox, s);

                                }
                        else
                                {
                        this.toolTipCClient.SetToolTip(this.listBox, "") ;
                                }

                        g.Dispose();
                }
        }


                private void ButtonDown(object sender, MouseEventArgs mea)
                {
                        int nDx = 0; //This will be the 0 based index of the row clicked on
                        nDx = mea.Y / listBox.ItemHeight; //Get Index of the item selected

                        if (mea.Button == MouseButtons.Left && listBox.Items.Count > nDx)
                        {
                                listBox.SetSelected(nDx,true);

                                if (mea.Clicks == 2)
                                {

                                        if (this.listBox.SelectedIndex != -1)
                                        {

                                                if (this.listBox.SelectedItem.ToString() !=
this.Text.Substring(33))

                                                {
                                                        PTalk form = new PTalk(this,
this.listBox.SelectedItem.ToString());

                                                        if
(!userarray.ContainsKey(this.listBox.SelectedItem.ToString()))
                                                        {

                                                                userarray.Add (
this.listBox.SelectedItem.ToString(), form);
```

```
                                                        form.Show();

                                                }
                                                else
                                                {

        ((PTalk)userarray[this.listBox.SelectedItem.ToString()]).Activate();

        ((PTalk)userarray[this.listBox.SelectedItem.ToString()]).WindowState =
FormWindowState.Normal;

                                                }

                                        }
                                        else
                                        {
MessageBox.Show("Do You Want To Talk To Yourself??", "Problem With Mybook IM
User",MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

                                        }

                                        }


                                }
                                }
                                else  if (mea.Button == MouseButtons.Right &&
listBox.Items.Count > nDx)
                                {
                                        listBox.SetSelected(nDx,true);


                                        popUpMenu =new ContextMenu();
                                        this.popUpMenu.MenuItems.Add(new
MyBookMenuItem("Private Talk", new EventHandler(this.contMenu_PTalk)));
                                        this.popUpMenu.MenuItems.Add(new
MyBookMenuItem("Send File", new EventHandler(this.contMenu_PTalk)));
                                        this.ContextMenu = popUpMenu;
                                }


                        }


                private void MenuExit_Click(object sender, EventArgs e)
                {

                        try

                        {
                Process [] processes = Process.GetProcessesByName ("javaw");
                                if (processes.Length == 1)
                                {

                                        javaw_process.Kill();
                                        System.Environment.Exit(-2147480000);

                                }
                                else
                                {
                                        System.Environment.Exit(-2147480000);
                                }

                        }
                        catch (Exception ex)
                        {
                                MessageBox.Show(ex.Message.ToString());
                        }
                }
        public void PTalk_Closing(object sender, System.ComponentModel.CancelEventArgs e)
                {

                                        e.Cancel = false;
```

```
        userarray.Remove(((String)((PTalk)sender).Text).Substring(19));

                }

                private void contMenu_PTalk(object sender, EventArgs e)
                {
                        try
                        {
                                MenuItem miClicked = (MenuItem)sender;
                                string item = miClicked.Text;

                                                if (this.listBox.SelectedIndex != -1)
                                                {
                                                        if (item == "Private Talk")
                                                        {
                                                                if
(this.listBox.SelectedItem.ToString() != this.Text.Substring(33) && item == "Private
Talk")

                                                                {
                                                                PTalk form = new PTalk(this,
this.listBox.SelectedItem.ToString());

                                                                        if
(!userarray.ContainsKey(this.listBox.SelectedItem.ToString()))
                                                                        {

                                                                                userarray.Add
( this.listBox.SelectedItem.ToString(), form);
                                                                                form.Show();

                                                                        }
                                                                        else
                                                                        {


        ((PTalk)userarray[this.listBox.SelectedItem.ToString()]).Activate();

        ((PTalk)userarray[this.listBox.SelectedItem.ToString()]).WindowState =
FormWindowState.Normal;
                                                                        }
                                                                }
                                                                else
                                                                {
MessageBox.Show("Do You Want To Talk To Yourself??", "Problem With Mybook IM
User",MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

                                                                }
                                                        }

                                                        else if (item == "Send File")
                                                        {


if (this.listBox.SelectedItem.ToString() != this.Text.Substring(33) && item == "Send
File")
                                                                        {
                                SendFile SendFileForm = new SendFile(this);
                                SendFileForm.Show();
                                SendFileForm.SendFileFrom.Text = usrname;
                                SendFileForm.SendFileTo.Text = listBox.Text;
                                                                        }
                                                else
                                                {
MessageBox.Show("Do You Want To Send File To Yourself??", "Problem With Mybook IM
User",MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

                                                }
                                                }
                                                        else
                                                        {

                                                                return;
```

```
                                                }


                                        }

                                else
                        {
                        return;
                        }
                }
                catch (Exception ex)
                {
                MessageBox.Show(ex.Message.ToString());
                }

}

                private void StartProcessing(object sender, EventArgs e)
                {
                        try
                        {
                                this.thread = new Thread(new ThreadStart(this.Connect));
                                this.thread.Priority = ThreadPriority.AboveNormal;
                                this.thread.Name = "Client Socket";
                                this.thread.Start();
                        }
                        catch (Exception ex)
                        {
                                MessageBox.Show(ex.Message.ToString());
                        }
                }

        [STAThread]
        public static void Main(string[] args)
                {

                        Process [] processes = Process.GetProcessesByName ("MyBookIM");
                        if (processes.Length != 1)
                        {

MessageBox.Show("An Other Instance Of MyBook IM Already Running!!", "Mybook IM
Application Already Running..",MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                                return;

                        }


                        try
                        {
                                CClient zz = new CClient();
                                zz.WindowState = FormWindowState.Normal;
                                zz.Show();
                                zz.Process1 += new ProcessHandler(zz.StartProcessing);
                                zz.nickName = zz.AuthorizeForm.Authusrname;
                                IPHostEntry hosts = Dns.Resolve("216.80.113.39");
                                IPAddress ip = hosts.AddressList[0];
                                zz.AuthServerPort = zz.AuthorizeForm.ServerPort;
                                zz.EndPoint = new IPEndPoint(ip,
Int32.Parse(zz.AuthServerPort.ToString()));
                                zz.OnProcess(null);
                                if (zz.ClientStatus != ClientStatus.Connected)
                                {
                                        try
                                        {

                                                zz.usrname = zz.AuthorizeForm.Authusrname;

                                                string userXP =
System.Security.Principal.WindowsIdentity.GetCurrent().Name.ToString();
                                                zz.userXPxx =
userXP.Substring(userXP.LastIndexOf("\\")+1);
                                                if (!File.Exists("c:\\Documents and
Settings\\"+zz.userXPxx+"\\.java.policy"))
                                                {
```

```
                                        File.Copy(".java.policy","c:\\Documents and
Settings\\"+zz.userXPxx+"\\.java.policy",true);
                                        }
                                        else
                                        {
                                                File.Delete("c:\\Documents and
Settings\\"+zz.userXPxx+"\\.java.policy");

       File.Copy(".java.policy","c:\\Documents and Settings\\"+zz.userXPxx+
"\\.java.policy",true);
                                        }

                                string directory = "c:\\public_html\\" + zz.usrname;

                                        if (!Directory.Exists(directory))
                                        {

                                                Directory.CreateDirectory(directory);
                                                ProcessStartInfo startInfo = new
ProcessStartInfo("javaw.exe","dhq.web.WebPeer -name "+zz.usrname+" -dhq mybook.uc.edu -
port 8081");

                                                startInfo.WorkingDirectory = "C:\\";
                                                zz.javaw_process =
Process.Start(startInfo);

                                        }
                                        else
                                        {

                                                ProcessStartInfo startInfo = new
ProcessStartInfo("javaw.exe","dhq.web.WebPeer -name "+zz.usrname+" -dhq mybook.uc.edu -
port 8081");
                                                startInfo.WorkingDirectory = "C:\\";
                                                zz.javaw_process =
Process.Start(startInfo);

                                        }

                                        zz.messageBox.Enabled=true;
                                        zz.Text="MyBook Instant Messenger Client:
"+zz.usrname;
                                        zz.MenuTools.Enabled = true;
                                        zz.MenuExit.Enabled = true;
                                        zz.MenuEdit.Enabled = true;
                                        zz.MenuHelp.Enabled = true;
                                        zz.ChatSendButton.Enabled = true;
                                        zz.comboBoxFontSize.Enabled = true;
                                        zz.fontComboBox.Enabled = true;
                                        zz.ButtonColor.Enabled = true;
                                        zz.m_bold.Enabled = true;
                                        zz.m_italic.Enabled = true;
                                        zz.m_under.Enabled = true;
                                        zz.messageBox.Focus();
                                        zz.MenuConnect.Enabled = false;

                                        while (true)
                                        {
                                                Application.Run();
                                        }

                                }
                                catch (Exception ex)
                                {

                                        Console.WriteLine(ex.ToString());
                                        zz.messageBox.Enabled=true;
                                        zz.Text="MyBook Instant Messenger Client:
"+zz.usrname;
                                        zz.MenuTools.Enabled = true;
                                        zz.MenuExit.Enabled = true;
                                        zz.MenuEdit.Enabled = true;
                                        zz.MenuHelp.Enabled = true;
                                        zz.ChatSendButton.Enabled = true;
                                        zz.comboBoxFontSize.Enabled = true;
```

```
                                        zz.fontComboBox.Enabled = true;
                                        zz.ButtonColor.Enabled = true;
                                        zz.m_bold.Enabled = true;
                                        zz.m_italic.Enabled = true;
                                        zz.m_under.Enabled = true;
                                        zz.messageBox.Focus();
                                        zz.MenuConnect.Enabled = false;
                                        MessageBox.Show("Application needs Java
Runtime Environment. Go To http://java.sun.com and download it there!\nYou might
encounter problems while sending or receiving files, if no JRE is present.", "Problem
With Java Runtime Environment",MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                                        while (true)
                                        {
                                                Application.Run();
                                        }
                                }

                        }


                        else
                        {

                                MessageBox.Show("You Are Not Connected To The
Server.", "Warning", MessageBoxButtons.OK, MessageBoxIcon.Error);
                        }


                }
        catch (Exception e)
        {
                MessageBox.Show(e.Message.ToString());
        }

}

        private void ChatSendButton_Click(object sender, System.EventArgs e)
        {

                if ((this.ClientStatus == ClientStatus.Connected) &&
(this.messageBox.Text.Trim().Equals("") == false))
                {

                        string Colorz =
TypeDescriptor.GetConverter(typeof(Color)).ConvertToString(ButtonColor.Color);
                        int myFontSize2 =
Int32.Parse(comboBoxFontSize.SelectedItem.ToString());
                        string message = "MSG^" + this.messageBox.Text.Trim()
+"^"+myFontSize2+"^"+fontComboBox.Text+"^"+Colorz +"^"+
this.m_bold.CheckState.ToString()+"^"+ this.m_italic.CheckState.ToString()+"^"+
this.m_under.CheckState.ToString();
                        this.SendData(message);
                        this.messageBox.Text = "";

                        this.messageBox.Focus();
                        }
                else
                {
                        this.messageBox.Focus();
                        return;
                }

        }

        private void MenuAbout_Click(object sender, System.EventArgs e)
        {
                About AboutForm = new About();
                AboutForm.ShowDialog();
        }

        private void MenuHelpTopic_Click(object sender, System.EventArgs e)
        {
                if (File.Exists("MBKIM_Help.txt"))
                {
```

```
        System.Diagnostics.Process.Start("wordpad.exe","MBKIM_Help.txt");
                        }
                        else
                        {
                                MessageBox.Show("Try To Reinstall MyBook IM To Obtain Help
File!!", "Missing 'Help.txt' File..",MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                        }
                }

                static int i = 0;
                private void MenuSaveLogFile_Click(object sender, System.EventArgs e)
                {
                        Process[] myProcesses;
                        // Returns array containing all instances of Wordpad.
                        myProcesses = Process.GetProcessesByName("Wordpad");


                        foreach (Process myProcess in myProcesses)
                        {
                                Console.WriteLine(myProcess.ToString());
                                MessageBox.Show("Please Exit WordPad Application...",
"WordPad application still running??",MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                                i=i+1;
                        }

                        if (i == 0)
                        {

                                if (File.Exists("MyLog.txt"))
                                {
                                        string formattedDate;
                                        string formattedTime;
                                        StreamWriter SW;
                                        formattedDate = DateTime.Now.ToString("MM/dd/yyyy");
                                        formattedTime = DateTime.Now.ToString("hh:mm:ss
tt");
                                        SW=File.AppendText("MyLog.txt");
                                        SW.WriteLine("Log Creation Time: "+formattedTime+",
Date: "+ formattedDate);
                                        SW.WriteLine(textBox.Text.Substring(113));
                                        SW.WriteLine("\r\n");
                                        SW.Close();

                                }

                                else
                                {
                                        StreamWriter SW;
                                        SW=File.CreateText("MyLog.txt");
                                        SW.Close();
                                        string formattedDate;
                                        string formattedTime;
                                        formattedDate = DateTime.Now.ToString("MM/dd/yyyy");
                                        formattedTime = DateTime.Now.ToString("hh:mm:ss
tt");
                                        SW=File.AppendText("MyLog.txt");
                                        SW.WriteLine("New Log File Creation Time:
"+formattedTime+", Date: "+ formattedDate+"\r\n");
                                        SW.WriteLine("\r\n");
                                        SW.WriteLine("Log Creation Time: "+formattedTime+",
Date: "+ formattedDate);
                                        SW.WriteLine(textBox.Text.Substring(113));
                                        SW.WriteLine("\r\n");
                                        SW.Close();

                                }

                        }
                        i=0;
                }

                static int g = 0;
                private void MenuOpenLogFile_Click(object sender, System.EventArgs e)
```

```
		{
			Process[] myProcesses;
			// Returns array containing all instances of Wordpad.
			myProcesses = Process.GetProcessesByName("Wordpad");


			foreach (Process myProcess in myProcesses)
			{
				Console.WriteLine(myProcess.ToString());
				MessageBox.Show("Please Exit WordPad Application...",
"WordPad application still running??",MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
				g=g+1;
			}

			if (g == 0)
			{

				if (File.Exists("MyLog.txt"))
				{

	System.Diagnostics.Process.Start("wordpad.exe","MyLog.txt");
				}
				else
				{
					StreamWriter SW;
					string formattedDate;
					string formattedTime;
					formattedDate = DateTime.Now.ToString("MM/dd/yyyy");
					formattedTime = DateTime.Now.ToString("hh:mm:ss
tt");
					SW=File.CreateText("MyLog.txt");
					SW.WriteLine("New Log File Creation Time:
"+formattedTime+", Date: "+ formattedDate+"\r\n");
					SW.Close();

	System.Diagnostics.Process.Start("wordpad.exe","MyLog.txt");

				}
			}
		}

		static int j = 0;
		private void MenuClearLogFile_Click(object sender, System.EventArgs e)
		{
			Process[] myProcesses;
			// Returns array containing all instances of Wordpad.
			myProcesses = Process.GetProcessesByName("Wordpad");

			foreach (Process myProcess in myProcesses)
			{
				Console.WriteLine(myProcess.ToString());
				MessageBox.Show("Please Exit NotePad Application...",
"WordPad application still running??",MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
				j=j+1;
			}

			if (j == 0)
			{

				File.Delete("MyLog.txt");
				StreamWriter SW;
				string formattedDate;
				string formattedTime;
				formattedDate = DateTime.Now.ToString("MM/dd/yyyy");
				formattedTime = DateTime.Now.ToString("hh:mm:ss tt");
				SW=File.CreateText("MyLog.txt");
				SW.WriteLine("New Log File Creation Time:
"+formattedTime+", Date: "+ formattedDate+"\r\n");
				SW.Close();

			}
			j=0;
		}
```

```
		public System.Diagnostics.Process p = new System.Diagnostics.Process();
		private void MainReceiveMessageBox_LinkClicked(object sender,
System.Windows.Forms.LinkClickedEventArgs e)
		{

			// Call Process.Start method to open a browser
			// with link text as URL.
			p = System.Diagnostics.Process.Start(e.LinkText);

		}

		public void StopWebProcess()
		{
			p.Kill();
		}

		public void MenuSendFile_Click(object sender, System.EventArgs e)
		{
			if (listBox.SelectedIndex != -1)
			{
				if (this.listBox.SelectedItem.ToString() !=
this.Text.Substring(33))

				{



						SendFile SendFileForm = new SendFile(this);
						SendFileForm.Show();
						SendFileForm.SendFileFrom.Text = usrname;
						SendFileForm.SendFileTo.Text = listBox.Text;


				}
				else
				{
MessageBox.Show("Do You Want To Send A File To Yourself??", "Problem With Mybook IM
User",MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

				}

			}
			else
			{
MessageBox.Show("Choose A User You Want To Send File To!", "User Not Chosen
..",MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
			}
		}

		public void OnChanged( object sender, System.EventArgs e )
		{

			messageBox.ForeColor = ButtonColor.Color ;

		}


		private void MenuShareFolder_Click(object sender, System.EventArgs e)
		{
			ShareFolder ShareFolderForm = new ShareFolder();
			ShareFolderForm.ShowDialog();
		}

		private void MenuSelAll_Click(object sender, System.EventArgs e)
		{

			if(textBox.SelectionLength >=0 && textBox.Focused == true)

			{
				// Select all text in the text box.
				textBox.SelectAll();

			}
```

```
                        // Determine if any text is selected in the TextBox
control.
                    else if (messageBox.SelectionLength >=0 && messageBox.Text != "" &&
messageBox.Focused == true)
                         {
                        if (textBox.SelectionLength >0)
                        {

                                textBox.SelectionLength =0;
                                messageBox.SelectAll();
                        }
                        else
                        {
                        messageBox.SelectAll();
                        }
                        }

                    else
                    {
                            return;
                    }

                }

                private void MenuCut_Click(object sender, System.EventArgs e)
                {
                    if (textBox.SelectionLength >0)
                    {
                            textBox.Copy();
                    }
                    else if (messageBox.SelectionLength >0)
                    {
                    messageBox.Cut();
                    }
                }

                private void MenuCopy_Click(object sender, System.EventArgs e)
                {
                    if (textBox.SelectionLength >0)
                    {
                            textBox.Copy();
                    }
                    else if (messageBox.SelectionLength >0)
                    {
                            messageBox.Copy();
                    }
                }

                private void MenuPaste_Click(object sender, System.EventArgs e)
                {

                    messageBox.Paste();

                }


                public delegate void ProcessHandler(object sender, EventArgs e);
                public event ProcessHandler Process1 = null;
                private void OnProcess(EventArgs e)
                {
                    if (this.Process1 != null)
                    {
                            this.Process1(this, e);
                    }
                    else
                    {
                            return ;
                    }
                }

                private void m_under_CheckedChanged(object sender, System.EventArgs e)
                {

                    FontStyle style = FontStyle.Regular;
```

```
        if( m_bold.Checked )
                style |= FontStyle.Bold;
        if( m_italic.Checked )
                style |= FontStyle.Italic;
        if( m_under.Checked )
                style |= FontStyle.Underline;

        string name = messageBox.SelectionFont.FontFamily.Name;
        float size = messageBox.SelectionFont.Size;

        messageBox.SelectionFont = new Font(
                name,
                size,
                style);

        messageBox.Focus();
}

private void m_bold_CheckedChanged(object sender, System.EventArgs e)
{
        FontStyle style = FontStyle.Regular;

        if( m_bold.Checked )
                style |= FontStyle.Bold;
        if( m_italic.Checked )
                style |= FontStyle.Italic;
        if( m_under.Checked )
                style |= FontStyle.Underline;

        string name = messageBox.SelectionFont.FontFamily.Name;
        float size = messageBox.SelectionFont.Size;

        messageBox.SelectionFont = new Font(
                name,
                size,
                style);

        messageBox.Focus();
}

private void m_italic_CheckedChanged(object sender, System.EventArgs e)
{


        FontStyle style = FontStyle.Regular;

        if( m_bold.Checked )
                style |= FontStyle.Bold;
        if( m_italic.Checked )
                style |= FontStyle.Italic;
        if( m_under.Checked )
                style |= FontStyle.Underline;

        string name = messageBox.SelectionFont.FontFamily.Name;
        float size = messageBox.SelectionFont.Size;

        messageBox.SelectionFont = new Font(
                name,
                size,
                style);


        messageBox.Focus();
}
public void OnTxtSelectionChanged( object sender, EventArgs e)
{


        if( messageBox.SelectionFont.Bold)
                m_bold.Checked = true;
        else
                m_bold.Checked = false;
```

```
                        if( messageBox.SelectionFont.Italic)
                                m_italic.Checked = true;
                        else
                                m_italic.Checked = false;

                        if( messageBox.SelectionFont.Underline)
                                m_under.Checked = true;
                        else
                                m_under.Checked = false;
                }

                private void MenuPrivateTalk_Click(object sender, System.EventArgs e)
                {

                        if (this.listBox.SelectedIndex != -1)
                        {
                                if (this.listBox.SelectedItem.ToString() !=
this.Text.Substring(33))

                                {
                                        PTalk form = new PTalk(this,
this.listBox.SelectedItem.ToString());

                                        if
(!userarray.ContainsKey(this.listBox.SelectedItem.ToString()))
                                        {
                                                userarray.Add (
this.listBox.SelectedItem.ToString(), form);
                                                form.Show();

                                        }
                                        else
                                        {

                                                return;
                                        }

                                }

                                else
                                {
                                        MessageBox.Show("Do You Want To Talk To Yourself??",
"Problem With Mybook IM User",MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

                                }
                        }
                        else
                        {
                                MessageBox.Show("Choose A User You Want To Talk To!", "User
Not Chosen ..",MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                        }

                }

                private void MenuHelpSlides_Click(object sender, System.EventArgs e)
                {

                        if (File.Exists("MBKIM_HelpSlides.pps"))
                        {
                                System.Diagnostics.Process.Start("MBKIM_HelpSlides.pps");
                        }
                        else
                        {
                                MessageBox.Show("Try To Reinstall MyBook IM To Obtain Help
File!!", "Missing 'MBKIM_HelpSlides.ppt' File..",MessageBoxButtons.OK,
MessageBoxIcon.Exclamation);
                        }

                }


        }

        internal sealed class PTalk : Form
```

```
	{
		public Container components = null;
		public RichTextBox PrivateTalkReceiveMessageBox = null;
		public TextBox PrivateTalkSendMessageBox = null;
		public Button PrivateTalkSendButton = null;
		public MainMenu mainPTMenu = null;
		public MyBookIM.MyBookMenuItem menuSendFile;
		public MyBookIM.MyBookMenuItem PMenuLogFile;
		public MyBookIM.MyBookMenuItem PMenuSaveLogFile;
		public MyBookIM.MyBookMenuItem PMenuOpenLogFile;
		public MyBookIM.MyBookMenuItem PMenuClearLogFile;
		public MyBookIM.MyBookMenuItem PMenuEdit;
		public MyBookIM.MyBookMenuItem PMenuCut;
		public MyBookIM.MyBookMenuItem PMenuCopy;
		public MyBookIM.MyBookMenuItem PMenuPaste;
		public MyBookIM.MyBookMenuItem PMenuAtstarpe2;
		public MyBookIM.MyBookMenuItem PMenuSelAll;
		private CClient chatClient = null;
		public string destNick = null;
		public PTalk() : base()
		{
		}
		internal PTalk(CClient chatClient, string destNick)
		{
			this.chatClient = chatClient;
			this.InitializeComponent();
			this.destNick = destNick;
			this.Text = "Private Talk With: " +destNick;

		}
		protected override void Dispose(bool disposing)
		{
			if ((disposing == true) && (this.components != null))
			{
				this.components.Dispose();
			}
			base.Dispose(disposing);
		}

		public void InitializeComponent()

		public void PMenuSendFile_Click (object sender, System.EventArgs e)
		{


				SendFile SendFileForm = new SendFile(this.chatClient);
				SendFileForm.Show();
				SendFileForm.SendFileFrom.Text = this.chatClient.usrname;
				SendFileForm.SendFileTo.Text = this.Text.Substring(19);


		}

		private void PrivateTalkSendMessageBox_KeyPress(object sender,
KeyPressEventArgs e)
		{
			try
			{
				if ((int)e.KeyChar == 13)

				{

					if ((this.chatClient.ClientStatus ==
ClientStatus.Connected) && (this.PrivateTalkSendMessageBox.Text.Trim().Equals("") ==
false))
					{
						e.Handled=true;

						if
(chatClient.command[0].ToUpper().Equals("QUITS") == false && chatClient.command[1] ==
this.Text.Substring(19))
						{
							string msg = "PRIVMSG^" +
this.destNick + "^" + this.PrivateTalkSendMessageBox.Text.Trim();
```

```
                                                this.chatClient.SendData(msg);

        this.PrivateTalkReceiveMessageBox.AppendText("<" + chatClient.usrname + "> " +
this.PrivateTalkSendMessageBox.Text + "\r\n");
                                                this.PrivateTalkSendMessageBox.Text =
"";

        this.PrivateTalkSendMessageBox.Focus();
                                                }
                                                else if
(chatClient.command[0].ToUpper().Equals("QUITS") == true && chatClient.command[1] !=
this.Text.Substring(19))
                                                {
                                                        string msg = "PRIVMSG^" +
this.destNick + "^" + this.PrivateTalkSendMessageBox.Text.Trim();

                                                this.chatClient.SendData(msg);

        this.PrivateTalkReceiveMessageBox.AppendText("<" + chatClient.usrname + "> " +
this.PrivateTalkSendMessageBox.Text + "\r\n");
                                                this.PrivateTalkSendMessageBox.Text =
"";

        this.PrivateTalkSendMessageBox.Focus();
                                                }
                                                else if
(chatClient.command[0].ToUpper().Equals("QUITS") == true && chatClient.command[1] ==
this.Text.Substring(19))
                                                {

        this.PrivateTalkReceiveMessageBox.AppendText("<" + chatClient.usrname + "> " +
this.PrivateTalkSendMessageBox.Text + "\r\n");

        this.PrivateTalkReceiveMessageBox.AppendText("<< " + chatClient.command[1] + " Has
Left Chat.. >>\r\n");
                                                this.PrivateTalkSendMessageBox.Text =
"";

        this.PrivateTalkSendMessageBox.Focus();
                                                }
                                                else if
(chatClient.command[0].ToUpper().Equals("QUITS") == false && chatClient.command[1] !=
this.Text.Substring(19))
                                                {
                                                        string msg = "PRIVMSG^" +
this.destNick + "^" + this.PrivateTalkSendMessageBox.Text.Trim();
                                                this.chatClient.SendData(msg);

        this.PrivateTalkReceiveMessageBox.AppendText("<" + chatClient.usrname + "> " +
this.PrivateTalkSendMessageBox.Text + "\r\n");
                                                this.PrivateTalkSendMessageBox.Text =
"";

        this.PrivateTalkSendMessageBox.Focus();
                                                }

                                        }
                                        else
                                        {
                                                return ;
                                        }
                                }
                                else
                                {
                                        return ;
                                }
                        }
                        catch (Exception ex)
                        {
                                MessageBox.Show(ex.Message.ToString());
                        }
                }
```

```
                static int m = 0;
                private void PMenuSaveLogFile_Click(object sender, System.EventArgs e)
                {
                        Process[] myProcesses;
                        // Returns array containing all instances of Wordpad.
                        myProcesses = Process.GetProcessesByName("Wordpad");

                        foreach (Process myProcess in myProcesses)
                        {
                                Console.WriteLine(myProcess.ToString());
                                MessageBox.Show("Please Exit WordPad Application...",
"WordPad application still running??",MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                                m=m+1;
                        }

                        if (m == 0)
                        {

                                if (File.Exists(destNick+"_Log.txt"))
                                {
                                string formattedDate;
                                string formattedTime;
                                StreamWriter SW;
                                formattedDate = DateTime.Now.ToString("MM/dd/yyyy");
                                formattedTime = DateTime.Now.ToString("hh:mm:ss tt");
                                SW=File.AppendText(destNick+"_Log.txt");
                                SW.WriteLine("Log Creation Time: "+formattedTime+", Date:
"+ formattedDate);
                                SW.WriteLine(PrivateTalkReceiveMessageBox.Text);
                                SW.WriteLine("\r\n");
                                SW.Close();

                                }

                                else
                                {
                                StreamWriter SW;
                                SW=File.CreateText(destNick+"_Log.txt");
                                SW.Close();
                                string formattedDate;
                                string formattedTime;
                                formattedDate = DateTime.Now.ToString("MM/dd/yyyy");
                                formattedTime = DateTime.Now.ToString("hh:mm:ss tt");
                                SW=File.AppendText(destNick+"_Log.txt");
                                SW.WriteLine("New Log File Creation Time:
"+formattedTime+", Date: "+ formattedDate+"\r\n");
                                SW.WriteLine("\r\n");
                                SW.WriteLine("Log Creation Time: "+formattedTime+", Date:
"+ formattedDate);
                                SW.WriteLine(PrivateTalkReceiveMessageBox.Text);
                                SW.WriteLine("\r\n");
                                SW.Close();

                                }

                        }
                        m=0;
                }
                static int f =0;
                private void PMenuOpenLogFile_Click(object sender, System.EventArgs e)

                {
                        Process[] myProcesses;
                        // Returns array containing all instances of Wordpad.
                        myProcesses = Process.GetProcessesByName("Wordpad");

                        foreach (Process myProcess in myProcesses)
                        {
                                Console.WriteLine(myProcess.ToString());
                                MessageBox.Show("Please Exit WordPad Application...",
"WordPad application still running??",MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                                f=f+1;
                        }
```

```
                        if (f == 0)
                        {

                                if (File.Exists(destNick+"_Log.txt"))
                                {

        System.Diagnostics.Process.Start("Wordpad.exe",destNick+"_Log.txt");
                                }
                                else
                                {
                                StreamWriter SW;
                                string formattedDate;
                                string formattedTime;
                                formattedDate = DateTime.Now.ToString("MM/dd/yyyy");
                                formattedTime = DateTime.Now.ToString("hh:mm:ss tt");
                                SW=File.CreateText(destNick+"_Log.txt");
                                SW.WriteLine("New Log File Creation Time:
"+formattedTime+", Date: "+ formattedDate+"\r\n");
                                SW.Close();

        System.Diagnostics.Process.Start("Wordpad.exe",destNick+"_Log.txt");

                                }
                        }
                f=0;
                }

                static int n = 0;
                private void PMenuClearLogFile_Click(object sender, System.EventArgs e)
                {
                        Process[] myProcesses;
                        // Returns array containing all instances of Wordpad.
                        myProcesses = Process.GetProcessesByName("Wordpad");

                        foreach (Process myProcess in myProcesses)
                        {
                                Console.WriteLine(myProcess.ToString());
                                MessageBox.Show("Please Exit WordPad Application...",
"WordPad application still running??",MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                                n=n+1;
                        }

                        if (n == 0)
                        {

                                File.Delete(destNick+"_Log.txt");
                                StreamWriter SW;
                                string formattedDate;
                                string formattedTime;
                                formattedDate = DateTime.Now.ToString("MM/dd/yyyy");
                                formattedTime = DateTime.Now.ToString("hh:mm:ss tt");
                                SW=File.CreateText(destNick+"_Log.txt");
                                SW.WriteLine("New Log File Creation Time:
"+formattedTime+", Date: "+ formattedDate+"\r\n");
                                SW.Close();

                        }
                        n=0;
                }
                private void PMenuSelAll_Click(object sender, System.EventArgs e)
                {

                        if(PrivateTalkReceiveMessageBox.SelectionLength >=0 &&
PrivateTalkReceiveMessageBox.Focused == true)

                        {
                        // Select all text in the text box.
                        PrivateTalkReceiveMessageBox.SelectAll();

                        }
                        // Determine if any text is selected in the TextBox control.
                        else if (PrivateTalkSendMessageBox.SelectionLength >=0 &&
PrivateTalkSendMessageBox.Text != "" && PrivateTalkSendMessageBox.Focused == true)
                        {
```

```
                        if (PrivateTalkReceiveMessageBox.SelectionLength >0)
                        {

                        PrivateTalkReceiveMessageBox.SelectionLength =0;
                        PrivateTalkSendMessageBox.SelectAll();
                        }
                        else
                        {
                        PrivateTalkSendMessageBox.SelectAll();
                        }
                }

                else
                {
                        return;
                }

        }

        private void PMenuCut_Click(object sender, System.EventArgs e)
        {
                if (PrivateTalkReceiveMessageBox.SelectionLength >0)
                {
                        PrivateTalkReceiveMessageBox.Copy();
                }
                else if (PrivateTalkSendMessageBox.SelectionLength >0)
                {
                        PrivateTalkSendMessageBox.Cut();
                }
        }

        private void PMenuCopy_Click(object sender, System.EventArgs e)
        {
                if (PrivateTalkReceiveMessageBox.SelectionLength >0)
                {
                        PrivateTalkReceiveMessageBox.Copy();
                }
                else if (PrivateTalkSendMessageBox.SelectionLength >0)
                {
                        PrivateTalkSendMessageBox.Copy();
                }
        }
        private void PMenuPaste_Click(object sender, System.EventArgs e)
        {

                PrivateTalkSendMessageBox.Paste();

        }



        public System.Diagnostics.Process pr = new System.Diagnostics.Process();
        private void PrivateTalkReceiveMessageBox_LinkClicked(object sender,
System.Windows.Forms.LinkClickedEventArgs e)
        {

                 //Call Process.Start method to open a browser
                 //with link text as URL.
                pr = System.Diagnostics.Process.Start(e.LinkText);

        }

        public void StopWebProcess()
        {
                pr.Kill();
        }


        public void PrivateTalkSendButton_Click(object sender, EventArgs e)
        {
                try
                {
```

```
                                if ((this.chatClient.ClientStatus ==
ClientStatus.Connected) && (this.PrivateTalkSendMessageBox.Text.Trim().Equals("") ==
false))
                                {
                                        if (chatClient.command[0].ToUpper().Equals("QUITS")
== false && chatClient.command[1] == this.Text.Substring(19))
                                        {
                                                string msg = "PRIVMSG^" + this.destNick +
"^" + this.PrivateTalkSendMessageBox.Text.Trim();

                                                this.chatClient.SendData(msg);

        this.PrivateTalkReceiveMessageBox.AppendText("<" + chatClient.usrname + "> " +
this.PrivateTalkSendMessageBox.Text + "\r\n");
                                                this.PrivateTalkSendMessageBox.Text = "";
                                                this.PrivateTalkSendMessageBox.Focus();
                                        }
                                        else if
(chatClient.command[0].ToUpper().Equals("QUITS") == true && chatClient.command[1] !=
this.Text.Substring(19))
                                        {
                                                string msg = "PRIVMSG^" + this.destNick +
"^" + this.PrivateTalkSendMessageBox.Text.Trim();
                                                this.chatClient.SendData(msg);

        this.PrivateTalkReceiveMessageBox.AppendText("<" + chatClient.usrname + "> " +
this.PrivateTalkSendMessageBox.Text + "\r\n");
                                                this.PrivateTalkSendMessageBox.Text = "";
                                                this.PrivateTalkSendMessageBox.Focus();
                                        }
                                        else if
(chatClient.command[0].ToUpper().Equals("QUITS") == true && chatClient.command[1] ==
this.Text.Substring(19))
                                        {

        this.PrivateTalkReceiveMessageBox.AppendText("<" + chatClient.usrname + "> " +
this.PrivateTalkSendMessageBox.Text + "\r\n");

        this.PrivateTalkReceiveMessageBox.AppendText("<< " + chatClient.command[1] + " Has
Left Chat.. >>\r\n");
                                                this.PrivateTalkSendMessageBox.Text = "";
                                                this.PrivateTalkSendMessageBox.Focus();
                                        }
                                        else if
(chatClient.command[0].ToUpper().Equals("QUITS") == false && chatClient.command[1] !=
this.Text.Substring(19))
                                        {
                                                 string msg = "PRIVMSG^" + this.destNick +
"^" + this.PrivateTalkSendMessageBox.Text.Trim();
                                                 this.chatClient.SendData(msg);

this.PrivateTalkReceiveMessageBox.AppendText("<" + chatClient.usrname + "> " +
this.PrivateTalkSendMessageBox.Text + "\r\n");
                                                 this.PrivateTalkSendMessageBox.Text = "";
                                                 this.PrivateTalkSendMessageBox.Focus();
                                        }

                                }
                                else
                                {
                                        this.PrivateTalkSendMessageBox.Focus();
                                        return;

                                }

                        }
                        catch (Exception ex)
                        {
                                MessageBox.Show(ex.Message.ToString());
                        }
                }
        }
}
```

About Form:

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;

namespace MyBookIM
{
	/// <summary>
	/// Summary description for About.
	/// </summary>
	public class About : System.Windows.Forms.Form
	{
		private System.Windows.Forms.Button button1;
		private System.Windows.Forms.Label label1;
		private System.Windows.Forms.LinkLabel MybookHomeLink;
		private System.Windows.Forms.ToolTip toolTip1;
		private System.ComponentModel.IContainer components;

		public About()
		{
			//
			// Required for Windows Form Designer support
			//
			InitializeComponent();

			//
			// TODO: Add any constructor code after InitializeComponent call
			//
		}

		/// <summary>
		/// Clean up any resources being used.
		/// </summary>
		protected override void Dispose( bool disposing )
		{
			if( disposing )
			{
				if(components != null)
				{
					components.Dispose();
				}
			}
			base.Dispose( disposing );
		}

		#region Windows Form Designer generated code
		/// <summary>
		/// Required method for Designer support - do not modify
		/// the contents of this method with the code editor.
		/// </summary>

		//Initialize components
		private void InitializeComponent()…
		#endregion

		//listens for Mybook.uc.edu link click
		private void MybookHomeLink_LinkClicked(object sender,
System.Windows.Forms.LinkLabelLinkClickedEventArgs e)
		{
			try
			{
				VisitLink();
			}
			catch (Exception ex )
			{
				MessageBox.Show("Unable To Open Link That Was Clicked.
Error:", ex.Message);
			}
		}
```

```
		public System.Diagnostics.Process p = new System.Diagnostics.Process();

		private void VisitLink()
		{
			// Change the color of the link text by setting LinkVisited to
true.
			MybookHomeLink.LinkVisited = true;
			toolTip1.SetToolTip(MybookHomeLink, "Go to http://mybook.uc.edu");

		//Call the Process.Start method to open the default browser
		//with a URL:
		p = System.Diagnostics.Process.Start("http://mybook.uc.edu");
		}
		public void StopWebProcess()
		{
			p.Kill();
		}




	}
}
```

Authorize Form:

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
using System.Data.OleDb;
using System.IO;
using System.Net;
using System.Threading;
using N = System.Net;
using System.Runtime.InteropServices;
using System.Data.SqlClient;
using System.Data.SqlTypes;
using System.Text;
using System.Net.Sockets;
using System.Security.Cryptography;


namespace MyBookIM
{
	/// <summary>
	/// Summary description for Authorize.
	/// </summary>
	public class Authorize : System.Windows.Forms.Form
	{
		public System.Windows.Forms.TextBox UserName;
		private System.Windows.Forms.Label UserNameLabel;
		private System.Windows.Forms.Label PasswordLabel;
		private System.Windows.Forms.Button AuthorizeButton;
		private System.Windows.Forms.Button CancelButton1;
		public System.Windows.Forms.Label AuthLabel;
		private System.Windows.Forms.Label WelcomeLabel;
		private System.Windows.Forms.Label Title;
		public System.Windows.Forms.TextBox Passwrd;
		private System.Windows.Forms.ToolTip toolTipAuthorize;
		private System.Windows.Forms.Button NewUserButton;
		private System.Windows.Forms.PictureBox pictureBox1;
		private System.Windows.Forms.PictureBox pictureBox2;
		private System.Windows.Forms.ComboBox ServerComboBox;
		public System.Windows.Forms.TextBox ServerPasswordTextBox;
		private System.Windows.Forms.Label labelCommunity;
		private System.Windows.Forms.Label LabelCommunityPassword;
		private System.Windows.Forms.Button ButtonNewCommunity;
		private System.ComponentModel.IContainer components;

		public Authorize()
```

```
		{
			//
			// Required for Windows Form Designer support
			//

			InitializeComponent();
			ServerComboBox.Text = "Main Public Community";

			//
			// TODO: Add any constructor code after InitializeComponent call
			//
		}

		/// <summary>
		/// Clean up any resources being used.
		/// </summary>
		protected override void Dispose( bool disposing )
		{
			if( disposing )
			{
				if(components != null)
				{
					components.Dispose();
				}
			}
			base.Dispose( disposing );
		}

		#region Windows Form Designer generated code
		/// <summary>
		/// Required method for Designer support - do not modify
		/// the contents of this method with the code editor.
		/// </summary>
		private void InitializeComponent()…
		#endregion

		public bool Result;

		//check if user's name and/or password are valid

		public bool Login1 (string uid, string @Password)
		{

			SqlConnection myConn = new SqlConnection ("Data
Source=216.80.113.39;initial catalog=Authortize;user id=MBKChat;password=@in@rs;");

			SqlCommand myComm = new SqlCommand("SELECT * FROM MybookIMUsers
WHERE User_Name = '" + uid + "' AND passw = @Password", myConn);

			//The array of bytes that will contain the encrypted value of
strPlainText
			byte[] hashedDataBytes;

			//The encoder class used to convert strPlainText to an array of
bytes
			UTF8Encoding encoder = new UTF8Encoding(false);

			//Create an instance of the MD5CryptoServiceProvider class
			MD5CryptoServiceProvider md5Hasher = new
MD5CryptoServiceProvider();

			//Call ComputeHash, passing in the plain-text string as an array of
bytes
			//The return value is the encrypted value, as an array of bytes
			hashedDataBytes =
md5Hasher.ComputeHash(encoder.GetBytes(Passwrd.Text + UserName.Text));

			//here we are encrypting password
			SqlParameter paramPwd;
			paramPwd = new SqlParameter("@Password", SqlDbType.Binary, 16);
			paramPwd.Value = hashedDataBytes;
			myComm.Parameters.Add(paramPwd);
			try
			{
```

```
                        myConn.Open();
                        SqlDataReader myReader = myComm.ExecuteReader();
                        Result = myReader.Read();
                        myReader.Close();
                        myConn.Close();

                }
                catch(Exception ed)
                {

                        //Disconnect();
                        MessageBox.Show("Try To Connect To Server Later Or Contact
System Administrator!!", "Problem With Mybook IM SQL Server Or Your Internet
Connection",MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                        Console.WriteLine("Exception occured:"+ed.ToString());
                        System.Environment.Exit(-2147480000);

                }

                return Result;


        }


        public string Authusrname ;
        public string ServerPort;
        //click on Authorize button and check if user name and/or password are
valid
        public void AuthorizeButton_Click(Object sender, EventArgs e)
        {

                if (Login1(UserName.Text, Passwrd.Text) == true &&
ServerComboBox.Text == "Main Public Community")
                {

                        this.Hide();
                        //this.Visible = false;
                        Authusrname = UserName.Text;
                        ServerPort = "8052";
                        this.Close();
                }
                else if (Login1(UserName.Text, Passwrd.Text) != true &&
ServerComboBox.Text == "Main Public Community")
                {
                        if (this.AuthLabel.Text == "     Authorization Failed! Try
Again.")
                        {
                                this.AuthLabel.Text = "      Try Again.
Authorization Failed!";
                        }
                        else
                        {
                                this.AuthLabel.ForeColor = Color.Red;
                                this.AuthLabel.Text = "     Authorization Failed!
Try Again.";
                        }
                }

                else if (Login1(UserName.Text, Passwrd.Text) == true &&
ServerComboBox.Text == "MyBook Community" && ServerPasswordTextBox.Text == "MyBook")
                {
                        this.Visible = false;
                        Authusrname = UserName.Text;
                        ServerPort = "7001";
                }
                else if (Login1(UserName.Text, Passwrd.Text) != true &&
ServerComboBox.Text == "MyBook Community" && ServerPasswordTextBox.Text == "MyBook")
                {
                        this.AuthLabel.ForeColor = Color.Red;
                        this.AuthLabel.Text = "     Authorization Failed! Try
Again.";
                }
```

```
                    else if (Login1(UserName.Text, Passwrd.Text) != true &&
ServerComboBox.Text == "MyBook Community" && ServerPasswordTextBox.Text != "MyBook")
                    {
                           this.AuthLabel.ForeColor = Color.Red;
                           this.AuthLabel.Text = "     Authorization Failed! Try
Again.";
                    }
                    else if (Login1(UserName.Text, Passwrd.Text) == true &&
ServerComboBox.Text == "MyBook Community" && ServerPasswordTextBox.Text != "MyBook")
                    {
                           this.AuthLabel.ForeColor = Color.Red;
                           this.AuthLabel.Text = "     Authorization Failed! Try
Again.";
                    }
                    else if (Login1(UserName.Text, Passwrd.Text) == true &&
ServerComboBox.Text == "Professors' Community" && ServerPasswordTextBox.Text ==
"Professor")
                    {
                           this.Visible = false;
                           Authusrname = UserName.Text;
                           ServerPort = "7002";
                    }
                    else if (Login1(UserName.Text, Passwrd.Text) == true &&
ServerComboBox.Text == "Professors' Community" && ServerPasswordTextBox.Text !=
"Professor")
                    {
                           this.AuthLabel.ForeColor = Color.Red;
                           this.AuthLabel.Text = "     Authorization Failed! Try
Again.";
                    }
                    else if (Login1(UserName.Text, Passwrd.Text) != true &&
ServerComboBox.Text == "Professors' Community" && ServerPasswordTextBox.Text ==
"Professor")
                    {
                           this.AuthLabel.ForeColor = Color.Red;
                           this.AuthLabel.Text = "     Authorization Failed! Try
Again.";
                    }
                    else if (Login1(UserName.Text, Passwrd.Text) != true &&
ServerComboBox.Text == "Professors' Community" && ServerPasswordTextBox.Text !=
"Professor")
                    {
                           this.AuthLabel.ForeColor = Color.Red;
                           this.AuthLabel.Text = "     Authorization Failed! Try
Again.";
                    }
                    else if (Login1(UserName.Text, Passwrd.Text) == true &&
ServerComboBox.Text == "Students' Community" && ServerPasswordTextBox.Text == "Student")
                    {
                           this.Visible = false;
                           Authusrname = UserName.Text;
                           ServerPort = "7003";
                    }
                    else if (Login1(UserName.Text, Passwrd.Text) == true &&
ServerComboBox.Text == "Students' Community" && ServerPasswordTextBox.Text != "Student")
                    {
                           this.AuthLabel.ForeColor = Color.Red;
                           this.AuthLabel.Text = "     Authorization Failed! Try
Again.";
                    }
                    else if (Login1(UserName.Text, Passwrd.Text) != true &&
ServerComboBox.Text == "Students' Community" && ServerPasswordTextBox.Text != "Student")
                    {
                           this.AuthLabel.ForeColor = Color.Red;
                           this.AuthLabel.Text = "     Authorization Failed! Try
Again.";
                    }
                    else if (Login1(UserName.Text, Passwrd.Text) == true &&
ServerComboBox.Text == "Students' Community" && ServerPasswordTextBox.Text == "Student")
                    {
                           this.AuthLabel.ForeColor = Color.Red;
                           this.AuthLabel.Text = "     Authorization Failed! Try
Again.";
                    }
```

```
		}
		public void Authorize_Closing(object sender,
System.ComponentModel.CancelEventArgs e)
		{

			e.Cancel = false;
			try

			{
				if (ServerPort != "4036")//"8052")

				{
					System.Environment.Exit(-2147480000);
				}
				else
				{
				return;
				}

			}
			catch (Exception ex)
			{
				MessageBox.Show(ex.Message.ToString());
			}

		}

		//exit application if Cancel button is clicked

		private void CancelButton1_Click(object sender, System.EventArgs e)
		{
		  System.Environment.Exit(-2147480000);
		}


		//New user registration button. Pops up new user registration form
		private void NewUserButton_Click(object sender, System.EventArgs e)
		{

			NewUserRegister NewUserRegisterForm = new NewUserRegister();
			NewUserRegisterForm.ShowDialog();
			if (AuthLabel.Text == "")
			{
				this.AuthLabel.Text = "";
				this.AuthLabel.ForeColor = Color.Red;
				this.UserName.Text = "";
				this.Passwrd.Text = "";
			}
			else
			{
				this.AuthLabel.Text = "";
				this.AuthLabel.ForeColor = Color.Red;
				this.UserName.Text = "";
				this.Passwrd.Text = "";
			}
		}

		private void ServerComboBox_SelectedIndexChanged(object sender,
System.EventArgs e)
		{
			if (ServerComboBox.Text == "Main Public Community")
			{
				ServerPasswordTextBox.Text = "";
				ServerPasswordTextBox.Enabled = false;

			}
			else
			{
				ServerPasswordTextBox.Text = "";
				ServerPasswordTextBox.Enabled = true;
			}
```

```
                }

                private void ButtonNewCommunity_Click(object sender, System.EventArgs e)
                {

                        NewCommunity NewCommunityForm = new NewCommunity();
                        NewCommunityForm.ShowDialog();
                        if (AuthLabel.Text == "")
                        {
                                this.AuthLabel.Text = "";
                                this.AuthLabel.ForeColor = Color.Red;
                                this.UserName.Text = "";
                                this.Passwrd.Text = "";
                        }
                        else
                        {
                                this.AuthLabel.Text = "";
                                this.AuthLabel.ForeColor = Color.Red;
                                this.UserName.Text = "";
                                this.Passwrd.Text = "";
                        }
                }

        }
```

New Community form:

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.IO;
using System.Data.OleDb;
using System.Data;
using System.Text;
using System.Threading;
using System.Net;
using N = System.Net;
using System.Runtime.InteropServices;
using System.Data.SqlClient;
using System.Data.SqlTypes;
using System.Security.Cryptography;



namespace MyBookIM
{


        /// <summary>
        /// Summary description for NewUserRegister.
        /// </summary>
        public class NewCommunity : System.Windows.Forms.Form
        {
                private System.Windows.Forms.Label UserNameLabel;
                private System.Windows.Forms.TextBox EmailTextBox;
                private System.Windows.Forms.Label EmailLabel;
                private System.Windows.Forms.TextBox UserNameTextBox;
                private System.Windows.Forms.Button CancelButton1;
                private System.Windows.Forms.Label WarningLabel;
                private System.Windows.Forms.Button RegisterButton;
                private System.Windows.Forms.ToolTip toolTip1NewUser;
                private System.ComponentModel.IContainer components;
                private System.Windows.Forms.TextBox ShortDescriptionTextBox;
                private System.Windows.Forms.Label ShortDescriptionLabel;
                private System.Windows.Forms.TextBox CommunityNameTextBox;
                private System.Windows.Forms.Label labelCommunityName;
                public Authorize AuthorizeForm;
                public NewCommunity()
                {
                        //
                        // Required for Windows Form Designer support
                        //
```

```
                        InitializeComponent();

                        //
                        // TODO: Add any constructor code after InitializeComponent call
                        //
                }

                /// <summary>
                /// Clean up any resources being used.
                /// </summary>
                protected override void Dispose( bool disposing )
                {
                        if( disposing )
                        {
                                if(components != null)
                                {
                                        components.Dispose();
                                }
                        }
                        base.Dispose( disposing );
                }

                #region Windows Form Designer generated code
                /// <summary>
                /// Required method for Designer support - do not modify
                /// the contents of this method with the code editor.
                /// </summary>

                //initializing components
                private void InitializeComponent()…
                #endregion

                //Cancel button
                private void CancelButton1_Click(object sender, System.EventArgs e)
                {
                        this.Close();

                }

                //check if user with such username is already registered or not
                private bool LookUpUser (String uid)
                {

                        SqlConnection myConn = new SqlConnection ("Data
Source=216.80.113.39;initial catalog=Authortize;user id=MBKChat;password=@in@rs;");

                        SqlCommand myComm = new SqlCommand("SELECT * FROM MybookIMUsers
WHERE User_Name = '" + uid + "'", myConn);

                        myConn.Open();
                        SqlDataReader myReader = myComm.ExecuteReader();
                        bool Result = myReader.Read();
                        myReader.Close();
                        myConn.Close();
                        return Result;

}
                //check if such email is already registered or not
                private bool LookUpEmail (String eid)
                {
                        SqlConnection myConn = new SqlConnection ("Data
Source=216.80.113.39;initial catalog=Authortize;user id=MBKChat;password=@in@rs;");

                        SqlCommand myComm = new SqlCommand("SELECT * FROM MybookIMUsers
WHERE Email = '" + eid + "'", myConn);

                                myConn.Open();
                                SqlDataReader myReader = myComm.ExecuteReader();
                                bool Result = myReader.Read();
                                myReader.Close();
                                myConn.Close();
                                return Result;

                }
```

```
                //check if email format is valid
                private bool isValidEmail(string str)
                {
                        return (str.IndexOf(".") > 2) && (str.IndexOf("@") > 0);

                }
                //check if password is longer than 5 symbols
                private bool isValidPwd(string str)
                {
                        return (str.Length > 5);

                }

                //Register button with lots of validation checks
                private void RegisterButton_Click(object sender, System.EventArgs e)
                {

                        if (UserNameTextBox.Text == (""))
                        {
                                this.WarningLabel.Text = "Please Fill Out User Name
Field!";
                        }
                        else if (UserNameTextBox.Text.Length > 40)
                        {
                                this.WarningLabel.Text = "Too Many Characters In User Name
Field!";
                        }
                        else if (CommunityNameTextBox.Text == (""))
                        {
                                this.WarningLabel.Text = "Please Fill Out Community Name
Field!";
                        }
                        else if (CommunityNameTextBox.Text.Length > 40)
                        {
                                this.WarningLabel.Text = "Too Many Characters In Community
Name Field!";
                        }
                        else if (ShortDescriptionTextBox.Text == (""))
                        {
                                this.WarningLabel.Text = "Please Fill Out Short Description
Field!";
                        }
                        else if (ShortDescriptionTextBox.Text.Length > 1000)
                        {
                                this.WarningLabel.Text = "Too Many Characters In Short
Description Field! (Max = 1000 Characters)";
                        }

                        else if (EmailTextBox.Text == (""))
                        {
                                this.WarningLabel.Text = "Please Fill Out E-Mail Field!";
                        }
                        else if (EmailTextBox.Text.Length > 40)
                        {
                                this.WarningLabel.Text = "Too Many Characters In Email
Field!";
                        }
                        else if (isValidEmail(EmailTextBox.Text) == false)
                        {
                                this.WarningLabel.Text = "E-Mail Format Not Valid!";
                        }

                        else if ((LookUpUser (UserNameTextBox.Text) == false) ||
(LookUpEmail (EmailTextBox.Text)== false))
                        {
                                this.WarningLabel.Text = "User Name And/Or Email Don't
Exist in DB! Only Registered Users Can Create New Community!";
                        }

                        else
                        {

                                try
```

```
                                {
                                        this.WarningLabel.Text = "Successful Registration!
Check Your E-Mail!";
                                        this.WarningLabel.ForeColor = Color.Green;
                                        MessageBox.Show("Answer Regarding Acceptence Of Your
New \r\nCommunity Will Be Sent To You Within 24 hours!");

                                {
                                        //here we connect to SQL server and save all data in
database
                                        string strDSN = "Data Source=216.80.113.39;initial
catalog=Authortize;user iD=MBKChat;password=@in@rs;";
                                        string strSQL = "INSERT INTO NewCommunity
(User_Name, Comm_Name, Short_Descr, Email) VALUES ('"+UserNameTextBox.Text+"',
'"+CommunityNameTextBox.Text+"', '"+ShortDescriptionTextBox.Text+"',
'"+EmailTextBox.Text+"')";

                                        // create Objects of SQLConnection and SQLCommand
                                        SqlConnection myConn = new SqlConnection(strDSN);
                                        SqlCommand myCmd = new SqlCommand( strSQL, myConn );
                                        //SqlCommand myComm = new SqlCommand("SELECT
User_Name FROM MybookIMUsers", myConn);

                                        try
                                        {


                                                myConn.Open();
                                                myCmd.ExecuteNonQuery();

                                        }
                                        catch (Exception ed)
                                        {
                                                MessageBox.Show("Try To Connect To Server
Later Or Contact System Administrator!!", "Problem With Mybook Instant Messenger SQL
Server",MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                                                Console.WriteLine("Exception
occured:"+ed.ToString());

                                                Application.Exit();
                                                //MessageBox.Show("Oooops. Error:\n{0}",
x.Message);
                                        }
                                        finally
                                        {

                                                myConn.Close();
                                                this.Close();


                                        }

                                }


                                }
                         finally{

                                }


                        }


                }
```

```
        }


}
```

New User Registration Form:

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.IO;
using System.Data.OleDb;
using System.Data;
using System.Text;
using System.Threading;
using System.Net;
using N = System.Net;
using System.Runtime.InteropServices;
using System.Data.SqlClient;
using System.Data.SqlTypes;
using System.Security.Cryptography;

namespace MyBookIM
{
        /// <summary>
        /// Summary description for NewUserRegister.
        /// </summary>
        public class NewUserRegister : System.Windows.Forms.Form
        {
                private System.Windows.Forms.Label UserNameLabel;
                private System.Windows.Forms.Label PasswLabel;
                private System.Windows.Forms.Label PasswAgLabel;
                private System.Windows.Forms.TextBox FirstNameTextBox;
                private System.Windows.Forms.TextBox LastNameTextBox;
                private System.Windows.Forms.TextBox AddressTextbox;
                private System.Windows.Forms.TextBox PasswAgainTextBox;
                private System.Windows.Forms.TextBox EmailTextBox;
                private System.Windows.Forms.Label FirstNameLabel;
                private System.Windows.Forms.Label LastNameLabel;
                private System.Windows.Forms.Label AddressLabel;
                private System.Windows.Forms.Label EmailLabel;
                private System.Windows.Forms.TextBox UserNameTextBox;
                private System.Windows.Forms.TextBox PasswTextBox;
                private System.Windows.Forms.Button CancelButton1;
                private System.Windows.Forms.Label WarningLabel;
                private System.Windows.Forms.Button RegisterButton;
                private System.Windows.Forms.ToolTip toolTip1NewUser;
                private System.ComponentModel.IContainer components;
                public Authorize AuthorizeForm;
                public NewUserRegister()
                {
                        //
                        // Required for Windows Form Designer support
                        //
                        InitializeComponent();

                        //
                        // TODO: Add any constructor code after InitializeComponent call
                        //
                }

                /// <summary>
                /// Clean up any resources being used.
                /// </summary>
                protected override void Dispose( bool disposing )
                {
                        if( disposing )
                        {
                                if(components != null)
```

```
                        {
                                components.Dispose();
                        }
                }
                base.Dispose( disposing );
        }

        #region Windows Form Designer generated code
        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>

        //initializing components
        private void InitializeComponent()…
        #endregion

        //Cancel button
        private void CancelButton1_Click(object sender, System.EventArgs e)
        {
                this.Close();

        }

        //check if user with such username is already registered or not
        private bool LookUpUser (String uid)
        {

                SqlConnection myConn = new SqlConnection ("Data
Source=216.80.113.39;initial catalog=Authortize;user id=MBKChat;password=@in@rs;");

                SqlCommand myComm = new SqlCommand("SELECT * FROM MybookIMUsers
WHERE User_Name = '" + uid + "'", myConn);
                myConn.Open();
                SqlDataReader myReader = myComm.ExecuteReader();
                bool Result = myReader.Read();
                myReader.Close();
                myConn.Close();
                return Result;

}
        //check if such email is already registered or not
        private bool LookUpEmail (String eid)
        {
                SqlConnection myConn = new SqlConnection ("Data
Source=216.80.113.39;initial catalog=Authortize;user id=MBKChat;password=@in@rs;");

                SqlCommand myComm = new SqlCommand("SELECT * FROM MybookIMUsers
WHERE Email = '" + eid + "'", myConn);

                        myConn.Open();
                        SqlDataReader myReader = myComm.ExecuteReader();
                        bool Result = myReader.Read();
                        myReader.Close();
                        myConn.Close();
                        return Result;

        }

        //check if email format is valid
        private bool isValidEmail(string str)
        {
                return (str.IndexOf(".") > 2) && (str.IndexOf("@") > 0);

        }
        //check if password is longer than 5 symbols
        private bool isValidPwd(string str)
        {
                return (str.Length < 3 || str.Length >40);

        }

        //Register button with lots of validation checks
        private void RegisterButton_Click(object sender, System.EventArgs e)
```

```
            {

                    if (UserNameTextBox.Text == (""))
                    {
                            this.WarningLabel.Text = "Please Fill Out User Name
Field!";
                    }
                    else if (UserNameTextBox.Text.Length > 20 ||
UserNameTextBox.Text.Length <3)
                    {
                            this.WarningLabel.Text = "User Name Should Be Bigger Than 3
Characters And Less Than 20!";
                    }
                    else if (PasswTextBox.Text == (""))
                    {
                            this.WarningLabel.Text = "Please Fill Out Password Field!";
                    }
                    else if (isValidPwd(PasswTextBox.Text) == true )
                    {
                            this.WarningLabel.Text = "Password Should Be Bigger Than 3
Characters And Less Than 40!!";
                    }
                    else if (PasswAgainTextBox.Text == (""))
                    {
                            this.WarningLabel.Text = "Please Fill Out Password Again
Field!";
                    }
                    else if (PasswTextBox.Text != PasswAgainTextBox.Text)
                    {
                            this.WarningLabel.Text = "Password Is Not Equal!";
                            PasswTextBox.Clear();
                            PasswAgainTextBox.Clear();
                    }
                    else if (PasswTextBox.Text == UserNameTextBox.Text)
                    {
                            this.WarningLabel.Text = "User Name And Password Are The
Same! Choose Different Password!";
                    }
                    else if (FirstNameTextBox.Text == (""))
                    {
                            this.WarningLabel.Text = "Please Fill Out First Name
Field!";
                    }
                    else if (FirstNameTextBox.Text.Length > 40)
                    {
                            this.WarningLabel.Text = "Too Many Characters In First Name
Field!";
                    }
                    else if (LastNameTextBox.Text == (""))
                    {
                            this.WarningLabel.Text = "Please Fill Out Last Name
Field!";
                    }
                    else if (LastNameTextBox.Text.Length > 40)
                    {
                            this.WarningLabel.Text = "Too Many Characters In Last Name
Field!";
                    }
                    else if (AddressTextbox.Text == (""))
                    {
                            this.WarningLabel.Text = "Please Fill Out Address Field!";
                    }
                    else if (AddressTextbox.Text.Length > 50)
                    {
                            this.WarningLabel.Text = "Too Many Characters In Address
Field!";
                    }
                    else if (EmailTextBox.Text == (""))
                    {
                            this.WarningLabel.Text = "Please Fill Out E-Mail Field!";
                    }
                    else if (EmailTextBox.Text.Length > 50)
                    {
```

```
                        this.WarningLabel.Text = "Too Many Characters In E-Mail
Field!";
                }
                else if (isValidEmail(EmailTextBox.Text) == false)
                {
                        this.WarningLabel.Text = "E-Mail Format Not Valid!";
                }
                else if (LookUpUser (UserNameTextBox.Text))
                {
                        this.WarningLabel.Text = "User Name '" +
UserNameTextBox.Text + "' Already Exists! Try Different One!";
                }
                else if (LookUpEmail (EmailTextBox.Text))
                {
                        this.WarningLabel.Text = "E-Mail: ' " + EmailTextBox.Text +
" ' Already Exists In Database!";
                }
                else
                {

                        try

                        {
                                this.WarningLabel.Text = "Successful Registration!
You May Log On Now!";
                                this.WarningLabel.ForeColor = Color.Green;

                                MessageBox.Show("Congratulations!!");

                        {
//here we connect to SQL server and save all data in database
                                string strDSN = "Data Source=216.80.113.39;initial
catalog=Authortize;user iD=MBKChat;password=@in@rs;";
                                string strSQL = "INSERT INTO MybookIMUsers
(User_Name, Passw, First_Name, Last_Name, Address, Email) VALUES
('"+UserNameTextBox.Text+"',  @Password,'"+FirstNameTextBox.Text+"',
'"+LastNameTextBox.Text+"', '"+AddressTextbox.Text+"', '"+EmailTextBox.Text+"')";

                        // create Objects of SQLConnection and SQLCommand
                        SqlConnection myConn = new SqlConnection(strDSN);
                        SqlCommand myCmd = new SqlCommand( strSQL, myConn );
                                //SqlCommand myComm = new SqlCommand("SELECT
User_Name FROM MybookIMUsers", myConn);

                                try
                                {

                                        //The array of bytes that will contain the
encrypted value of strPlainText
                                        byte[] hashedDataBytes;
                                        //The encoder class used to convert
strPlainText to an array of bytes
                                        UTF8Encoding encoder = new
UTF8Encoding(false);
                                        //Create an instance of the
MD5CryptoServiceProvider class
                                        MD5CryptoServiceProvider md5Hasher = new
MD5CryptoServiceProvider();
                                        //Call ComputeHash, passing in the plain-
text string as an array of bytes
                                        //The return value is the encrypted value,
as an array of bytes
                                        hashedDataBytes =
md5Hasher.ComputeHash(encoder.GetBytes(PasswTextBox.Text + UserNameTextBox.Text));
                                        SqlParameter paramPwd;
                                        paramPwd = new SqlParameter("@Password",
SqlDbType.Binary, 16);
                                        paramPwd.Value = hashedDataBytes;
                                        myCmd.Parameters.Add(paramPwd);
                                        myConn.Open();
                                        myCmd.ExecuteNonQuery();

                                }
                                catch (Exception ed)
```

```
                                        {
                                                MessageBox.Show("Try To Connect To Server
Later Or Contact System Administrator!!", "Problem With Mybook Instant Messenger SQL
Server",MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                                                Console.WriteLine("Exception
occured:"+ed.ToString());

                                                Application.Exit();

                                        }
                                        finally
                                        {

                                                myConn.Close();
                                                this.Close();

                                        }

                                        }
                                        }

                         finally
                                {

                                }

                        }

                }

        }
}
```

OkSendFile Form:

```
using System;
using System.Threading;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.IO;
using System.Net;
using System.Net.Sockets;

namespace MyBookIM
{
        /// <summary>
        /// Summary description for OkSendFile.
        /// </summary>
        public class OkSendFile : System.Windows.Forms.Form
        {
                public System.Windows.Forms.Button OKButton;
                public System.Windows.Forms.Label InfoLabel;
                private SendFile SendFileForm;
                private CClient cccc = null;
                private ProgressBarFrm PBF = null;
                /// <summary>
                /// Required designer variable.
                /// </summary>
                private System.ComponentModel.Container components = null;

                public OkSendFile(CClient cccc)
                {
                        //
                        // Required for Windows Form Designer support
                        //
                        this.SendFileForm = SendFileForm;
                        this.cccc = cccc;
                        this.PBF = PBF;

                        InitializeComponent();
                        //
```

```
                        // TODO: Add any constructor code after InitializeComponent call
                        //
                }

                /// <summary>
                /// Clean up any resources being used.
                /// </summary>
                protected override void Dispose( bool disposing )
                {
                        if( disposing )
                        {
                                if(components != null)
                                {
                                        components.Dispose();
                                }
                        }
                        base.Dispose( disposing );
                }

                #region Windows Form Designer generated code
                /// <summary>
                /// Required method for Designer support - do not modify
                /// the contents of this method with the code editor.
                /// </summary>
                private void InitializeComponent()…
                #endregion

                private void OKButton_Click(object sender, System.EventArgs e)
                {

                        try
                        {
                                long total=cccc.File_size;
                                string sendusr = this.cccc.Senderusr;
                                string recusr = this.cccc.Receiverusr;
                                ProgressBarFrm PB = new ProgressBarFrm();
                                PB.Show();
                                SendingThread s = new
SendingThread(this.cccc.File_path,total,cccc.reciever_ipaddress,PB,
this.cccc,sendusr,recusr);
                                Thread t = new Thread(new ThreadStart(s.sendfile));
                                t.Start();

                        }
                        catch(Exception ed)
                        {
                                Console.WriteLine("A Exception occured in
transfer"+ed.ToString()) ;

                        }
                        this.Close();
                }

        }
}


PrivateTalkNew Form:

using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Net.Sockets;
using System.Runtime.InteropServices;



namespace MyBookIM
{
        /// <summary>
        /// Summary description for PrivateTalk.
        /// </summary>
```

```
	///

	public class PrivateTalkNew : System.Windows.Forms.Form
	{
		public System.Windows.Forms.TextBox PrivateTalkReceiveMessageBox;
		public System.Windows.Forms.TextBox PrivateTalkSendMessageBox;
		private System.Windows.Forms.Button PrivateTalkSendButton;
		public string PUsr;
		private System.Windows.Forms.MainMenu mainMenu1;
		private System.Windows.Forms.MenuItem MenuSendFile;
		private System.Windows.Forms.MenuItem MenuSaveLog;
		/// <summary>
		/// Required designer variable.
		/// </summary>
		private System.ComponentModel.Container components = null;
		public CClient CC;

		public PrivateTalkNew()
		{
			//
			// Required for Windows Form Designer support
			//
			InitializeComponent();

			//
			// TODO: Add any constructor code after InitializeComponent call
			//
		}

		/// <summary>
		/// Clean up any resources being used.
		/// </summary>
		protected override void Dispose( bool disposing )
		{
			if( disposing )
			{
				if(components != null)
				{
					components.Dispose();
				}
			}
			base.Dispose( disposing );
		}

		#region Windows Form Designer generated code
		/// <summary>
		/// Required method for Designer support - do not modify
		/// the contents of this method with the code editor.
		/// </summary>
		private void InitializeComponent()…
		#endregion

private void PrivateTalkSendButton_Click(object sender, System.EventArgs e)
		{

			if (PrivateTalkSendMessageBox.Text!="")
			{
				string message = "PRIVMSG:" +
this.PrivateTalkSendMessageBox.Text.Trim();
				CC.SendData(message);
				this.PrivateTalkSendMessageBox.Text = "";
				this.PrivateTalkSendMessageBox.Focus();

			}
			else
			{
				return ;
			}


		}

		private void MenuExit_Click(object sender, System.EventArgs e)
		{
```

```
                        this.Close();
                        //CClient.flag = CClient.flag -1;

                }


                public string message;
                private void PrivateTalkSendMessageBox_KeyPress(object sender,
KeyPressEventArgs e)
                {
                        try
                        {
                                if ((int)e.KeyChar == 13)
                                {
                                        if (PrivateTalkSendMessageBox.Text!="")
                                        {
                                                e.Handled=true;
                                                message = "PRIVMSG:" +
this.PrivateTalkSendMessageBox.Text.Trim();
                                                //this.SendData(message);
                                                this.PrivateTalkSendMessageBox.Text = "";
                                                this.PrivateTalkSendMessageBox.Focus();
                                        }
                                        else
                                        {
                                                return ;
                                        }
                                }
                                else
                                {
                                        return ;
                                }
                        }
                        catch (Exception ex)
                        {
                                MessageBox.Show(ex.Message.ToString());
                        }
                }



                private void MenuSaveLog_Click(object sender, System.EventArgs e)
                {
                        MessageBox.Show(PrivateTalkReceiveMessageBox.Text);
                }

                private void PrivateTalkNew_Load(object sender, System.EventArgs e)
                {

                }


        }


}



ProgressBarFrm Form:

using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;

namespace MyBookIM
{
        /// <summary>
        /// Summary description for .
        /// </summary>
        public class  ProgressBarFrm : System.Windows.Forms.Form
```

```
	{
		public System.Windows.Forms.Button OkButton;
		//public System.Windows.Forms.ProgressBar ProgressBarSF;
		//public System.Windows.Forms.Button CancelDownlButton;
		//public System.Windows.Forms.ToolTip toolTipProgressBar;
		//public System.Windows.Forms.Label PrBarLabel;
		private System.ComponentModel.IContainer components;

		public ProgressBarFrm()
		{
			//
			// Required for Windows Form Designer support
			//
			InitializeComponent();
			this.SetStyle(ControlStyles.UserPaint, true);
			this.SetStyle(ControlStyles.AllPaintingInWmPaint, true);
			this.SetStyle(ControlStyles.DoubleBuffer, true);
			this.UpdateStyles();
			//
			// TODO: Add any constructor code after InitializeComponent call
			//
		}

		/// <summary>
		/// Clean up any resources being used.
		/// </summary>
		protected override void Dispose( bool disposing )
		{
			if( disposing )
			{
				if(components != null)
				{
					components.Dispose();
				}
			}
			base.Dispose( disposing );
		}

		#region Windows Form Designer generated code
		/// <summary>
		/// Required method for Designer support - do not modify
		/// the contents of this method with the code editor.
		/// </summary>
		private void InitializeComponent()…
		#endregion

		private void OkButton_Click(object sender, System.EventArgs e)
		{
		this.Close();
		}

	}
}
```

ReceivingThread class:

```
using System;
using System.Windows.Forms;
using System.Threading;
using System.IO;
using System.Diagnostics;
using System.Net;
using System.Net.Sockets;
using System.Text;
using System.Runtime;
using System.Reflection;
using System.Collections;
using System.Drawing;
using System.ComponentModel;
using System.Data;
using System.Text.RegularExpressions ;
using System.Xml.Serialization;
using System.Timers;
```

```
namespace MyBookIM
{
	/// <summary>
	/// Summary description for ReceivingThread.
	/// </summary>
	public sealed class ReceivingThread
	{
		string myFileName;
		FileStream fout;
		NetworkStream nfs;
		string fileSize;
		Socket socketForClient;
		TcpListener tcpListener;
		Form PB;

		System.Windows.Forms.Button OkButton;
		System.Windows.Forms.ProgressBar ProgressBarSF;
		System.ComponentModel.IContainer components;
		System.Windows.Forms.Button CancelDownlButton;
		System.Windows.Forms.ToolTip toolTipProgressBar;
		System.Windows.Forms.Label PrBarLabel;

		public void OkButton_Click(object sender, System.EventArgs e)
		{
			PB.Close();
		}

		private void CancelDownlButton_Click(object sender, System.EventArgs e)
		{
			try
			{
				this.socketForClient.Shutdown(SocketShutdown.Both);
				this.socketForClient.Close();
				if (this.socketForClient.Connected)
				{
					Console.WriteLine("Winsock error: " +
Convert.ToString(System.Runtime.InteropServices.Marshal.GetLastWin32Error()) );
				}


				PB.Text = "Download Aborted!";
				Thread.Sleep(5000);
				PB.Close();

			}
			catch (Exception ex)
			{

				tcpListener.Stop();
				Console.WriteLine(ex.ToString());
				PB.Text = "Download Aborted!";
				Thread.Sleep(5000);
				PB.Close();

			}
		}


		public ReceivingThread(string fileName, string fsize, Form prbar)
		{

			myFileName = fileName;
			fileSize = fsize;
			PB = prbar;

			try
			{
				fout = new FileStream(myFileName ,FileMode.Create ,
FileAccess.Write) ;

				System.Resources.ResourceManager resources = new
System.Resources.ResourceManager(typeof(ProgressBarFrm));
```

```
                        this.ProgressBarSF = new
System.Windows.Forms.ProgressBar();
                        this.OkButton = new System.Windows.Forms.Button();
                        this.CancelDownlButton = new System.Windows.Forms.Button();
                        this.toolTipProgressBar = new
System.Windows.Forms.ToolTip();
                        this.PrBarLabel = new System.Windows.Forms.Label();
                        PB.SuspendLayout();

                }

                catch (Exception exx)
                {
                        Console.WriteLine(exx.ToString());
                        PB.Close();
                        MessageBox.Show("Can't OverWrite The File. Rename New File
Or Remove\nRead Only Attributes To The File You Want To OverWrite!", "Problem With File
OverWriting.. ",MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                }


        }

        //
        // TODO: Add constructor logic here
        //

        public void getfile()
        {
                try

                {

                        tcpListener = new TcpListener(4036);
                        tcpListener.Start();


                        socketForClient = tcpListener.AcceptSocket();
                        nfs  = new NetworkStream(socketForClient);
                        StreamWriter streamWriter = new StreamWriter(nfs);
                        if ((socketForClient.Connected)&& (myFileName !=
""))
                        {

                        {
                                int pcc =0;
                                int pbpc =0;
                                float pc =0;
                                long rby=0 ;
                                int elapsed_time;
                                int UploadSpeed = 0;
                                DateTime start = System.DateTime.Now;
                                long size = long.Parse(fileSize);

                                while(rby<size)
                                {
                                        byte[] buffer = new byte[1024] ;
                                        //Read from the Network Stream
                                        int i =
nfs.Read(buffer,0,buffer.Length) ;
                                        if (rby<size && i == 0)
                                        {


        this.socketForClient.Shutdown(SocketShutdown.Both);
                                                this.socketForClient.Close();
                                                if
(this.socketForClient.Connected)
                                                {

        Console.WriteLine("Winsock error: " +
Convert.ToString(System.Runtime.InteropServices.Marshal.GetLastWin32Error()) );
                                                }
```

```
                                                        PB.Text = "Sender Aborted
File Sending!";
                                                        Thread.Sleep(5000);
                                                        PB.Close();

                                                }
                                                else
                                                {


                                                        fout.Write(buffer,0,(int)i) ;
                                                        fout.Flush();
                                                        rby=rby+i ;
                                                        DateTime end =
System.DateTime.Now;
                                TimeSpan duration = end - start;
                                elapsed_time = (int)(duration.TotalMilliseconds/1000)+1;
                                //then that is how long between start & end time

                        UploadSpeed = (int)((double)(rby/1024)/(double)elapsed_time);
                        pc = (float)( ((double)rby/(double)size ) * 100.00);
                        pcc = (int)( ((double)rby/(double)size ) * 100.00);
                        pbpc =(int)(Math.Floor(pc/8)*8);
                                                        if (pc==100)
                                                        {
                                                                pbpc = 100;
                                                        }

                                                        if (pc < 100)
                                                        {


                        PB.Text = "Downloaded " + pcc + " % of File";
                        PrBarLabel.Text = "Download Speed: "+UploadSpeed.ToString()+"
Kb/s\r\nElapsed Time: " + elapsed_time.ToString()  + " Seconds";
                        ProgressBarSF.Value = pbpc;
                                                                if (pc == pbpc)
                                                                {

        ProgressBarSF.PerformStep();
                                                                }
                                                        OkButton.Enabled = false;

                                                        }
                                                        else
                                                        {
                                        PB.Text = "Downloaded " + pcc + " % of File";
                                                                PrBarLabel.Text =
"Download Speed: "+UploadSpeed.ToString()+" Kb/s\r\nElapsed Time: " +
elapsed_time.ToString()  + " Seconds";
                                                        ProgressBarSF.Value = pbpc;

        ProgressBarSF.PerformStep();
                                                        OkButton.Enabled = true;

        CancelDownlButton.Enabled = false;
                                                                Thread.Sleep(5000);
                                                                PB.Close();

                                                        }
                                                }

                                        }
                                        fout.Close();
                                        }
                                        streamWriter.Flush() ;


                                }


                        }

                catch(Exception exx)
```

```
                        {
                                Console.WriteLine(exx.ToString());

                        }


                }


        }
}
```

SendFile Form:

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.IO;
using System.Net;
using System.Net.Sockets;

namespace MyBookIM
{
        /// <summary>
        /// Summary description for SendFile.
        /// </summary>
        public class SendFile : System.Windows.Forms.Form
        {
                public System.Windows.Forms.TextBox SendFileFrom;
                public System.Windows.Forms.TextBox SendFilePath;
                public string Filepath;
                private System.Windows.Forms.Button BrowseButton;
                private System.Windows.Forms.Button SendFileButton;
                private System.Windows.Forms.Button CancelSendFile;
                private System.Windows.Forms.Label SendFileToLabel;
                private System.Windows.Forms.Label SendFileFromLabel;
                private System.Windows.Forms.Label SendFilePathLabel;
                public System.Windows.Forms.TextBox SendFileTo;
                private System.Windows.Forms.MainMenu _MainMenu;
                private CClient cc = null;
                public string FileChosen;
                public string FileNfo;
                private System.Windows.Forms.ToolTip toolTipSendFile;
                private System.ComponentModel.IContainer components;
                public SendFile(CClient cc)
                {
                        //
                        // Required for Windows Form Designer support
                        //
                        this.cc = cc;
                        InitializeComponent();
                        //
                        // TODO: Add any constructor code after InitializeComponent call
                        //
                }

                /// <summary>
                /// Clean up any resources being used.
                /// </summary>
                protected override void Dispose( bool disposing )
                {

                        if( disposing )
                        {
                                if(components != null)
                                {
```

```
                                components.Dispose();
                        }
                }
                base.Dispose( disposing );
        }

        #region Windows Form Designer generated code
        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()…
        #endregion


        private void CancelSendFile_Click(object sender, System.EventArgs e)
        {
                this.Close();
        }

        private void BrowseButton_Click(object sender, System.EventArgs e)
        {
        {
                OpenFileDialog fdlg = new OpenFileDialog();
                fdlg.Title = "Mybook IM Open & Send File Dialog";
                fdlg.InitialDirectory = @"c:\public_html\" + cc.Text.Substring(33);
                fdlg.Filter = "All documents (*.*)| *.*|MBK File (*.mbk)|
*.mbk|Jpeg Image (*.jpg)|*.jpg|Bitmap Image (*.bmp)|*.bmp|Gif Image (*.gif)|*.gif|Word
Document (*.doc)| *.doc|Excel Document (*.xls)| *.xls|Power Point Document (*.ppt)|
*.ppt|Video file (*.avi)| *.avi|Mp3 File (*.mp3)| *.mp3";
                fdlg.RestoreDirectory = true;

                if(fdlg.ShowDialog() == DialogResult.OK)
                {

                        SendFilePath.Text = fdlg.FileName ;

                }
        }


        }

        public void SendFileButton_Click(object sender, System.EventArgs e)
        {

                if (SendFileTo.Text != SendFileFrom.Text)
                {
                        if (SendFilePath.Text != "")
                                try
                                {
                                {
                                        FileChosen  =
SendFilePath.Text.Substring(SendFilePath.Text.LastIndexOf("\\") + 1);
                                        this.cc.File_path = SendFilePath.Text;
                                        this.cc.Senderusr = SendFileFrom.Text;
                                        this.cc.Receiverusr = SendFileTo.Text;
                                        FileInfo fi = new
FileInfo(SendFilePath.Text);
                                        FileNfo = fi.Length.ToString();
                                        this.cc.File_size= long.Parse(FileNfo);

                                        if (this.cc.File_size != 0)
                                        {
                                                string msg = "SENDFIL^" +
SendFileTo.Text+ "^"  + FileChosen.ToString()+ "^" +FileNfo;
                                                this.cc.SendData(msg);
                                                this.Close();
                                        }
                                        else
                                        {
                                                MessageBox.Show("File Size is 0
Bytes. Choose An Other File!", "Problem With Mybook Instant Messenger
User..",MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
```

```
                                        }


                                        }
                                        }
                                        catch (Exception ex)
                                        {
                                                Console.WriteLine(ex.ToString());
                                        MessageBox.Show("Such File Or File Path Doesn't
Exist. Choose An Other File!", "Problem With Mybook Instant Messenger
User..",MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                                        }
                                else
                                {
                                        MessageBox.Show("Please Choose File For Sending
Using 'Browse' Button!", "File Is Not Chosen..",MessageBoxButtons.OK,
MessageBoxIcon.Exclamation);
                                }
                        }
                        else
                        {
                        MessageBox.Show("Do You Want To Send File To Yourself??", "Problem
With Mybook IM User",MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                        }
                        }
        }
}
```

SendFileReceiver Form:

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Net;
using System.Net.Sockets;
using System.IO;
using System.Threading;

namespace MyBookIM
{
        /// <summary>
        /// Summary description for SendFileReceiver.
        /// </summary>
        public class SendFileReceiver : System.Windows.Forms.Form
        {
                public System.Windows.Forms.Label labelFileInfo;
                public System.Windows.Forms.Button buttonDownload;
                public System.Windows.Forms.Button buttonCancel;
                private System.Windows.Forms.ToolTip toolTip1;
                public SendFile SendFileForm;
                private CClient ccc = null;
                private OkSendFile OkSendFileForm;
                private System.ComponentModel.IContainer components;
                public Form PB;
                public SendFileReceiver(CClient ccc)
                {
                        //
                        // Required for Windows Form Designer support
                        //
                        this.ccc = ccc;
                        this.SendFileForm = SendFileForm;
                        this.OkSendFileForm = OkSendFileForm;
                        InitializeComponent();
                        //
                        // TODO: Add any constructor code after InitializeComponent call
                        //
```

```
		}
		/// <summary>
		/// Clean up any resources being used.
		/// </summary>
		protected override void Dispose( bool disposing )
		{
			if( disposing )
			{
				if(components != null)
				{
					components.Dispose();
				}
			}
			base.Dispose( disposing );
		}

		#region Windows Form Designer generated code
		/// <summary>
		/// Required method for Designer support - do not modify
		/// the contents of this method with the code editor.
		/// </summary>
		private void InitializeComponent()
		#endregion

		private void buttonDownload_Click(object sender, System.EventArgs e)
		{
			// Displays a SaveFileDialog so the user can save the file
			SaveFileDialog saveFileDialog1 = new SaveFileDialog();
			saveFileDialog1.Filter = "All documents (*.*)| *.*|MBK File
(*.mbk)| *.mbk|Jpeg Image (*.jpg)|*.jpg|Bitmap Image (*.bmp)|*.bmp|Gif Image
(*.gif)|*.gif|Word Document (*.doc)| *.doc|Excel Document (*.xls)| *.xls|Power Point
Document (*.ppt)| *.ppt|Video file (*.avi)| *.avi|Mp3 File (*.mp3)| *.mp3";
			saveFileDialog1.FileName = ccc.File_Name;
			saveFileDialog1.Title = "Mybook IM Save File";
			saveFileDialog1.InitialDirectory = @"c:\public_html\" +
ccc.Text.Substring(33);
			//send message to sender if clicks download button
			if(saveFileDialog1.ShowDialog() == DialogResult.OK)
			{
				String strHostName = "";
				strHostName = Dns.GetHostName ();
				IPHostEntry ipEntry = Dns.GetHostByName (strHostName);
				IPAddress [] addr = ipEntry.AddressList;
				ccc.reciever_ipaddress = addr[0].ToString();
				string msg = "WANTSDL^" + ccc.SF_Sender+ "^" +
ccc.File_Name + "^" + ccc.SF_FileSize + "^" + ccc.reciever_ipaddress;
				this.ccc.SendData(msg);
				ProgressBarFrm PBa = new ProgressBarFrm();
				PBa.Show();
				ReceivingThread st = new
ReceivingThread(saveFileDialog1.FileName,ccc.SF_FileSize,PBa);
				Thread tt = new Thread(new ThreadStart(st.getfile));
				tt.Start();
				this.Close();
			}
			else
			{
			this.Close();
			}
		}
		private void buttonCancel_Click(object sender, System.EventArgs e)
		{
			this.Close();
		}
	}
}
```

SendingThread class:

```
using System;
using System.IO;
using System.Diagnostics;
```

```
using System.Net;
using System.Net.Sockets;
using System.Text;
using System.Runtime;
using System.Threading;
using System.Reflection;
using System.Collections;
using System.Drawing;
using System.Windows.Forms;
using System.ComponentModel;
using System.Data;
using System.Text.RegularExpressions ;
using System.Xml.Serialization;
using System.Timers;


namespace MyBookIM
{
	/// <summary>
	/// Summary description for SendingThread.
	/// </summary>

	public sealed class SendingThread
	{


		FileStream fin;
		long total;
		NetworkStream networkStream;
		string myPath;
		string ipAdd;
		TcpClient myclient;
		Form PB;
		string senderuser;
		string receiveruser;


		System.Windows.Forms.Button OkButton;
		System.Windows.Forms.ProgressBar ProgressBarSF;
		System.ComponentModel.IContainer components;
		System.Windows.Forms.Button CancelDownlButton;
		System.Windows.Forms.ToolTip toolTipProgressBar;
		System.Windows.Forms.Label PrBarLabel;



		public void OkButton_Click(object sender, System.EventArgs e)
		{
			PB.Close();

		}
		public CClient cccc = null;
		public SendFile SendFileForm;
		string msg1;
		private void CancelDownlButton_Click(object sender, System.EventArgs e)
		{
			this.networkStream.Close();
			this.networkStream.Flush();
			PB.Text = "Download Aborted!";
			Thread.Sleep(5000);
			PB.Close();

		}

		public SendingThread(string path, long t, string ipAdrese, Form ProBar,
CClient passed_cccc, string sendu, string receiveu)
		{

			PB = ProBar;
			myPath = path;
			total = t;
			fin = new FileStream(myPath,FileMode.Open , FileAccess.Read) ;
			ipAdd = ipAdrese;
			senderuser = sendu;
```

```
                        receiveruser = receiveu;


                        System.Resources.ResourceManager resources = new
System.Resources.ResourceManager(typeof(ProgressBarFrm));
                        this.ProgressBarSF = new System.Windows.Forms.ProgressBar();
                        this.OkButton = new System.Windows.Forms.Button();
                        this.CancelDownlButton = new System.Windows.Forms.Button();
                        this.toolTipProgressBar = new System.Windows.Forms.ToolTip();
                        this.PrBarLabel = new System.Windows.Forms.Label();
                        this.cccc = passed_cccc;
                        this.SendFileForm = SendFileForm;
                        PB.SuspendLayout();


                }

                public void sendfile()
                {
                        try
                        {
                                myclient = new TcpClient(ipAdd, 4036);
                                networkStream = myclient.GetStream();
                                DateTime start = System.DateTime.Now;

                                long rdby=0 ;
                                int len=0 ;
                                int pcc = 0;
                                float pc = 0;
                                int pbpc=0;
                                int elapsed_time;
                                int UploadSpeed = 0;
                                byte[] buffed = new byte[1024] ;


                                Debug.WriteLine("Writing "+total+" bytes to the net
connection");

                                while(rdby < total && networkStream.CanWrite && pc <100)
                                {
                                        //Read from the File (len contains the number of
bytes read)
                                        len =fin.Read(buffed,0,buffed.Length) ;

                                        // USe cc.senddat to send the buffed

                                        //Write the Bytes on the Socket

                                        networkStream.Write(buffed, 0,len);
                                        networkStream.Flush();

                                        //Increase the bytes Read counter
                                        rdby=rdby+len ;
                                        DateTime end = System.DateTime.Now;
                                        TimeSpan duration = end - start;
                                        elapsed_time =
(int)(duration.TotalMilliseconds/1000)+1;
                                        //then that is how long between start & end time
                                        //MessageBox.Show(elapsed_time.ToString());
                                         UploadSpeed =
(int)((double)(rdby/1024)/(double)elapsed_time);

                                pc = (float)( ((double)rdby/(double)total ) * 100.00);
                                pcc = (int)( ((double)rdby/(double)total ) * 100.00);
                                pbpc =(int)(Math.Floor(pc/8)*8);

                                        if (pc==100)
                                        {
                                                pbpc = 100;
                                        }


                                        if (pc < 100)
```

```
                                        {


                                                PB.Text = "Uploaded " + pcc + " % of File ";
                                                PrBarLabel.Text = "Upload Speed:
"+UploadSpeed.ToString()+" Kb/s\r\nElapsed Time: " + elapsed_time.ToString()  + "
Seconds";
                                                ProgressBarSF.Value = pbpc;
                                                if (pc == pbpc)
                                                {
                                                        ProgressBarSF.PerformStep();
                                                }
                                                OkButton.Enabled = false;


                                        }
                                        else
                                        {
                                                PB.Text = "Uploaded " + pcc + " % of File";
                                                PrBarLabel.Text = "Upload Speed:
"+UploadSpeed.ToString()+" Kb/s\r\nElapsed Time: " + elapsed_time.ToString()  + "
Seconds";
                                                ProgressBarSF.Value = pbpc;
                                                ProgressBarSF.PerformStep();
                                                OkButton.Enabled = true;
                                                CancelDownlButton.Enabled = false;

                                                Thread.Sleep(5000);
                                                PB.Close();

                                        }



                                }

                                Debug.WriteLine("Done writing the file "+myPath);

                                fin.Close();
                        }
                        catch(Exception exx)
                        {
                                //MessageBox.Show(exx.ToString()) ;
                                Console.WriteLine(exx.ToString());

if (exx.ToString().StartsWith("System.Net.Sockets.SocketException: A connection attempt
failed because the connected party did not properly respond after a period of time"))
                                {
                                        try
                                        {


                                                OkButton.Enabled = true;
                                                CancelDownlButton.Enabled = false;
                                                PB.Text = "Receiver Behind The FireWall!";
                                                Thread.Sleep(3000);
                                                PB.Close();
MessageBox.Show("Can't Connect To Receiver Directly. Sending File Using DHQ Tunneling..",
"Problem With File Sending",MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

        msg1 = "SYSMSG^" + this.receiveruser+ "^"  + "There were problems in file
transfer. Try to download file using this
link:\nhttp://localhost:8081/"+this.senderuser+"/"+this.myPath.Substring(this.myPath.Last
IndexOf("\\") + 1);
                                                cccc.SendData(msg1);

                                        }
                                        catch (Exception ex)
                                        {
                                                MessageBox.Show(ex.ToString());
                                        }
```

```
                        }
                        else



                        {
                                try
                                {

if (exx.ToString().StartsWith("System.Net.Sockets.SocketException: No connection could be
made because the target machine"))
                                        {
                                        OkButton.Enabled = true;
                                        CancelDownlButton.Enabled = false;
                                        PB.Text = "Receiver Behind The FireWall!";
                                        Thread.Sleep(3000);
                                        PB.Close();
                                        MessageBox.Show("Can't Connect To Receiver
Directly. Sending File Using DHQ Tunneling..", "Problem With File
Sending",MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
msg1 = "SYSMSG^" + this.receiveruser+ "^"  + "There were problems in file transfer. Try
to download file using this
link:\nhttp://localhost:8081/"+this.senderuser+"/"+this.myPath.Substring(this.myPath.Last
IndexOf("\\") + 1);

                                                cccc.SendData(msg1);
                                        }
                                        else
                                        {


                                        OkButton.Enabled = true;
                                        CancelDownlButton.Enabled = false;
                                        PB.Text = "Receiver Aborted File Sending!";
                                        Thread.Sleep(5000);
                                        PB.Close();
                                        }
                                        }
                                catch (Exception ex)
                                {
                                        MessageBox.Show(ex.ToString());
                                }


                        }
                }
        }
}


}
```

## Server Side:

ServerObj class:

```
using System;
using System.IO;
using System.Text;
using System.Threading;
using System.Reflection;
using System.Net;
using System.Net.Sockets;
using System.Windows.Forms;



public sealed class ServerObj
{
	public ObjectList objList = null;
	private System.Collections.ArrayList syncList = null;
	private AsyncCallback onConnect = null;
	private AsyncCallback registerClient = null;
	private AsyncCallback onReceive = null;
	private AsyncCallback onSend = null;
	private ManualResetEvent itsDone = null;
	private ManualResetEvent ready = null;
	private IPEndPoint endPoint = null;
	private Socket socket = null;
	private Encoding ASCII = null;
	public ServerObj()
	{
		this.objList = new ObjectList();
		this.syncList = new System.Collections.ArrayList();
		this.onConnect = new AsyncCallback(this.ClientConnection);
		this.registerClient = new AsyncCallback(this.ClientRegistration);
		this.onReceive = new AsyncCallback(this.Receive);
		this.onSend = new AsyncCallback(this.Send);
		this.itsDone = new ManualResetEvent(false);
		this.ready = new ManualResetEvent(false);
		IPHostEntry hosts = Dns.GetHostByName(Dns.GetHostName());
		this.endPoint = new IPEndPoint(hosts.AddressList[0], 8052);
		this.ASCII = Encoding.ASCII;
	}
	public void Start()
	{
		try
		{
			this.socket = new Socket(AddressFamily.InterNetwork,
SocketType.Stream, ProtocolType.Tcp);
			this.socket.Bind(this.endPoint);
			this.socket.Listen(1000);
			for (; ;)
			{
				this.itsDone.Reset();
				this.socket.BeginAccept(this.onConnect, this.socket);
				this.itsDone.WaitOne();
			}
		}
		catch (Exception e)
		{
			throw e;
		}
	}
	public void Stop()
	{
		try
		{
			string sht = "SHUTDOWN";
			byte[] shutdown = this.ASCII.GetBytes(sht.ToCharArray());
			for (int i = 0; i < this.objList.Count; i++)
			{
				((Client)this.objList[i]).Socket.Send(shutdown,
shutdown.Length, SocketFlags.None);
```

```
                        ((Client)this.objList[i]).Shutdown();
                    }
                    this.objList.Clear();
                    this.objList = null;
                    this.syncList.Clear();
                    this.syncList = null;
                    this.socket.Close();
                }
                catch (Exception e)
                {
                    throw e;
                }
            }
            private void ClientConnection(IAsyncResult ar)
            {
                try
                {
                    this.itsDone.Set();
                    Socket sock = ((Socket)ar.AsyncState).EndAccept(ar);
                    string cnf = "OK";
                    byte[] conf = this.ASCII.GetBytes(cnf.ToCharArray());
                    Client client = new Client(sock, sock.RemoteEndPoint);
                    sock.Send(conf, conf.Length, SocketFlags.None);
                    client.Socket.BeginReceive(client.buffer, 0, Client.bufferSize,
SocketFlags.None, this.registerClient, client);
                }
                catch (Exception e)
                {
                    throw e;
                }
            }
            private void ClientRegistration(IAsyncResult ar)
            {
                Client client = null;
                try
                {
                    client = (Client)ar.AsyncState;
                    int bytesRead = client.Socket.EndReceive(ar);
                    if (bytesRead > 0)
                    {
                        string fullCommand = this.ASCII.GetString(client.buffer, 0,
bytesRead);
                        string[] command = fullCommand.Split(new char[] {'^'});
                        if (command[0].ToUpper().Equals("AUTH") == true)
                        {
                            for (int i = 0; i < this.objList.Count; i++)
                            {
                                if
(((Client)this.objList[i]).NickName.ToUpper().Equals(command[1].ToUpper()) == true)
                                {
                                    string errCode = "100";
                                    byte[] errVal =
this.ASCII.GetBytes(errCode.ToCharArray());
                                    client.Socket.Send(errVal,
errVal.Length, SocketFlags.None);
                                    client.Shutdown();
                                    client = null;
                                    return ;
                                }
                                else
                                {
                                    continue;
                                }
                            }
                            string connConfirm = "001";
                            byte[] connConf =
this.ASCII.GetBytes(connConfirm.ToCharArray());
                            client.NickName = command[1];
                            client.Socket.BeginSend(connConf, 0,
connConf.Length, SocketFlags.None, new AsyncCallback(this.SendWlc), client);
                            string jMessage = "JOINS^" + command[1];
                            byte[] jns =
this.ASCII.GetBytes(jMessage.ToCharArray());
                            for (int i = 0; i < this.objList.Count; i++)
```

```
                                        {
                                                ((Client)this.objList[i]).Socket.Send(jns,
jns.Length, SocketFlags.None);
                                        }
                                        this.ready.WaitOne();
                                        string iList = this.GetActiveUserList();
                                        if (iList.Trim().Equals("") == true)
                                        {
                                                iList = "LIST^EMPTY";
                                        }
                                        else
                                        {
                                                iList = "LIST^" + iList;
                                        }
                                        byte[] iLst =
this.ASCII.GetBytes(iList.ToCharArray());
                                        client.Socket.Send(iLst, iLst.Length,
SocketFlags.None);
                                        this.syncList.Add(client);
                                        this.syncList.Sort();
                                        this.objList.Clear();
                                        for (int i = 0; i < this.syncList.Count; i++)
                                        {
                                                this.objList.Add((Client)this.syncList[i]);
                                        }
                                        client.Socket.BeginReceive(client.buffer, 0,
Client.bufferSize, SocketFlags.None, this.onReceive, client);
                                }
                                else
                                {
                                        client.Shutdown();
                                        client = null;
                                }
                        }
                        else
                        {
                                client.Shutdown();
                                client = null;
                        }
                }
                catch (Exception e)
                {
                        throw e;
                }
        }
        private void SendWlc(IAsyncResult ar)
        {
                try
                {
                        Client client = (Client)ar.AsyncState;
                        client.Socket.EndSend(ar);
                        this.ready.Set();
                }
                catch (Exception e)
                {
                        throw e;
                }
        }
        private void Receive(IAsyncResult ar)
        {
                Client client = null;
                try
                {
                        client = (Client)ar.AsyncState;
                        int bytesRead = client.Socket.EndReceive(ar);
                        if (bytesRead > 0)
                        {
                                string fullCommand = this.ASCII.GetString(client.buffer, 0,
bytesRead);
                                string[] command = fullCommand.Split(new char[] {'^'});
                                if (command[0].ToUpper().Equals("MSG") == true)
                                {
```

```
                                        string message = command[0] + "^" + client.NickName
+ "^" + command[1] + "^" +command[2]+"^"+ command[3]+"^" +command[4]+"^" +command[5]+"^"
+command[6]+"^" +command[7];
                                        byte[] msg =
this.ASCII.GetBytes(message.ToCharArray());
                                        for (int i = 0; i < this.objList.Count; i++)
                                        {

       ((Client)this.objList[i]).Socket.BeginSend(msg, 0, msg.Length, SocketFlags.None,
this.onSend, (Client)this.objList[i]);
                                        }
                                }
                                else if (command[0].ToUpper().Equals("SENDFIL") == true)
                                {
                                        int index = 1;
                                        for (int i = 0; i < this.objList.Count; i++)
                                        {
                                                if
(((Client)this.objList[i]).NickName.ToUpper().Equals(command[1].ToUpper()) == true)
                                                {
                                                        index = i;
                                                        break;
                                                }
                                                else
                                                {
                                                        continue;
                                                }
                                        }
                                        if (index != -1)
                                        {
                                                string sfmsg = command[0] + "^" +
client.NickName+ "^" + command[2]+ "^" + command[3];
                                                byte[] sendmsg =
this.ASCII.GetBytes(sfmsg.ToCharArray());

       ((Client)this.objList[index]).Socket.BeginSend(sendmsg, 0, sendmsg.Length,
SocketFlags.None, this.onSend, (Client)this.objList[index]);
                                                client.Socket.BeginReceive(client.buffer, 0,
Client.bufferSize, SocketFlags.None, this.onReceive, client);
                                        }
                                }
                                else if (command[0].ToUpper().Equals("WANTSDL") == true)
                                {
                                        int index = 1;
                                        for (int i = 0; i < this.objList.Count; i++)
                                        {
                                                if
(((Client)this.objList[i]).NickName.ToUpper().Equals(command[1].ToUpper()) == true)
                                                {
                                                        index = i;
                                                        break;
                                                }
                                                else
                                                {
                                                        continue;
                                                }
                                        }
                                        if (index != -1)
                                        {
                                                string dlmsg = command[0] + "^" +
client.NickName+ "^" + command[2]+ "^" + command[3] + "^" + command[4];
                                                byte[] sendmsg =
this.ASCII.GetBytes(dlmsg.ToCharArray());

       ((Client)this.objList[index]).Socket.BeginSend(sendmsg, 0, sendmsg.Length,
SocketFlags.None, this.onSend, (Client)this.objList[index]);
                                                client.Socket.BeginReceive(client.buffer, 0,
Client.bufferSize, SocketFlags.None, this.onReceive, client);
                                        }
                                }
                                else if (command[0].ToUpper().Equals("SENDFSZ") == true)
                                {
                                        int index = -1;
                                        for (int i = 0; i < this.objList.Count; i++)
```

```
                                        {
                                                if
(((Client)this.objList[i]).NickName.ToUpper().Equals(command[1].ToUpper()) == true)
                                                {
                                                        index = i;
                                                        break;
                                                }
                                                else
                                                {
                                                        continue;
                                                }
                                        }
                                        if (index != -1)
                                        {
string fsmsg = command[0] + "^" + client.NickName + "^" + command[2];
byte[] fsizemsg = this.ASCII.GetBytes(fsmsg.ToCharArray());
((Client)this.objList[index]).Socket.BeginSend(fsizemsg, 0, fsizemsg.Length,
SocketFlags.None, this.onSend, (Client)this.objList[index]);
client.Socket.BeginReceive(client.buffer, 0, Client.bufferSize, SocketFlags.None,
this.onReceive, client);
                                        }
                                }
                                else if (command[0].ToUpper().Equals("SYSMSG") == true)
                                {
                                        int index = -1;
                                        for (int i = 0; i < this.objList.Count; i++)
                                        {
if (((Client)this.objList[i]).NickName.ToUpper().Equals(command[1].ToUpper()) == true)
                                                {
                                                        index = i;
                                                        break;
                                                }
                                                else
                                                {
                                                        continue;
                                                }
                                        }
                                        if (index != -1)
                                        {
string symsg = command[0] + "^" + client.NickName + "^" + command[2];
byte[] sysmsg = this.ASCII.GetBytes(symsg.ToCharArray());
((Client)this.objList[index]).Socket.BeginSend(sysmsg, 0, sysmsg.Length,
SocketFlags.None, this.onSend, (Client)this.objList[index]);
client.Socket.BeginReceive(client.buffer, 0, Client.bufferSize, SocketFlags.None,
this.onReceive, client);
                                        }
                                }
                                else if (command[0].ToUpper().Equals("PRIVMSG") == true)
                                {
                                        int index = -1;
                                        for (int i = 0; i < this.objList.Count; i++)
                                        {
if (((Client)this.objList[i]).NickName.ToUpper().Equals(command[1].ToUpper()) == true)
                                                {
                                                        index = i;
                                                        break;
                                                }
                                                else
                                                {
                                                        continue;
                                                }
                                        }
                                        if (index != -1)
                                        {
string prmsg = command[0] + "^" + client.NickName + "^" + command[2];
byte[] privmsg = this.ASCII.GetBytes(prmsg.ToCharArray());
((Client)this.objList[index]).Socket.BeginSend(privmsg, 0, privmsg.Length,
SocketFlags.None, this.onSend, (Client)this.objList[index]);
                                                client.Socket.BeginReceive(client.buffer, 0,
Client.bufferSize, SocketFlags.None, this.onReceive, client);
                                        }


                                        else
```

```
                                        {
                        string errCode = "102";
                        byte[] errVal = this.ASCII.GetBytes(errCode.ToCharArray());
                        client.Socket.BeginSend(errVal, 0, errVal.Length, SocketFlags.None,
this.onSend, client);
                                        }
                                }
                                else if (command[0].ToUpper().Equals("QUIT") == true)
                                {
                                        if (this.syncList.Contains(client) == true)
                                        {
                                                int index = this.objList.IndexOf(client);
                                                if (index != -1)
                                                {
                                                        this.syncList.RemoveAt(index);
                                                        this.syncList.Sort();
                                                        this.objList.Clear();
                                                for (int i = 0; i < this.syncList.Count; i++)
                                                        {

        this.objList.Add((Client)this.syncList[i]);
                                                        }
                                        string qms = "QUITS^" + client.NickName;
                                        byte[] qmsg = this.ASCII.GetBytes(qms.ToCharArray());
                                        for (int i = 0; i < this.objList.Count; i++)
                                                        {
((Client)this.objList[i]).Socket.BeginSend(qmsg, 0, qmsg.Length, SocketFlags.None,
this.onSend, (Client)this.objList[i]);
                                                        }
                                                        client.Shutdown();
                                                        client = null;
                                                }
                                                else
                                                {
                                                        return ;
                                                }
                                        }
                                        else
                                        {
                                                return ;
                                        }
                                }
                                else
                                {
                                        string errCode = "101";
byte[] errVal = this.ASCII.GetBytes(errCode.ToCharArray());
client.Socket.BeginSend(errVal, 0, errVal.Length, SocketFlags.None, this.onSend, client);
                                }
                        }
                        else
                        {
                                if (this.syncList.Contains(client) == true)
                                {
                                        int index = this.objList.IndexOf(client);
                                        if (index != -1)
                                        {
                                                this.syncList.RemoveAt(index);
                                                this.syncList.Sort();
                                                this.objList.Clear();
                                        for (int i = 0; i < this.syncList.Count; i++)
                                                {

        this.objList.Add((Client)this.syncList[i]);
                                                }
                        string qms = "QUITS^" + client.NickName;
                        byte[] qmsg = this.ASCII.GetBytes(qms.ToCharArray());
                        for (int i = 0; i < this.objList.Count; i++)
                                                {

        ((Client)this.objList[i]).Socket.BeginSend(qmsg, 0, qmsg.Length, SocketFlags.None,
this.onSend, (Client)this.objList[i]);
                                                }
```

```
                                        client.Shutdown();
                                        client = null;
                                }
                                else
                                {
                                        return ;
                                }
                        }
                        else
                        {
                                client.Shutdown();
                                client = null;
                        }
                }
        }
        catch (Exception)
        {
                if (this.syncList.Contains(client) == true)
                {
                        int index = this.objList.IndexOf(client);
                        if (index != -1)
                        {
                                this.syncList.RemoveAt(index);
                                this.syncList.Sort();
                                this.objList.Clear();
                                for (int i = 0; i < this.syncList.Count; i++)
                                {
                                        this.objList.Add((Client)this.syncList[i]);
                                }
                                string qms = "QUITS^" + client.NickName;
                        byte[] qmsg = this.ASCII.GetBytes(qms.ToCharArray());
                        for (int i = 0; i < this.objList.Count; i++)
                                {

        ((Client)this.objList[i]).Socket.BeginSend(qmsg, 0, qmsg.Length, SocketFlags.None,
this.onSend, (Client)this.objList[i]);
                                }
                                client.Shutdown();
                                client = null;
                        }
                        else
                        {
                                return ;
                        }
                }
                else
                {
                        client.Shutdown();
                        client = null;
                }
        }
}
private void Send(IAsyncResult ar)
{
        Client client = null;
        try
        {
                client = (Client)ar.AsyncState;
                client.Socket.EndSend(ar);
                client.Socket.BeginReceive(client.buffer, 0, Client.bufferSize,
SocketFlags.None, this.onReceive, client);
        }
        catch (Exception)
        {
                if (this.syncList.Contains(client) == true)
                {
                        int index = this.objList.IndexOf(client);
                        if (index != -1)
                        {
                                this.syncList.RemoveAt(index);
                                this.syncList.Sort();
                                this.objList.Clear();
                                for (int i = 0; i < this.syncList.Count; i++)
                                {
```

```
                                        this.objList.Add((Client)this.syncList[i]);
                                }
                                string qms = "QUITS^" + client.NickName;
                        byte[] qmsg = this.ASCII.GetBytes(qms.ToCharArray());
                                for (int i = 0; i < this.objList.Count; i++)
                                {

        ((Client)this.objList[i]).Socket.BeginSend(qmsg, 0, qmsg.Length, SocketFlags.None,
this.onSend, (Client)this.objList[i]);
                                }
                                client.Shutdown();
                                client = null;
                        }
                        else
                        {
                                return ;
                        }
                }
                else
                {
                        client.Shutdown();
                        client = null;
                }
        }
    }
    private string GetActiveUserList()
    {
        try
        {
            string users = "";
            for (int i = 0; i < this.objList.Count; i++)
            {
                users += ((Client)this.objList[i]).NickName;
                if (i != (this.objList.Count - 1))
                {
                    users += "^";
                }
                else
                {
                    break;
                }
            }
            return users;
        }
        catch (Exception e)
        {
            throw e;
        }
    }
}
```

MBK_IMServer Form:

```
using System;
using System.IO;
using System.Text;
using System.Runtime;
using System.Threading;
using System.Reflection;
using System.Collections;
using System.Drawing;
using System.Windows.Forms;
using System.ComponentModel;

public sealed class MBK_IMServer : Form
{
    private Container components = null;
    private ListBox ListBox = null;
    private ServerObj obj = null;
    private Thread thread;
    public MBK_IMServer()
    {
        this.components = new Container();
```

```
            obj = new ServerObj();
            obj.objList.Change += new ObjectListEventHandler(this.Obj_Change);
            this.thread = new Thread(new ThreadStart(this.StartListening));
            this.thread.Priority = ThreadPriority.AboveNormal;
            this.thread.Start();

            this.InitializeComponent();

        }
        protected override void Dispose(bool disposing)
        {
            if ((disposing == true) && (this.components != null))
            {
                this.components.Dispose();
            }
            base.Dispose(disposing);
        }
        protected override void OnClosed(EventArgs e)
        {
            try
            {
                this.thread.Abort();
                GC.Collect();
                GC.WaitForPendingFinalizers();
            }
            catch (ThreadAbortException)
            {
                return ;
            }
            catch (Exception ex)
            {
                throw ex;
            }
            base.OnClosed(e);
        }
        public void InitializeComponent()…

        private void StartListening()
        {
            try
            {
                this.obj.Start();
            }
            catch (ThreadAbortException)
            {
                this.obj.Stop();
                return ;
            }
            catch (Exception)
            {
                this.obj.Stop();
            }
        }
        private void Obj_Change(object sender, ObjectListEventArgs e)
        {
            try
            {
                if (e.ChangeType.Equals("Add") == true)
                {
                this.ListBox.BeginUpdate();
                this.ListBox.Items.Add("Client Connected:
"+e.Current.ToString().Substring(17));
                        this.ListBox.EndUpdate();


                }
                else if (e.ChangeType.Equals("Clear") == true)
                {
                        this.ListBox.Items.Clear();
                }
                else
                {
                        return ;
                }
```

```
                }
                catch (Exception ex)
                {
                        MessageBox.Show(ex.Message.ToString());
                }
        }
        [STAThread]
        public static void Main(string[] args)
        {
                try
                {
                        Application.Run(new MBK_IMServer());
                }
                catch (Exception e)
                {
                        MessageBox.Show(e.Message.ToString());
                }
        }
}
```

ObjectListEventArgs class:

```
using System;
using System.Reflection;
using System.Collections;

public class ObjectListEventArgs : EventArgs
{
    private object current;
    private string changeType;
    public ObjectListEventArgs(object current, string changeType)
    {
        this.current = current;
       this.changeType = changeType;
    }
    public object Current
    {
        get
        {
            return this.current;
        }
    }
    public string ChangeType
    {
        get
        {
            return this.changeType;
        }
    }
}

public delegate void ObjectListEventHandler(object sender, ObjectListEventArgs e);

public class ObjectList : MarshalByRefObject, IEnumerator, IEnumerable, ICloneable
{
    private object[] obj;
    private object[] temp = null;
    public event ObjectListEventHandler Change;
    private int index;
    private int IEnumIndex;
    public ObjectList()
    {
        this.obj = new object[0];
       this.index = 0;
    }
    public virtual object this[int idx]
    {
        get
        {
            try
            {
                if (idx >=0 && idx < this.obj.Length)
                {
```

```
                return this.obj[idx];
            }
            else
            {
                throw new IndexOutOfRangeException("Wrong Index");
            }
        }
        catch (Exception e)
        {
            throw e;
        }
    }
    set
    {
        try
        {
            if (idx >= 0 && idx < this.obj.Length)
            {
                this.obj[idx] = value;
            }
            else
            {
                throw new IndexOutOfRangeException("Wrong Index");
            }
        }
        catch (Exception e)
        {
            throw e;
        }
    }
}
protected virtual void OnChange(ObjectListEventArgs e)
{
    try
    {
        if (this.Change != null)
        {
            this.Change(this, e);
        }
    }
    catch (Exception ex)
    {
        throw ex;
    }
}
public virtual int Add(object o)
{
    try
    {
        if (this.obj.Length > 0)
        {
            this.temp = new object[this.obj.Length];
            for (int i = 0; i < this.obj.Length; i++)
            {
                this.temp[i] = this.obj[i];
            }
            ++this.index;
            this.obj = null;
            this.obj = new object[this.index];
            for (int i = 0; i < this.temp.Length; i++)
            {
                this.obj[i] = this.temp[i];
            }
            this.obj[this.obj.Length - 1] = o;
            this.temp = null;
            ObjectListEventArgs e = new ObjectListEventArgs(o, "Add");
            this.OnChange(e);
            return (this.index - 1);
        }
        else
        {
            ++this.index;
            this.obj = new object[this.index];
            this.obj[0] = o;
```

```
            ObjectListEventArgs e = new ObjectListEventArgs(o, "Add");
            this.OnChange(e);
            return 0;
        }
    }
    catch (Exception e)
    {
        throw e;
    }
}
public virtual void AddRange(params object[] o)
{
    try
    {
        for (int i = 0; i < o.Length; i++)
        {
            this.Add(o[i]);
        }
    }
    catch (Exception e)
    {
        throw e;
    }
}
public virtual int Remove(object o)
{
    try
    {
        if (this.obj.Length <= 0)
        {
            throw new IndexOutOfRangeException("Wrong Index");
        }
        else
        {
            --this.index;
            ObjectListEventArgs e = null;
            int tmp = -1;
            int sVal = 0;
            bool IsBreak = false;
            this.temp = new object[this.index];
             for (int i = 0; i < this.obj.Length; i++)
            {
                 if (this.obj[i].Equals(o) == false)
                {
                     temp[sVal] = this.obj[i];
                    ++sVal;
                 }
                else
                {
                    if (IsBreak == false)
                    {
                        tmp = i;
                        e = new ObjectListEventArgs(this.obj[i], "Remove");
                        IsBreak = true;
                        continue;
                    }
                    else
                    {
                        temp[sVal] = this.obj[i];
                        ++sVal;
                    }
                }
             }
            this.obj = null;
            this.obj = new object[this.index];
            for (int i = 0; i < this.obj.Length; i++)
            {
                this.obj[i] = this.temp[i];
            }
            this.temp = null;
            return tmp;
        }
    }
    catch (Exception e)
```

```
    {
        throw e;
    }
}
public virtual void RemoveAt(int idx)
{
    try
    {
        if (idx < 0 || idx >= this.obj.Length || this.obj.Length <= 0)
        {
            throw new IndexOutOfRangeException("Wrong Index");
        }
        else
        {
            --this.index;
            ObjectListEventArgs e = null;
           this.temp = new object[this.index];
           int tmp = 0;
           for (int i = 0; i < this.obj.Length; i++)
           {
               if (idx != i)
               {
                   temp[tmp] = this.obj[i];
                  tmp++;
               }
               else
               {
                   e = new ObjectListEventArgs(obj[i], "RemoveAt");
                   continue;
               }
           }
           this.obj = null;
           this.obj = new object[this.index];
           for (int i = 0; i < this.temp.Length; i++)
           {
               this.obj[i] = this.temp[i];
           }
           this.temp = null;
           this.OnChange(e);
        }
    }
    catch (Exception e)
    {
        throw e;
    }
}
public virtual void Clear()
{
    try
    {
        this.obj = new object[0];
        this.index = 0;
        ObjectListEventArgs e = new ObjectListEventArgs(null, "Clear");
        this.OnChange(e);
    }
    catch (Exception e)
    {
        throw e;
    }
}
public virtual object[] ToObjectArray()
{
    try
    {
        if (this.obj.Length <= 0)
        {
            throw new Exception();
        }
        else
        {
            object[] retVal = new object[this.obj.Length];
           for (int i = 0; i < this.obj.Length; i++)
           {
               retVal[i] = this.obj[i];
```

```
            }
            return retVal;
        }
    }
    catch (Exception e)
    {
        throw e;
    }
}
public void Reverse()
{
    try
    {
        this.temp = new object[this.obj.Length];
        int tmp = 0;
        for (int i = (this.obj.Length - 1); i >= 0; i--)
        {
            this.temp[tmp] = this.obj[i];
           ++tmp;
        }
        this.obj = null;
        this.obj = new object[this.index];
        for (int i = 0; i < this.temp.Length; i++)
        {
            this.obj[i] = this.temp[i];
        }
        this.temp = null;
    }
    catch (Exception e)
    {
        throw e;
    }
}
public bool Contains(object o)
{
    try
    {
        bool IsContain = false;
        if (this.obj.Length <= 0)
        {
            throw new Exception();
        }
        else
        {
            for (int i = 0; i < this.obj.Length; i++)
           {
                if (this.obj[i].Equals(o) == true)
               {
                    IsContain = true;
                   break;
                }
               else
                {
                    continue;
                }
            }
        }
        return IsContain;
    }
    catch (Exception e)
    {
        throw e;
    }
}
public int IndexOf(object o)
{
    try
    {
        int retVal = -1;
        for (int i = 0; i < this.obj.Length; i++)
        {
            if (this.obj[i].Equals(o) == true)
           {
                retVal = i;
```

```
                break;
            }
            else
            {
                continue;
            }
        }
        return retVal;
    }
    catch (Exception e)
    {
        throw e;
    }
}
public int IndexOf(object o, int idx)
{
    try
    {
        int retVal = -1;
        for (int i = idx; i < this.obj.Length; i++)
        {
            if (this.obj[i].Equals(o) == true)
            {
                retVal = i;
                break;
            }
            else
            {
                continue;
            }
        }
        return retVal;
    }
    catch (Exception e)
    {
        throw e;
    }
}
public int LastIndexOf(object o)
{
    try
    {
        int retVal = -1;
        for (int i = (this.obj.Length - 1); i >= 0; i--)
        {
            if (this.obj[i].Equals(o) == true)
            {
                retVal = i;
                break;
            }
            else
            {
                continue;
            }
        }
        return retVal;
    }
    catch (Exception e)
    {
        throw e;
    }
}
public int LastIndexOf(object o, int idx)
{
    try
    {
        int retVal = -1;
        for (int i = idx; i >= 0; i--)
        {
            if (this.obj[i].Equals(o) == true)
            {
                retVal = i;
                break;
            }
```

```
            else
            {
                continue;
            }
        }
        return retVal;
    }
    catch (Exception e)
    {
        throw e;
    }
}
public ObjectList GetShallowClone()
{
    try
    {
        ObjectList tmpRet = this.MemberwiseClone() as ObjectList;
        if (tmpRet != null)
        {
            return tmpRet;
        }
        else
        {
            return null;
        }
    }
    catch (Exception e)
    {
        throw e;
    }
}
public virtual int Count
{
    get
    {
        return this.index;
    }
}
void IEnumerator.Reset()
{
    this.IEnumIndex = -1;
}
bool IEnumerator.MoveNext()
{
    ++this.IEnumIndex;
    if (this.IEnumIndex >= this.obj.Length)
    {
        return false;
    }
    else
    {
        return true;
    }
}
object IEnumerator.Current
{
    get
    {
        return this.obj[IEnumIndex];
    }
}
IEnumerator IEnumerable.GetEnumerator()
{
    IEnumerator RetVal = this as IEnumerator;
    if (RetVal != null)
    {
        return RetVal;
    }
    else
    {
        return null;
    }
}
object ICloneable.Clone()
```

```
    {
        try
        {
            if (this.obj.Length <= 0 || this.obj == null)
            {
                return null;
            }
            else
            {
                ObjectList tmp = new ObjectList();
               for (int i = 0; i < this.obj.Length; i++)
               {
                   tmp.Add(this.obj[i]);
               }
               return tmp;
            }
        }
        catch (Exception e)
        {
            throw e;
        }
    }
}
```