A Mini Project report on

**BizzChat – A Chat Application**

Submitted by

**Navya R S, 14CO126**

**Sheetal Shalini, 14CO142**

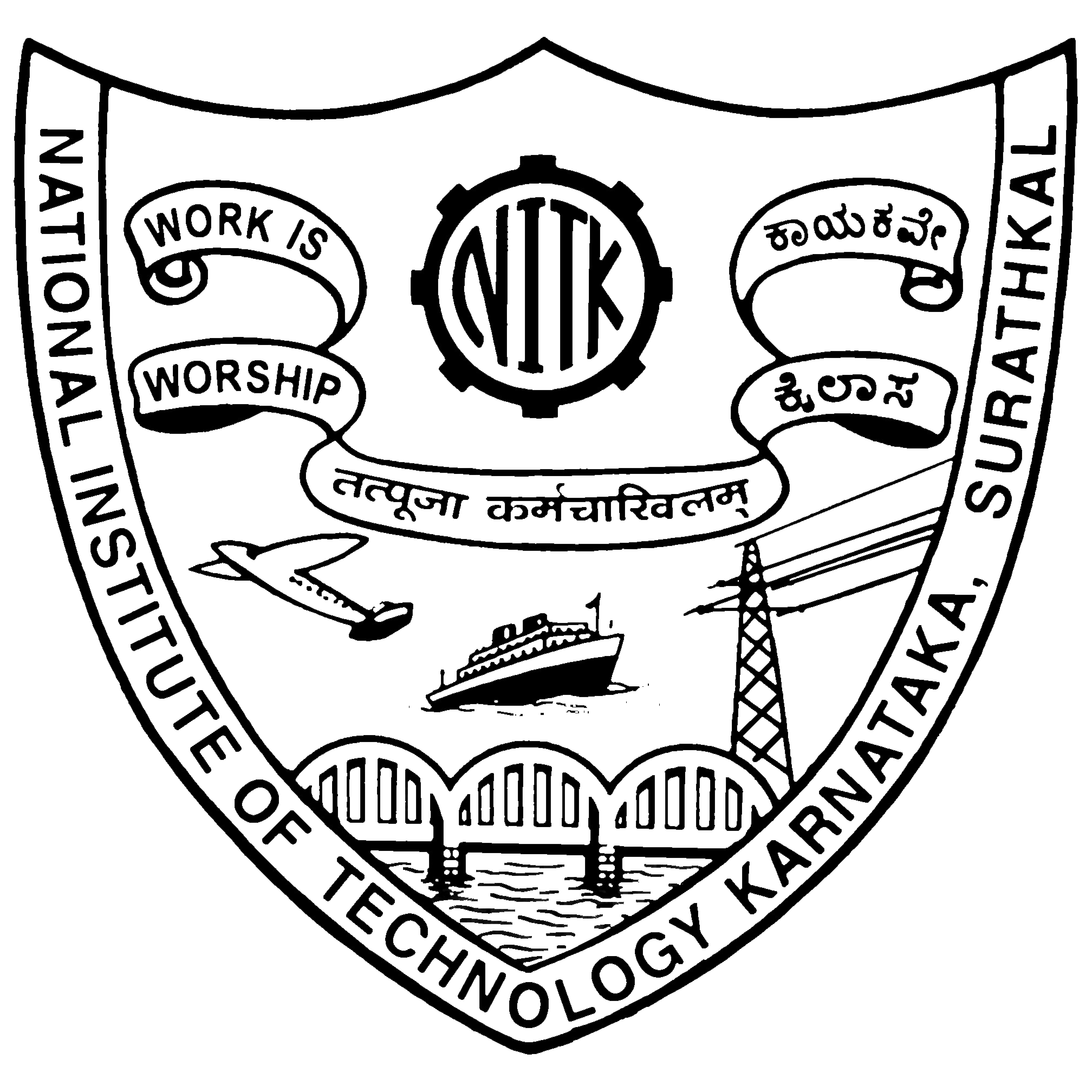
**S R Rimitha, 14CO137**

In partial fulfilment for the award of the degree of

**BACHELOR OF TECHNOLOGY**

in

**COMPUTER SCIENCE & ENGINEERING**



**Department of Computer Science and Engineering**

**National Institute of technology Karnataka, Surathkal**

**22nd April 2016**

**BIZZCHAT – A CHAT APPLICATION**

A platform for people to connect

**CERTIFICATE**

This is to certify that the project work under titled as Chat Application is the bonfire work of Navya R S(14CO126), Sheetal Shalini(14CO142) and S R Rimitha(14CO137) during this project submission as a partial fulfillment of the requirement for the Object Oriented Programming project of Bachelor in Technology IV Semester, of the National Institute of Technology Karnataka, Surathkal.

Project Supervisor H.O.D of Computer Dept.

**ACKNOWLEDGEMENT**

My heartfelt gratitude goes to **Ms. Sharvari**, department of Computer Science and Engineering, NITK, Surathkal, for giving me the guidance and motivation to carry out this project which was imperative in the completion of this project. Their knowledge in this subject and constructive criticism helped me steer the project to its success.

1. Navya R S

14CO126

1. Sheetal Shalini

14CO142

1. S R Rimitha

14CO137

**ABSTRACT**

BizzChat is a chat application for business as well as personal purposes. It facilitates multiple users to communicate with each other when they are connected to the same network. Initially, the server is set up by mentioning the port number. Once this is done, multiple clients can connect to it by entering the port number of the server and its own IP address. Connection establishment is then done, after which the clients can start chatting with each other. Once they enter their message in the chat textbox and click on the send symbol button, their message will be sent to the other client, and the same will appear on their screen, as well as that of the receiver. The users can also transfer audio, video, image and other type of files by dropping them onto the ‘Drag and Drop’ section provided on the chat window. As soon as the files are dropped there, they are put onto the socket of the receiver and is then sent to him and ready for download. Backup provisions have also been provided, wherein chats can be stored and saved in any directory of the computer, as chosen by the user. These messages can then be retrieved later for future use. Numerous styling as well as profile updation techniques have also been provided in the Settings window of the chat application. These functionalities and attributes are meant to improve the user interface as well as his/her experience, and provide all the necessary requirements that a user would look for in a chat application.

**Table of contents**

|  |  |  |
| --- | --- | --- |
| Sl No. | Topic | Page No. |
| 1 | Introduction | 1 |
| 2 | Product Overview | 4 |
| 3 | Networking | 5 |
| 4 | Graphical User Interface | 9 |
| 5 | Implementation and Workflow | 18 |
| 6 | Conclusion | 19 |
| 7 | References | 20 |

**INTRODUCTION**

On the Internet, chatting is talking to other people who are using the Internet at the same time you are. In some cases, a private chat can be arranged between two parties who meet initially in a group chat. Chat Application is primarily meant based group chat to refer to direct one-on-one chat or text. DVS Technologies provide both type of chat application. DVS Technologies chat application facilitates real-time communication, with its superior feature set and unprecedented reliability. DVS Technologies chat application increases the efficiency of the communication process.

Teleconferencing or Chatting, is a method of using technology to bring people and ideas “together” despite of the geographical barriers. The technology has been available for years but the acceptance it was quit recent. Our project is an example of a chat server. It is made up of 2 applications the client application, which runs on the user’s Pc and server application, which runs on any Pc on the network. To start chatting client should get connected to server where they can practice two kinds of chatting, public one (message is broadcasted to all connected users) and private one (between any 2 users only) and during the last one security measures were taken.

BizzChat is a chat application which helps us to communicate with others by sending and receiving instant messages. It also facilitates all kinds of file transfer like image, audio, video and so on. There is a client and server which is set up when the application is run for the first time. If we choose to set up a server, then we need to mention the port number, and if we choose to set up a client, then we need to mention the port number of the client as well as the IP address in order to connect to the server. Once the connection has been established, the chat window opens up for the client, and he or she can chat with other clients connected to the same network. The chat window has several attributes associated with it, which will be explained in the technicalities section.

We have come up with a chat application for this project. The general features expected from a chat application are:

1. Multiple user’s sustainability.
2. Private Messaging
3. Option to create Private rooms/Public rooms
4. File Sharing
5. GitHub integration
6. The ability to "mention" someone
7. Quoting
8. Add formatting attributes like Headings, Code, strong, emphasize and so on.

A fast chat application with a good UI/UX is all what the user wants. From a client’s perspective, they need a live chat that can quickly get their questions answered or problems resolved. So, efficiency is the biggest need.  
  
From a company’s perspective, if you run a small online shop or niche service, you may need a live chat service that allows you to [interact proactively with visitors](http://www.comm100.com/livechat/features/proactive-chat-invitation.aspx) to help convert browsers into buyers.  
If our company is a large enterprise, we may be looking for advanced features such as scheduling and reporting capabilities. While virtually all live-chat software enables us to interact with visitors with basic chat and message functions, there are a ton of other features and needs that we may, or may not, need to consider. Apart from these customer-demand features, the basic functionalities that a user looks for, in a chat application are:

1. Messaging
2. Group Messaging capabilities
3. Block/deny people
4. Add people
5. Voice messaging
6. Video messaging
7. Available on all devices
8. Attractive UI

Our implementation of the chat application is a basic model, which includes the basic and necessary functionalities like client-to-client messaging, setting up client and server as well as file transfers which include audio, video and images. Backup provisions have also been provided. The UI of the app has been designed with utmost detailing. We have incorporated features like changing and updating the profile, profile picture, changing background features like background color as well as other font and styling details.

All these technicalities have been described in the sections to come. The main sections which follow are Networking and GUI handling of the application.

**PRODUCT OVERVIEW**

BizzChat is a chat application which has the following attributes:

1. A basic client-server connection setup.
2. Multiple clients can chat with each other, by sharing the same network.
3. The chat window appears to a client as soon as he/she gets connected.
4. The chat window contains the username of the current user.
5. He/she can send messages to other clients, connected to the same network.
6. There are provisions for file transfers including images, audio and video.
7. Dragging and dropping files is also possible.
8. The Setting tab contains various features like updating profile, changing username and profile picture, changing font, styling and background details and taking backup of the chat. When the profile picture of a user is changed, the same is updated on the send button icon of the other client chatting with this user.
9. There are provisions such that when a file is dropped onto the ‘Drag and Drop Files’ section, it is automatically put onto the socket of the other client, and is then sent to him/her.
10. Any number of clients can be connected to the same server, and can communicate with each other.
11. The setting up of client and server is an important application of Networking. Its concepts have been explained in details in the networking section.
12. The GUI part contains J-Frames and other embodiments of user-interfacing, which have been explained in detail in the GUI section.
13. The app meets the basic functionalities of an application, as well as meets the necessary requirements of the user.

**NETWORKING:**

**SERVER:**

In [computing](https://en.wikipedia.org/wiki/Computing), a **server** is a [computer program](https://en.wikipedia.org/wiki/Computer_program) or a [device](https://en.wikipedia.org/wiki/Computer) that provides functionality for other programs or devices, called "[clients](https://en.wikipedia.org/wiki/Client_(computing))". This [architecture](https://en.wikipedia.org/wiki/Systems_architecture) is called the client-server model, and a single overall computation is distributed across multiple processes or devices. Servers can provide various functionalities, often called "services", such as sharing data or [resources](https://en.wikipedia.org/wiki/System_resource) among multiple clients, or performing computation for a client. A single server can serve multiple clients, and a single client can use multiple servers. A client process may run on the same device or may connect over a network to a server on a different device. In this system the client and server run on the same system.

The function of a computer server is to store, retrieve and send computer files and data to other computers on a network.

**CLIENT:**

 A **client** is a [computer](http://www.computerhope.com/jargon/c/computer.htm) that retrieves information from or uses resources provided by the [server](http://www.computerhope.com/jargon/s/server.htm) or main computer. It is the requesting program in a client-server relationship. For example, the user of a Web browser is effectively making client requests for pages from servers all over the Web. The browser itself is a client in its relationship with the computer that is getting and returning the requested HTML file. The computer handling the request and sending back the HTML file is a server. Each client connects to a central server or [mainframe](http://www.computerhope.com/jargon/m/mainfram.htm) that allows processing to be done on the **client side** instead of the server side and reduces the processing requirements of the server.

**CONNECTION ESTABLISHMENT:**

**SOCKET:**

A socket is one endpoint of a two-way communication link between two programs running on the network. A socket is bound to a port number so that the TCP layer can identify the application that data is destined to be sent to.

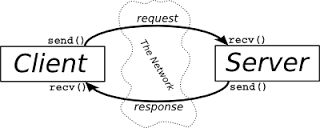
Normally, a server runs on a specific computer and has a socket that is bound to a specific port number. The server just waits, listening to the socket for a client to make a connection request.

On the client-side: The client knows the hostname of the machine on which the server is running and the port number on which the server is listening. To make a connection request. The client also needs to identify itself to the server so it binds to a local port number that it will use during this connection. This is usually assigned by the system.

If everything goes well, the server accepts the connection. Upon acceptance, the server gets a new socket bound to the same local port and also has its remote endpoint set to the address and port of the client. It needs a new socket so that it can continue to listen to the original socket for connection requests while tending to the needs of the connected client.

On the client side, if the connection is accepted, a socket is successfully created and the client can use the socket to communicate with the server.

The client and server can now communicate by writing to or reading from their sockets.

****

**TCP/IP :**

The **Transmission Control Protocol** (**TCP**) is a core [protocol](https://en.wikipedia.org/wiki/Communications_protocol) of the [Internet protocol suite](https://en.wikipedia.org/wiki/Internet_protocol_suite). It originated in the initial network implementation in which it complemented the [Internet Protocol](https://en.wikipedia.org/wiki/Internet_Protocol) (IP). Therefore, the entire suite is commonly referred to as *TCP/IP*. TCP provides [reliable](https://en.wikipedia.org/wiki/Reliability_(computer_networking)), ordered, and [error-checked](https://en.wikipedia.org/wiki/Error_detection_and_correction) delivery of a stream of [octets](https://en.wikipedia.org/wiki/Octet_(computing)) between applications running on hosts communicating over an IP network.4

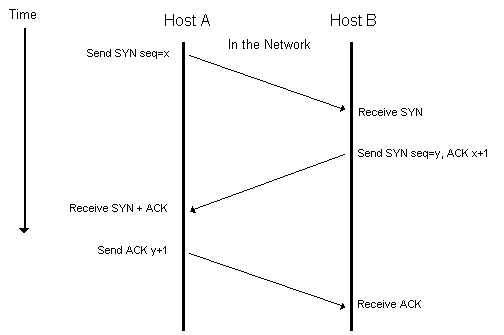
The Transmission Control Protocol provides a communication service at an intermediate level between an application program and the Internet Protocol. It provides host-to-host connectivity at the Transport Layer of the Internet model. An application does not need to know the particular mechanisms for sending data via a link to another host, such as the required packet fragmentation on the transmission medium. At the transport layer, the protocol handles all handshaking and transmission details and presents an abstraction of the network connection to the application.

At the lower levels of the protocol stack, due to network congestion, traffic load balancing, or other unpredictable network behaviour, IP packets may be lost, duplicated, or delivered out of order. TCP detects these problems, requests retransmission of lost data, rearranges out-of-order data, and even helps minimize network congestion to reduce the occurrence of the other problems. If the data still remains undelivered, its source is notified of this failure. Once the TCP receiver has reassembled the sequence of octets originally transmitted, it passes them to the receiving application. Thus, TCP abstracts the application's communication from the underlying networking details.

The following scenario occurs when a TCP connection is established:

1. The server must be prepared to accept an incoming connection. This is normally done by calling socket, bind, and listen and is called a passive open.
2. The client issues an active open by calling connect. This causes the client TCP to send a "synchronize" (SYN) segment, which tells the server the client's initial sequence number for the data that the client will send on the connection. Normally, there is no data sent with the SYN; it just contains an IP header, a TCP header, and possible TCP options (which we will talk about shortly).
3. The server must acknowledge (ACK) the client's SYN and the server must also send its own SYN containing the initial sequence number for the data that the server will send on the connection. The server sends its SYN and the ACK of the client's SYN in a single segment.
4. The client must acknowledge the server's SYN.

The minimum number of packets required for this exchange is three; hence, this is called TCP's three-way handshake.



While it takes three segments to establish a connection, it takes four to terminate a connection.

1. One application calls close first, and we say that this end performs the *active close*. This end's TCP sends a FIN segment, which means it is finished sending data.
2. The other end that receives the FIN performs the *passive close*. The received FIN is acknowledged by TCP. The receipt of the FIN is also passed to the application as an end-of-file (after any data that may have already been queued for the application to receive), since the receipt of the FIN means the application will not receive any additional data on the connection.
3. Sometime later, the application that received the end-of-file will close its socket. This causes its TCP to send a FIN.
4. The TCP on the system that receives this final FIN (the end that did the active close) acknowledges the FIN.

**FILE TRANSFER:**

File transfer happens between the server and the client by writing and reading from the socket. The file that has to be sent is converted into bytes and written on the socket. This is sent over the network and is written on the other endpoint. There it is read from the socket and is interpreted. Documents, PDFs, images, music, videos etc can be transferred over the network.

**CODING FOR NETWORK ESTABLISHMENT:**

In the java program **java.net**is for importing the ServerSocket and the Socket class. These classes are used for defining the server connection port number and also are used for accepting connections from the client.

The server always keeps listening. For this, the concept of multithreading is used. Server, as soon as it gets connected to a client redirects it to another port and continues listening to address other requests.

Client sends the request to the server and gets connected to another client via the server as server holds the records of which all clients are active or present in the network etc.

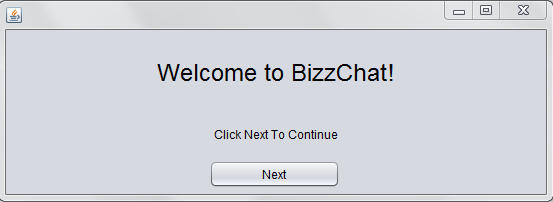
**GRAPHICAL USER INTERFACE:**

**INTRODUCTION:**

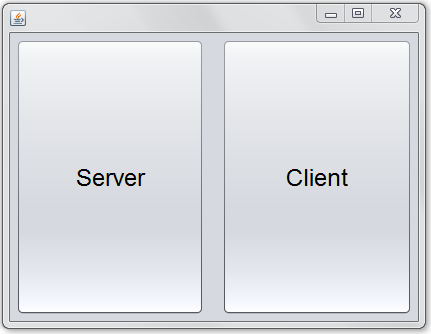
This section of the document mainly focuses on the Graphical User Interface of the Bizzchat Application. Since the application is built using NetBeans, we have majorly used JFrames and few inbuilt java classes to help with the implementation of the interface. It is to be noted that the java inbuilt class used swing. The major interface was designed using the design option given in NetBeans JFrame and the minor messages was displayed using the JOptionPane of the java swing class. It is also to be noted here that some of the interfaces was also hardcoded by giving the value for each and every component like size, colour, text boxes etc,. The text displayed are majorly JLabel fields while some may use JTextField too. The buttons in the application are implemented using JButtons and are made to listen to the mouse click by defining their action.

**AN OVERALL VIEW OF THE GRAPHICAL USER INTERFACE:**

The screenshot below displays the starting tab of the Application is a basic JFrame displaying a welcome message. This tab will appear twice. Once when the application is run for the server and once when the application is run for the client.



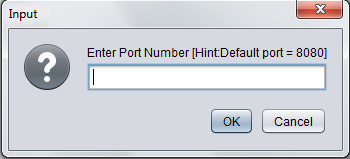
The starting tab of the application is followed by the following JFrame which lets the user choose either Server or Client in order to determine the type of user they are. The options can be chosen by clicking on the corressponding buttons.



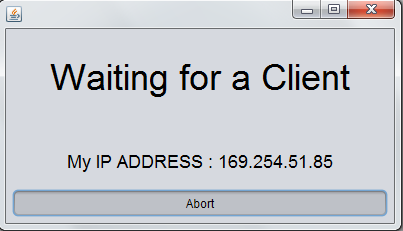
This is followed by a list of events that is different for the server and the client. Hence we not move onto show the Server side followed by the Client side.

**SERVER SIDE:**

In order to establish a secure connection between the server and the client, the port number is absolutely necessary. In this application, the user can determine what port number he/she requires.

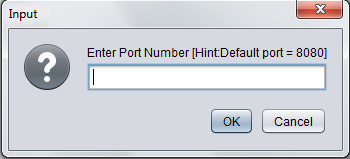


After inputting the port number, the application displays a JFrame containing the IP Address of the Server so that the client can connect to it using this address. This address is generated at random.

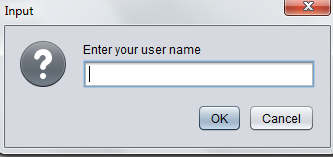


**CLIENT SIDE:**

The client side has the same question of port number in the JFrame displayed first.



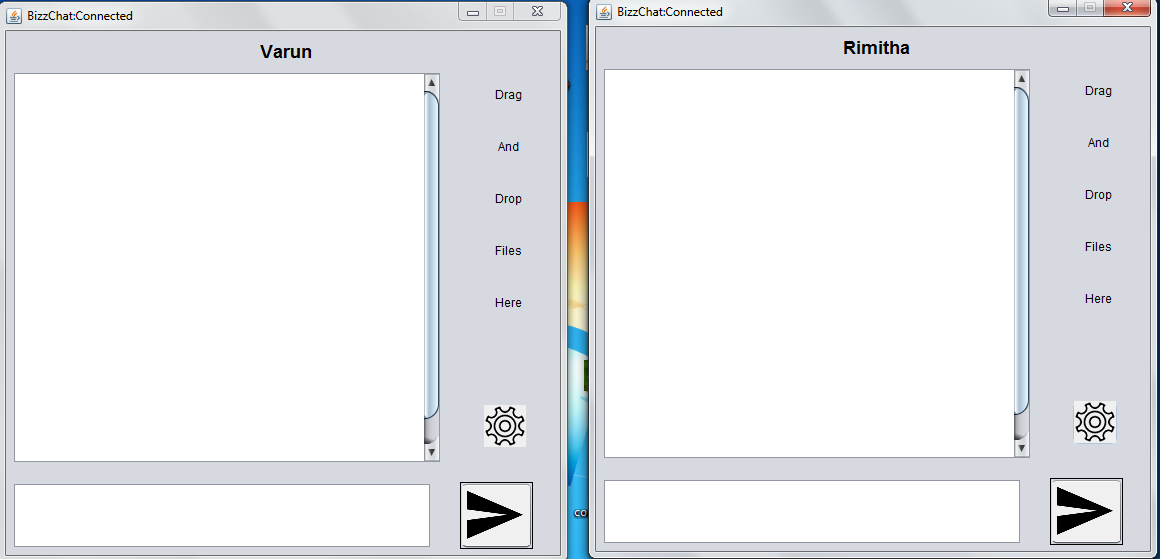
After successful establishment of connection, this is followed by the JFrame asking for the usernames of the two chat users. A screenshot is shown below:



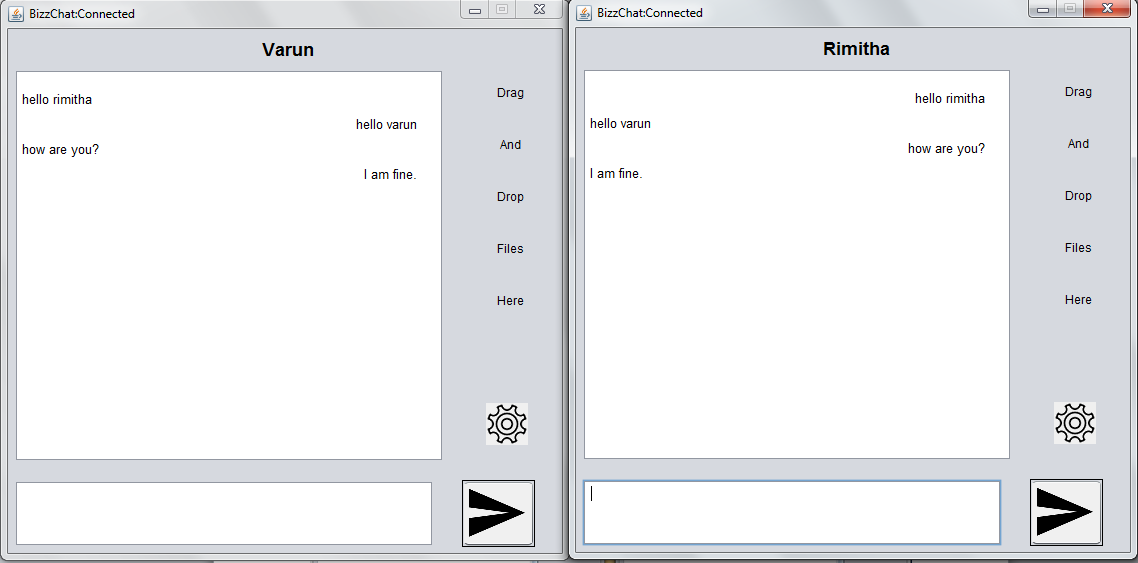
**CHAT BOX:**

The following screenshots shows how the chat box looks at three different states.

1)After the establishment of the connection

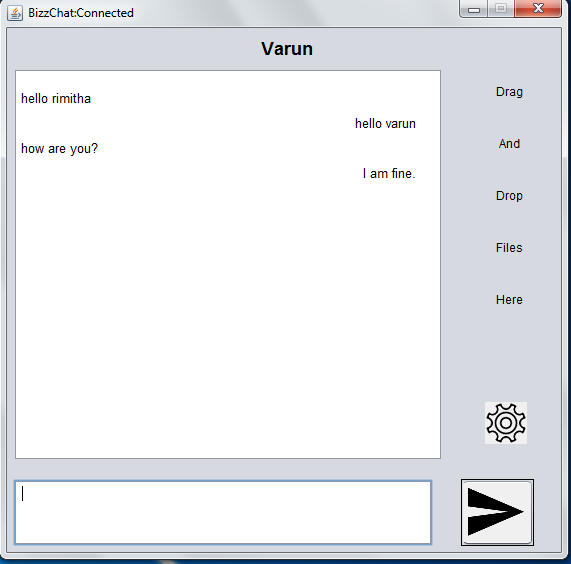


2)After some amount of chatting done between the users:



3)A clear view of the chat window of one user:

This is shown in order to highlight the spacing between the sender’s and the receiver’s messages. Like one can see the sender’s messages are aligned to the left whereas the receiver’s is aligned to the right.



**SETTINGS:**

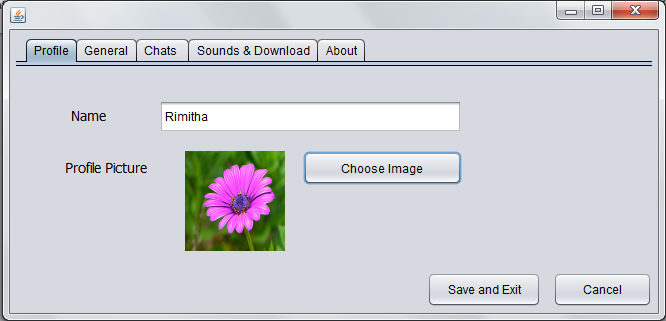
This particular section deals with the settings functionality of each and every chat. We are focusing more on the User Interface part of the setting.

Each chat window has a settings option which can be enabled by clicking on the wheel button.This settings has the following options:

1. Profile
2. General
3. Chats
4. Sounds and Downloads
5. About

**PROFILE:**

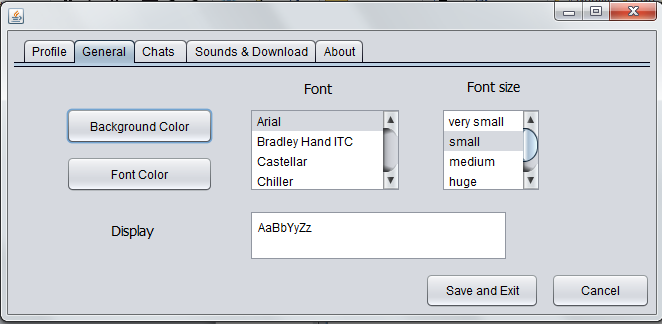
This JFrame displays the name of the user and allows the user to pick a profile picture for themselves using the JFileChooser function.



**GENERAL:**

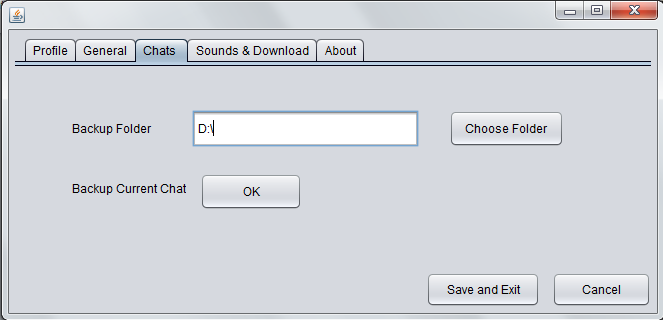
This tab gives the user a variety of options related to the aesthetic sense of the chat application. This includes:

1. Background Colour
2. Font Colour
3. Font type
4. Font size



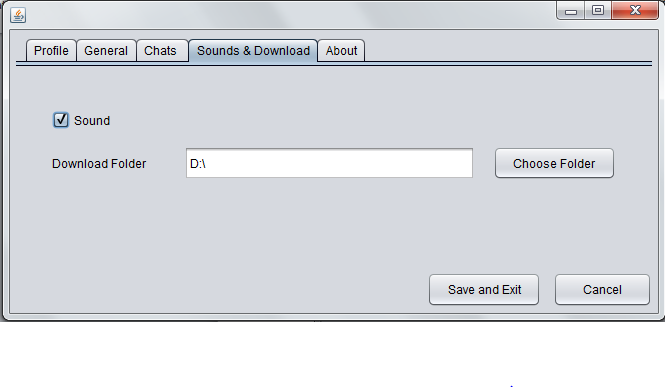
**CHATS:**

This Jframe allows the user to choose the folder in which he/she wants to store the chat backup. It must be noted here that the list of folders available are shown using the ShowDialogBox function and the folder is chosen using the FileChooser function.

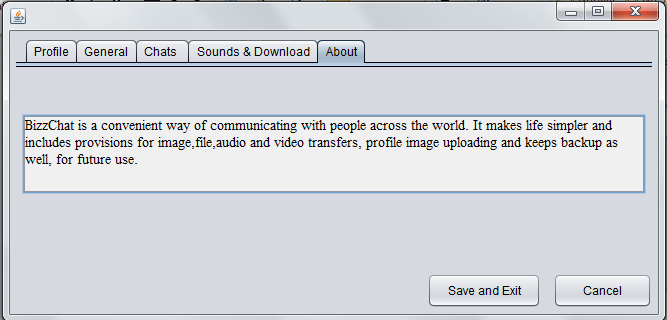


**SOUNDS AND DOWNLOADS:**

This Jframe allows the user to choose the folder in which he/she wants to store the files that are transferred. It must be noted here that the list of folders available are shown using the ShowDialogBox function and the folder is chosen using the FileChooser function.



**ABOUT:**

This is a basic JFrame that is used to display the essential information of the application. It could be timely edited by the creators of the application.

**IMPLEMENTION DETAILS OF SETTINGS:**

It must be noted here that all the JFrames of the settings option have used:

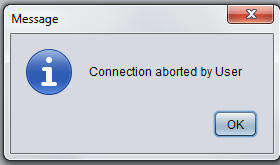
1. JPanel
2. JtextField
3. JButton
4. JLabel
5. JCheckBox

The Jpanel is used to create different panels for different options of the settings. Jtextfield is used to display short messages like the path of the backup folders of the about the developers information. JButton is used to create all the necessary buttons of this option. JLabel is used to display all the essential information and headers in each and every Frame. And the JCheckBox to create a check box for sounds.

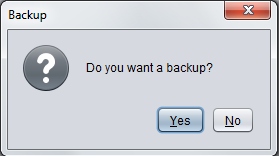
**ADDITONAL MESSAGES:**

Every application needs to alert the user if some kind of an event has occurred of if some kind or an event is going to occur. This part of the application is dealt by the use of JOptionPane which is widely used to display short messages to the user. Some of them are listed below with their screenshots:

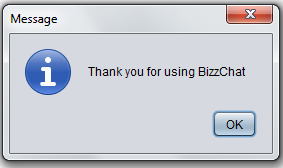
1. The following JOptionPane is displayed when one of the chat users decided to terminate their window. This is to notify the other user that the chat has been terminated.



2)This JOptionPane is used to ask every chat user if he/she requires a data backup or not at the end of every chat right before terminating:



3) This is the JOptionPane displayed when the user terminates the application before establishing a full fledged connection:



**IMPLEMENTATION AND WORKFLOW**

The workflow of the system is constituted by the following:

1. The user runs the application.
2. A J-Frame comes up, with client and server buttons. If we wish to set up the server, we click on that button. The server window opens, and we enter the port name. Multiple clients can be connected to this server.
3. Once a client logs in, he/she can establish a connection with this server by entering the port number of the server, as well as its own IP address.
4. Once the connection is made, the client can start chatting with other clients connected to the same network.
5. If any user wants to share any files like image, audio or video, they can drop the file onto the ‘Drag and Drop Files’ section on the chat window. These files are automatically transferred to the other client socket and are ready for download by the other client.
6. The user can also make changes like changing the background color, font size, styling and color. Formatting of text into Bold, Italics, Underline are also possible by clicking on the Font tab in the Setting window. The settings window can be open up by clicking on the settings wheel symbol near the send button on the chat window. This window includes tabs like Font, Backup, Download, Profile and About us.
7. Backup provisions are also made, wherein the previous chats of the user can be viewed, and the new chat can be stored, for future reference. The folder to which the backup has to be stored can be chosen from the drop down menu in the Backup tab of the Settings window.
8. In order to send a message to the other client, the user can enter the message in the chat textbox and then click on the ‘Send’ symbol button. The message will appear on the chat screen of that user and also of the client to whom the message has been sent. It appears on the left side of the screen for the sender, but on the right side for the receiver. When the receiver then replies to that message, it appears on the left side on his window, but on the right side for the initial sender.

**CONCLUSION**

Once a system is fully implemented & being operated by end user & the maintenance function begins, system maintenance is the monitoring to evaluating &modifying of operational information system to make desirable or necessary improvements. For example, the implementation of a new system usually results in the phenomenon known as the learning curve. The person who operates &uses the system will make mistakes simply because they are not familiar with it .Though such errors usually diminished experience is gained with a new system, they do point out areas where system may be improved. Maintenance is also necessary for the failures and problems that arise during the operation of a system .the maintenance activity includes a post implementation of a system review process to ensure that newly implemented system meet the system development objective established for them . Errors in the development of a system must be corrected by the maintenance process .this includes a periodic review or audit of a system to ensure on operating system &meeting its objective. A project is successfully completed only when there is contribution of all the members of a team. Each member in the project has a required objective to accomplish .In same way to complete this project we divided our work in equal manner .Instead of selecting a particular task to do individually, we divided a single task in multiple subtasks so that we all can work together on the same phase or task of the project. Doing so no team member had to wait to show his or her capabilities.

Installing and operating a newly designed system or modifying an established application requires a detailed record of that system’s design. Documentation is important in diagnosing errors and making changes. Documentation serves as a method of communication among the people responsible for developing, implementing and maintaining a system.

Hence, through this project, we have implemented a chat system and explained its various functionalities including the networking as well as GUI details. This app is meant to provide ease of convenience to the users, along with fulfilling their basic needs. Thank You!

**REFERENCES**

1. Research paper on “Enhanced Chat Application” by “Avinash Bamane, Parikshit Bhoyar, Ashish Dugar & Lineesh Antony”.
2. Journal on “Design of Chatting Application Based on Android Bluetooth” by “Nikita Mahajan, Garima Verma, Gayatri Erale, Sneha Bonde and Divya Arya”.
3. Conference Publication on “The Design Method of Network Chat System Based on Socket and Cloud Computing” by “[Minqiang Cai](http://ieeexplore.ieee.org/search/searchresult.jsp?searchWithin=%22Authors%22:.QT.Minqiang%20Cai.QT.&newsearch=true) ; Sch. of Inf. & Eng., Huzhou Teacher Coll., Huzhou, China”.
4. “Chats for all: A user survey to improve chats' interaction” by “Rocío Calvo, Ana Iglesias and Lourdes Moreno”.
5. Research paper on “A Secure Chat Application Based on Pure Peer-to-Peer Architecture” by “Mohamad Afendee Mohamed, Abdullah Muhammed and Mustafa Man”.
6. Paper on “Performance of Video-Chat Applications Under Congestion” by “Omer Boyaci, Andrea G. Forte and Henning Schulzrinne”.