GROUP PROJECT – BSCS 21043

CHAT APPLICATION

Group number 2

Summary

This report is about the chat application. As the Group Project of Visual Programming we have prepared a chat application using Microsoft Visual studio. The program is written in C# language. We have used some references to prepare the code. Further studied about creating a socket and the network or the connection between server and user. Used the lecture materials of Visual programming And Data communication And Networks.

Communication in our current age largely digital and the most popular form of digital communication is instant messaging. So we have designed a chat application using visual studio.

CONTENT

1.Introduction

Communication is a mean for people to exchange messages. Communication began as early with the introduction of television, telegraph and then telephony. People use different type of chat applications to communicate in present. Chat refers to the process of communicating, interacting or exchanging messages over the Internet.

Our project is an example of a chat application which communicates between the server and the client. It is made up of 2 applications; namely the client application and the server application. We have included a IP Address for the program and server will be connected to a particular port. Then client can give a user name and it has to connect to the same network which user is connected. Here we have used a password. If the user and the client is considering about the security and privacy they can use a password to protect their information. Program is written as two different programs and connection between those two will be happened after the execution. The chat session is based on socket programming.

A socket is an endpoint in communication between two computers across a computer network. It is uniquely identified by an IP address, a port number, and a communications protocol. A socket can send and receive data over the network. Data is generally sent in blocks of few kilobytes at a time for efficiency. Each of these blocks are called a packet.

All packets that travel on the internet must use the Internet Protocol. This means that the source IP address, destination address must be included in the packet. Packets also contain a port number. A port is simply a number between 1 and 65,535 that is used to differentiate higher protocols. Ports are important when it comes to programming your own network applications because no two applications can use the same port.

A network protocol defines rules and conventions for communication between network devices. Network protocols include mechanisms for devices to identify and make connections with each other, as well as formatting rules that specify how data is packaged into sending and receiving messages. Some protocols also support message acknowledgment and data compression designed for reliable and/or high-performance network communication.

The Internet Protocol (IP) family contains a set of related protocols like TCP, UDP, HTTP and FTP, all integrated with IP to provide additional capabilities.

We have used TCP protocol to implement our chat application. TCP (Transmission Control Protocol) works with the Internet Protocol (IP), which defines how computers send packets of data to each other. TCP is a connection-oriented protocol, which means a connection is established and maintained until the application programs at each end have finished exchanging messages. It determines how to break application data into packets that networks can deliver, sends packets to and accepts packets from the network layer.

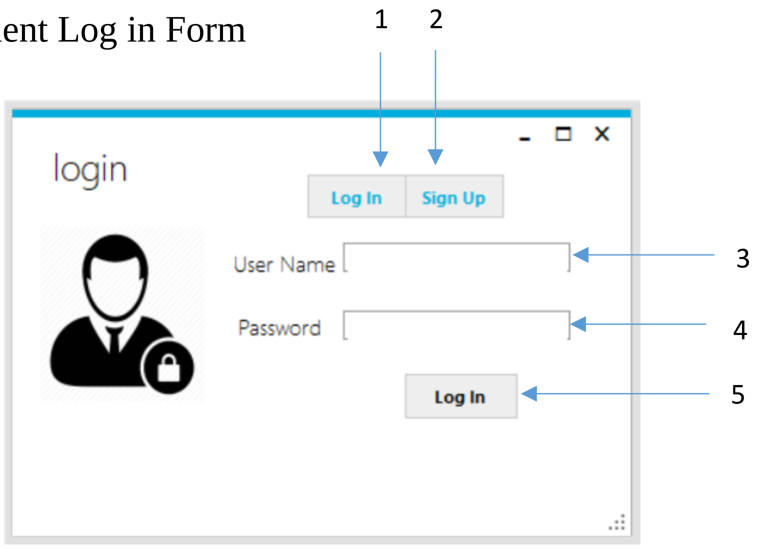
And also we used SMTP protocol to implement our application. SMTP (Simple Mail Transfer Protocol) is the standard protocol for email services on a TCP/IP network. SMTP provides the ability to send and receive email messages. SMTP is an application-layer protocol that enables the transmission and delivery of email over the Internet.

We have designed the program using Microsoft Visual Studio 2017 and the interface is created using it. The program is written in C# language. In the interface user can see the information about both client and server. Microsoft Visual Studio is a Integrated Development Environment (IDE) developed by Microsoft to develop GUI (Graphical User Interface), console, Web applications, web apps, mobile apps, cloud, and web services etc. With the help of this IDE, you can create managed code as well as native code. It uses the various platforms of Microsoft software development software like Windows store, Microsoft Silverlight, and Windows API etc. It is not a language specific IDE as you can use this to write code in C#, C++, VB (Visual Basic), Python, JavaScript, and many more languages. It is available for Windows as well as for macOS. Visual Studio automatically analyzes your code to point out errors and offer suggestions while you type. Step through your code line by line to quickly find problems in your code. Spend less time on setup and configuration means Visual Studio creates projects with relevant tools and runtimes.

2.Procedure

2.1Explanation of the forms

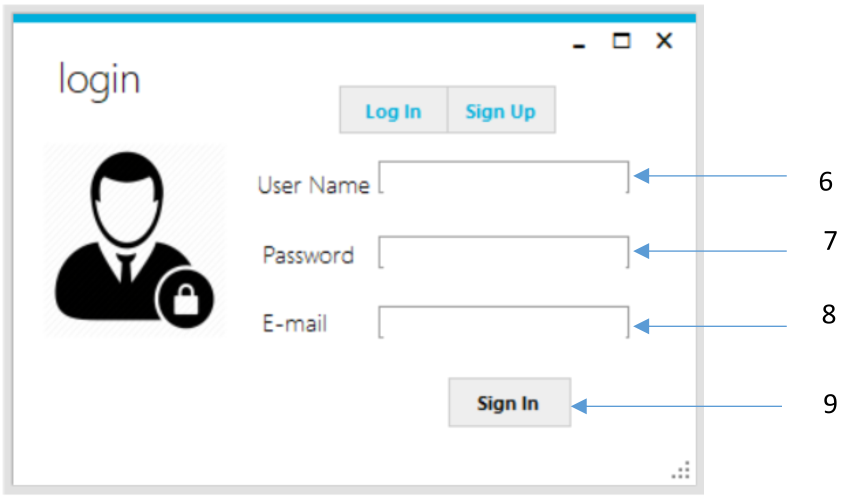
Client Log in Form



If user has signed up with the program before, he or she can log in to the program by providing the User name to the text box [3] and Password to the text box [4] and clicking on the Log In button [5].Both text box [3] and text box [4] should be filled before user clicks on the login button. Otherwise an error message will be generated.

If user use the program for the first time he or she should sign up before logging in to the program.

User can get the sign up window by clicking on the sign up tab [2].



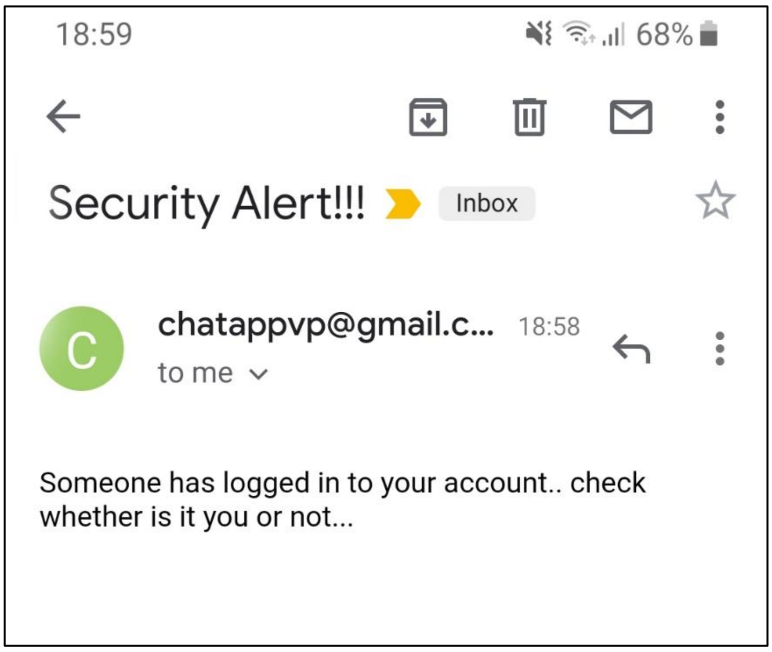
When user clicks on the sign up tab above form will be displayed. User has to give a suitable user name to the text box [6] and suitable password to the text box [7] and his or her e mail to the text box [8]. Then user should click on the sign in button [9]. All the fields should be filled when user click on the sign in button.

If user provides an existing user name an error message will be displayed.

Once clicked on the sign-in button it will be automatically switched to the Log in tab. Then user can log in to his or her newly created account by giving the user name and password.

When the user logs into the account, the below security alert will be sent to his or her e-mail account automatically to check whether it is a fake logging or not.

Security alert-



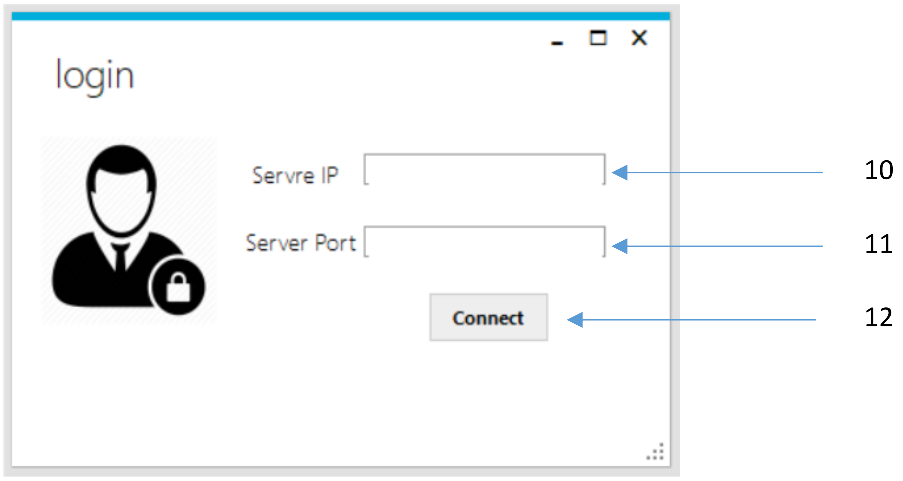
18:59 68%

Security Alert!!! Inbox

chatappvp@gmail.c... 18:58

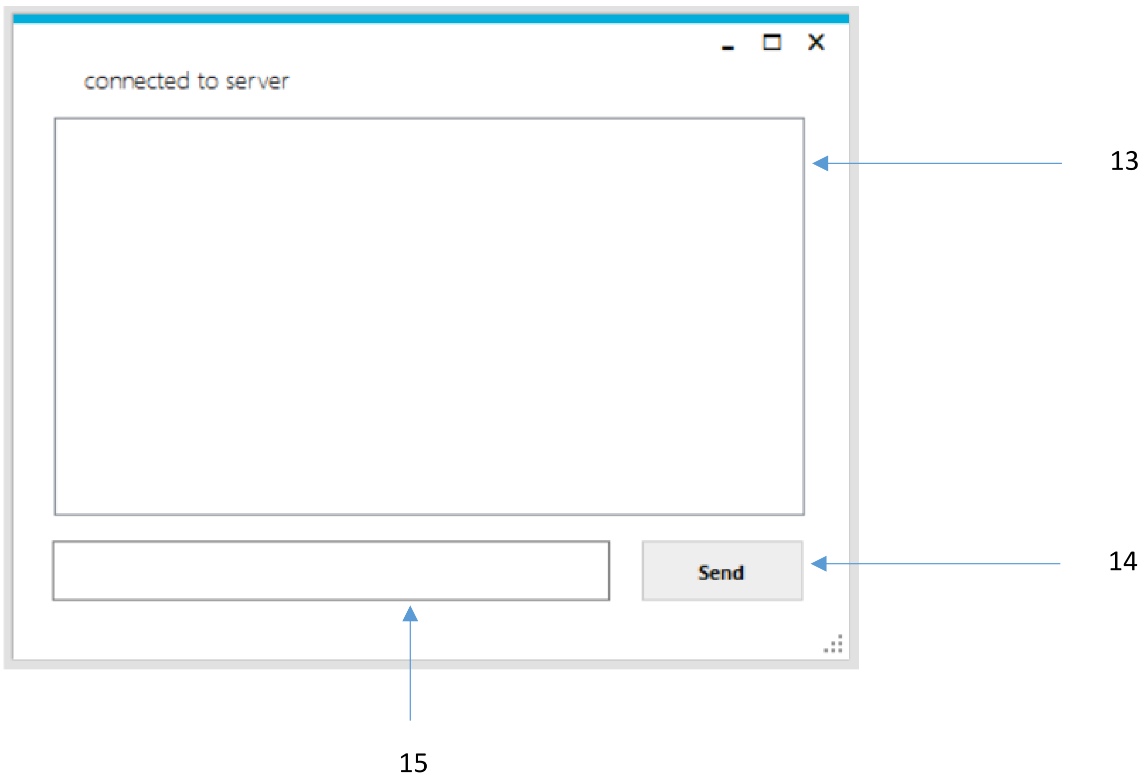
to me

Someone has logged in to your account.. check whether is it you or not...



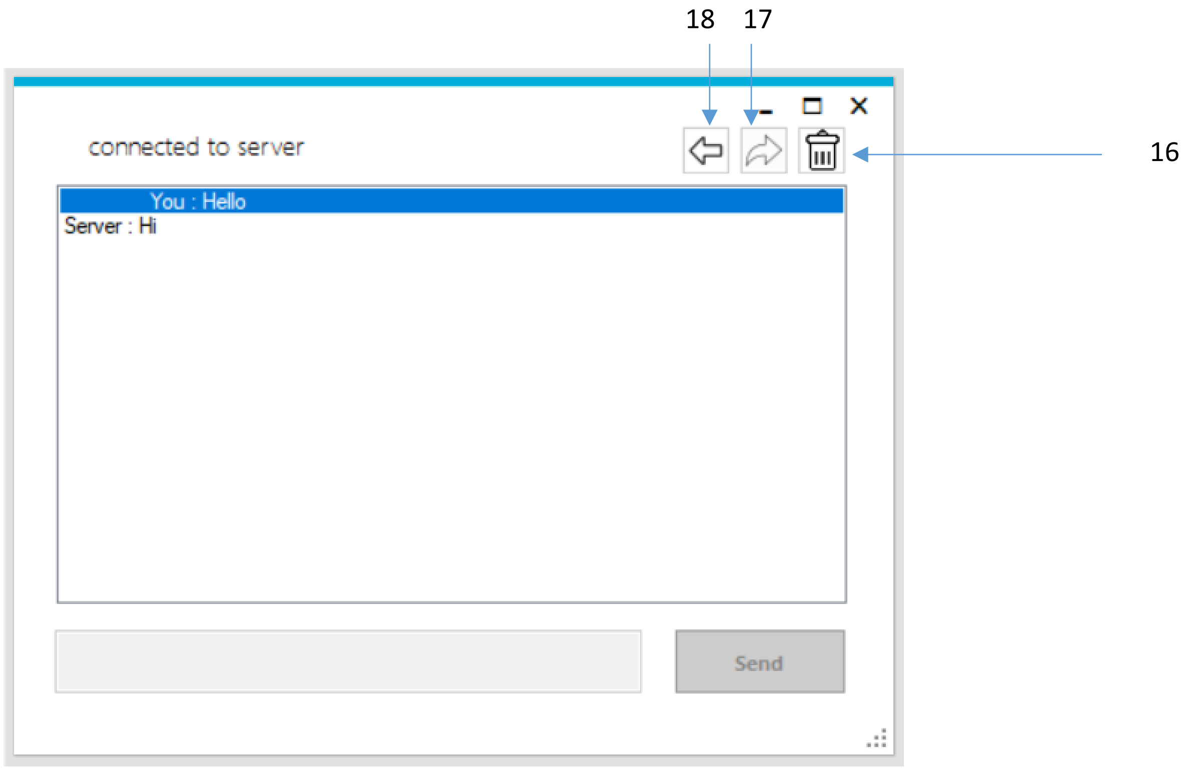
After user clicked on the Log in button [5], above form will be displayed. Now user should enter the Sever IP address to the text box [10] and server port number to the text box [11] and click on the connect button [12]. For the client to be connected to the server, the server should have been already executed. Otherwise it will not be connected and an error message will be displayed.

Client messaging form



Once connected, above form will be displayed. User can type messages in text box [15] and send them to the server by clicking send button [14]. Sending and receiving messages will be displayed in the list box [13].

If the user has logged in to the program and chat before, the previous chat will be displayed.

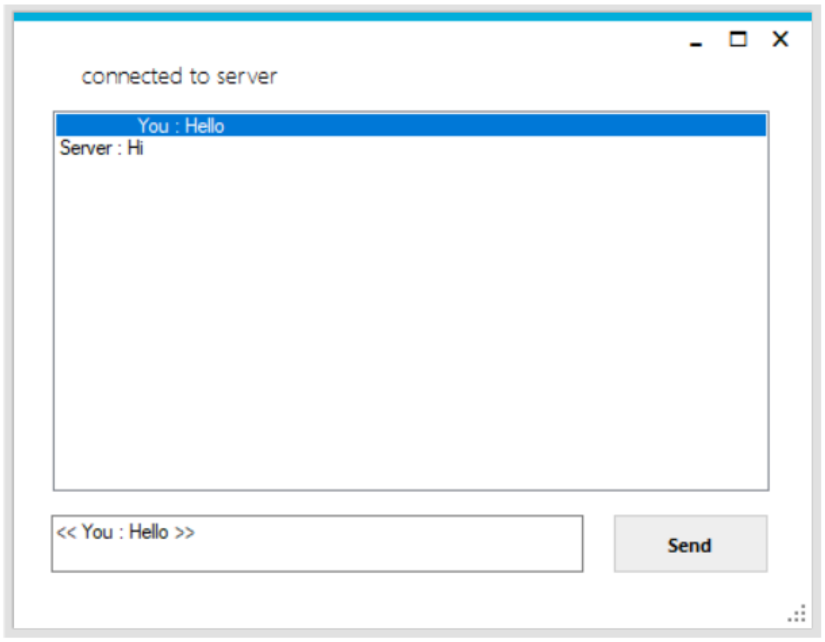


You : Hello
Server : Hi

If user double click on the particular message, button [16], button [17], button [18] will be displayed.

User can delete selected message by clicking on button [16]. User can reply to the selected message by clicking on button [17] and user can undo the action by clicking on button [18].

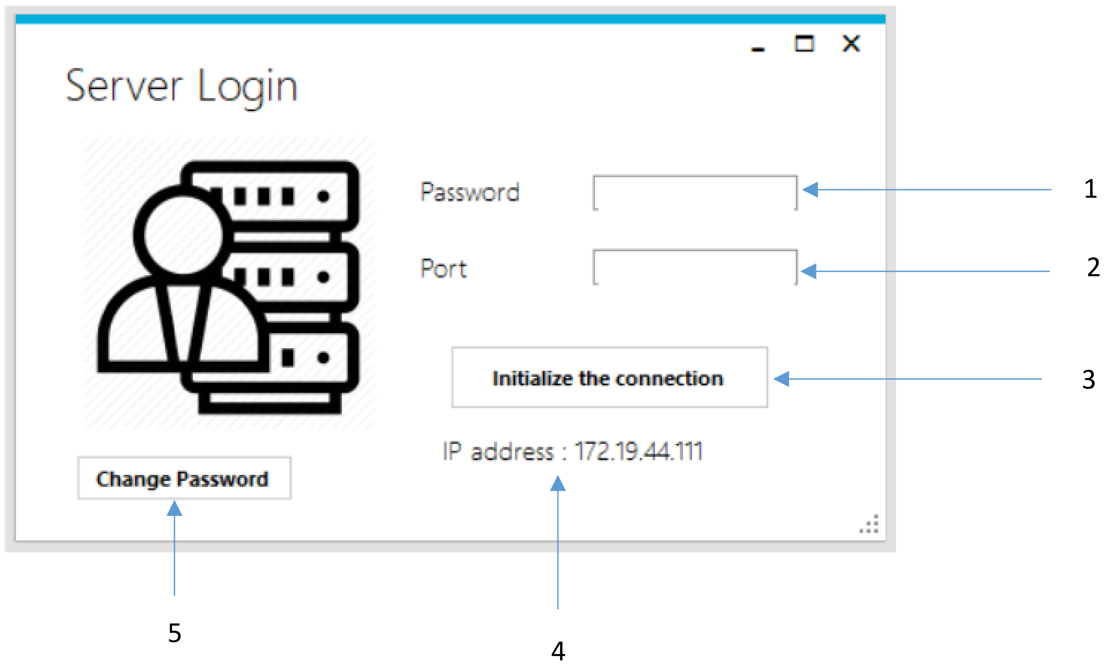
If user clicks on the button [17], selected message will be shown in the text box and now user can type a reply for it and send.



You : Hello
Server : Hi

<< You : Hello >>

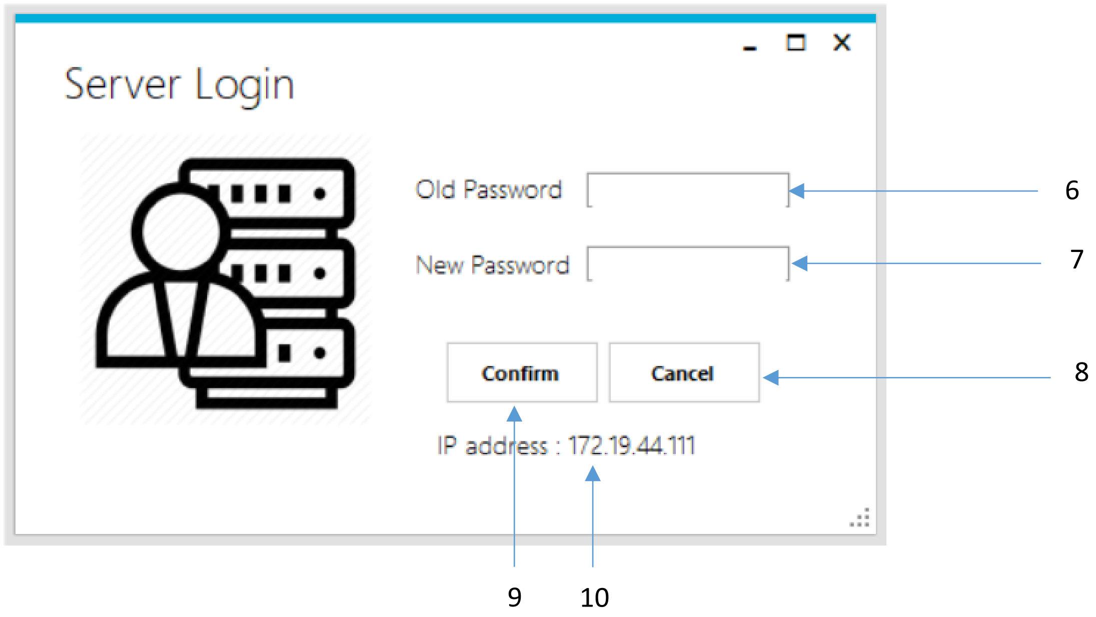
Server Login Form



Above is the Server Login form and in order to initiate the connection, server side user has to enter the password to text box [1] and port number to text box [2] and then click on the button [3].

Sever IP address is shown in the label [4].

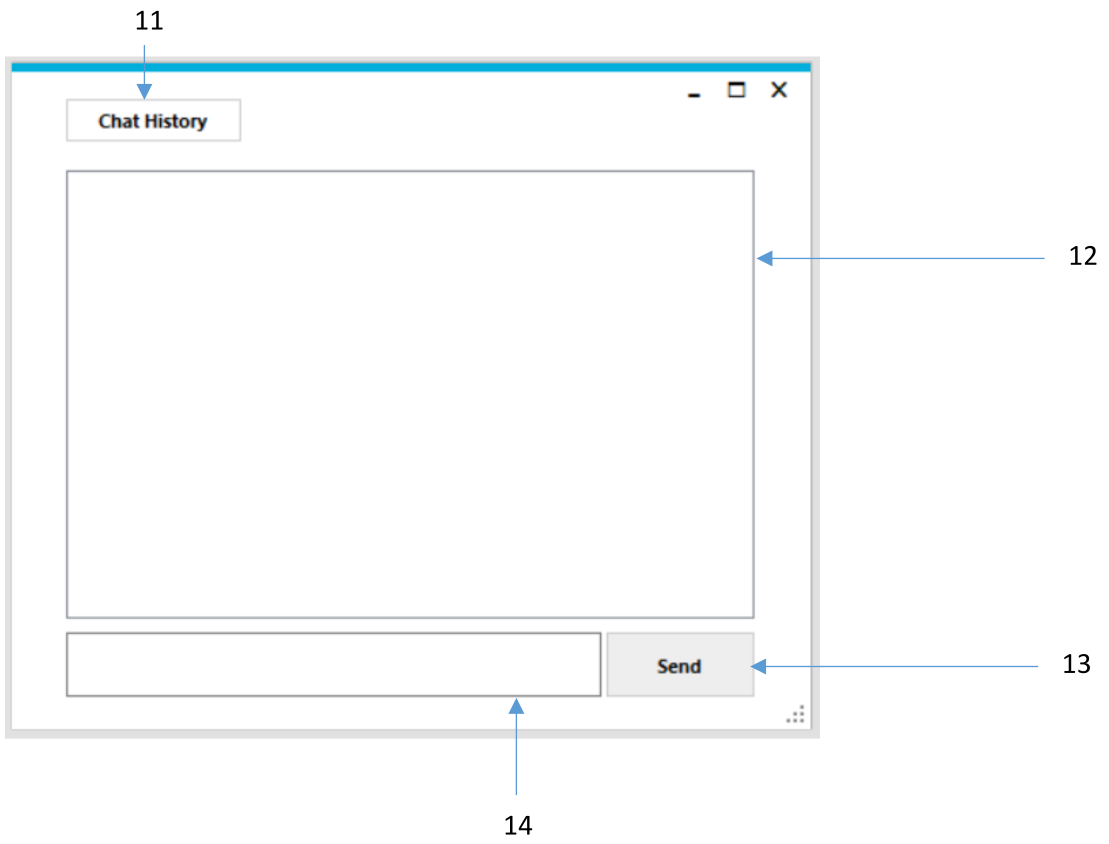
If server side user wants to change the password, user should click on the change password button [5].



When user clicked on the change password button [5] above figure will be shown. To change the password user should first enter current password to the text box [6] and new password to the text box [7] and click on the confirm button [9]. If user doesn't want to change the current password he or she can return to the previous state by clicking on the cancel button.

The label [10] is showing the Server IP address.

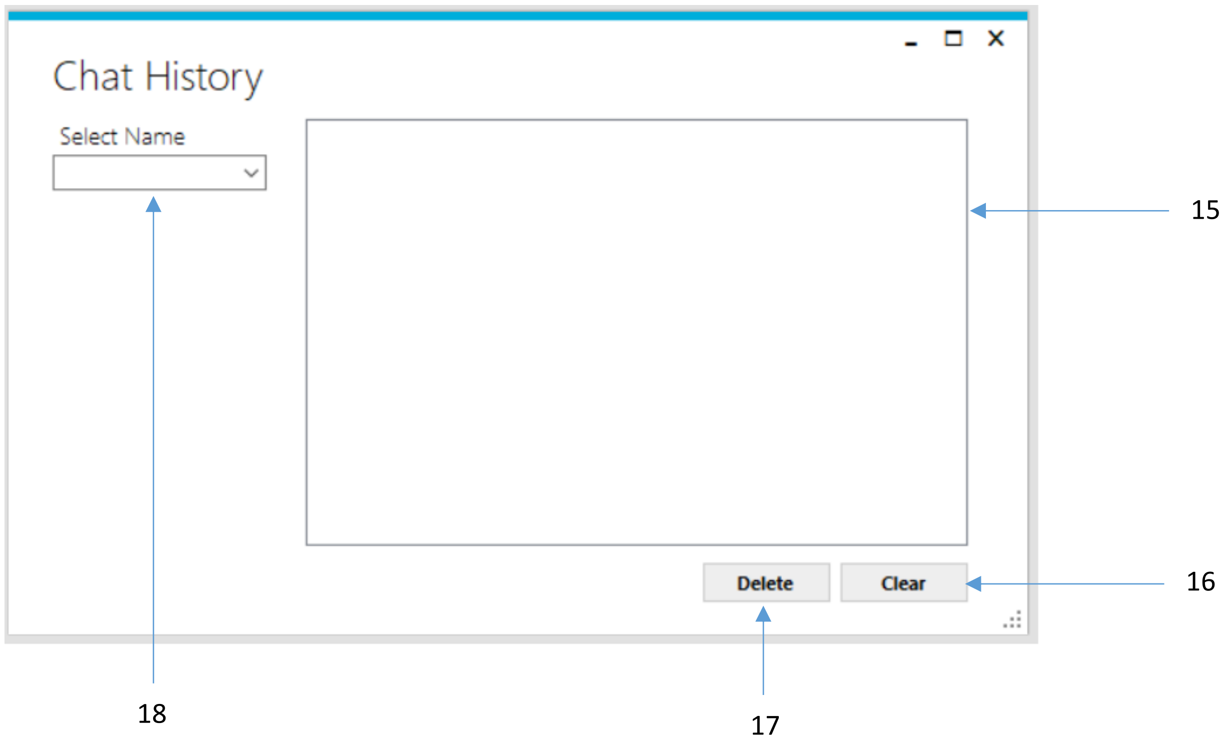
Server messaging form



Once connected the client, above form will be displayed. User can type messages in text box [14] and send them to the client by clicking send button [13]. Sending and receiving messages will be displayed in the list box [12].

Same as the client side message window if the user double click on the particular message, deleting, replying and returning back buttons will be displayed and the server side user can perform the same tasks also in the server side.

If server wants to get all the previous chats with any client, user should click on the chat history button [11].



If user clicks on the chat history button above form will be displayed.

All the previous client's names will be shown in the combo box [18] and when user clicked on particular name, chat history according to that client will be loaded in to the list box[15].

User can clear the list box [15] by clicking on the clear button [16] and delete the entire chat history by clicking on the delete button [17].

2.2 Code explanation

Server Login code-

```
IPAddress[] localIP = Dns.GetHostAddresses(Dns.GetHostName());
foreach (IPAddress address in localIP)
{
    if (address.AddressFamily == AddressFamily.InterNetwork)
    {
        lblIP.Text = "IP address : " + address.ToString();
    }
}
```

Get the IP address and assign it the array [localIP]

It is shown in the label [lblIP].

Form load-

```
private void Form2_Load(object sender, EventArgs e)
{
    using (System.IO.StreamReader file = new System.IO.StreamReader("password.txt", true))
        password = file.ReadLine();
}
```

Server password has written in to a text file named password.txt. when form 2 loads program can read the current password by StreamReader.

Button Initiate the connection-

```
private void btnLog_Click(object sender, EventArgs e)
{
    if (txtPW.Text==password)
    {
        if (Convert.ToInt32(txtPort.Text) > 1024 || Convert.ToInt32(txtPort.Text) > 65535)
        {
            Form1 f1 = new Form1();
            f1.Show();
            this.Hide();
        }
        else
        {
            MessageBox.Show("Invlid Port No","Error");
        }
    }
    else
    {
        MessageBox.Show("Invlid Password", "Error");
    }
}
```

Default password - 1234

Check the user entered password with the current password

Check whether the entered port number is true one or not. If it's true show the Form1 (Messaging form)

Change password button -

```
private void btnChange_Click(object sender, EventArgs e)
{
    if(txtOldPass.Text==password)
    {
        if (txtNewPass.Text != "")
        {
            password = txtNewPass.Text;
            MessageBox.Show("Password updated sucessfully", "Error");
            System.IO.File.WriteAllText("password.txt", password);
            lblPass.Visible = true;
            lblPort.Visible = true;
            txtPW.Visible = true;
            txtPort.Visible = true;
            btnLog.Visible = true;


            lblNewPass.Visible = false;
            lblOldPass.Visible = false;
            txtNewPass.Visible = false;
            txtOldPass.Visible = false;
            btnCancel.Visible = false;
            btnChange.Visible = false;
        }
        else
        {
            MessageBox.Show("New Password can't be Empty", "Error");
        }
    }
    else
    {
        MessageBox.Show("Current Password is Incorrect", "Error");
    }
}
```

When user change the server password, it first check the given password with the current password. If condition true, new password will be written in to the text file password.txt

Server Messaging code –

Form Load -

```
private void Form1_Load(object sender, EventArgs e)
{

    delBtn.Visible = false;
    repBtn.Visible = false;
    backBtn.Visible = false;
    connect();
}
private void connect()
{
    TcpListener listener = new TcpListener(IPAddress.Any, int.Parse(portNo));
    listener.Start();
    client = listener.AcceptTcpClient();
    STR = new StreamReader(client.GetStream());
    STW = new StreamWriter(client.GetStream());
    STW.AutoFlush = true;
    backgroundWorker1.RunWorkerAsync();
    backgroundWorker2.WorkerSupportsCancellation = true;
}
```

Start listning until client connect through the given port number via any IP address.

Accept TCP client

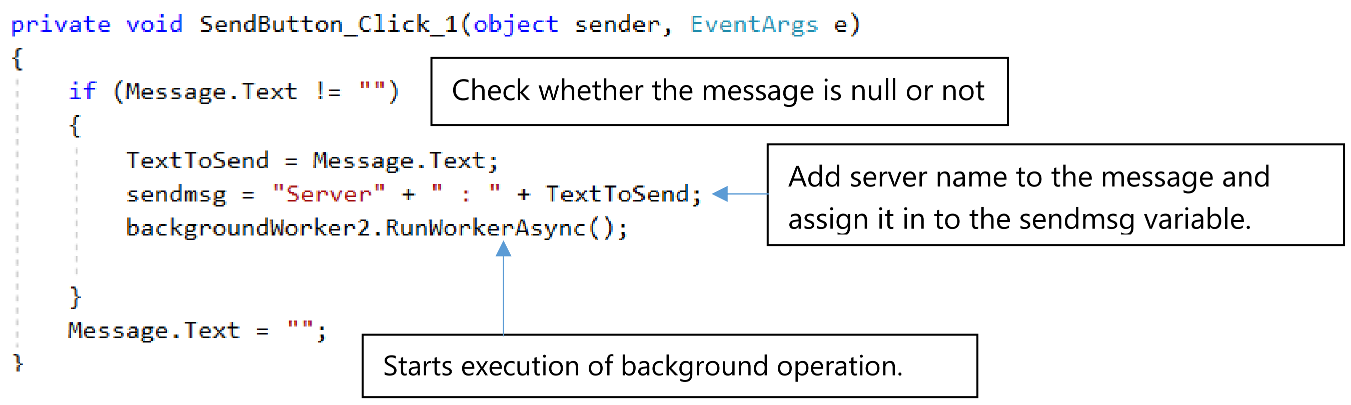
Two variables for catch, read and write client

Starts execution of background operation.

Flush the buffer

Set value to true to supports asynchronous cancellation.

Send button -



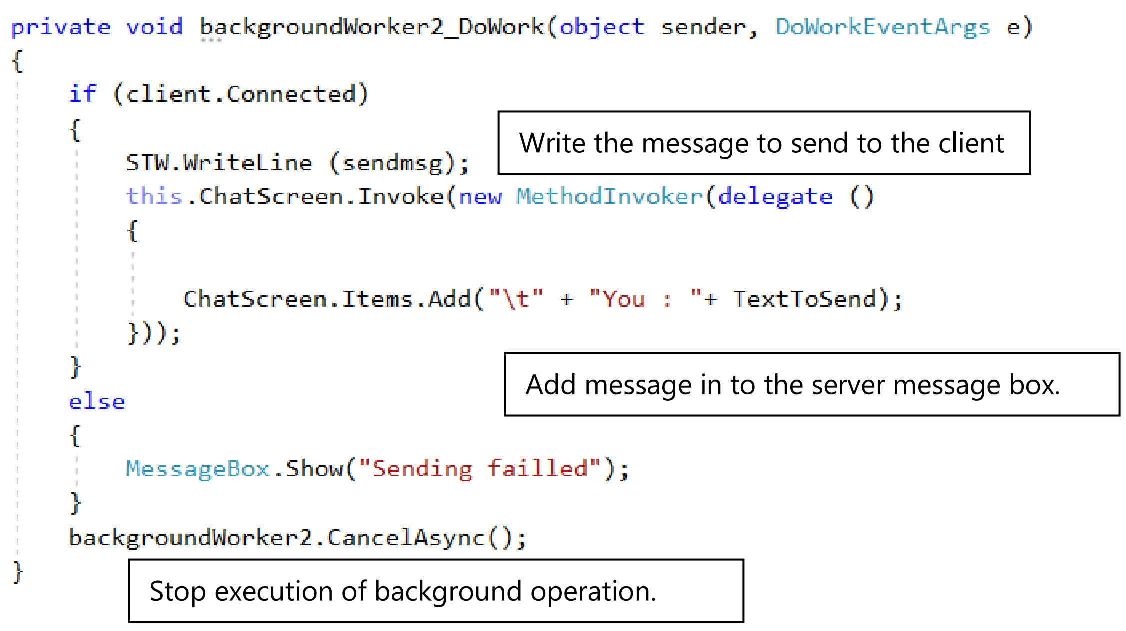
```
private void SendButton_Click_1(object sender, EventArgs e)
{
    if (Message.Text != "")
    {
        TextToSend = Message.Text;
        sendmsg = "Server" + " : " + TextToSend;
        backgroundWorker2.RunWorkerAsync();

    }
    Message.Text = "";
}
```

Check whether the message is null or not

Add server name to the message and assign it in to the sendmsg variable.

Starts execution of background operation.

BackGroundWorker2 –



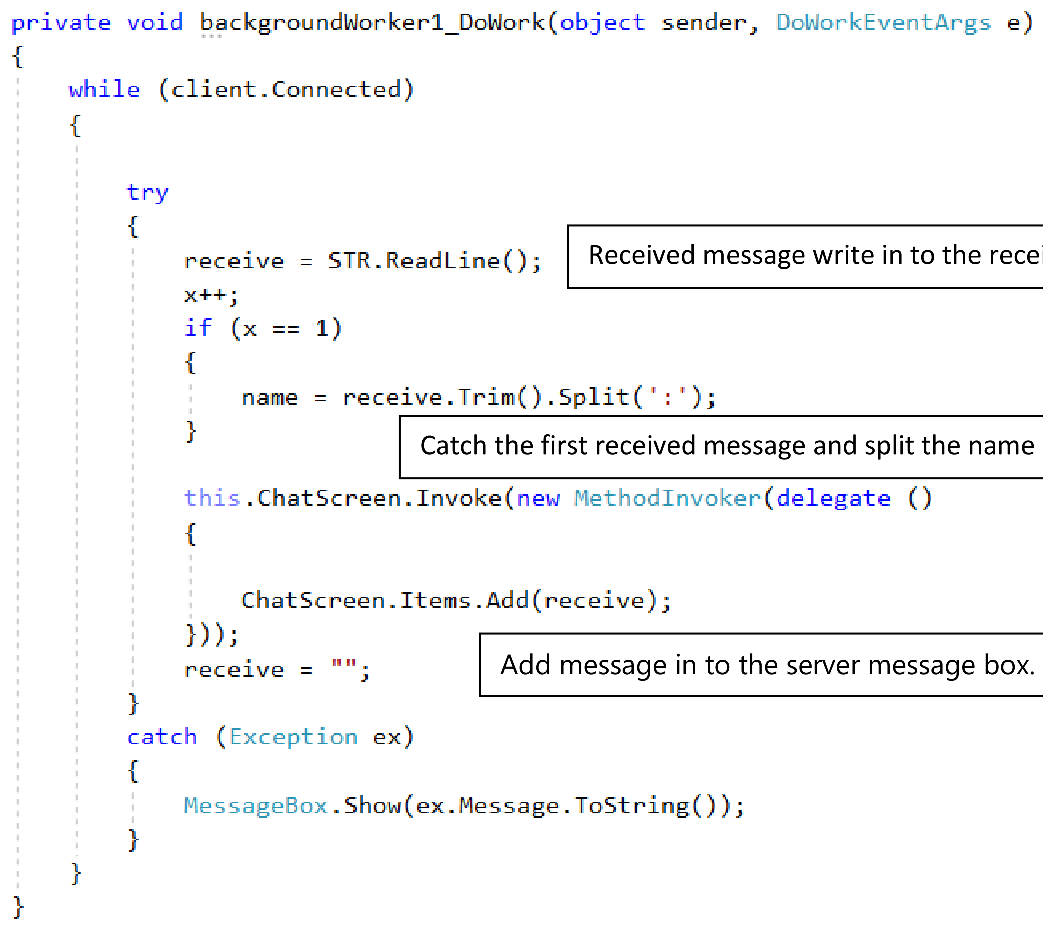
```
private void backgroundWorker2_DoWork(object sender, DoWorkEventArgs e)
{
    if (client.Connected)
    {
        STW.WriteLine (sendmsg);
        this.ChatScreen.Invoke(new MethodInvoker(delegate ()
        {

            ChatScreen.Items.Add("\t" + "You : "+ TextToSend);
        }));
    }
    else
    {
        MessageBox.Show("Sending failled");
    }
    backgroundWorker2.CancelAsync();
}
```

Write the message to send to the client

Add message in to the server message box.

Stop execution of background operation.

BackGroundWorker1 –



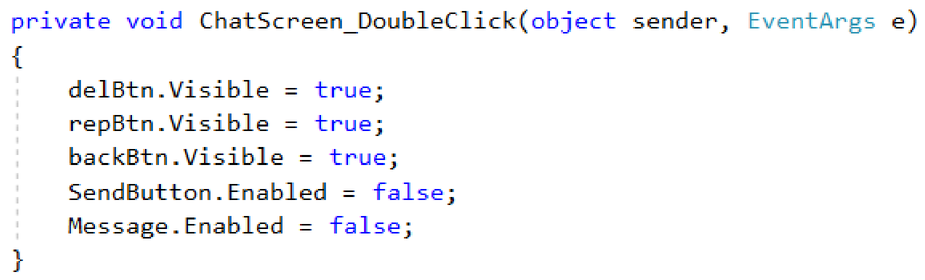
```
private void backgroundWorker1_DoWork(object sender, DoWorkEventArgs e)
{
    while (client.Connected)
    {

        try
        {
            receive = STR.ReadLine();
            x++;
            if (x == 1)
            {
                name = receive.Trim().Split(':');
            }

            this.ChatScreen.Invoke(new MethodInvoker(delegate ()
            {

                ChatScreen.Items.Add(receive);
            }));
            receive = "";
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message.ToString());
        }
    }
}
```

Received message write in to the receive variable

Catch the first received message and split the name of the client.

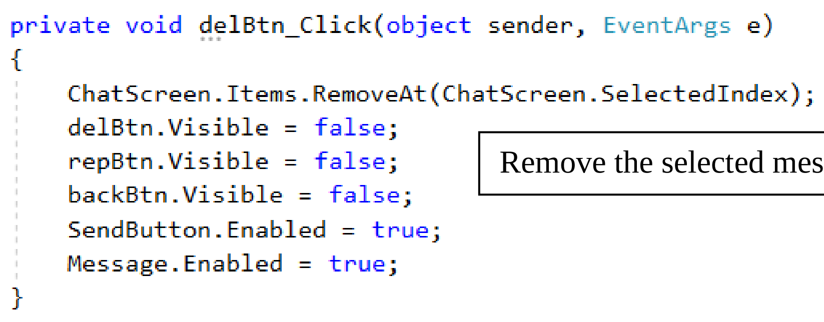
Add message in to the server message box.

Double click the items in list box-



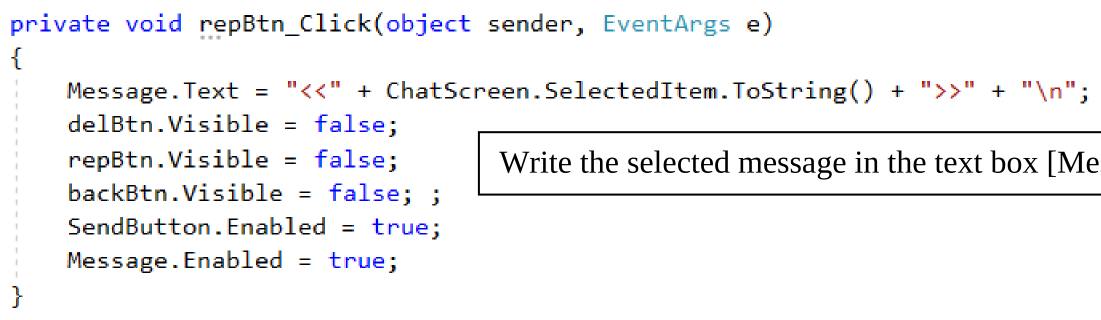
```
private void ChatScreen_DoubleClick(object sender, EventArgs e)
{
    delBtn.Visible = true;
    repBtn.Visible = true;
    backBtn.Visible = true;
    SendButton.Enabled = false;
    Message.Enabled = false;
}
```

Delte button, reply button, back buttton -



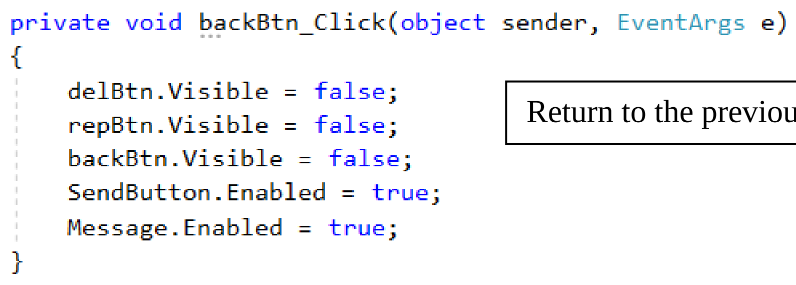
```
private void delBtn_Click(object sender, EventArgs e)
{
    ChatScreen.Items.RemoveAt(ChatScreen.SelectedIndex);
    delBtn.Visible = false;
    repBtn.Visible = false;
    backBtn.Visible = false;
    SendButton.Enabled = true;
    Message.Enabled = true;
}
```

Remove the selected message from the list box



```
private void repBtn_Click(object sender, EventArgs e)
{
    Message.Text = "<<" + ChatScreen.SelectedItem.ToString() + ">>" + "\n";
    delBtn.Visible = false;
    repBtn.Visible = false;
    backBtn.Visible = false; ;
    SendButton.Enabled = true;
    Message.Enabled = true;
}
```

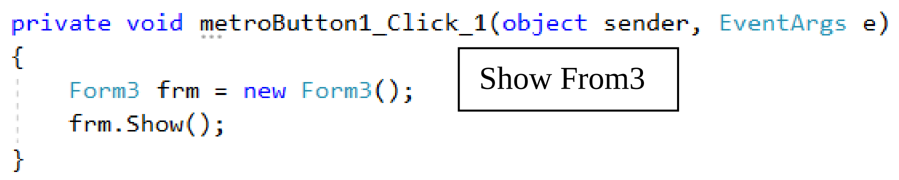
Write the selected message in the text box [Message]



```
private void backBtn_Click(object sender, EventArgs e)
{
    delBtn.Visible = false;
    repBtn.Visible = false;
    backBtn.Visible = false;
    SendButton.Enabled = true;
    Message.Enabled = true;
}
```

Return to the previous state

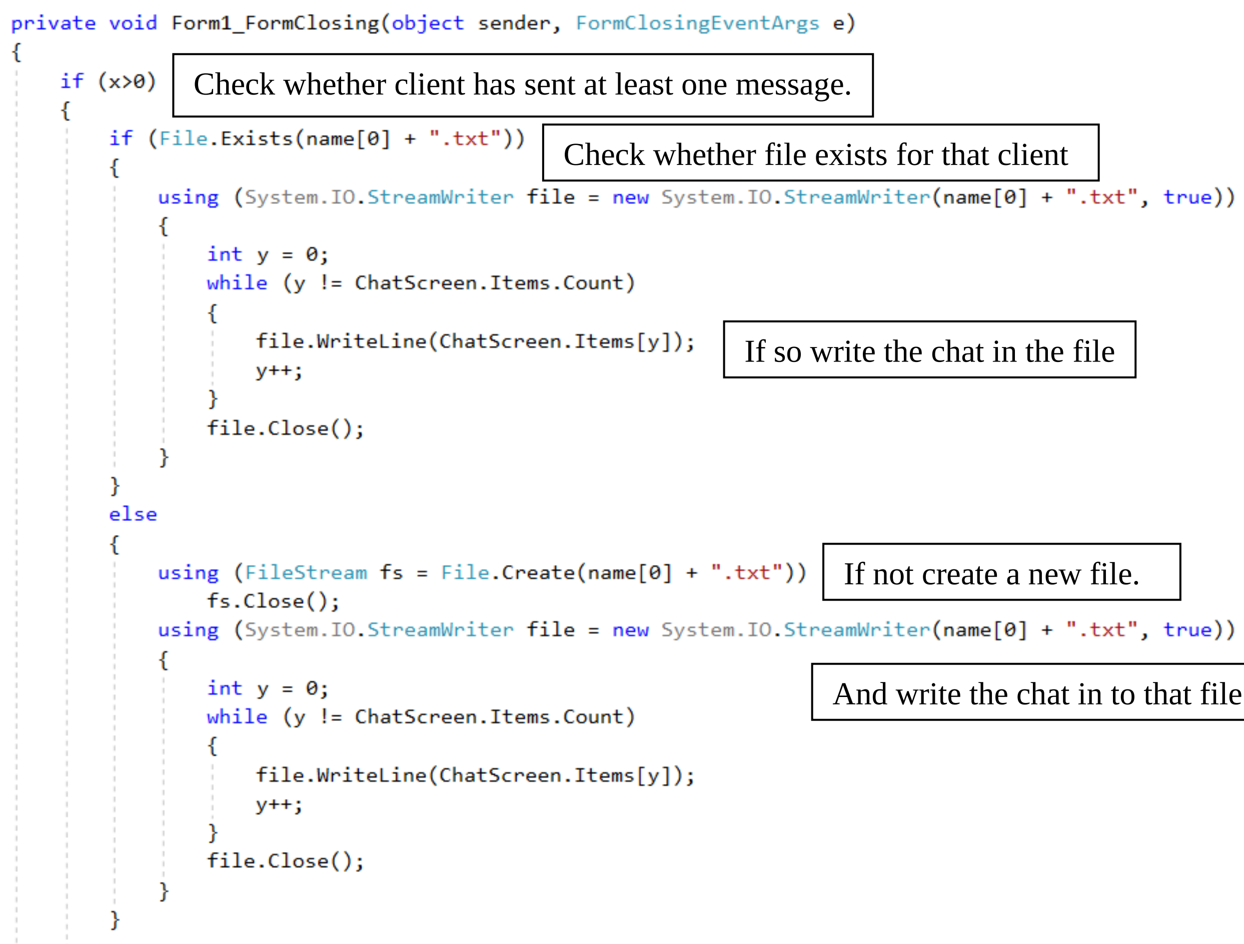
Get previous chat history –



```
private void metroButton1_Click_1(object sender, EventArgs e)
{
    Form3 frm = new Form3();
    frm.Show();
}
```

Show From3

Write chat to the client file –



```
private void Form1_FormClosing(object sender, FormClosingEventArgs e)
{
    if (x>0)
    {
        if (File.Exists(name[0] + ".txt"))
        {
            using (System.IO.StreamWriter file = new System.IO.StreamWriter(name[0] + ".txt", true))
            {
                int y = 0;
                while (y != ChatScreen.Items.Count)
                {
                    file.WriteLine(ChatScreen.Items[y]);
                    y++;
                }
                file.Close();
            }
        }
        else
        {
            using (FileStream fs = File.Create(name[0] + ".txt"))
                fs.Close();
            using (System.IO.StreamWriter file = new System.IO.StreamWriter(name[0] + ".txt", true))
            {
                int y = 0;
                while (y != ChatScreen.Items.Count)
                {
                    file.WriteLine(ChatScreen.Items[y]);
                    y++;
                }
                file.Close();
            }
        }
```

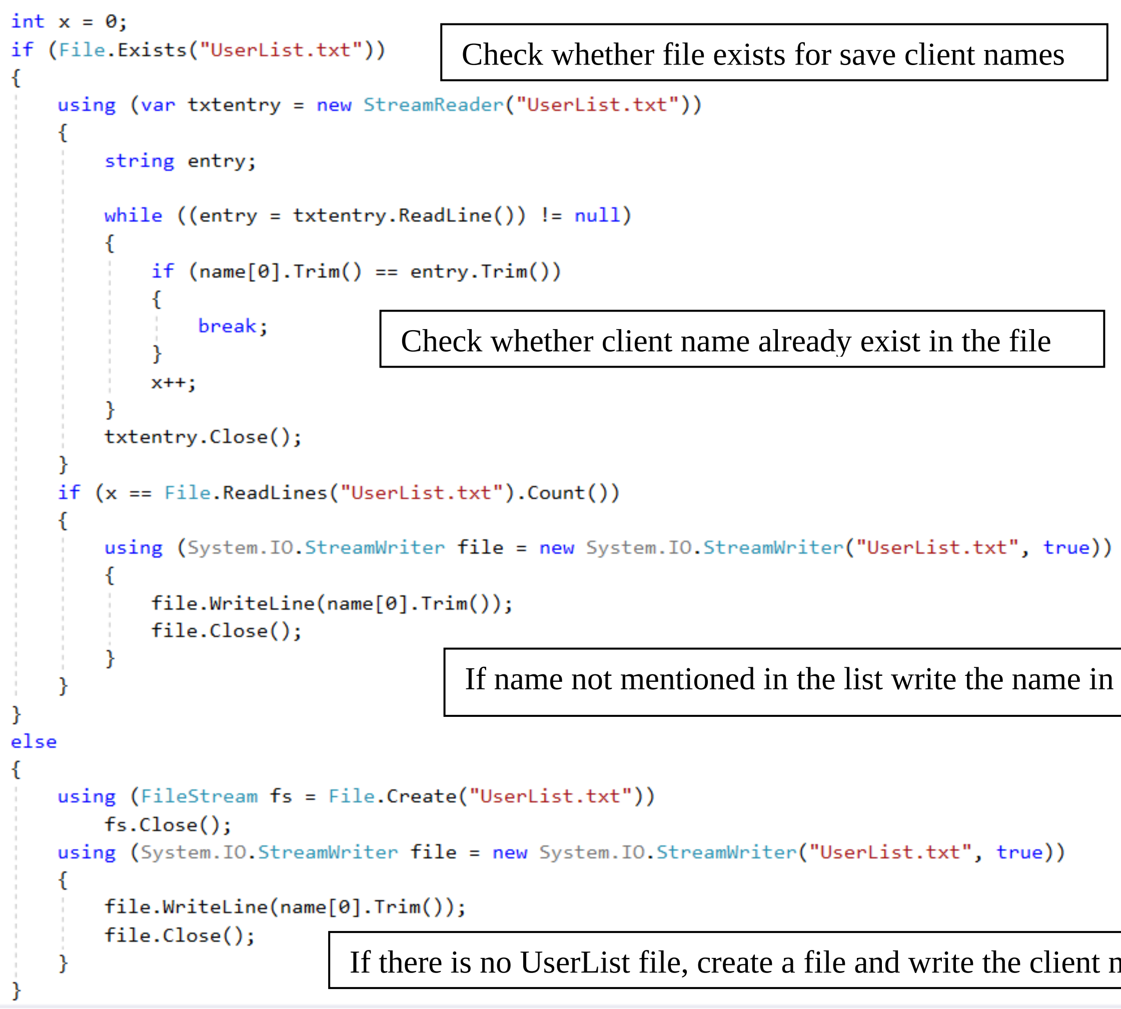
Check whether client has sent at least one message.

Check whether file exists for that client

If so write the chat in the file

If not create a new file.

And write the chat in to that file



```
int x = 0;
if (File.Exists("UserList.txt"))
{
    using (var txtentry = new StreamReader("UserList.txt"))
    {
        string entry;

        while ((entry = txtentry.ReadLine()) != null)
        {
            if (name[0].Trim() == entry.Trim())
            {
                break;
            }
            x++;
        }
        txtentry.Close();
    }
    if (x == File.ReadLines("UserList.txt").Count())
    {
        using (System.IO.StreamWriter file = new System.IO.StreamWriter("UserList.txt", true))
        {
            file.WriteLine(name[0].Trim());
            file.Close();
        }
    }
}
else
{
    using (FileStream fs = File.Create("UserList.txt"))
        fs.Close();
    using (System.IO.StreamWriter file = new System.IO.StreamWriter("UserList.txt", true))
    {
        file.WriteLine(name[0].Trim());
        file.Close();
    }
}
```

Check whether file exists for save client names

Check whether client name already exist in the file

If name not mentioned in the list write the name in to the file.

If there is no UserList file, create a file and write the client name

Server chat history code-

Form Load event-

```
private void Form3_Load(object sender, EventArgs e)
{
    if (File.Exists("UserList.txt"))
    {
        using (var txtentry = new StreamReader("UserList.txt"))
        {
            string entry;

            while ((entry = txtentry.ReadLine()) != null)
            {
                cmbList.Items.Add(entry.Trim());

            }
            txtentry.Close();
        }

    }
    else
    {
        MessageBox.Show("No previous Chat history...");
    }
}
```

Check whether UserList file exists and if so add the client names to the combo box [cmbList]

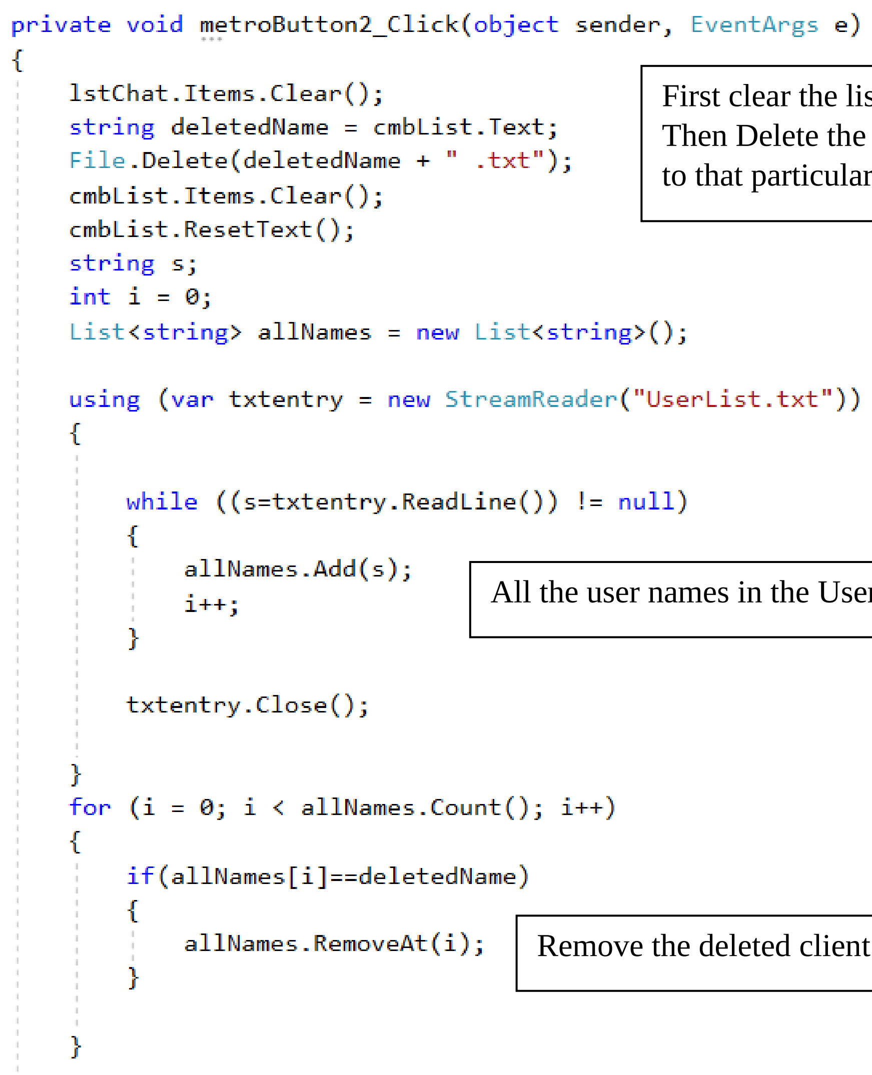
Selecting items in combo box-

Select the file according to the selected user name.

```
private void listBox1_SelectedIndexChanged(object sender, EventArgs e)
{
    lstChat.Items.Clear();
    using (var txtentry2 = new StreamReader(cmbList.SelectedItem.ToString().Trim() + " .txt"))
    {
        string entry;

        while ((entry = txtentry2.ReadLine()) != null)
        {
            lstChat.Items.Add(entry);

        }
        txtentry2.Close();
    }
}
```

When user clicked on the name listed in a combo box, chat history according to that client will be shown in the listbox [lstChat].

Deleting chat history –



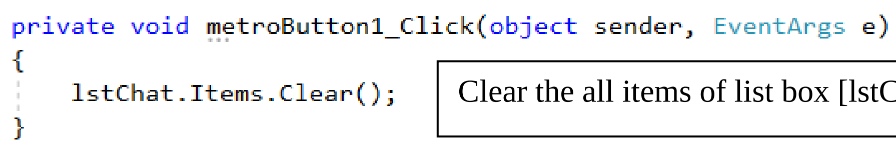
```
private void metroButton2_Click(object sender, EventArgs e)
{
    lstChat.Items.Clear();
    string deletedName = cmbList.Text;
    File.Delete(deletedName + " .txt");
    cmbList.Items.Clear();
    cmbList.ResetText();
    string s;
    int i = 0;
    List<string> allNames = new List<string>();

    using (var txtentry = new StreamReader("UserList.txt"))
    {

        while ((s=txtentry.ReadLine()) != null)
        {
            allNames.Add(s);
            i++;
        }

        txtentry.Close();

    }
    for (i = 0; i < allNames.Count(); i++)
    {
        if(allNames[i]==deletedName)
        {
            allNames.RemoveAt(i);
        }

    }
```

First clear the list box [lstChat]. Then Delete the chat file according to that particular client.

All the user names in the UserList.txt write in to an array.

Remove the deleted client name from the array

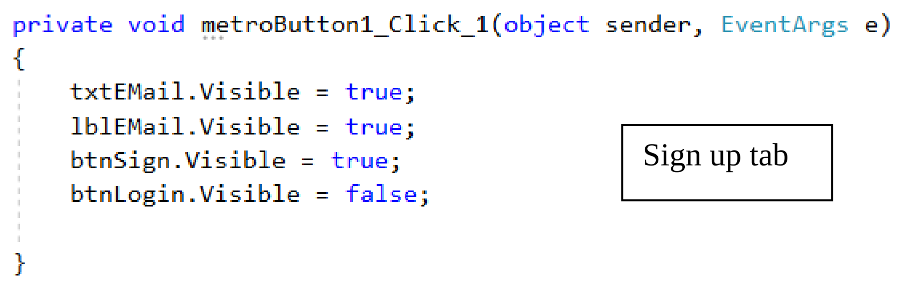
Crear the list box –



```
private void metroButton1_Click(object sender, EventArgs e)
{
    lstChat.Items.Clear();
}
```

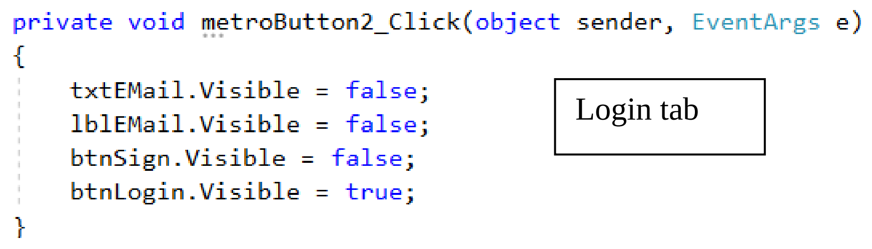
Clear the all items of list box [lstChat]

Client code



```
private void metroButton1_Click_1(object sender, EventArgs e)
{
    txtEMail.Visible = true;
    lblEMail.Visible = true;
    btnSign.Visible = true;
    btnLogin.Visible = false;

}
```

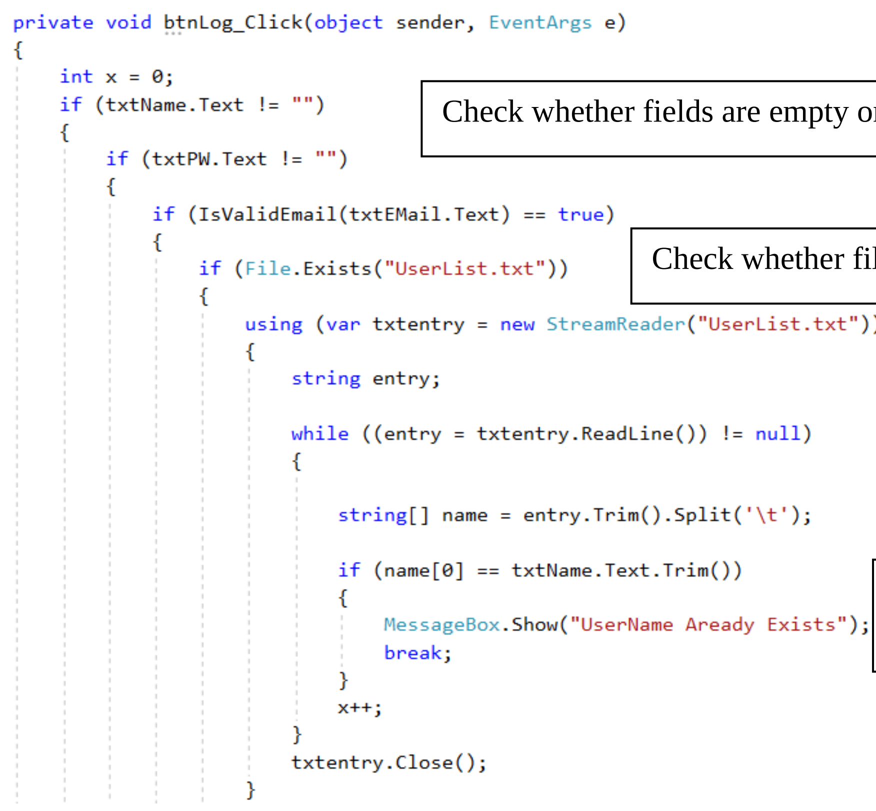
Sign up tab



```
private void metroButton2_Click(object sender, EventArgs e)
{
    txtEMail.Visible = false;
    lblEMail.Visible = false;
    btnSign.Visible = false;
    btnLogin.Visible = true;
}
```

Login tab

Sign in button –

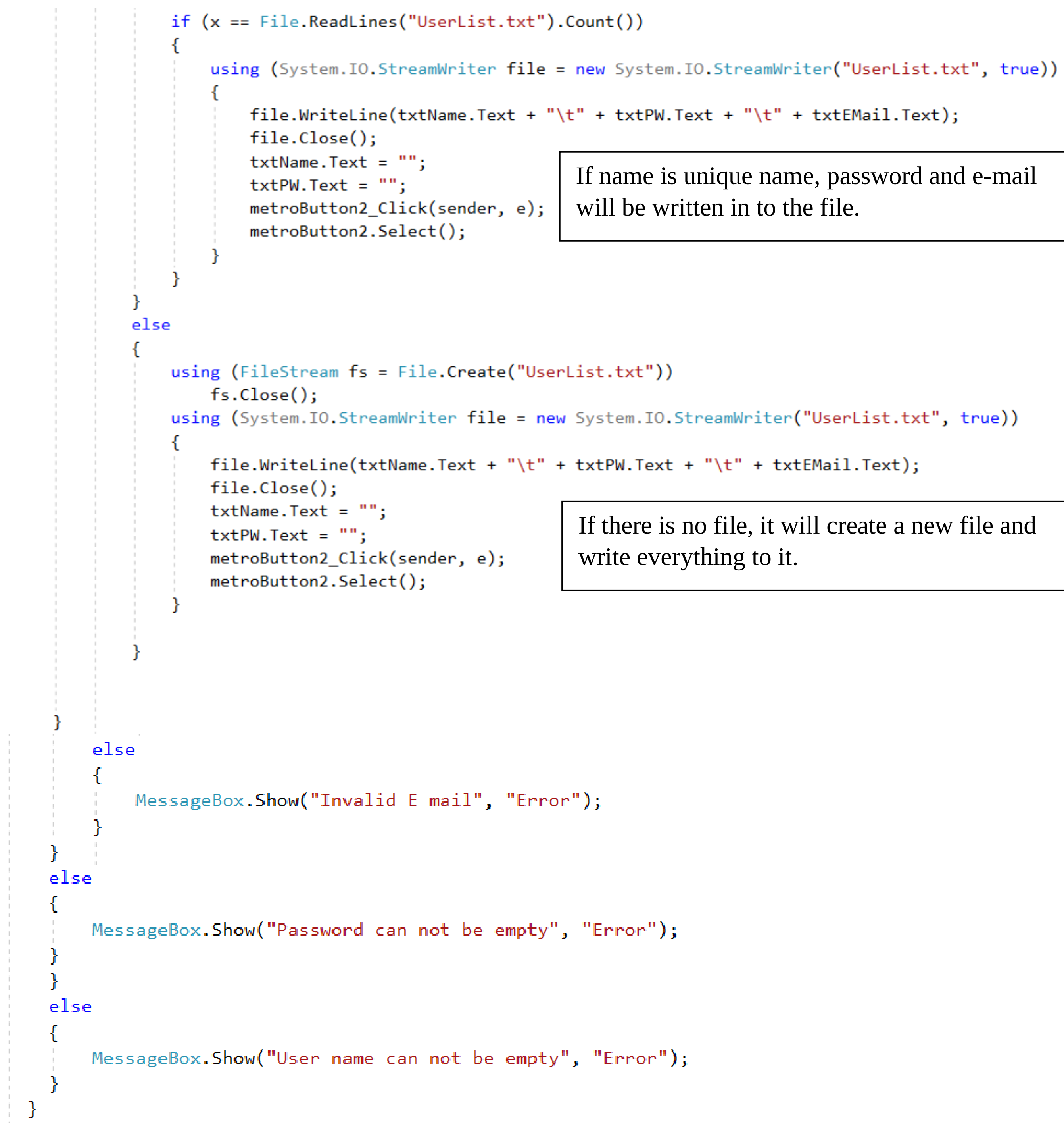


```
private void btnLog_Click(object sender, EventArgs e)
{
    int x = 0;
    if (txtName.Text != "")
    {
        if (txtPW.Text != "")
        {
            if (IsValidEmail(txtEMail.Text) == true)
            {
                if (File.Exists("UserList.txt"))
                {
                    using (var txtentry = new StreamReader("UserList.txt"))
                    {
                        string entry;

                        while ((entry = txtentry.ReadLine()) != null)
                        {

                            string[] name = entry.Trim().Split('\t');

                            if (name[0] == txtName.Text.Trim())
                            {
                                MessageBox.Show("UserName Aready Exists");
                                break;
                            }
                            x++;
                        }
                        txtentry.Close();
                    }
```

Check whether fields are empty or not and email in correct format.

Check whether file is exists or not

Compare given user name with existing user names.



```
                if (x == File.ReadLines("UserList.txt").Count())
                {
                    using (System.IO.StreamWriter file = new System.IO.StreamWriter("UserList.txt", true))
                    {
                        file.WriteLine(txtName.Text + "\t" + txtPW.Text + "\t" + txtEMail.Text);
                        file.Close();
                        txtName.Text = "";
                        txtPW.Text = "";
                        metroButton2_Click(sender, e);
                        metroButton2.Select();
                    }
                }
            }
            else
            {
                using (FileStream fs = File.Create("UserList.txt"))
                    fs.Close();
                using (System.IO.StreamWriter file = new System.IO.StreamWriter("UserList.txt", true))
                {
                    file.WriteLine(txtName.Text + "\t" + txtPW.Text + "\t" + txtEMail.Text);
                    file.Close();
                    txtName.Text = "";
                    txtPW.Text = "";
                    metroButton2_Click(sender, e);
                    metroButton2.Select();
                }

            }


    }
        else
        {
            MessageBox.Show("Invalid E mail", "Error");
        }
    }
    else
    {
        MessageBox.Show("Password can not be empty", "Error");
    }
    }
    else
    {
        MessageBox.Show("User name can not be empty", "Error");
    }
}
```

If name is unique name, password and e-mail will be written in to the file.

If there is no file, it will create a new file and write everything to it.

Log in button –

```
private void btnLogin_Click(object sender, EventArgs e)
{

    using (var txtentry = new StreamReader("UserList.txt"))
    {
        string entry;int x = 0;

        while ((entry = txtentry.ReadLine()) != null)
        {

            string[] name = entry.Trim().Split('\t');

                if (name[0] == txtName.Text.Trim())
                {
                    if (name[1] == txtPW.Text.Trim())
                    {
```

Check whether user name and password is true or not

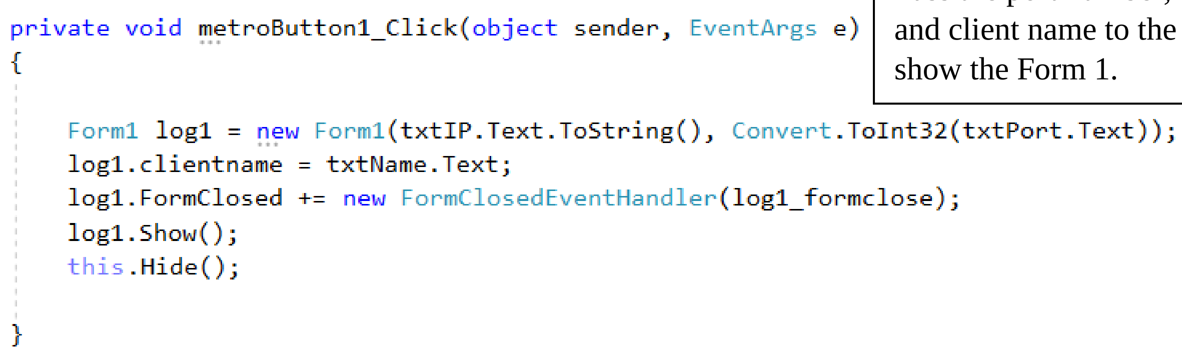
```
 try
 {
     string email = name[2].Trim();
     MailMessage mail = new MailMessage();
     SmtpClient SmtpServer = new SmtpClient("smtp.gmail.com");

     mail.From = new MailAddress("chatappvp@gmail.com");
     mail.To.Add(email);
     mail.Subject = "Security Alert!!!";
     mail.Body = "Someone has logged in to your account.. check whether is it you or not...";

     SmtpServer.Port = 587;
     SmtpServer.Credentials = new System.Net.NetworkCredential("chatappvp@gmail.com", "Chatapp123");
     SmtpServer.EnableSsl = true;

     SmtpServer.Send(mail);
 }
 catch (Exception ex)
 {
     MessageBox.Show(ex.ToString());
 }

 break;
```

If it is true send an email to the client that mentioning someone has logged in to his or her account.

```
                    }
                }
            x++;
        }

        if (x == File.ReadLines("UserList.txt").Count())
        {
            MessageBox.Show("Invalid User Name");
            txtPW.Text = "";
            txtName.Text = "";
        }

        txtentry.Close();
    }
}
```

Message box will be shown if it's invalid user name or password.

Connect button –

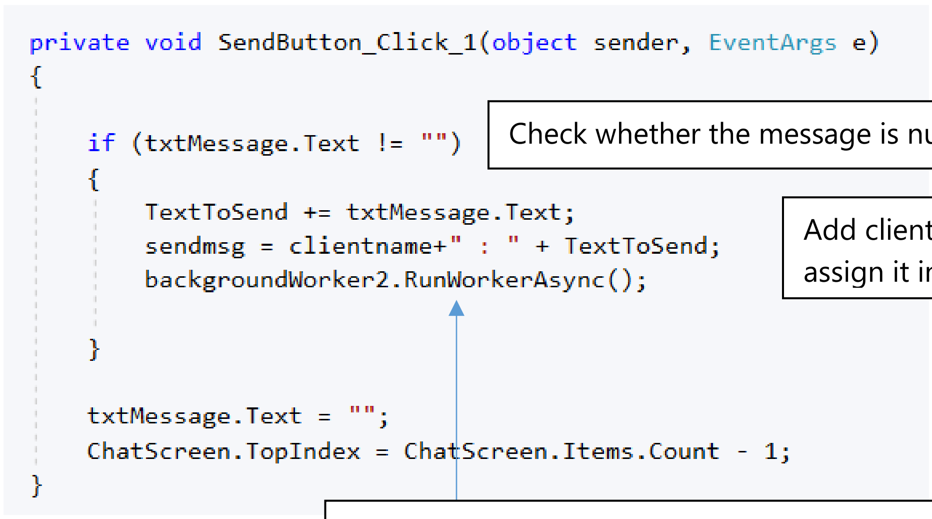


```
private void metroButton1_Click(object sender, EventArgs e)
{

    Form1 log1 = new Form1(txtIP.Text.ToString(), Convert.ToInt32(txtPort.Text));
    log1.clientname = txtName.Text;
    log1.FormClosed += new FormClosedEventHandler(log1_formclose);
    log1.Show();
    this.Hide();

}
```

Pass the port number, IP address and client name to the Form1 and show the Form 1.

Client side messaging form –

Send button -



```
private void SendButton_Click_1(object sender, EventArgs e)
{

    if (txtMessage.Text != "")
    {
        TextToSend += txtMessage.Text;
        sendmsg = clientname+" : " + TextToSend;
        backgroundWorker2.RunWorkerAsync();

    }

    txtMessage.Text = "";
    ChatScreen.TopIndex = ChatScreen.Items.Count - 1;
}
```

Check whether the message is null or not

Add client name to the message and assign it in to the sendmsg variable.

Starts execution of background operation.

Form load –

```
private void Form1_Load(object sender, EventArgs e)
{

    delBtn.Visible = false;
    repBtn.Visible = false;
    backBtn.Visible = false;
    connect();

    if (File.Exists(clientname + ".txt"))
    {
        using (var txtentry = new StreamReader(clientname + ".txt"))

        {
            string entry;
            while ((entry = txtentry.ReadLine()) != null)
            {
                ChatScreen.Items.Add(entry);
            }
            txtentry.Close();
        }
    }
    else
    {
        using (FileStream fs = File.Create(clientname + ".txt"))
            fs.Close();
    }
}
```

Call connect function

Check whether client has previous chat history

Write chat history in to client message box

If there is no such a file create a new one.

BackGroundWorker2 –

```
private void backgroundWorker2_DoWork(object sender, DoWorkEventArgs e)
{
    if (client.Connected)
    {
        STW.WriteLine (sendmsg);
        this.ChatScreen.Invoke(new MethodInvoker(delegate ()
        {
            ChatScreen.Items.Add("\t" + "You :  " + TextToSend);
        }));

        writefile();
    }
    else
    {
        MessageBox.Show("Sending failled");
    }
    backgroundWorker2.CancelAsync();
}
```

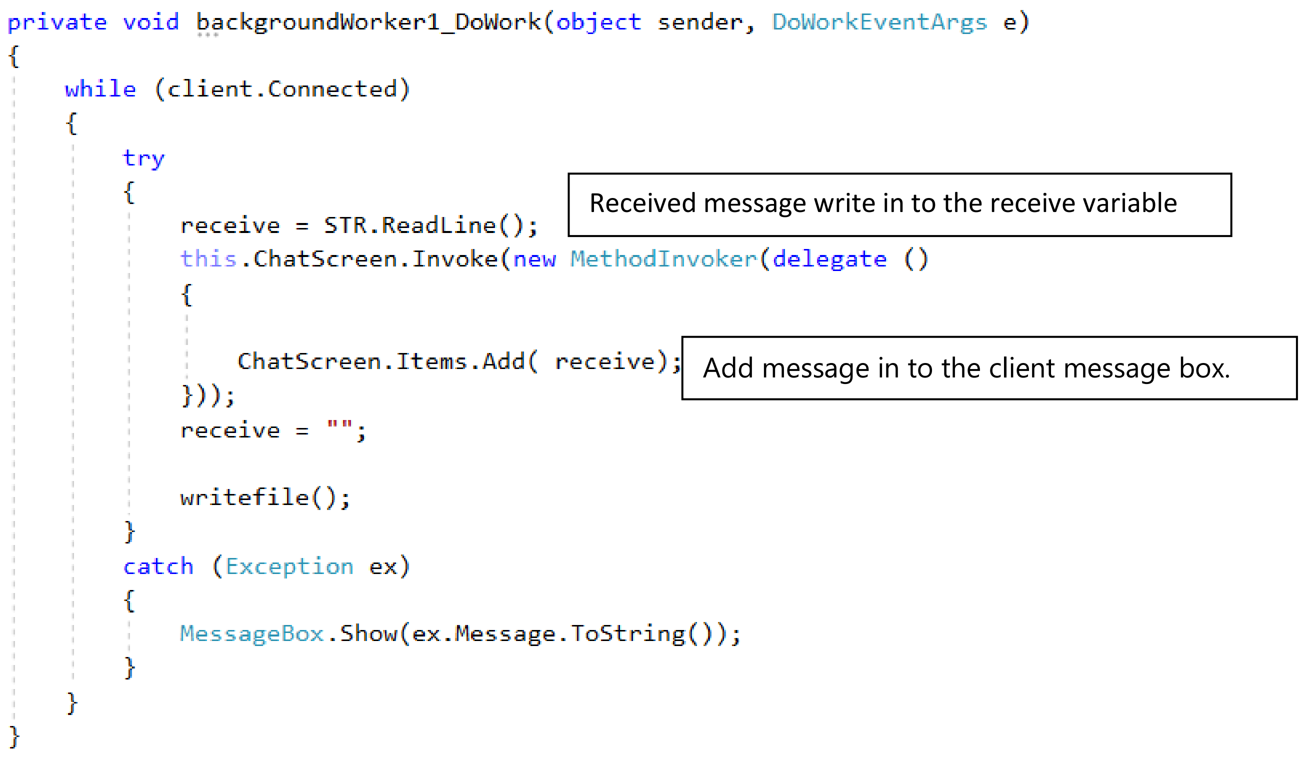
Write the message to send to the server

Add message in to the client message box.

Call writefile() function

Stop execution of background operation.

BackGroundWorker1 –

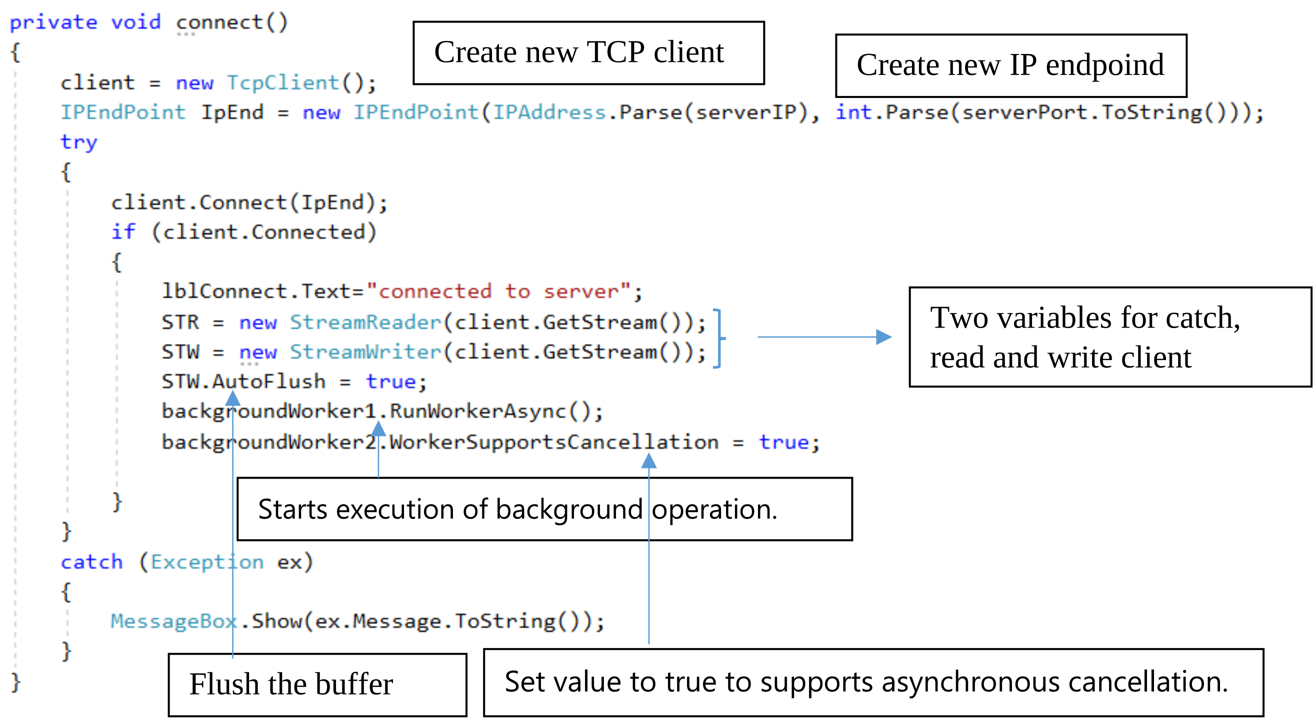


```
private void backgroundWorker1_DoWork(object sender, DoWorkEventArgs e)
{
    while (client.Connected)
    {
        try
        {
            receive = STR.ReadLine();
            this.ChatScreen.Invoke(new MethodInvoker(delegate ()
            {

                ChatScreen.Items.Add( receive);
            }));
            receive = "";

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

Received message write in to the receive variable

Add message in to the client message box.

Connect() –



```
private void connect()
{
    client = new TcpClient();
    IPEndPoint IpEnd = new IPEndPoint(IPAddress.Parse(serverIP), int.Parse(serverPort.ToString()));
    try
    {
        client.Connect(IpEnd);
        if (client.Connected)
        {
            lblConnect.Text="connected to server";
            STR = new StreamReader(client.GetStream());
            STW = new StreamWriter(client.GetStream());
            STW.AutoFlush = true;
            backgroundWorker1.RunWorkerAsync();
            backgroundWorker2.WorkerSupportsCancellation = true;

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

Create new TCP client

Create new IP endpoind

Two variables for catch, read and write client

Starts execution of background operation.

Flush the buffer

Set value to true to supports asynchronous cancellation.

List box item click –

```
private void ChatScreen_DoubleClick(object sender, EventArgs e)
{
    delBtn.Visible = true;
    repBtn.Visible = true;
    backBtn.Visible = true;
    SendButton.Enabled = false;
    txtMessage.Enabled = false;

}
```

Delte button, reply button, back buttton –

```
private void btnReply_Click(object sender, EventArgs e)
{
    txtMessage.Text = "<<"+ ChatScreen.SelectedItem.ToString().Trim() + ">>"+"\n";
    delBtn.Visible = false;
    repBtn.Visible = false;
    backBtn.Visible = false;
    SendButton.Enabled = true;
    txtMessage.Enabled = true;
}
```

Return to the previous state

```
private void delBtn_Click(object sender, EventArgs e)
{
    DialogResult dialogResult = MessageBox.Show(
        "Are you sure you want to delete?", "Delete Message", MessageBoxButtons.YesNo);
    if (dialogResult == DialogResult.Yes)
    {
        ChatScreen.Items.RemoveAt(ChatScreen.SelectedIndex);
        writefile();
        delBtn.Visible = false;
        repBtn.Visible = false;
        backBtn.Visible = false;
        SendButton.Enabled = true;
        txtMessage.Enabled = true;
    }

}
```

Remove the selected message from the list box and call write() function

```
private void btnReturn_Click(object sender, EventArgs e)
{
    delBtn.Visible = false;
    repBtn.Visible = false;
    backBtn.Visible = false;
    SendButton.Enabled = true;
    txtMessage.Enabled = true;
}
```

Write the selected message in the text box [txtMessage]

3.Final code

3.1 Server code

Login form –

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Net;
using System.Net.Sockets;
using System.IO;

namespace gp1
{
    public partial class Form1 : MetroFramework.Forms.MetroForm
    {
        private TcpClient client;
        public StreamReader STR;
        public StreamWriter STW;
        public string receive;
        public string TextToSend;
        string sendmsg;
        string portNo;
        string[] name;
        int x = 0;
        public Form1(string port)
        {
            portNo = port;
            InitializeComponent();
        }
        private void backgroundWorker1_DoWork(object sender, DoWorkEventArgs e)
        {

            while (client.Connected)
            {
                try
                {
                    //read the recieve message from client
                    receive = STR.ReadLine();
                    x++;
                    if (x == 1)
                    {
                        name = receive.Trim().Split(':');
                    }

                    this.ChatScreen.Invoke(new MethodInvoker(delegate ()
                    {

                        ChatScreen.Items.Add(receive);
                    }));
                    receive = "";
                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.Message.ToString());
                }
            }
        }

        private void backgroundWorker2_DoWork(object sender, DoWorkEventArgs e)
        {
            if (client.Connected)
            {
                //write the send message from server in the server chat screen
                STW.WriteLine (sendmsg);
                this.ChatScreen.Invoke(new MethodInvoker(delegate ()
                {

                    ChatScreen.Items.Add("\t" + "You : "+ TextToSend);
                }));
            }
            else
            {
                MessageBox.Show("Sending failled");
            }
            backgroundWorker2.CancelAsync();
        }


        private void SendButton_Click_1(object sender, EventArgs e)
        {
            if (Message.Text != "")
            {
                //send the message from server to client
                TextToSend = Message.Text;
                sendmsg = "Server" + " : " + TextToSend;
                backgroundWorker2.RunWorkerAsync();

            }
            Message.Text = "";
        }

        private void Form1_Load(object sender, EventArgs e)
        {

            delBtn.Visible = false;
            repBtn.Visible = false;
            backBtn.Visible = false;
            connect();
        }
        private void connect()
        {
            TcpListener listener = new TcpListener(IPAddress.Any, int.Parse(portNo));
            listener.Start();
            client = listener.AcceptTcpClient();
            STR = new StreamReader(client.GetStream());
            STW = new StreamWriter(client.GetStream());
            STW.AutoFlush = true;
            backgroundWorker1.RunWorkerAsync();
            backgroundWorker2.WorkerSupportsCancellation = true;
        }

        private void metroButton1_Click(object sender, EventArgs e)
        {
            connect();
        }

        private void btnReply_Click(object sender, EventArgs e)
        {
            Message.Text = "<<" + ChatScreen.SelectedItem.ToString() + ">>" + "\n";
            delBtn.Visible = false;
```

```
            repBtn.Visible = false;
            backBtn.Visible = false;
            SendButton.Enabled = true;
            Message.Enabled = true;
        }

        private void btnDelete_Click(object sender, EventArgs e)
        {
            ChatScreen.Items.RemoveAt(ChatScreen.SelectedIndex);
            delBtn.Visible = false;
            repBtn.Visible = false;
            backBtn.Visible = false;
            SendButton.Enabled = true;
            Message.Enabled = true;
        }

        private void btnReturn_Click(object sender, EventArgs e)
        {
            delBtn.Visible = false;
            repBtn.Visible = false;
            backBtn.Visible = false;
            SendButton.Enabled = true;
            Message.Enabled = true;
        }

        private void ChatScreen_DoubleClick(object sender, EventArgs e)
        {
            delBtn.Visible = true;
            repBtn.Visible = true;
            backBtn.Visible = true;
            SendButton.Enabled = false;
            Message.Enabled = false;
        }

        private void btnPrvChat_Click(object sender, EventArgs e)
        {
            Form3 frm = new Form3();
            frm.Show();
        }

        private void Form1_FormClosing(object sender, FormClosingEventArgs e)
        {
            if(x>0)
            {
                if (File.Exists(name[0] + ".txt"))
                {
                    //save the chat history in the text file
                    using (System.IO.StreamWriter file = new System.IO.StreamWriter(name[0] +
".txt", true))
                    {
                        int y = 0;
                        while (y != ChatScreen.Items.Count)
                        {
                            file.WriteLine(ChatScreen.Items[y]);
                            y++;
                        }
                        file.Close();
                    }
                }
                else
                {
                    using (FileStream fs = File.Create(name[0] + ".txt"))
                        fs.Close();
                    using (System.IO.StreamWriter file = new System.IO.StreamWriter(name[0] +
".txt", true))
                    {
                        int y = 0;
                        while (y != ChatScreen.Items.Count)
                        {
                            file.WriteLine(ChatScreen.Items[y]);
                            y++;
                        }
                        file.Close();
                    }
                }

                int x = 0;
                if (File.Exists("UserList.txt"))
                {
                    using (var txtentry = new StreamReader("UserList.txt"))
                    {
                        string entry;

                        while ((entry = txtentry.ReadLine()) != null)
                        {
                            if (name[0].Trim() == entry.Trim())
                            {
                                break;
                            }
                            x++;
                        }
                        txtentry.Close();
                    }
                    if (x == File.ReadLines("UserList.txt").Count())
                    {
                        using (System.IO.StreamWriter file = new
System.IO.StreamWriter("UserList.txt", true))
                        {
                            file.WriteLine(name[0].Trim());
                            file.Close();
                        }
                    }
                }
                else
                {
                    using (FileStream fs = File.Create("UserList.txt"))
                        fs.Close();
                    using (System.IO.StreamWriter file = new
System.IO.StreamWriter("UserList.txt", true))
                    {
                        file.WriteLine(name[0].Trim());
                        file.Close();
                    }
                }
            }
        }

        private void delBtn_Click(object sender, EventArgs e)
        {
            ChatScreen.Items.RemoveAt(ChatScreen.SelectedIndex);
            delBtn.Visible = false;
            repBtn.Visible = false;
            backBtn.Visible = false;
            SendButton.Enabled = true;
            Message.Enabled = true;
        }

        private void repBtn_Click(object sender, EventArgs e)
        {
            Message.Text = "<<" + ChatScreen.SelectedItem.ToString() + ">>" + "\n";
            delBtn.Visible = false;
            repBtn.Visible = false;
            backBtn.Visible = false; ;
            SendButton.Enabled = true;
            Message.Enabled = true;
        }

        private void backBtn_Click(object sender, EventArgs e)
        {
            delBtn.Visible = false;
            repBtn.Visible = false;
            backBtn.Visible = false;
            SendButton.Enabled = true;
            Message.Enabled = true;
        }

        private void metroButton1_Click_1(object sender, EventArgs e)
        {
            Form3 frm = new Form3();
            frm.Show();
        }

    }
}
```

Messaging form –

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Net;
using System.Net.Sockets;
using System.IO;

namespace gp1
{
    public partial class Form1 : MetroFramework.Forms.MetroForm
    {
        private TcpClient client;
        public StreamReader STR;
        public StreamWriter STW;
        public string receive;
        public string TextToSend;
        string sendmsg;
        string portNo;
        string[] name;
        int x = 0;
        public Form1(string port)
        {
            portNo = port;
            InitializeComponent();
        }
        private void backgroundWorker1_DoWork(object sender, DoWorkEventArgs e)
        {

            while (client.Connected)
            {
                try
                {
                    //read the recieve message from client
                    receive = STR.ReadLine();
                    x++;
                    if (x == 1)
                    {
                        name = receive.Trim().Split(':');
                    }

                    this.ChatScreen.Invoke(new MethodInvoker(delegate ()
                    {

                        ChatScreen.Items.Add(receive);
                    }));
                    receive = "";
                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.Message.ToString());
                }
            }
        }

        private void backgroundWorker2_DoWork(object sender, DoWorkEventArgs e)
        {
            if (client.Connected)
            {
                //write the send message from server in the server chat screen
                STW.WriteLine (sendmsg);
                this.ChatScreen.Invoke(new MethodInvoker(delegate ()
                {

                    ChatScreen.Items.Add("\t" + "You : "+ TextToSend);
                }));
            }
            else
            {
                MessageBox.Show("Sending failled");
            }
            backgroundWorker2.CancelAsync();
        }


        private void SendButton_Click_1(object sender, EventArgs e)
        {
            if (Message.Text != "")
            {
                //send the message from server to client
                TextToSend = Message.Text;
                sendmsg = "Server" + " : " + TextToSend;
                backgroundWorker2.RunWorkerAsync();

            }
            Message.Text = "";
        }

        private void Form1_Load(object sender, EventArgs e)
        {

            delBtn.Visible = false;
            repBtn.Visible = false;
            backBtn.Visible = false;
            connect();
        }
        private void connect()
        {
            TcpListener listener = new TcpListener(IPAddress.Any, int.Parse(portNo));
            listener.Start();
            client = listener.AcceptTcpClient();
            STR = new StreamReader(client.GetStream());
            STW = new StreamWriter(client.GetStream());
```

```
            STW.AutoFlush = true;
            backgroundWorker1.RunWorkerAsync();
            backgroundWorker2.WorkerSupportsCancellation = true;
        }

        private void metroButton1_Click(object sender, EventArgs e)
        {
            connect();
        }

        private void btnReply_Click(object sender, EventArgs e)
        {
            Message.Text = "<<" + ChatScreen.SelectedItem.ToString() + ">>" + "\n";
            delBtn.Visible = false;
            repBtn.Visible = false;
            backBtn.Visible = false;
            SendButton.Enabled = true;
            Message.Enabled = true;
        }

        private void btnDelete_Click(object sender, EventArgs e)
        {
            ChatScreen.Items.RemoveAt(ChatScreen.SelectedIndex);
            delBtn.Visible = false;
            repBtn.Visible = false;
            backBtn.Visible = false;
            SendButton.Enabled = true;
            Message.Enabled = true;
        }

        private void btnReturn_Click(object sender, EventArgs e)
        {
            delBtn.Visible = false;
            repBtn.Visible = false;
            backBtn.Visible = false;
            SendButton.Enabled = true;
            Message.Enabled = true;
        }

        private void ChatScreen_DoubleClick(object sender, EventArgs e)
        {
            delBtn.Visible = true;
            repBtn.Visible = true;
            backBtn.Visible = true;
            SendButton.Enabled = false;
            Message.Enabled = false;
        }

        private void btnPrvChat_Click(object sender, EventArgs e)
        {
            Form3 frm = new Form3();
            frm.Show();
        }

        private void Form1_FormClosing(object sender, FormClosingEventArgs e)
        {
            if(x>0)
            {
                if (File.Exists(name[0] + ".txt"))
                {
                    //save the chat history in the text file
                    using (System.IO.StreamWriter file = new System.IO.StreamWriter(name[0] +
".txt", true))
                    {
                        int y = 0;
                        while (y != ChatScreen.Items.Count)
                        {
                            file.WriteLine(ChatScreen.Items[y]);
                            y++;
                        }
                        file.Close();
                    }
                }
                else
                {
                    using (FileStream fs = File.Create(name[0] + ".txt"))
                        fs.Close();
                    using (System.IO.StreamWriter file = new System.IO.StreamWriter(name[0] +
".txt", true))
                    {
                        int y = 0;
                        while (y != ChatScreen.Items.Count)
                        {
                            file.WriteLine(ChatScreen.Items[y]);
                            y++;
                        }
                        file.Close();
                    }
                }

                int x = 0;
                if (File.Exists("UserList.txt"))
                {
                    using (var txtentry = new StreamReader("UserList.txt"))
                    {
                        string entry;

                        while ((entry = txtentry.ReadLine()) != null)
                        {
                            if (name[0].Trim() == entry.Trim())
                            {
                                break;
                            }
                            x++;
                        }
                        txtentry.Close();
                    }
                    if (x == File.ReadLines("UserList.txt").Count())
                    {
                        using (System.IO.StreamWriter file = new
System.IO.StreamWriter("UserList.txt", true))
                        {
                            file.WriteLine(name[0].Trim());
                            file.Close();
                        }
                    }
                }
                else
                {
                    using (FileStream fs = File.Create("UserList.txt"))
                        fs.Close();
                    using (System.IO.StreamWriter file = new
System.IO.StreamWriter("UserList.txt", true))
                    {
                        file.WriteLine(name[0].Trim());
                        file.Close();
                    }
                }
            }
        }

        private void delBtn_Click(object sender, EventArgs e)
```

```
        {
            ChatScreen.Items.RemoveAt(ChatScreen.SelectedIndex);
            delBtn.Visible = false;
            repBtn.Visible = false;
            backBtn.Visible = false;
            SendButton.Enabled = true;
            Message.Enabled = true;
        }

        private void repBtn_Click(object sender, EventArgs e)
        {
            Message.Text = "<<" + ChatScreen.SelectedItem.ToString() + ">>" + "\n";
            delBtn.Visible = false;
            repBtn.Visible = false;
            backBtn.Visible = false; ;
            SendButton.Enabled = true;
            Message.Enabled = true;
        }

        private void backBtn_Click(object sender, EventArgs e)
        {
            delBtn.Visible = false;
            repBtn.Visible = false;
            backBtn.Visible = false;
            SendButton.Enabled = true;
            Message.Enabled = true;
        }

        private void metroButton1_Click_1(object sender, EventArgs e)
        {
            Form3 frm = new Form3();
            frm.Show();
        }

    }
}
```

Chat history form –

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.IO;

namespace gp1
{
    public partial class Form3 : MetroFramework.Forms.MetroForm
    {
        public Form3()
        {
            InitializeComponent();
        }

        //get the previous chat history of the server
        private void Form3_Load(object sender, EventArgs e)
        {
            if (File.Exists("UserList.txt"))
            {
                using (var txtentry = new StreamReader("UserList.txt"))
                {
                    string entry;

                    while ((entry = txtentry.ReadLine()) != null)
                    {
                        cmbList.Items.Add(entry.Trim());
                    }
                    txtentry.Close();
                }
            }
            else
            {
                MessageBox.Show("No previous Chat history...");
            }
        }

        private void listBox1_SelectedIndexChanged(object sender, EventArgs e)
        {

            lstChat.Items.Clear();
            using (var txtentry2 = new StreamReader(cmbList.SelectedItem.ToString().Trim() +
" .txt"))
            {
                string entry;

                while ((entry = txtentry2.ReadLine()) != null)
                {
                    lstChat.Items.Add(entry);

                }
                txtentry2.Close();
            }
        }

        private void metroButton1_Click(object sender, EventArgs e)
        {
            lstChat.Items.Clear();
        }

        private void metroButton2_Click(object sender, EventArgs e)
        {
            lstChat.Items.Clear();
            string deletedName = cmbList.Text;
            File.Delete(deletedName + " .txt");
            cmbList.Items.Clear();
            cmbList.ResetText();
            string s;
            int i = 0;
            List<string> allNames = new List<string>();

            using (var txtentry = new StreamReader("UserList.txt"))
            {

                while ((s=txtentry.ReadLine()) != null)
                {
                    allNames.Add(s);
                    i++;
                }

                txtentry.Close();

            }
            for (i = 0; i < allNames.Count(); i++)
```

```
                {
                    if(allNames[i]==deletedName)
                    {
                        allNames.RemoveAt(i);
                    }

                }

                String[] newNames = new string[allNames.Count];
                int j = 0;
                foreach (String name in allNames)
                {
                    newNames[j++] = name;
                }

                using (var txtentry = new StreamWriter("UserList.txt"))
                {
                    for(i=0;i<allNames.Count();i++)
                    {
                        txtentry.WriteLine(newNames[i]);
                    }
                    txtentry.Close();
                }


                using (var txtentry = new StreamReader("UserList.txt"))
                {
                    string entry;
                    while ((entry = txtentry.ReadLine()) != null)
                    {
                        cmbList.Items.Add(entry.Trim());

                    }
                    txtentry.Close();
                }
            }
        }
    }
```

3.2 client code

Login form –

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.IO;
using System.Net.Mail;

namespace gp1
{
    public partial class login : MetroFramework.Forms.MetroForm
    {
        public login()
        {
            InitializeComponent();
        }

        private void metroButton1_Click(object sender, EventArgs e)
        {

            /*When client log to the Form1, then login form will be hidden.
             After server close the Form1, then login form will be closed.*/
            Form1 log1 = new Form1(txtIP.Text.ToString(), Convert.ToInt32(txtPort.Text));
            log1.clientname = txtName.Text;
            log1.FormClosed += new FormClosedEventHandler(log1_formclose);
            log1.Show();
            this.Hide();

        }
        private void log1_formclose(object sender,FormClosedEventArgs arg)
        {
            this.Close();

        }

        private void btnLog_Click(object sender, EventArgs e)
        {
            int x = 0;
            if (txtName.Text != "")
            {
                if (txtPW.Text != "")
                {
                    if (IsValidEmail(txtEMail.Text) == true)
                    {
                        if (File.Exists("UserList.txt"))
                        {
                            using (var txtentry = new StreamReader("UserList.txt"))
                            {
                                string entry;

                                while ((entry = txtentry.ReadLine()) != null)
                                {

                                    string[] name = entry.Trim().Split('\t');

                                    if (name[0] == txtName.Text.Trim())
                                    {
                                        MessageBox.Show("UserName Aready Exists");
                                        break;
                                    }
                                    x++;
                                }
                                txtentry.Close();
                            }
                            if (x == File.ReadLines("UserList.txt").Count())
                            {
                                using (System.IO.StreamWriter file = new
System.IO.StreamWriter("UserList.txt", true))
                                {
                                    file.WriteLine(txtName.Text + "\t" + txtPW.Text + "\t" +
txtEMail.Text);

                                    file.Close();
                                    txtName.Text = "";
                                    txtPW.Text = "";
                                    metroButton2_Click(sender, e);
                                    metroButton2.Select();
                                }
                            }
                        }
                        else
                        {
                            using (FileStream fs = File.Create("UserList.txt"))
                                fs.Close();
                            using (System.IO.StreamWriter file = new
System.IO.StreamWriter("UserList.txt", true))
                            {
                                file.WriteLine(txtName.Text + "\t" + txtPW.Text + "\t" +
txtEMail.Text);

                                file.Close();
                                txtName.Text = "";
                                txtPW.Text = "";
                                metroButton2_Click(sender, e);
                                metroButton2.Select();
                            }

                        }


                  }
                  else
                  {
                      MessageBox.Show("Invalid E mail", "Error");
                  }
              }
              else
              {
                  MessageBox.Show("Password can not be empty", "Error");
              }
              }
              else
              {
                  MessageBox.Show("User name can not be empty", "Error");
              }
          }


        private void btnLogin_Click(object sender, EventArgs e)
        {

            using (var txtentry = new StreamReader("UserList.txt"))
            {
                string entry;int x = 0;
                while ((entry = txtentry.ReadLine()) != null)
                {

                    string[] name = entry.Trim().Split('\t');

                        if (name[0] == txtName.Text.Trim())
                        {
                            if (name[1] == txtPW.Text.Trim())
                            {
                                metroButton1.Visible = false;
                                metroButton2.Visible = false;
                                metroLabel3.Visible = false;
                                metroLabel4.Visible = false;
                                txtName.Visible = false;
                                txtPW.Visible = false;
                                btnLogin.Visible = false;
                                btnSign.Visible = false;

                                metroLabel1.Visible = true;
                                metroLabel2.Visible = true;
                                txtPort.Visible = true;
                                txtIP.Visible = true;
                                btnConnect.Visible = true;

                                try
                                {
                                    string email = name[2].Trim();
                                    MailMessage mail = new MailMessage();
                                    SmtpClient SmtpServer = new SmtpClient("smtp.gmail.com");

                                    mail.From = new MailAddress("chatappvp@gmail.com");
                                    mail.To.Add(email);
                                    mail.Subject = "Security Alert!!!";
                                    mail.Body = "Someone has logged in to your account..
check whether is it you or not...";

                                    SmtpServer.Port = 587;
                                    SmtpServer.Credentials = new
System.Net.NetworkCredential("chatappvp@gmail.com", "Chatapp123");
                                    SmtpServer.EnableSsl = true;

                                    SmtpServer.Send(mail);
                                }
                                catch (Exception ex)
                                {
                                    MessageBox.Show(ex.ToString());
                                }

                                break;

                            }
                            else
                            {

                                MessageBox.Show("Invalid Password");
                                break;
                            }
                        }
                    x++;
                }

                if (x == File.ReadLines("UserList.txt").Count())
                {
                    MessageBox.Show("Invalid User Name");
                    txtPW.Text = "";
                    txtName.Text = "";
                }

                txtentry.Close();
            }
        }

        private void metroButton1_Click_1(object sender, EventArgs e)
        {
            txtEMail.Visible = true;
            lblEMail.Visible = true;
            btnSign.Visible = true;

            btnLogin.Visible = false;

        }

        private void metroButton2_Click(object sender, EventArgs e)
        {
            txtEMail.Visible = false;
            lblEMail.Visible = false;
            btnSign.Visible = false;
            btnLogin.Visible = true;
        }

        //Check the validity of the E-Mail address
        bool IsValidEmail(string email)
        {
            try
            {
                var addr = new System.Net.Mail.MailAddress(email);
                return addr.Address == email;
            }
            catch
            {
                return false;
            }
        }
    }
}
```

Messaging form –

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Net;
using System.Net.Sockets;
using System.IO;

namespace gp1
{
    public partial class Form1 : MetroFramework.Forms.MetroForm
    {
        private TcpClient client;
        public StreamReader STR;
        public StreamWriter STW;
        public string receive;
        public string TextToSend;

        public string clientname;
        public string sendmsg;

        public string serverIP;
        public int  serverPort;

        public Form1(string s,int p)
        {
            InitializeComponent();
            serverIP= s;
            serverPort = p;
            IPAddress[] localIP = Dns.GetHostAddresses(Dns.GetHostName());

        }

        private void backgroundWorker1_DoWork(object sender, DoWorkEventArgs e)
        {
            while (client.Connected)
            {
                try
                {
                    //Get the recieve message from server
                    receive = STR.ReadLine();
                    this.ChatScreen.Invoke(new MethodInvoker(delegate ()
                    {

                        ChatScreen.Items.Add( receive);
                    }));
                    receive = "";

                    writefile();
                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.Message.ToString());
                }
            }
        }
        private void backgroundWorker2_DoWork(object sender, DoWorkEventArgs e)
        {
            if (client.Connected)
            {
                STW.WriteLine (sendmsg);
                this.ChatScreen.Invoke(new MethodInvoker(delegate ()
                {
                    ChatScreen.Items.Add("\t" + "You : " + TextToSend);
                }));

                writefile();
            }
            else
            {
                MessageBox.Show("Sending failled");
            }
            backgroundWorker2.CancelAsync();
        }


        private void SendButton_Click_1(object sender, EventArgs e)
        {

            if (txtMessage.Text != "")
            {
                TextToSend = txtMessage.Text;
                sendmsg = clientname+" : " + TextToSend;
                backgroundWorker2.RunWorkerAsync();

            }

            txtMessage.Text = "";
            ChatScreen.TopIndex = ChatScreen.Items.Count - 1;
        }


        private void connect()
        {
            client = new TcpClient();
            IPEndPoint IpEnd = new IPEndPoint(IPAddress.Parse(serverIP),
int.Parse(serverPort.ToString()));
            try
            {
                client.Connect(IpEnd);
                if (client.Connected)
                {
                    lblConnect.Text="connected to server";
                    STR = new StreamReader(client.GetStream());
                    STW = new StreamWriter(client.GetStream());
                    STW.AutoFlush = true;
                    backgroundWorker1.RunWorkerAsync();
                    backgroundWorker2.WorkerSupportsCancellation = true;

                }

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

            }
        }

        private void Form1_Load(object sender, EventArgs e)
        {

            delBtn.Visible = false;
            repBtn.Visible = false;
            backBtn.Visible = false;
            connect();

            if (File.Exists(clientname + ".txt"))
            {
                using (var txtentry = new StreamReader(clientname + ".txt"))

                {
                    string entry;
                    while ((entry = txtentry.ReadLine()) != null)
                    {
                        ChatScreen.Items.Add(entry);
                    }
                    txtentry.Close();
                }
            }
            else
            {
                using (FileStream fs = File.Create(clientname + ".txt"))
                    fs.Close();
            }
        }

        void writefile()
        {
            File.Delete(clientname + ".txt");
            using (FileStream fs = File.Create(clientname+".txt"))
                fs.Close();
            using (System.IO.StreamWriter file = new
System.IO.StreamWriter(clientname+".txt", true))
            {
                int x = 0;
                while (x != ChatScreen.Items.Count)
                {
                    file.WriteLine(ChatScreen.Items[x]);
                    x++;
                }
                file.Close();
            }
        }

        private void ChatScreen_DoubleClick(object sender, EventArgs e)
        {
            delBtn.Visible = true;
            repBtn.Visible = true;
            backBtn.Visible = true;
            SendButton.Enabled = false;
            txtMessage.Enabled = false;

        }

        private void btnReturn_Click(object sender, EventArgs e)
        {
            delBtn.Visible = false;
            repBtn.Visible = false;
            backBtn.Visible = false;
            SendButton.Enabled = true;
            txtMessage.Enabled = true;
        }

        private void btnDelete_Click(object sender, EventArgs e)
        {
            ChatScreen.Items.RemoveAt(ChatScreen.SelectedIndex);
            writefile();
            delBtn.Visible = false;
            repBtn.Visible = false;
            backBtn.Visible = false;
            SendButton.Enabled = true;
            txtMessage.Enabled = true;
        }

        private void btnReply_Click(object sender, EventArgs e)
        {
            txtMessage.Text = "<<"+ ChatScreen.SelectedItem.ToString().Trim() + ">>"+"\n";
            delBtn.Visible = false;
            repBtn.Visible = false;
            backBtn.Visible = false;
            SendButton.Enabled = true;
            txtMessage.Enabled = true;
        }

        private void ChatScreen_SelectedIndexChanged(object sender, EventArgs e)
        {

        }

        private void delBtn_Click(object sender, EventArgs e)
        {
            DialogResult dialogResult = MessageBox.Show("Are you sure you want to delete?",
"Delete Message", MessageBoxButtons.YesNo);
```

```
                if (dialogResult == DialogResult.Yes)
                {
                    ChatScreen.Items.RemoveAt(ChatScreen.SelectedIndex);
                    writefile();
                    delBtn.Visible = false;
                    repBtn.Visible = false;
                    backBtn.Visible = false;
                    SendButton.Enabled = true;
                    txtMessage.Enabled = true;
                }
                else if (dialogResult == DialogResult.No)
                {

                }

            }

            private void metroButton2_Click(object sender, EventArgs e)
            {
                delBtn.Visible = false;
                repBtn.Visible = false;
                backBtn.Visible = false;
                SendButton.Enabled = true;
                txtMessage.Enabled = true;
            }

            private void metroButton1_Click(object sender, EventArgs e)
            {
                txtMessage.Text = "<< " + ChatScreen.SelectedItem.ToString().Trim() + " >>\n";

                delBtn.Visible = false;
                repBtn.Visible = false;
                backBtn.Visible = false;
                SendButton.Enabled = true;
                txtMessage.Enabled = true;
            }
        }
    }
```

4.Conclusion

The main objective of the project is to develop a Secure and efficient Chat Application. Here in this project we have considered the privacy and the data of the users. So we have included a method to avoid that issue. In the program we take email address of the user. If he/she logged into the chat that person will receive a security alert.

In some chat applications user cannot access any client information. But in our program server can access all the chat details. So it will easy to control the clients.

Also we can improve our program by using some functionalities. First thing is currently the chat can be happen between only two people (client and server). So it can be improved to multiclient program.

Second thing is we can use a real-time database which make it easy to implement reactive applications because it can keep your information up to date and you can access the database and run the chat application anytime, anywhere.

Although, this project is made for Small Scale communication but if we implement new features in this system, it will be able to compete with other online chatting applications.

References

- https://www2.le.ac.uk/offices/ld/resources/writing/writing-resources/reports
- http://csharp.net-informations.com/communications/csharp-chat-server-programming.htm
- https://code.msdn.microsoft.com/windowsdesktop/Chat-Application-using-0a279e3f
- https://pusher.com/tutorials/chat-aspnet
- https://www.codeproject.com/Tips/607801/SimpleplusChatplusprogramplusinplusC-23
- https://stackoverflow.com/questions/18894738/create-chat-application-using-c
- https://www.dreamincode.net/forums/topic/33396-basic-clientserver-chat-application-in-c%23/