UNIVERSITY OF INFORMATION TECHNOLOGY
FACULTY OF COMPUTER NETWORK AND COMMUNICATION

REPORT
Subject: Basic Network Programming
Semester II (2021 – 2022)

Report Lab 6

Student: Võ Anh Kiệt

Student ID Number: 20520605

Class: NT106.M21.ANTN

University of Information Technology

Lecturer: Đỗ Thị Hương Lan

Hồ Chí Minh City, April 2022

Report Lab 6

Student Information

Full Name: Võ Anh Kiệt

Student ID Number: 20520605

Class: ANTN2020

Device Information

CPU: Intel core i5 – 8250U @ 1.60 GHz

Ram: 16 GB DDR3L

SSD M2 SATA: 500GB

HDD: 1000GB

Display chip name: UHD 620

Link Video: https://drive.google.com/drive/folders/1G1qsWBf-PgL7-475xJT-gs_FyKqysJly?usp=sharing

Create the Encryption and Decryption class

Using RSA 2048 and key “voanhkiet”

```
public static class myTransformer1
{
    1 reference
    public static string Encrypt(string value)
    {
        byte[] plaintext = Encoding.Unicode.GetBytes(value);

        CspParameters cspParams = new CspParameters();
        cspParams.KeyContainerName = "voanhkiet";
        using (RSACryptoServiceProvider RSA = new RSACryptoServiceProvider(2048, cspParams))
        {
            byte[] encryptedData = RSA.Encrypt(plaintext, false);
            return Convert.ToBase64String(encryptedData);
        }
    }

    1 reference
    public static string Decrypt(string value)
    {
        byte[] encryptedData = Convert.FromBase64String(value);

        CspParameters cspParams = new CspParameters();
        cspParams.KeyContainerName = "voanhkiet";
        using (RSACryptoServiceProvider RSA = new RSACryptoServiceProvider(2048, cspParams))
        {
            byte[] decryptedData = RSA.Decrypt(encryptedData, false);
            return Encoding.Unicode.GetString(decryptedData);
        }
    }

}
```

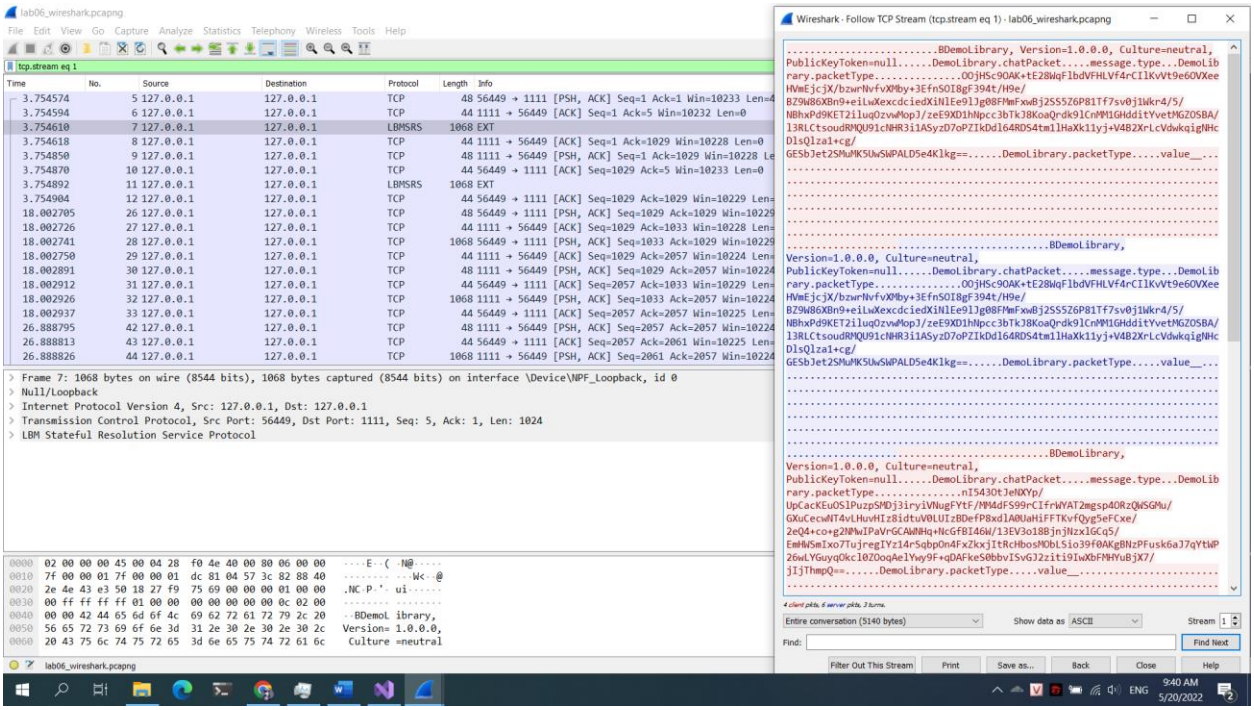
Encrypt before sending

```
1 reference
private void button3_Click(object sender, EventArgs e)
{
    if (roomSelect.connectGetSet == false)
    {
        return;
    }
    string enc = myTransformer1.Encrypt(textBox1.Text);
    roomSelect.myConnectRoomGetSet.sendText(enc);
    textBox1.Text = "";
}
```

Decrypt when receiving

```
switch(packet.type)
{
    case packetType.CHATMESSAGE:
        string message = ((chatPacket)packet).message;
        message = myTransformer1.Decrypt(message);
        outputText(message);
        break;
}
```

Catch the packet of message



The data have been encrypted before sending and decrypted when recieve

Inheritance from Lab 3

Create chatApp has multi-room, multi-client replacing 4 tasks

DemoLibrary

Set up the type for the object

```
5 references
public enum packetType
{
    EMPTY,
    NICKNAME,
    CHATMESSAGE,
}
```

Set up the class packet

```
public class packet
{
    public packetType type = packetType.EMPTY;
}
```

Set up the chatPacket – inheritance class of packet

```
public class chatPacket : packet
{
    public string message = String.Empty;
    1 reference
    public chatPacket(string message)
    {
        this.type = packetType.CHATMESSAGE;
        this.message = message;
    }
}
```

Set up the nicknamePacket – inheritance class of packet

```
public class nicknamePacket : packet
{
    public string nickName = String.Empty;
    1 reference
    public nicknamePacket (string nickname)
    {
        this.type = packetType.NICKNAME;
        this.nickName = nickname;
    }
}
```

simpleServer

serverClass

Declare the variable and the constructor

```
class server
{
    int portSimpleServer;
    TcpListener tcpListener;
    static List<client> clients = new List<client>();

    1 reference
    public server(string ipAddress, int port)
    {
        portSimpleServer = port;
        IPAddress ip = IPAddress.Parse(ipAddress);
        tcpListener = new TcpListener(ip, port);
    }
```

Set up the start function

```
public void start()
{
    tcpListener.Start();

    Console.WriteLine("Port: " + Convert.ToString(portSimpleServer));

    while(true)
    {
        Socket socket = tcpListener.AcceptSocket();
        client client = new client(socket);
        clients.Add(client);
        client.start();
    }
}
```

Set up the socket method to create the socket to use for chatting

```
public static void socketMethod(client dataFromClient)
{
    try
    {
        BinaryFormatter binaryFormatter = new BinaryFormatter();
        Socket socket = dataFromClient.Socket;
        NetworkStream stream = dataFromClient.Stream;
        BinaryReader binaryReader = dataFromClient.Reader;
        dataFromClient.sendText(dataFromClient, "Successful Connection");

        int numberInputBytes;
        while((numberInputBytes = binaryReader.ReadInt32()) != 0 )
        {
            byte[] bytes = binaryReader.ReadBytes(numberInputBytes);
            MemoryStream memoryStream = new MemoryStream(bytes);
            packet packet = binaryFormatter.Deserialize(memoryStream) as packet;

            switch (packet.type)
            {
                case packetType.NICKNAME:
                    string nickName = ((nicknamePacket)packet).nickName;
                    dataFromClient.setupNickName(nickName);
                    break;

                case packetType.CHATMESSAGE:
                    string message = ((chatPacket)packet).message;
                    Console.WriteLine("<" + dataFromClient.NickName + ">: " + message);
                    foreach(client element in clients)
                    {
                        element.sendText(dataFromClient, message);
                    }
                    break;
            }
```

clientClass

Declare the constructor and the GetSet function

```
public class client
{
    5 references
    public Socket Socket { get; private set; }
    4 references
    public NetworkStream Stream { get; private set; }
    2 references
    public BinaryReader Reader { get; private set; }
    4 references
    public BinaryWriter Writer { get; private set; }
    4 references
    public string NickName { get; private set; }

    private Thread thread;

    1 reference
    public client(Socket socket)
    {
        Socket = socket;
        Stream = new NetworkStream(Socket, true);
        Writer = new BinaryWriter(Stream, Encoding.UTF8);
        Reader = new BinaryReader(Stream, Encoding.UTF8);
    }
```

Set up the start and stop function

```
1 reference
public void start()
{
    thread = new Thread(new ThreadStart(socketMethod));
    thread.Start();
}

2 references
public void stop(bool aThread = false)
{
    Socket.Close();
    if (thread.IsAlive)
        thread.Abort();
}
```

Set up the sendFunc and the socket Method for using

```
1 reference
public void sendFunc(packet data)
{
    MemoryStream memoryStream = new MemoryStream();
    BinaryFormatter binaryFormatter = new BinaryFormatter();
    binaryFormatter.Serialize(memoryStream, data);
    byte[] buffer = memoryStream.GetBuffer();

    Writer.Write(buffer.Length);
    Writer.Write(buffer);
    Writer.Flush();
}

1 reference
private void socketMethod()
{
    server.socketMethod(this);
}
```

Set up the nicknameFunc and sendText

```
1 reference
public void setupNickName(string nickName)
{
    this.NickName = nickName;
}

2 references
public void sendText(client dataFromClient, string text)
{
    if(Socket.Connected == false)
    {
        return;
    }
    string message = "*" + text + "*";
    if (dataFromClient.NickName !=  null)
    {
        message = dataFromClient.NickName + ": " + text;
    }
    chatPacket chat = new chatPacket(message);
    sendFunc(chat);
}
```

Set up the main function in the program class

To change the the port, dumplicate the server file and then go the sln file and change the number, do it for 5 times to get 5 ports

```
static void Main()
{
    Application.EnableVisualStyles();
    Application.SetCompatibleTextRenderingDefault(false);
    server simpleServer = new server("127.0.0.1", 1111); // 1111, 2222, 3333, 4444, 5555
    simpleServer.start();
    simpleServer.stop();
}
```

simpleClient

declare the variable and sendFunc

```
public class myConnect
{
    private clientForm form;
    private TcpClient tcpClient;
    private NetworkStream stream;
    private BinaryWriter writer;
    private BinaryReader reader;
    private Thread thread;

    2 references
    public void sendFunc(packet data)
    {
        MemoryStream memoryStream = new MemoryStream();
        BinaryFormatter binaryFormatter = new BinaryFormatter();
        binaryFormatter.Serialize(memoryStream, data);
        byte[] buffer = memoryStream.GetBuffer();

        writer.Write(buffer.Length);
        writer.Write(buffer);
        writer.Flush();
    }
```

Set up the function sendtext, nickname and the outputText

```
public void sendText(string str)
{
    if (tcpClient.Connected == false)
    {
        return;
    }
    chatPacket chat = new chatPacket(str);
    sendFunc(chat);
}
1 reference
private void setUpNickName(string nickname)
{
    nicknamePacket chatPac = new nicknamePacket(nickname);
    sendFunc(chatPac);
}
private delegate void AppendTextDelegate(string str);
5 references
private void outputText(string text)
{
    form.Invoke(new AppendTextDelegate(form.appendText), new object[] { text });
}
```

Set up the process with the responding server

```
private void processSeverRespone()
{
    try
    {
        BinaryFormatter binaryFormatter = new BinaryFormatter();
        int numberInputByte;
        while ((numberInputByte = reader.ReadInt32()) != 0)
        {
            byte[] bytes = reader.ReadBytes(numberInputByte);
            MemoryStream memoryStream = new MemoryStream(bytes);
            packet packet = binaryFormatter.Deserialize(memoryStream) as packet;

            switch(packet.type)
            {
                case packetType.CHATMESSAGE:
                    string message = ((chatPacket)packet).message;
                    outputText(message);
                    break;
            }
        }
    }
    catch (Exception exc)
    {
        outputText("Caution: " + exc.Message);
    }
}
```

Make the connection function to connect with the server

```
1 reference
public bool makeConnect(clientForm cform, string hostname, int port, string nickname)
{
    try
    {

        form = cform;
        tcpClient = new TcpClient();
        tcpClient.Connect(hostname, port);
        stream = tcpClient.GetStream();
        writer = new BinaryWriter(stream, Encoding.UTF8);
        reader = new BinaryReader(stream, Encoding.UTF8);

        setUpNickName(nickname);

        thread = new Thread(new ThreadStart(processSeverRespone));
        thread.Start();
    }
    catch (Exception exc)
    {
        outputText("Exception: " + exc.Message);
        return false;
    }

    return true;
}
```

Make the disconnection function to disconnect with the server

```
1 reference
public void makeDisconnect()
{
    try
    {
        reader.Close();
        writer.Close();
        tcpClient.Close();
        thread.Abort();
    }
    catch (Exception exc)
    {
        outputText("Caution: " + exc.Message);
    }
    outputText("Disconnect");
}
```

myRoom

Set up the multiroom for the chatApp

```
public class myRoom
{
    private string name;
    private string address;
    private int port;
    private bool connect;
    private myConnect myConnectRoom;

    5 references
    public myRoom(string name, string address, int port)
    {
        this.name = name;
        this.address = address;
        this.port = port;
        this.connect = false;
        this.myConnectRoom = new myConnect();
    }
```

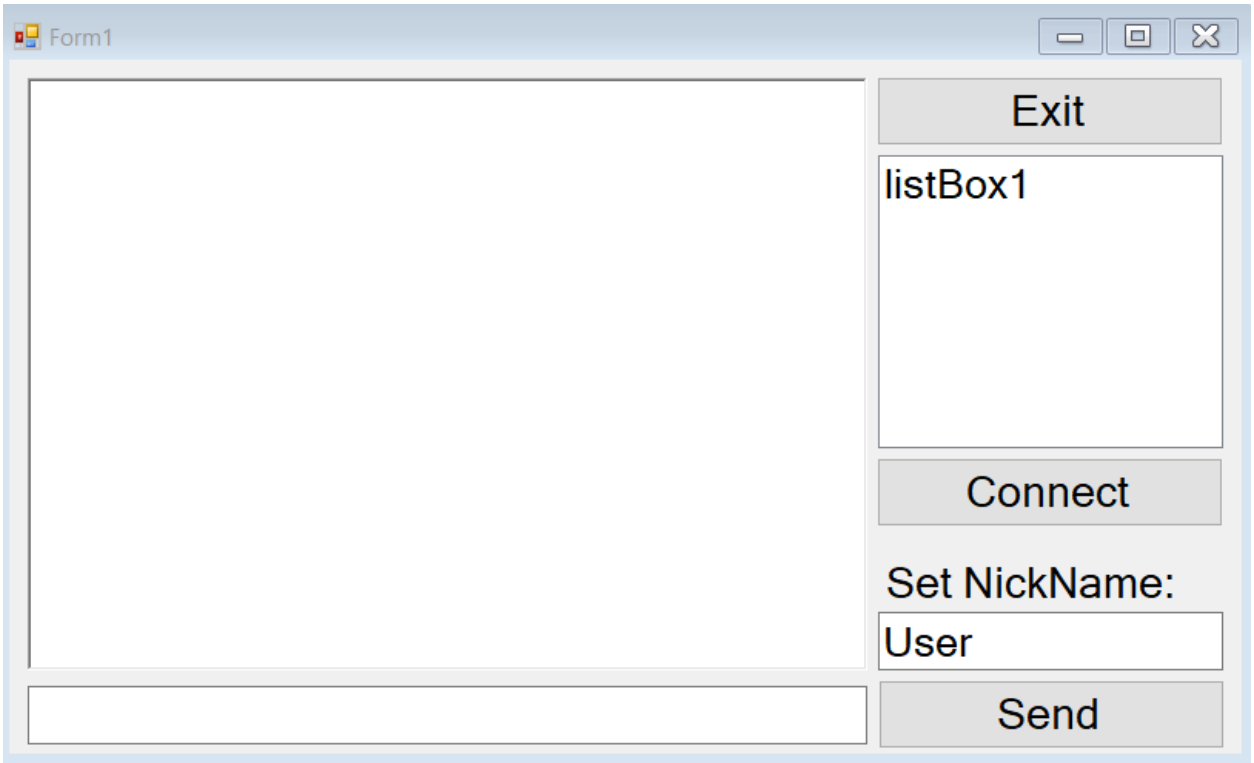
Name, address, port, connect and connectRoom class to get and set value

```
5 references
public string nameGetSet
{
    get
    {
        return name;
    }
    set
    {
        name = value;
    }
}

1 reference
public string addressGetSet
{
    get
    {
        return address;
    }
    set
    {
        address = value;
    }
}
```

Create the connection in main program

```
0 references
static void Main()
{
    Application.EnableVisualStyles();
    Application.SetCompatibleTextRenderingDefault(false);
    Application.Run(new login());
    myConnect connect = new myConnect();

}
```

Form for the chat room



Set up the client form

```
public partial class clientForm : Form
{
    private myRoom room1;
    private myRoom room2;
    private myRoom room3;
    private myRoom room4;
    private myRoom room5;
    private myRoom roomSelect = null;

    1 reference
    public clientForm()
    {
        InitializeComponent();
        room1 = new myRoom("Room 1", "127.0.0.1", 1111);
        room2 = new myRoom("Room 2", "127.0.0.1", 2222);
        room3 = new myRoom("Room 3", "127.0.0.1", 3333);
        room4 = new myRoom("Room 4", "127.0.0.1", 4444);
        room5 = new myRoom("Room 5", "127.0.0.1", 5555);

    }
```

Append the text function and and show the connect

```
1 reference
public void appendText(string str)
{
    richTextBox1.Text += str + "\n";
    richTextBox1.SelectionStart = richTextBox1.Text.Length;
    richTextBox1.ScrollToCaret();
}
5 references
string checkConnect(myRoom room)
{
    bool value1 = room.connectGetSet;

    if (value1)
    {
        return " (connected)";
    }
    else
    {
        return "";
    }
}
2 references
```

Refresh the list

```
2 references
void refreshTheList()
{
    listBox1.Items.Clear();

    string[] myRoomsList = new string[5];

    myRoomsList[0] = room1.nameGetSet + checkConnect(room1);
    myRoomsList[1] = room2.nameGetSet + checkConnect(room2);
    myRoomsList[2] = room3.nameGetSet + checkConnect(room3);
    myRoomsList[3] = room4.nameGetSet + checkConnect(room4);
    myRoomsList[4] = room5.nameGetSet + checkConnect(room5);

    listBox1.Items.AddRange(myRoomsList);
}
```

Set up the connection room

```
1 reference
void connectRoom(myRoom room, bool disconnect)
{
    if(room.connectGetSet == false)
    {
        bool connect = room.myConnectRoomGetSet.makeConnect(this, room.addressGetSet, room.portGetSet, te
        if(connect == true)
        {
            button1.Text = "Disconnect";
            room.connectGetSet = true;
            roomSelect.myConnectRoomGetSet.sendText("*join the room*");
        }
    }
    else if (disconnect == true)
    {
        roomSelect.myConnectRoomGetSet.sendText("*left the room*");
        room.myConnectRoomGetSet.makeDisconnect();
        room.connectGetSet = false;
        button1.Text = "Connect";
    }
    refreshTheList();
}
```

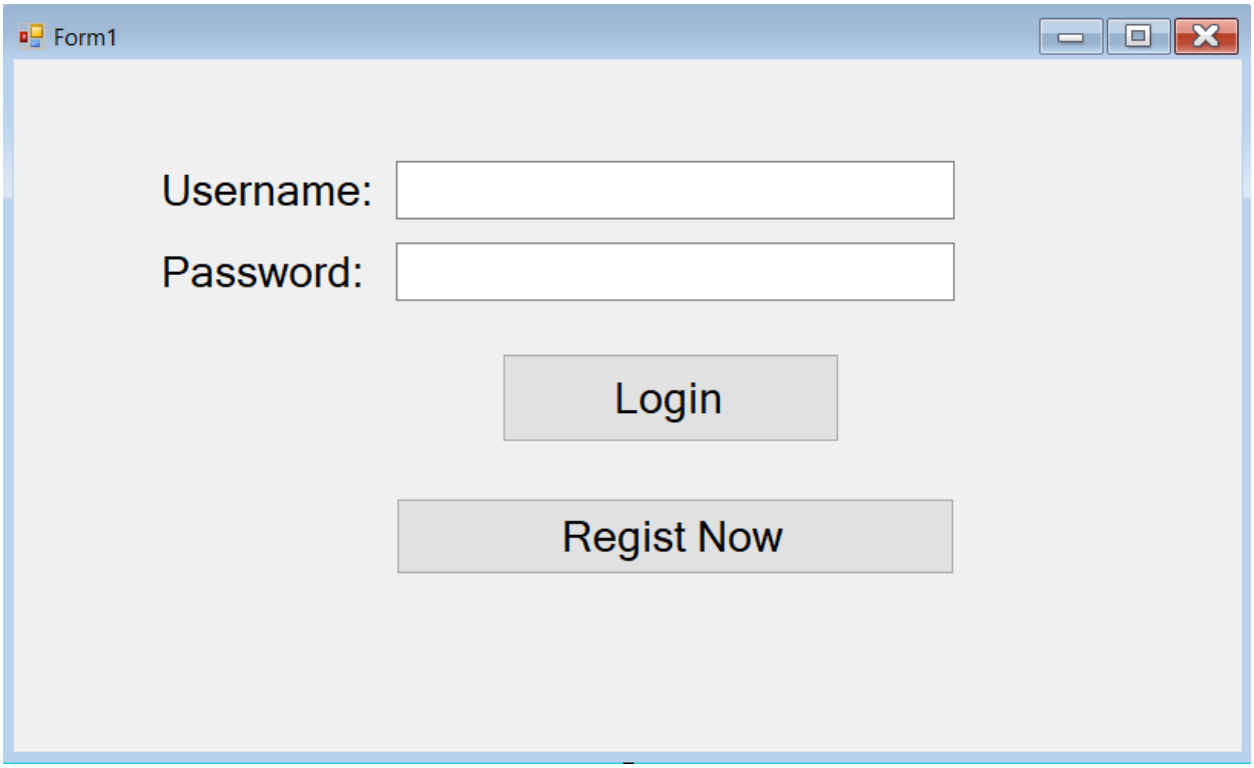
```
private void listBox1_SelectedIndexChanged(object sender, EventArgs e)
{
    int value1 = listBox1.SelectedIndex;
    switch(value1)
    {
        case 0:
            roomSelect = room1;
            break;
        case 1:
            roomSelect = room2;
            break;
        case 2:
            roomSelect = room3;
            break;
        case 3:
            roomSelect = room4;
            break;
        case 4:
            roomSelect = room5;
            break;
    }

    if (roomSelect.connectGetSet == false)
    {
        button1.Text = "Connect";
    }
    else
```

Set up the click

```
1 reference
private void button1_Click(object sender, EventArgs e)
{
    connectRoom(roomSelect, true);
}

1 reference
private void clientForm_Load(object sender, EventArgs e)
{
    refreshTheList();
}

1 reference
private void button3_Click(object sender, EventArgs e)
{
    if (roomSelect.connectGetSet == false)
    {
        return;
    }
    roomSelect.myConnectRoomGetSet.sendText(textBox1.Text);
    textBox1.Text = "";
}
```

```
1 reference
private void button2_Click(object sender, EventArgs e)
{
    this.Hide();
    login tmpForm = new login();
    tmpForm.Closed += (s, args) => this.Close();
    tmpForm.Show();
}
```

Login form



Declarate the variable

```
OleDbConnection dbConnect = new OleDbConnection(@"Provider=Microsoft.Jet.OLEDB.4.0;Data Source=D:\excercise
OleDbCommand dbCommand = new OleDbCommand();
OleDbDataAdapter dbDataAdapter = new OleDbDataAdapter();
3 references
public login()
{
    InitializeComponent();
}
```

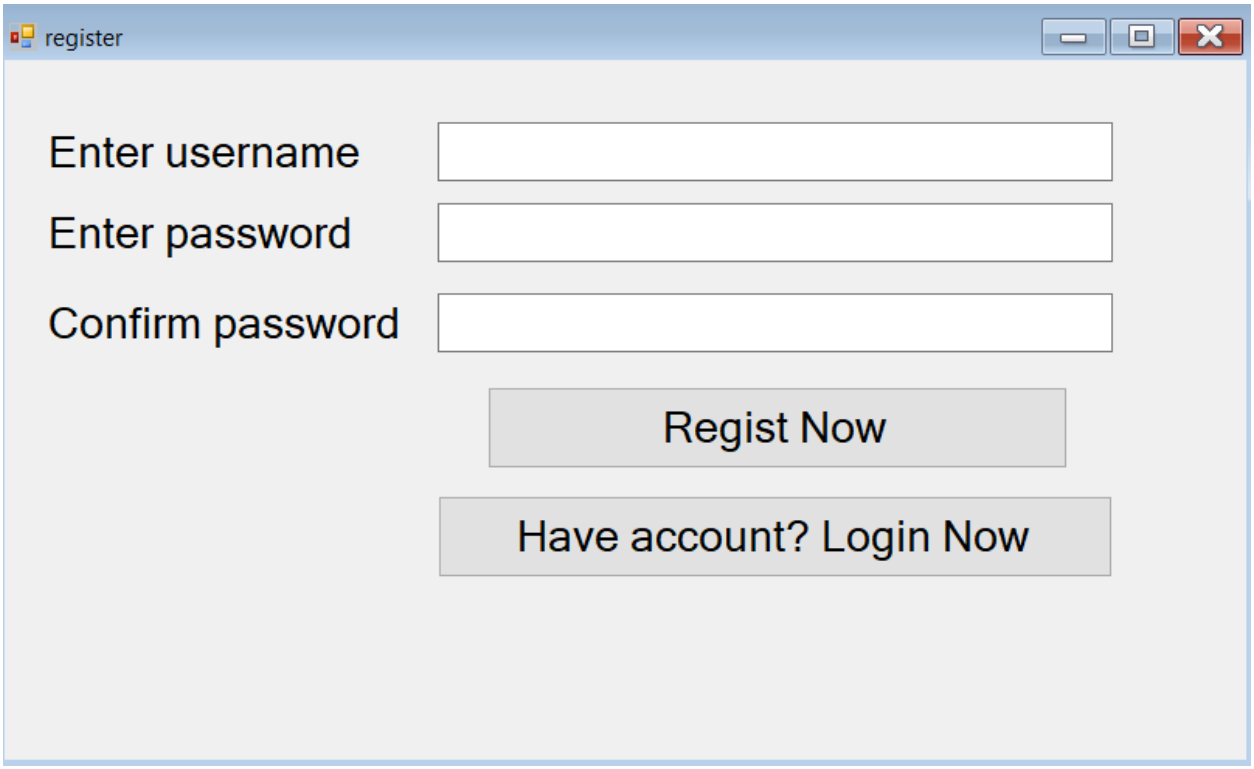
Get the data from the database and verify the login

```
1 reference
private void button1_Click(object sender, EventArgs e)
{
    dbConnect.Open();
    string login = "SELECT * FROM Table1 WHERE username= '" + enterUser.Text + "' and password= '" + enter
    dbCommand = new OleDbCommand(login, dbConnect);
    OleDbDataReader dbDataReader = dbCommand.ExecuteReader();

    if (dbDataReader.Read() == true)
    {
        this.Hide();
        var tmpForm = new clientForm();
        tmpForm.Closed += (s, args) => this.Close();
        tmpForm.Show();
        dbConnect.Close();
    }
    else
    {
        MessageBox.Show("Invalid Username or Password, Please Try Again");
        enterUser.Text = "";
        enterPass.Text = "";
        enterUser.Focus();
        dbConnect.Close();
    }

}
```

Go to the register

```
1 reference
private void button2_Click(object sender, EventArgs e)
{
    this.Hide();
    register tmpForm = new register();
    tmpForm.Closed += (s, args) => this.Close();
    tmpForm.Show();
}
```

Register form



Declare the variable

```
OleDbConnection dbConnect = new OleDbConnection(@"Provider=Microsoft.Jet.OLEDB.4.0;Data Source=D:\excer
OleDbCommand dbCommand = new OleDbCommand();
OleDbDataAdapter dbDataAdapter = new OleDbDataAdapter();
1 reference
public register()
{
    InitializeComponent();
}
```

Get the registration information and append it into access file 2002 – 2003 (mdb)

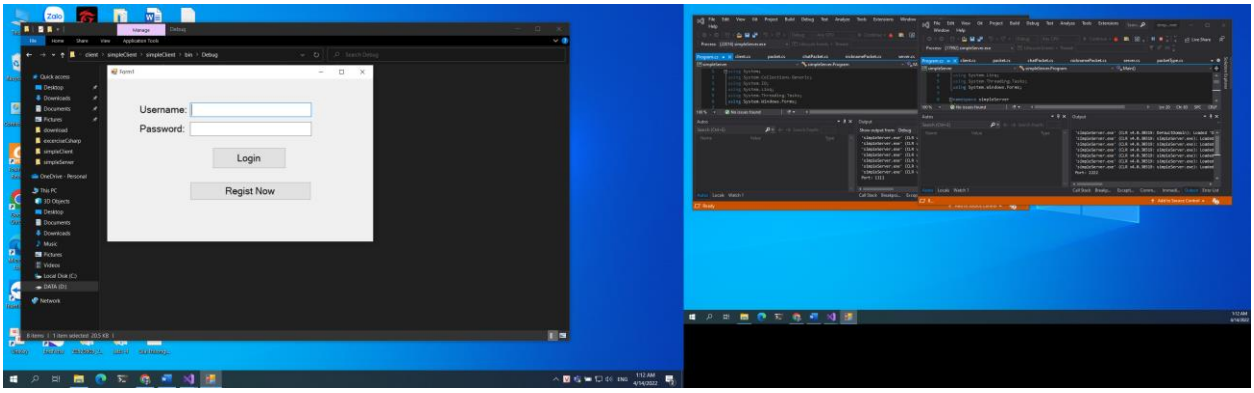
```
private void button1_Click(object sender, EventArgs e)
{
    if (enterUser.Text == "" && enterPass.Text == "" && confirmPass.Text == "")
    {
        MessageBox.Show("Username and Password fields are empty");
    }
    else if (enterPass.Text == confirmPass.Text)
    {
        dbConnect.Open();
        string register = "INSERT INTO Table1 VALUES ('" + enterUser.Text + "','" + enterPass.Text + "')";
        dbCommand = new OleDbCommand(register, dbConnect);
        dbCommand.ExecuteNonQuery();
        dbConnect.Close();

        enterUser.Text = "";
        enterPass.Text = "";
        confirmPass.Text = "";
        MessageBox.Show("Successfully Regist!");
    }
    else
    {
        MessageBox.Show("Password does not match!");
        enterPass.Text = "";
        confirmPass.Text = "";
        enterPass.Focus();
    }

}
```

Back to the login

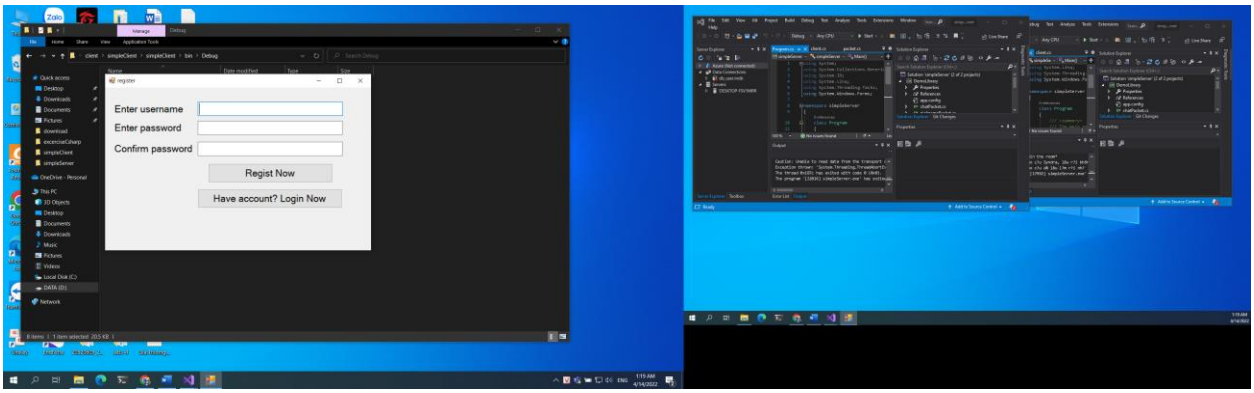
```
1 reference
private void button2_Click(object sender, EventArgs e)
{
    this.Hide();
    login tmpForm = new login();
    tmpForm.Closed += (s, args) => this.Close();
    tmpForm.Show();
}
```

Deploy the chatApp

Login



Register



Using chatApp