TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA CÔNG NGHỆ PHẦN MỀM

ĐỒ ÁN MÔN LẬP TRÌNH TRỰC QUAN

BÁO CÁO LẬP TRÌNH TRỰC QUAN

Nhóm sinh viên :
Lê Hữu Thắng - 16521098
Hồ Nguyễn Minh Triết - 16521294
Lớp : IT008.I21
Giảng viên hướng dẫn : Huỳnh Tuấn Anh

Thành phố Hồ Chí Minh –ngày 15 tháng 06 năm 2018

MỤC LỤC

LỜI CẢM ƠN :

Nhóm em xin gửi lời cảm ơn chân thành đến các thầy cô của trường Đại học Công Nghệ Thông Tin,khoa Công nghệ phần mềm đã tạo điều kiện cho nhóm được thực hiện đề tài môn học Lập trình trực quan.Đặc biệt nhóm em xin gửi lời cảm ơn sâu sắc đến thầy Huỳnh Tuấn Anh.Thầy đã giảng dạy nhiệt tình,hướng dẫn,sửa chữa và góp ý để giúp nhóm em hoàn thành tốt đề tài
Trong quá trình thực hiện đề tài môn học cũng như quá trình làm báo cáo không thể tránh khỏi sai sót do trình độ cũng như kinh nghiệm thực tiễn của nhóm em còn hạn chế,rất mong thầy bỏ qua.Nhóm em rất mong được nhận ý kiến đóng góp từ thầy để có thêm nhiều kinh nghiệm và sẽ hoàn thành tốt các đề tài sắp tới.Nhóm em xin chân thành cảm ơn ạ.

CHƯƠNG 1 : MỞ ĐẦU

1.1 Lý do chọn đề tài

Hiện nay Internet,đặc biệt là mạng LAN nói riêng đã và đang có những tiến bộ vượt bậc và ngày càng phổ biến hơn trong cuộc sống hiện đại.Nếu trước đây để có thể liên lạc,trao đổi hàng hóa với nhau,đặc biệt là giữa những khoảng cách xa thì con người cần rất nhiều thời gian thì ngày nay,chỉ cần thao tác click chuột,người ta đã có thể làm mọi thứ 1 cách nhanh chóng mà không cần di chuyển nhiều.Nhu cầu thông tin liên lạc và trao đổi qua mạng ngày càng lớn hơn.Chính vì vậy mà chương trình Chat trên mạng LAN được xây dựng để đáp ứng phần nào nhu cầu cần thiết ấy

1.2 Mục đích đề tài

Xây dựng chương trình Chat hoạt động trong mạng LAN với các chức năng cơ bản :gửi tin nhắn public,chat riêng với bạn bè hoặc với chính server,chat nhóm,load người dùng online theo thời gian thực

1.3 Đối tượng và phạm vi nghiên cứu

1.3.1 Đối tượng nghiên cứu

Tìm hiểu về Lập trình Socket trong C# để xây dựng nên chương trình

1.3.2 Phạm vi nghiên cứu

Chương trình được xây dựng với khả năng gửi được văn bản qua lại giữa các User thông qua sự điều khiển,điều hướng của 1 Server

CHƯƠNG 2 : KIẾN THỨC ỨNG DỤNG

2.1 Sơ lược về Lập trình Socket

2.1.1 Khái niệm về Địa chỉ (IP) và Cổng (Port)

Địa chỉ IP (IP viết tắt của Internet Protocol - giao thức Internet) là số định dạng cho một phần cứng mạng, các thiết bị sử dụng địa chỉ IP để liên lạc với nhau qua mạng dựa trên IP như mạng Internet.Hiểu nôm na,địa chỉ IP chính có thể xem là địa chỉ nhà riêng.
+VD: Khi bạn muốn gửi bưu kiện đến cho bạn bè ở nơi khác,bạn cần phải biết địa chỉ chính xác chứ không thể ghi tên rồi chờ gói bưu kiện sẽ đến tay bạn bè của bạn
Cổng (Port) xác định duy nhất một quá trình (process) trên một máy trong mạng. Hay nói cách khác là cách mà phân biệt giữa các ứng dụng.
+VD: Khi máy bạn chạy nhiều ứng dụng mạng như Yahoo,Firefox, game online….Chương trình Yahoo sử dụng cổng (port) 5150 thì khi ai đó gửi tin nhắn đến, máy bạn nó sẽ dựa vào port để nhận biết đó là chương trình Yahoo (port 5150) chứ không phải là chương trình khác. Sau đó thông tin sẽ đc xử lý và hiển thị tin nhắn lên.

2.1.2 Khái niệm Socket

Socket là 1 trong những kĩ thuật cơ bản nhất trong truyền thông mạng máy tính,được sử dụng để 1 tiến trình "nói chuyện" với 1 tiến trình khác
Nhiều ứng dụng thông dụng hiện nay sử dụng kĩ thuật Socket như : trình duyệt web, mail,client,..
Có 2 loại Socket :
+Stream Socket: Dựa trên giao thức TCP (Tranmission Control Protocol) việc truyền dữ liệu chỉ thực hiện giữa 2 quá trình **đã thiết lập kết nối**. Giao thức này đảm bảo dữ liệu được truyền đến nơi nhận một cách đáng tin cậy, đúng thứ tự nhờ vào cơ chế quản lý luồng lưu thông trên mạng và cơ chế chống tắc nghẽn.
+Datagram Socket: Dựa trên giao thức UDP (User Datagram Protocol) việc truyền dữ liệu **không yêu cầu có sự thiết lập kết nối giữa 2 quá trình**. Ngược lại với giao thức TCP thì dữ liệu được truyền theo giao thức UDP không được tin cậy, có thể không đúng trình tự và lặp lại. Tuy nhiên vì nó không yêu cầu thiết lập kết nối không phải có những cơ chế phức tạp nên tốc độ nhanh…ứng dụng cho các ứng dụng truyền dữ liệu nhanh như gọi điện thoại,trình đa phương tiện,…

- ➔ Một TCP/IP Socket gồm một địa chỉ IP kết hợp với một port.Xác định duy nhất một tiến trình (process) trên mạng. Hay nói cách khác Luồng thông tin trên mạng dựa vảo IP là để xác định máy một máy trên mạng còn port xác định 1 tiến trình trên 1 máy.

2.1.3 Các lớp .NET cơ bản trong lập trình mạng

Namespace System.Net
Lớp IPAdress
Trong .NET, IPAddress là 1 lớp dùng để mô tả địa chỉ IP. Đây là lớp rất cơ bản được sử dụng khi chúng ta thao tác (truyền) vào các lớp như IPEndpoint,UDP,TCP,Socket,…
Bảng 2.1.3a : Các thành phần của lớp IPAdress

Thành viên Static	**Mô tả**

Any	Cung cấp một địa chỉ IP (thường là 0.0.0.0) để chỉ ra rằng Server phải lắng nghe các hoạt động của Client trên tất cả các Card mạng (sử dụng khi xây dựng Server). Thuộc tính này chỉ đọc.
Broadcast	Cung cấp một địa chỉ IP quảng bá (Broadcast, thường là 255.255.255.255), ở dạng số long.
Loopback	Trả về một địa chỉ IP lặp (IP Loopback, ví dụ 127.0.0.1).
AddressFamily	Trả về họ địa chỉ của địa chỉ IP hiện hành. Nếu địa chỉ ở dạng IPv4 thì kết quả là Internetwork, và InternetworkV6 nếu là địa chỉ IPv6.
Phương thức	**Mô tả**
IPAddress(Int64)	Tạo địa chỉ IP từ một số long.
IPAddress(Byte[])	Tạo địa chỉ IP từ một mảng Byte.
GetAddressByte ()	Chuyển địa chỉ thành mảng Byte.
HostToNetworkOrder()	Đảo thứ tự Byte của một số cho đúng với thứ tự Byte trong địa chỉ IPAddress.
IsLoopback()	Cho biết địa chỉ có phải là địa chỉ lặp hay không?

Lớp IPEndpoint

Trong mạng,để 2 trạm có thể trao đổi thông tin với nhau thì chúng phải biết địa chỉ (IP) và cổng (port) mà 2 bên dùng để trao đổi. Lớp IPAddress mới chỉ cung cấp cho ta 1 nửa là địa chỉ IP,như vậy vẫn còn thiếu nửa còn lại đó là cổng (port)-> Lớp IPEndpoint chính là lớp chứa đựng cả 2 Đối tượng IPEndpoint sẽ được dùng sau này để truyền trực tiếp cho các đối tượng UDP,TCP,…

Bảng 2.1.3b : Các thành viên của lớp IPEndpoint

Phương thức khởi tạo	**Mô tả**
IPEndPoint(Int64, Int32)	Tạo một đối tượng mới của lớp **IPEndPoint,** tham số truyền vào là địa **chỉ IP** (ở dạng số) và **cổng** sẽ dùng để giao tiếp.

IPEndPoint(IPAddress, Int32)	Tạo một đối tượng mới của lớp **IPEndPoint,** Tham số truyền vào là một địa chỉ **IPAddress** và số hiệu cổng dùng để giao tiếp.
Thuộc tính	**Mô tả**
Address	Trả về hoặc thiết lập địa chỉ IP cho Endpoint (trả về một đối tượng IPAddress).
AddressFamily	Lấy về loại giao thức mà Endpoint này đang sử dụng.
Port	Lấy hoặc gán số hiệu cổng của Endpoint.
Phương thức	**Mô tả**
Create()	Tạo một Endpoint từ một địa chỉ socket (socket address).
ToString()	Trả về địa chỉ IP và số hiệu cổng theo khuôn dạng địa chỉ: cổng. Ví dụ: "192.168.1.1:8080"

Namespace System.Net.Sockets

Lớp Socket

Bảng 2.1.3c : Các thành phần của lớp Socket

Phương thức khởi tạo	Mô tả
Socket(AddressFamily,SocketType,ProtocolType)	Tạo 1 Socket mới với địa chỉ được chỉ định,kiểu socket và giao thức truyền tin
Socket(SocketType,ProtocolType)	Tạo 1 Socket mới với kiểu socket và giao thức truyền tin được truyền vào
Thuộc tính	Mô tả
AddressFamily	Lấy về địa chỉ của Socket
Connected	Lấy về giá trị cho biết Socket có được kết nối với máy chủ tùy vào hoạt động Nhận hay Gửi cuối cùng
ProtocolType	Lây về kiểu giao thức truyền tin
ReceiveBufferSize	Lấy hoặc thiết lập kích thước của buffer nhận của Socket
SendBufferSize	Lấy hoặc thiết lập kích thước của buffer gửi của Socket
Phương thức	Mô tả
BeginAccept(AsyncCallback,Object)	Bắt đầu 1 quá trình hoạt động bất đồng bộ để chấp nhận 1 nỗ lực kết nối đến
BeginConnect(EndPoint,AsyncCallback,Object)	Bắt đầu 1 yêu cầu bất đồng bộ cho 1 kết nối máy chủ từ xa
BeginReceive(Byte[],Int32,Int32,SocketFlags, AsyncCallback,Object)	Bắt đầu nhận dữ liệu bất đồng bộ từ 1 Socket đã kết nối

BeginSend(Byte[],Int32,Int32,SocketFlags, AsyncCallback,Object)	Bắt đầu gửi dữ liệu bất đồng bộ đến 1 Socket đã kết nối
Bind(EndPoint)	Liên kết Socket với IPEndPoint cục bộ
Close()	Đóng kết nối Socket và giải phóng tất cả các tài nguyên liên quan đến Socket đó
Dispose()	Giải phóng toàn bộ tài nguyên được sử dụng bởi 1 thể hiện hiện tại của lớp Socket
EndAccept(IASyncResult)	Sự bất đồng bộ chấp nhận 1 nỗ lực kết nối đến và tạo ra 1 Socket mới để xử lý giao tiếp máy chủ từ xa
EndConnect(IASyncResult)	Kết thúc 1 yêu cầu kết nối bất đồng bộ đang xử lý
EndReceive(IASyncResult)	Kết thúc trạng thái đọc bất đồng bộ đang được xử lý
EndSend(IASyncResult)	Kết thúc hành động gửi bất đồng bộ đang xử lý
Listen()	Đặt Socket trong trạng thái lắng nghe
RemoteEndpoint()	Trả về Endpoint từ xa

Lớp TcpClient

Để lập trình theo giao thức TCP,MS.NET cung cấp hai lớp có tên là TcpClient và TcpListener

Bảng 2.1.3d : Các thành phần của lớp TcpClient

Phương thức khởi tạo	**Mô tả**
TcpClient()	Tạo một đối tượng **TcpClient**. Chưa đặt thông số gì.
TcpClient(IPEndPoint)	Tạo một **TcpClient** và gắn cho nó một EndPoint cục bộ. (Gán địa chỉ máy cục bộ và số hiệu cổng để sử dụng trao đổi thông tin về sau)
TcpClient(String,Int32)	Tạo một đối tượng **TcpClient** và kết nối đến một máy có địa chỉ và số hiệu cổng được truyền vào. RemoteHost có thể là địa chỉ IP chuẩn hoặc tên máy.
Các thuộc tính	**Mô tả**
Available	Cho biết số byte đã nhận về từ mạng và có sẵn để đọc.
Client	Trả về Socket ứng với TCPClient hiện hành.
Connected	Trạng thái cho biết đã kết nối được đến Server hay chưa?
Các hàm thành phần	**Mô tả**
Close()	Giải phóng đối tượng **TcpClient** nhưng không đóng kết nối.

Connect(RemoteHost, RemotePort)	Kết nối đến một máy TCP khác có Tên và số hiệu cổng.
GetStream()	Trả về **NetworkStream** để từ đó giúp ta gửi hay nhận dữ liệu. (Thường làm tham số khi tạo **StreamReader** và StreamWriter để gửi và nhận dữ liệu dưới dạng xâu ký tự) . Khi đã gắn vào **StreamReader** và **StreamWriter** rồi thì ta có thể gửi và nhận dữ liệu thông qua các phương thức Readline, writeline tương ứng của các lớp này.

Lớp TcpListener

Là một lớp cho phép người lập trình có thể xây dựng các ứng dụng Server.Ứng dụng Server khác với ứng dụng Client ở chỗ nó luôn luôn lắng nghe và chấp nhận các kết nối khác đến từ Client

Bảng 2.1.3d : Các thành phần của lớp TcpListener

Phương thức khởi tạo	**Mô tả**
TcpListener (Int32)	Tạo một **TcpListener** và lắng nghe tại cổng chỉ định.
TcpListener (IPEndPoint)	Tạo một **TcpListener** với giá trị Endpoint truyền vào.
TcpListener(IPAddress,Int32)	Tạo một **TcpListener** và lắng nghe các kết nối đến tại địa chỉ IP và cổng chỉ định.
Phương thức	**Mô tả**
AcceptSocket( )	Chấp nhận một yêu cầu kết nối đang chờ.
AcceptTcpClient()	Chấp nhận một yêu cầu kết nối đang chờ. (Ứng dụng sẽ dừng tại lệnh này cho đến khi nào có một kết nối đến – "Blocking").
Pending()	Cho biết liệu có kết nối nào đang chờ đợi không
Start()	Bắt đầu lắng nghe các yêu cầu kết nối.
Stop()	Dừng việc nghe.

2.2 Sơ lược về luồng (Thread) trong .NET

Thread hay còn gọi là tiểu trình là khái niệm khá quen thuộc trong lập trình. Thread cho phép chương trình thực hiện đồng thời nhiều tác vụ, và giúp quá trình tương tác với người dùng không bị gián đoạn, lập trình song song và là kĩ thuật không thể thiếu trong các ứng dụng về mạng

Một luồng (**Thread)** là một chuỗi liên tiếp những sự thực thi trong chương trình. Trong một chương trình C#, việc thực thi bắt đầu bằng phương thức main() và tiếp tục cho đến khi kết thúc

hàm main(). Cấu trúc này rất hay cho những chương trình có một chuỗi xác định những nhiệm vụ liên tiếp.
Ở đây chúng ta sẽ chỉ đề cập tới lớp Thread
Lớp thread
Lớp sơ đẳng nhất trong tất cả các lớp thuộc Namespace System.Threading là lớp Thread. Lớp này tượng trưng cho một vỏ bọc hướng đối tượng bao quanh một lộ trình thi hành trong lòng một AppDomain nào đó.
Bao gồm một số hàm thực thi (cả static lẫn shared) cho phép bạn tạo mới từ luồng hiện hành cũng như cho Sleep,Stop hay Kill một luồng nào đó
Bảng 2.2.1 Các thành phần static của lớp Thread

Các thành phần Static	**Mô tả**
CurrentThread	Thuộc tính read-only này trả về một quy chiếu về luồng hiện đang chạy.
GetData()	Đi lấy vị trí từ slot được khai báo trên luồng hiện hành đối với domain hiện hành trong luồng.
SetData()	Cho đặt để trị lên slot được khai báo trên luồng hiện hành đối với domain hiện hành trong luồng
GetDomain() ***GetDomainID()***	Đi lấy một qui chiếu về AppDomain hiện hành (hoặc mã nhận diện ID của domain này) mà luồng hiện đang chạy trên đó.
Sleep()	Cho ngưng luồng hiện hành trong một thời gian nhất định được khai báo.

Bảng 2.2.2 Các thành viên cấp đối tượng của lớp Thread

Các lớp thành viên	**Mô tả**
IsAlive	Thuộc tính này trả về một trị boolean cho biết liệu xem luồng đã khởi đông hay chưa.
IsBackground	Đi lấy hoặc đặt để giá trị cho biết liệu xem luồng là một luồng nền hay không.
Name	Thuộc tính này cho phép bạn thiết lập một tên văn bản mang tính thân thiện đối với luồng.
Priority	Đi lấy hoặc đặt để ưu tiên của một luồng. Có thể được gán một trị lấy từ enumeration **ThreadPriority** (chẳng hạn **Normal, Lowest, Highest, BelowNormal, AboveNormal**).

ThreadState	Đi lấy hoặc đặt để tình trạng của luồng. Có thể được gán từ enumeration **ThreadState** (chẳng hạn **Unstarted, Running, WaitSleepJoin, Suspended, SuspendRequested, AbortRequested, Stopped**).
Interrup()	Cho ngưng chạy luồng hiện hành.
Join()	Yêu cầu luồng chờ đối với luồng bị ngưng chạy.
Resume()	Tiếp tục lại đối với một luồng bị ngưng chạy.
Start()	Cho bắt đầu thi hành luồng được khai báo bởi delegate ThreadStart.
Suspend()	Cho ngưng chạy một luồng. Nếu luồng đã bị ngưng rồi, một triệu gọi hàm **Suspend()** sẽ không có tác dụng.

CHƯƠNG 3 : HIỆN THỰC HÓA

1.Chuẩn bị:

a.Database

Do tính chất của 1 phần mềm Chat nên nhóm cần sử dụng 1 database lưu trữ. Mục đích của việc dùng database này nhằm để lưu trữ thông tin người dùng cũng như kiểm tra đăng nhập,đăng ký từ phía Client và lưu lại lịch sử Chat riêng tư giữa các Client với nhau

b.Client

Đây là phần mềm chính cho người dùng sử dụng.Bao gồm những tính năng như đăng ký,đăng nhập,chat public toàn bộ các client có trong server,chat riêng giữa các client khác nhau,tham gia nhóm chat sẵn có

c.Server

Là cốt lõi của phần mềm Lan Chat.Server có nhiệm vụ chấp nhận các kết nối khác từ Client cũng như lắng nghe những hành động từ phía các Client.Tùy theo hành động mà sẽ điều hướng thích hợp,đưa ra các bước chỉ thị tiếp theo thỏa yêu cầu từ phía Client đưa ra

d.Đa ngôn ngữ

Đây là tính năng thêm của phần mềm.Hỗ trợ nhiều ngôn ngữ khác nhau.Ở đây nhóm phát triển 2 ngôn ngữ chính đó là Tiếng Việt và Tiếng Anh cho người dùng

2.Hiện thực hóa phần mềm

2.1.Cơ sở dữ liệu

2.1.1 Mô hình dữ liệu

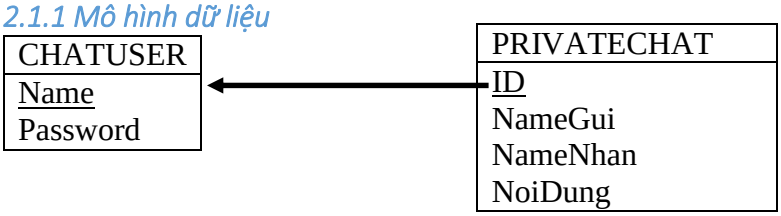


Table CHATUSER

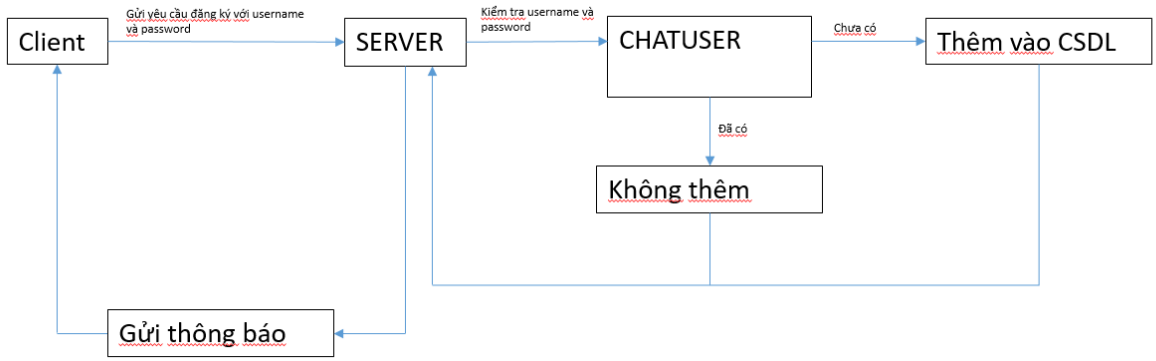
STT	Tên	Kiểu dữ liệu	Diễn giải
1	Name	Nvarchar(100)	Tên đăng nhập của người dùng,đồng thời cũng là khóa chính
2	Password	Nvarchar(100)	Mật khẩu người dùng

Table PRIVATECHAT

STT	Tên	Kiểu dữ liệu	Diễn giải
1	ID	int	Khóa chính,tự động tăng
2	NameGui	Nvarchar(100)	Người gửi tin nhắn
3	NameNhan	Nvarchar(100)	Người nhận tin nhắn
4	NoiDung	Nvarchar(1000)	Nội dung tin nhắn gửi

2.1.2 Phân tích thành phần xử lý

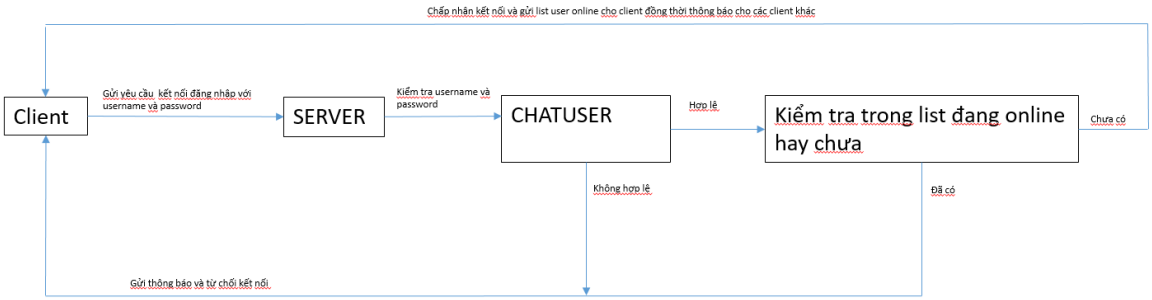
a.Xử lý đăng ký



Mô tả:

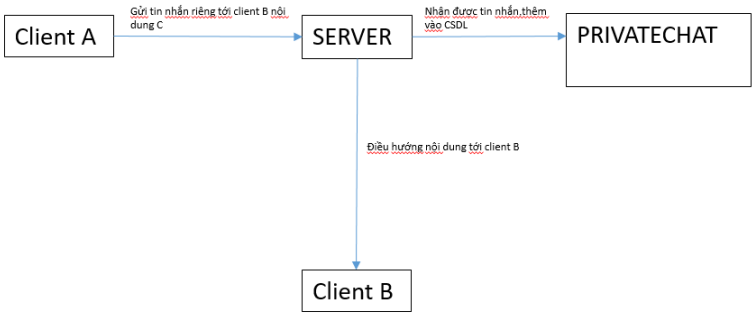
Client sẽ gửi thông điệp yêu cầu đăng ký với username và password.Server nhận được sẽ kiểm tra username đã tồn tại trong CSDL hay chưa ,nếu chưa thì thêm còn nếu đã tồn tại thì không thêm.Sau đó trả về thông điệp cho Client biết là việc đăng ký có thành công hay không

b.Xử lý đăng nhập



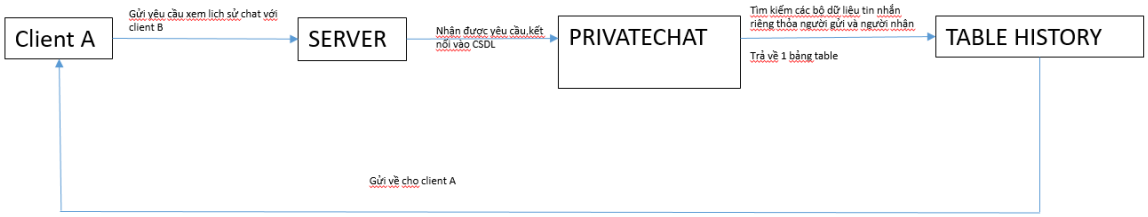
Mô tả
Client gửi yêu cầu kết nối với username và password.Server nhận được sẽ kiểm tra username và password có hợp lệ hay không.Sau đó tiếp tục kiểm tra xem trong list client đã tồn tại user đang online này chưa.Nếu đăng nhập thành công,Server sẽ lấy list User online gửi tới cho client và thông báo cho các client khác đã có thêm 1 client đăng nhập vào phòng chat

c.Xử lý tin nhắn riêng giữa 2 client



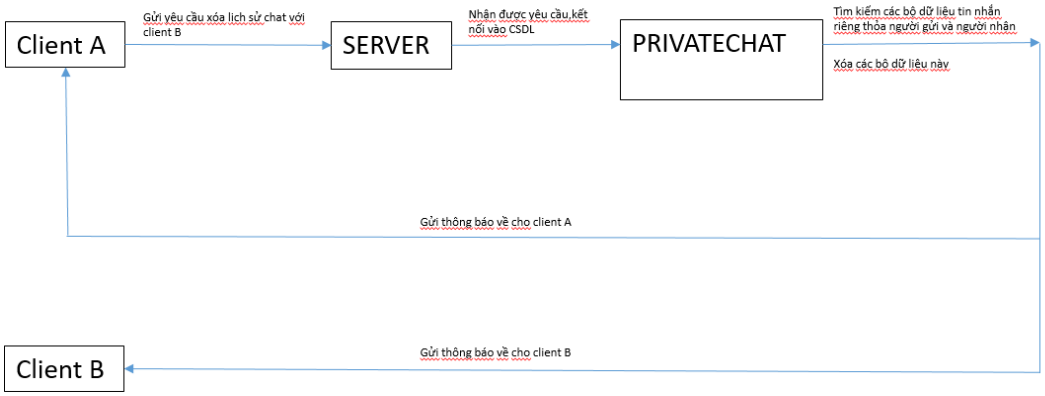
Mô tả
Client gửi thông điệp tin nhắn riêng với nội dung :người gửi|người nhận|nội dung tin nhắn.Server nhận được sẽ lưu xuống CSDL tương ứng

d.Xử lý yêu cầu xem lịch sử tin nhắn



Mô tả
Client gửi thông điệp yêu cầu xem tin nhắn riêng của mình với client khác.Server nhận được yêu cầu sẽ tìm kiếm trong CSDL những bộ dữ liệu thỏa điều kiện người gửi và người nhận của client.Sau đó trả về 1 table tương ứng,gửi lại cho client

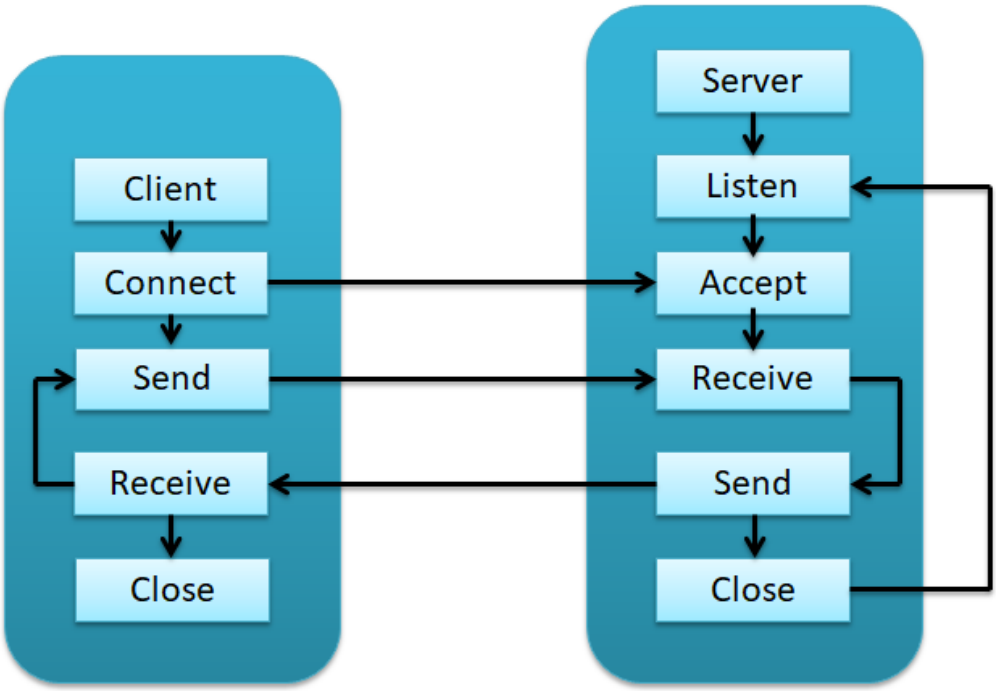
e.Xử lý yêu cầu xóa lịch sử tin nhắn



Mô tả
Client gửi thông điệp yêu cầu xóa tin nhắn riêng của mình với client khác.Server nhận được sẽ thực hiện hành động xóa các bộ dữ liệu thỏa điều kiện người gửi và người nhận của client trong CSDL.Sau đó thông báo việc xóa lịch sử thành công cho client gửi yêu cầu và người được client gửi yêu cầu nhắn tin tới.

2.2 Server và Client

Mô hình ứng dụng

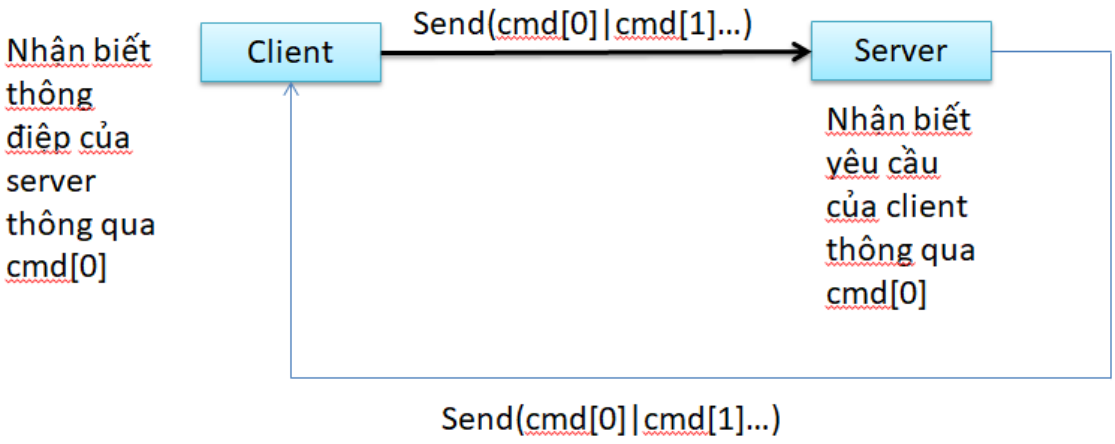


Mô hình ứng dụng

Cách hoạt động :

Trong quá trình lắng nghe.Server/Client sẽ nhận được chuỗi các dữ liệu dưới dạng tin nhắn (cmd[0]|cmd[1]…)

Tùy theo cmd[0] mà nhận biết được các yêu cầu và đưa ra các hành vi thích hợp



Mô hình hoạt động

2.2.1.Server

Class User.cs : xây dựng 1 class mới dựa trên những thuộc tính và phương thức sẵn có của lớp TcpClient

Thuộc tính

```
        private TcpClient client;
        private byte[] readBuffer = new byte[READ_BUFFER_SIZE];
        private string strName;
        const int READ_BUFFER_SIZE = 255;
        // Thuộc tính tên xác định người dùng kết nối
        public string Name
        {
            get
            {
                return strName;
            }
            set
            {
                strName = value;
            }
        }
```

Sự kiện

```
public delegate void LineReceive(User sender, string Data);
public event LineReceive LineReceived;
```

Phương thức

```
//Hàm khởi tạo thread đọc dữ liệu
        public User(TcpClient client)
        {
            this.client = client;
            //Bắt đầu thread đọc bất đồng bộ và dữ liệu sẽ được lưu xuống readbuffer
            this.client.GetStream().BeginRead(readBuffer, 0, READ_BUFFER_SIZE, new
AsyncCallback(StreamReceiver), null);
        }
```

```
        //sử dụng StreamReader và StreamWriter để gửi nhận dữ liệu mà không cần bước
chuyển đổi qua lại mảng byte
        public void SendData(string Data)
        {
            // đảm bảo không còn luồng nào khác cố gắng sử dụng stream tại cùng thời điểm
            lock (client.GetStream())
            {
                StreamWriter writer = new StreamWriter(client.GetStream());
                writer.Write(Data + (char)13 + (char)10);
                //Đảm bảo dữ liệu được gửi đi
                //Giải phóng tài nguyên
                writer.Flush();
            }
        }

        //Hàm gọi lại cho TcpClient.GetStream.Begin.
        //Bắt đầu đọc không đồng bộ từ stream
        private void StreamReceiver(IAsyncResult ar)
        {
            int BytesRead;
            string strMessage;

            try
            {
                // Ensure that no other threads try to use the stream at the same time.
                lock (client.GetStream())
                {
                    // Finish asynchronous read into readBuffer and get number of bytes
read.
                    BytesRead = client.GetStream().EndRead(ar);
                }
                // Convert the byte array the message was saved into, minus one for the
                // Chr(13).
                strMessage = Encoding.UTF8.GetString(readBuffer, 0, BytesRead - 1);
                LineReceived(this, strMessage);
                // Ensure that no other threads try to use the stream at the same time.
                lock (client.GetStream())
                {
                    // Start a new asynchronous read into readBuffer.
                    client.GetStream().BeginRead(readBuffer, 0, READ_BUFFER_SIZE, new
AsyncCallback(StreamReceiver), null);
                }
            }
            catch (Exception e)
            {
            }
        }
```

Class Server : là class thực hiện các thao tác lên CSDL

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.SqlClient;
using System.IO;
using System.Linq;
using System.Text;
using System.Windows.Forms;
```

```
namespace Server
{
    class Server
    {
        SqlConnection connect;
        string source = @"Data Source=REBORN;Initial Catalog=Chat;Integrated
Security=True";


        //Kiem tra xem User da ton tai hay chua
        public bool CheckUserExist(string name)
        {
            DataTable data = new DataTable();
            bool x;
            connect = new SqlConnection();
            connect.ConnectionString = source;
            connect.Open();
            SqlCommand cmd = new SqlCommand("CheckThemUser", connect);
            cmd.CommandType = System.Data.CommandType.StoredProcedure;
            SqlParameter p = new SqlParameter("@Name", name);
            cmd.Parameters.Add(p);
            SqlDataAdapter adapter = new SqlDataAdapter(cmd);
            adapter.Fill(data);
            if (data.Rows.Count > 0)
            {
                x = true;
            }
            else x = false;
            connect.Close();
            return x;
        }

        //Them moi User
        public void AddUser(string name,string pass)
        {

            connect = new SqlConnection();
            connect.ConnectionString = source;
            connect.Open();
            SqlCommand cmd = new SqlCommand("ThemUser", connect);
            cmd.CommandType = System.Data.CommandType.StoredProcedure;
            SqlParameter p = new SqlParameter("@Name", name);
            cmd.Parameters.Add(p);
            p = new SqlParameter("@Pass", pass);
            cmd.Parameters.Add(p);
            cmd.ExecuteNonQuery();
            connect.Close();

        }

        //Kiem tra dang nhap co dung
        public bool CheckLogin(string name,string pass)
        {
            DataTable data = new DataTable();
            bool x;
            connect = new SqlConnection();
            connect.ConnectionString = source;
```

```
            connect.Open();
            SqlCommand cmd = new SqlCommand("CheckLogin", connect);
            cmd.CommandType = System.Data.CommandType.StoredProcedure;
            SqlParameter p = new SqlParameter("@Name", name);
            cmd.Parameters.Add(p);
            p = new SqlParameter("@Pass", pass);
            cmd.Parameters.Add(p);
            SqlDataAdapter adapter = new SqlDataAdapter(cmd);
            adapter.Fill(data);
            if (data.Rows.Count > 0)
            {
                x= true;
            }
            else x= false;
            connect.Close();
            return x;
        }

        //Them vao lich su chat
        public void AddHistory(string from,string to,string mess)
        {
            connect = new SqlConnection();
            connect.ConnectionString = source;
            connect.Open();
            SqlCommand cmd = new SqlCommand("InsertPrivateChat", connect);
            cmd.CommandType = System.Data.CommandType.StoredProcedure;
            SqlParameter p = new SqlParameter("@From", from);
            cmd.Parameters.Add(p);
            p = new SqlParameter("@To",to);
            cmd.Parameters.Add(p);
            p = new SqlParameter("@Mess", mess);
            cmd.Parameters.Add(p);
            cmd.ExecuteNonQuery();
            connect.Close();
        }
        //Tra ve lich su chat theo yeu cau cua client
        public DataTable getHistory(string from,string to)
        {
            DataTable datatable=new DataTable();
            connect = new SqlConnection();
            connect.ConnectionString = source;
            connect.Open();
            SqlCommand cmd = new SqlCommand("History", connect);
            cmd.CommandType = System.Data.CommandType.StoredProcedure;
            SqlParameter p = new SqlParameter("@From", from);
            cmd.Parameters.Add(p);
            p = new SqlParameter("@To", to);
            cmd.Parameters.Add(p);
            SqlDataAdapter adapter = new SqlDataAdapter(cmd);
            adapter.Fill(datatable);
            connect.Close();
            return datatable;
        }
        //Xoa lich su chat
        public void DeleteHistory(string from,string to)
        {
            connect = new SqlConnection();
            connect.ConnectionString = source;
```

```
            connect.Open();
            SqlCommand cmd = new SqlCommand("DeleteHistory", connect);
            cmd.CommandType = System.Data.CommandType.StoredProcedure;
            SqlParameter p = new SqlParameter("@From", from);
            cmd.Parameters.Add(p);
            p = new SqlParameter("@To", to);
            cmd.Parameters.Add(p);
            cmd.ExecuteNonQuery();
            connect.Close();
        }
    }
}
```

Form Main

Thuộc tính

```
        DataTable table = new DataTable();
        Server sv = new Server();
        bool flag = true; //Cờ hiệu ngôn ngữ
        MultiLang ml = new MultiLang();//Đa ngôn ngữ
        const int PORT_NUM = 2018;
        private Hashtable clients = new Hashtable();
        User client;
        private TcpListener listener;
        private Thread listenerThread;
        private PrivateChat pChat;
        private List<PrivateChat> listpChat = new List<PrivateChat>(); //Danh sách form
chat riêng với client
```

Hàm Load form có nhiệm vụ bắt đầu 1 luồng Thread chấp nhận nỗ lực kết nối từ client và lắng nghe thông điệp nhận từ client

```
  private void Main_Load(object sender, EventArgs e)
        {
            this.Icon = Properties.Resources.Home;
            label1.Text = "SERVER UIT LAN CHAT";
            label2.Text = "ChatBox";
            menuStrip1.BackColor = Color.Transparent;
            //Tạo luồng lắng nghe và bắt đầu luồng lắng nghe
            listenerThread = new Thread(new ThreadStart(DoListen));
            listenerThread.Start();
        }


        //Hàm lắng nghe kết nối
        void DoListen()
        {
            try
            {
                // Lắng nghe kết nối ở địa chỉ IP với port chỉ định
                listener = new TcpListener(System.Net.IPAddress.Any, PORT_NUM);
                listener.Start();
                do
                {
                    // Create a new user connection using TcpClient returned by
                    // TcpListener.AcceptTcpClient()
                    client = new User(listener.AcceptTcpClient());
                    // Create an event handler to allow the UserConnection to communicate
                    // with the window.
```

```
                    client.LineReceived += new LineReceive(OnLineReceived);
                    //AddHandler client.LineReceived, AddressOf OnLineReceived;
                } while (true);
            }
            catch
            {
            }
        }
```

Thông điệp nhận được từ client tùy theo yêu cầu mà đưa ra các điều hướng hợp lý
Các yêu cầu từ client gửi đến server sẽ được đáp ứng

```
       private void OnLineReceived(User sender, string data)
        {
            string[] cmd;
            // Message parts are divided by "|"  Break the string into an array
accordingly.
            cmd = data.Split((char)124);

            // dataArray(0) is the command.
            switch (cmd[0])
            {
                //Đăng ký tài khoản
                case "SIGNUP":
                    if (sv.CheckUserExist(cmd[1]) == true)
                    {
                        ReplyToSender("EXISTNAME|", sender);
                    }
                    else
                    {
                        sv.AddUser(cmd[1], cmd[2]);
                        ReplyToSender("SUCCESS|", sender);
                    }

                    break;
                //Đăng nhập
                case "CONNECT":
                    ConnectUser(cmd[1],cmd[2], sender);
                    break;
                //Chat public
                case "MESSAGE":
                    string v = cmd[1] + " : " + cmd[2];
                    UpdateMess(v);
                    Send("REFRESHCHAT|" + v);
                    break;
                //Chat riêng giữa server và client
                case "pMessage":
                    ChatPrivateWithUser(cmd);
                    break;
                //Nhận yêu cầu chat riêng tư giữa client A và gửi yêu cầu này đến client
B
                case "PRIVATECHAT":
                    Send("FROMTO|" + cmd[1] + "|" + cmd[2]);
                    break;
                //server da nhan duoc tin nhan rieng giua 2 nguoi va bat dau dieu huong
                case "PrivateFromTo":
                    sv.AddHistory(cmd[1], cmd[2], cmd[3]);
                    Send("PrivateMess|" + cmd[1] + "|" + cmd[2] + "|" + cmd[3]);
                    break;
```

```
                //nhận được yêu cầu join group chat
                case "JOINGROUP":
                    SendToClients("JOINSUCCESS|" + cmd[1] + "|" + cmd[2] + " vừa vào
phòng chat "+cmd[1],sender);
                    break;
                //Điều hướng tin nhắn group chat đến các client tham gia trong group chat
phù hợp
                case "GroupChat":
                    SendToClients("GroupMess|" + cmd[1] + "|" + cmd[2], sender);
                    break;
                //nhận được yêu cầu thoát group chat và thông báo cho các client còn lại
                case "QuitGroupChat":
                    SendToClients("GroupMess|" + cmd[1] + "|" + cmd[2]+cmd[3], sender);
                    break;
                //nhận được yêu cầu xem lịch sử chat giữa 2 client từ client A và trả lời
lại theo yêu cầu của client A
                case "History":
                    ReplyHistoryChat(cmd, sender);
                    break;
                //nhận yêu cầu xóa lịch sử chat.server thực hiện và gửi thông báo đến 2
client liên quan
                case "DeleteHistory":
                    sv.DeleteHistory(cmd[1], cmd[2]);
                    Send("Delete|" + cmd[1] +"|"+ cmd[2]);
                    break;
                //ngắt kết nối với client
                case "DISCONNECT":
                    DisconnectUser(sender);
                    break;
            }
        }
```

Các hàm liên quan để gửi lại cho client

```
        //Tra loi client gui yeu cau
        private void ReplyToSender(string strMessage, User sender)
        {
            sender.SendData(strMessage);
        }


        //Gui cho toan bo client ngoai tru client gui yeu cau
        private void SendToClients(string strMessage, User sender)
        {
            User client;
            // All entries in the clients Hashtable are UserConnection so it is possible
            // to assign it safely.
            foreach (DictionaryEntry entry in clients)
            {
                client = (User)entry.Value;
                // Exclude the sender.
                if (client.Name != sender.Name)
                {
                    client.SendData(strMessage);
                }
            }
        }
```

```
        //Gui cho tat ca cac client
        private void Send(string strMessage)
        {
            User client;
            // All entries in the clients Hashtable are UserConnection so it is possible
            // to assign it safely.
            foreach (DictionaryEntry entry in clients)
            {
                client = (User)entry.Value;
                client.SendData(strMessage);
            }
        }
```

Sự kiện đăng nhập kết nối

Kiểm tra xem tên đăng nhập có hợp lệ hay không.Nếu có sẽ kiểm tra tiếp tài khoản này có đang được kết nối hay không.Nếu có thì từ chối.Ngược lại thì chấp nhận kết nối

Sau khi đăng nhập server nhận được sẽ them vào list danh sách online và gửi lại cho client

```
   //Ham kiem tra xem da co ten dang nhap hay chua va tra lai ket qua cho client yeu cau
dang nhap
        private void ConnectUser(string userName,string pass, User sender)
        {
            if (clients.Contains(userName))
            {
                ReplyToSender("REFUSE", sender);
            }
            else
            {
                bool x = sv.CheckLogin(userName, pass);
                if (x == true)
                {
                    sender.Name = userName;
                    chatBox.Text += userName + " vừa vào phòng chat \r\n";
                    clients.Add(userName, sender);
                    Add(userName);
                    // Send a JOIN to sender, and notify all other clients that sender
joined
                    ReplyToSender("JOIN", sender);
                    GuiUserOnline();
                    SendToClients("NOTI|" + sender.Name + " vừa vào phòng chat ",
sender);
                }
                if (x == false)
                {
                    ReplyToSender("WRONG", sender);
                }
            }
        }
```

Các hàm cần thiết để gửi list online

```
//Them vao listview
        private void Add(string name)
        {
            ListViewItem item = new ListViewItem(name);
            item.SubItems.Add("online"); // status
            item.Tag = client;
            userlist.Items.Add(item);
        }
/// <summary>
```

```
        ///Gui List User Online
        /// </summary>
        void GuiUserOnline()
        {
            string u = string.Empty;
            for (int j = 0; j < userlist.Items.Count; j++)
            {
                u += userlist.Items[j].SubItems[0].Text + "|";
            }
            Send("USERS|" + u.TrimEnd('|'));
        }
```

Client yêu cầu xem lịch sử chat

```
//Nhan yeu cau xem lich su chat cua client
        void ReplyHistoryChat(string []cmd,User sender)
        {
            string Gui = cmd[1];
            string Nhan = cmd[2];
            table = sv.getHistory(Gui, Nhan);
            string temp = string.Empty;
            for (int i = 0; i < table.Rows.Count; i++)
            {
                string send = table.Rows[i]["NameGui"].ToString();
                string message = table.Rows[i]["NoiDung"].ToString();
                temp += send + " : " + message + "|";
            }
            ReplyToSender("DETAIL|" + Nhan + "|" + temp.TrimEnd('|'), sender);
        }
```

Server chat riêng với client

Server sẽ gửi yêu cầu nhắn tin riêng tới client.Trong quá trình nhắn tin,nếu server đã đóng form PrivateChat nhưng vẫn nhận được tin nhắn riêng thì sẽ tạo form PrivateChat mới để nhận tin nhắn

```
//Gửi yêu cầu chat riêng với client
        private void privateChatToolStripMenuItem_Click(object sender, EventArgs e)
        {
            foreach (var client in from ListViewItem item in userlist.SelectedItems
select (User)item.Tag)
            {
                string to = userlist.SelectedItems[0].SubItems[0].Text;
                var value = listpChat.SingleOrDefault(r => r.getTo() == to);
                if (value == null) //Neu chua co form
                {
                    client.SendData("Chat|" + to);
                    pChat = new PrivateChat(this);
                    pChat.SetTo(to);
                    pChat.Show();
                    listpChat.Add(pChat);
                }

            }
        }

//Chat riêng tư với client
        private void ChatPrivateWithUser(string []cmd)
        {
            var value = listpChat.SingleOrDefault(r => r.getTo() == cmd[1]);
            if (value != null) //Neu trong list ton tai form
```

```
                value.chatBox.Text += cmd[1] + " : " + cmd[2] + "\r\n";
            else
            {
                this.Invoke(() =>
                {
                    pChat = new PrivateChat(this);
                    pChat.SetTo(cmd[1]);
                    pChat.Show();
                    listpChat.Add(pChat);
                    pChat.chatBox.Text += cmd[1] + " : " + cmd[2] + "\r\n";
                });

            }
        }
```

Sự kiện ngắt kết nối với client (client thoát khỏi phòng chat)

```
        void DisconnectUser(User sender)
        {
            clients.Remove(sender.Name); // Xoa khoi bang bam client
            for (int i = 0; i < userlist.Items.Count; i++)
            {
                var item = userlist.Items[i].Tag as User;
                if (item.Name == sender.Name) //Tim trong list view client vua moi thoat
de xoa khoi list view
                {
                    if (userlist.Items[i].SubItems[0].Text != string.Empty)
                    {
                        string s = userlist.Items[i].SubItems[0].Text + " đã thoát ra";
                        UpdateMess(s);
                        Send("REFRESHCHAT|" + s); // thong bao cho cac client con lai
                    }
                    userlist.Items.RemoveAt(i); // Xoa khoi list view
                    //Gui lai list online user cho cac client con lai
                    GuiUserOnline();
                }
            }
        }
```

Server chủ động disconnect Client

```
private void disconnectToolStripMenuItem_Click(object sender, EventArgs e)
        {
            foreach (var client in from ListViewItem item in userlist.SelectedItems
select (User)item.Tag)
            {
                client.SendData("BAN");
            }
        }
```

2.2.2 Client

Thuộc tính

```
#region Thuoc tinh
        bool flag = true; //Cờ hiệu ngôn ngữ
        MultiLang ml = new MultiLang();
```

```
        private TcpClient client ;
        private PrivateChat pChat;
        private PrivateUser uChat;
        private GroupChat gChat;
        private List<GroupChat> listgChat = new List<GroupChat>();
        private List<PrivateUser> listuChat = new List<PrivateUser>();
        private List<PrivateChat> listpChat = new List<PrivateChat>();
        private bool checkuChat;//gán cờ nếu để kiểm tra xem uChat đã đóng hay chưa
        private bool Check;//Gán cờ để nhận biết xem pChat đã đóng hay chưa
        const int PORT_NUM = 2018;
        const int READ_BUFFER_SIZE = 255;
        string ip="127.0.0.1";
        string user;
        private byte[] readBuffer = new byte[READ_BUFFER_SIZE];
#endregion
```

Hàm Connect được gọi trong sự kiện Load có tác dụng kết nối đến server

```
private void Connect()
        {


            try
            {
                this.Hide();
                Login frmLogin = new Login();
                frmLogin.ShowDialog(this);
                ip = frmLogin.IP();
                // The TcpClient is a subclass of Socket, providing higher level
                // functionality like streaming.
                // TcpClient sẽ kết nối đến server có ip=ip và port num chỉ định
                client = new TcpClient(ip,PORT_NUM);

                // Bắt đầu luồng Receive để tránh tình trạng lag giao diện người dùng
                client.GetStream().BeginRead(readBuffer, 0, READ_BUFFER_SIZE, new
AsyncCallback(Receive), null);
                // Make sure the window is showing before popping up connection dialog.

                //Gửi tên tài khoản đăng nhập và mật khẩu đến server để kiểm tra xem có
hợp lệ
                SendData("CONNECT|" + frmLogin.textBox1.Text + "|" +
frmLogin.textBox3.Text);
                //Gán name
                user = frmLogin.textBox1.Text;
                labelName.Text = " : "+user;
                frmLogin.Dispose();
                this.Show();
            }
            catch (Exception Ex)
            {
                MessageBox.Show("Không thể kết nối với máy chủ. Vui lòng thực hiện lại
việc đăng nhập.",
                       this.Text, MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                this.Dispose();
            }
        }
```

Gọi hàm kết nối

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

```
            menuStrip1.BackColor = Color.Transparent;
            Connect();

        }
```

Khi đã kết nối thành công đến server,ta đặt luồng lắng nghe cho client

```
private void Receive(IAsyncResult ar)
        {
            int BytesRead;
            string strMessage;

            try
            {
                //Kết thúc đọc bất đồng bộ vào readBuffer và trả về số byte đã đọc
                BytesRead = client.GetStream().EndRead(ar);
                if (BytesRead < 1)
                {
                    return;
                }
                // Convert the byte array the message was saved into, minus two for the
                // Chr(13) and Chr(10)
                strMessage = Encoding.UTF8.GetString(readBuffer, 0, BytesRead - 2);
                ProcessCommands(strMessage);
                //Bắt đầu đọc bất đồng bộ vào readBuffer
                client.GetStream().BeginRead(readBuffer, 0, READ_BUFFER_SIZE, new
AsyncCallback(Receive), null);

            }
            catch (Exception e)
            {
                MessageBox.Show("Server đã đóng!");

            }
        }
```

Tùy theo thông điệp mà server trả về sẽ đưa ra các hành vi thích hợp

```
private void ProcessCommands(string strMessage)
        {
            string[] cmd;
            // Message parts are divided by "|"  Break the string into an array
accordingly.
            cmd = strMessage.Split('|');
            // dataArray(0) is the command.
            switch (cmd[0])
            {
                //Đăng nhập sai tên tài khoản hoặc mật khẩu
                case "WRONG":
                    MessageBox.Show("Sai tên tài khoản hoặc mật khẩu!");
                    Application.Exit();
                    break;
                //Server chấp nhận kết nối
                case "JOIN":
                    // Server acknowledged login.
                    DisplayText(user + " vừa vào phòng chat");
                    break;
                //Tài khoản này đang được đăng nhập nên server từ chối
                case "REFUSE":
                    MessageBox.Show("Đã có người dùng tên tài khoản này!");
                    Application.Exit();
```

```
                    break;
                //Nhận thông báo có client khác đã đăng nhập vào phòng chat
                case "NOTI":
                    DisplayText(cmd[1]);
                    break;
                //Nhận được chuỗi list người online,tách chuỗi để show lên listview
Userlist
                case "USERS":
                    //Server sent Online Users List.
                    LoadUser(cmd);
                    break;
                //Chat public
                case "REFRESHCHAT":
                    ChatPublic(cmd);
                    break;
                // nhan duoc yeu cau nhan tin rieng tu server
                case "Chat":
                    RequestPrivate(cmd);
                    break;
                case "pMessage": //Nhan duoc tin nhan private cua server
                    PrivateMessFromServer(cmd);
                    break;
                //Tin nhan public tu server
                case "BROAD":
                    DisplayText(cmd[1]);
                    break;
                case "FROMTO":
                    RequestPrivateClient(cmd);
                    break;
                case "PrivateMess": // nhan duoc tin nhan rieng tu giua 2 User do server
gui ve va bat dau xu ly
                    PrivateMess(cmd);
                    break;
                //xem lich su chat
                case "DETAIL":
                    History(cmd);
                    break;
                //Nhan thong bao da xoa lich su chat
                case "Delete":
                    DelHistory(cmd);
                    break;
                //Thong bao join group chat thanh cong
                case "JOINSUCCESS":
                    GroupChat(cmd);
                    break;
                case "GroupMess":
                    GroupMess(cmd);
                    break;
                //Server disconnect client
                case "BAN": // Server disconnect client
                    Application.Exit();
                    break;
            }
        }
```

Load người dùng online.

```
//Load người dùng online
        void LoadUser(string []cmd)
```

```
        {
            UserList.Items.Clear();
            for (int i = 1; i < cmd.Length; i++)
            {
                if (cmd[i] != "USERS" | cmd[i] != "online")
                {
                    UserList.Items.Add(cmd[i]);
                }
            }
        }
```

Các thao tác của client

```
  //Chat public
        void ChatPublic(string[] cmd)
        {
            string temp = cmd[1].Split(':')[0];
            string from = temp.Substring(0, temp.Length - 1);
            if (user != from)
                DisplayText(cmd[1]);
        }
        //Nhan duoc yeu cau chat rieng tu server
        void RequestPrivate(string[] cmd)
        {
            if (listpChat.Count == 0)
            {
                this.Invoke(() =>
                {
                    pChat = new PrivateChat(this);
                    pChat.SetFrom(cmd[1]);
                    pChat.Show();
                    listpChat.Add(pChat);
                });
            }

        }
        //Nhan tin nhan rieng tu server
        void PrivateMessFromServer(string[] cmd)
        {
            if (listpChat.Count == 0) // lúc này chưa có form pChat vì form pChat trước
đã bị đóng lại
             {
                        this.Invoke(() =>
                        {
                            pChat = new PrivateChat(this);
                            //Check = false;
                            pChat.SetFrom(cmd[2]);
                            pChat.Show();
                            listpChat.Add(pChat);
                            pChat.richTextBox1.Text += "Server : " + cmd[1] + "\r\n";
                        });
             }
            else
                    {
                        var p = listpChat.SingleOrDefault(r => r.getFrom() == cmd[2]);
                        if (p != null)
                            p.richTextBox1.Text += "Server : " + cmd[1] + "\r\n";
                    }
        }
```

```
        //Nhan yeu cau chat rieng tu tu client khac
        void RequestPrivateClient(string[] cmd)
        {
            string fromUser = cmd[1];
            string toUser = cmd[2];
            if (toUser == user)
            {

                var value = listuChat.SingleOrDefault(r => r.From() == cmd[2]); // kiem
tra xem co ton tai form nao chua
                if (value == null)
                {
                    this.Invoke(() =>
                    {
                        uChat = new PrivateUser(this);
                        uChat.SetFrom(toUser);
                        uChat.SetTo(fromUser);
                        uChat.Text = "From :" + toUser + ":To:" + fromUser;
                        uChat.Show();
                        listuChat.Add(uChat);
                    });
                }
                else value.Show();

            }
        }

        //Tin nhan rieng giua 2 client
        void PrivateMess(string[] cmd)
        {
            string pfrom = cmd[1]; //gui tu
            string pto = cmd[2]; // gui den

            if (pto == user) //kiểm tra xem có đúng là tin nhắn gửi tới mình hay không
            {
                var value = listuChat.SingleOrDefault(r => r.To() == cmd[1]);
                if (value != null)
                    value.richTextBox1.Text += cmd[1] + " : " + cmd[3] + "\r\n";
                else
                {
                    this.Invoke(() =>
                    {
                        uChat = new PrivateUser(this);
                        uChat.SetFrom(pto);
                        uChat.SetTo(pfrom);
                        uChat.Text = "From :" + pto + ":To:" + pfrom;
                        uChat.Show();
                        checkuChat = false;
                        listuChat.Add(uChat);
                        uChat.richTextBox1.Text += cmd[1] + " : " + cmd[3] + "\r\n";

                    });
                }
                //}

            }
        }
```

```
        //Xem lich su chat
        void History(string[] cmd)
        {
            string receive = cmd[1];
            string mess = string.Empty;
            labelHistory.Text = receive;
            txtHistory.Text = string.Empty;
            for (int i = 2; i < cmd.Length; i++)
            {
                txtHistory.Text += cmd[i] + "\r\n";
            }
        }


        //Xoa lich su chat
        void DelHistory(string []cmd)
        {
            if (user == cmd[1] || user == cmd[2])
                MessageBox.Show("Đã xóa cuộc trò chuyện giữa " + cmd[1] + " và " +
cmd[2], "Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Information);
        }


        //Tham gia chat nhom thanh cong
        void GroupChat(string[] cmd)
        {
            var values = listgChat.SingleOrDefault(r => r.getName() == cmd[1]);
            if (values != null)
            {
                values.gchatBox.Text += cmd[2] + "\r\n";
            }
        }

        //Tin nhan chat nhom
        void GroupMess(string[] cmd)
        {
            var y = listgChat.SingleOrDefault(r => r.getName() == cmd[1]);
            if (y != null)
                y.gchatBox.Text += cmd[2] + "\r\n";
        }
```

Form PrivateChat có chức năng chat riêng với Server

```
private void PrivateChat_Load(object sender, EventArgs e)
        {
            Text = "Private with Server";
            wp.URL = Application.StartupPath + @"\Sounds\ding.mp3";
            wp.controls.play();
        }


        private void PrivateChat_FormClosing(object sender, FormClosingEventArgs e)
        {
            pChat.RemovePrivate(this);
        }
```

```
        private void textBox1_KeyDown(object sender, KeyEventArgs e)
        {
            if (textBox1.Text != string.Empty)
            {
                if (e.KeyCode == Keys.Enter)
                {
                    e.SuppressKeyPress = true;
                    button1.PerformClick();
                }
            }
        }

        private void button1_Click(object sender, EventArgs e)
        {
            if (textBox1.Text != string.Empty)
            {
                pChat.SendData("pMessage|" + from + "|" + textBox1.Text);

                richTextBox1.Text += from + " : " + textBox1.Text + "\r\n";
                textBox1.Text = string.Empty;
            }
        }

        private void richTextBox1_LinkClicked(object sender, LinkClickedEventArgs e)
        {
            Process p = Process.Start("Chrome.exe", e.LinkText);
        }
```

Form PrivateUser có chức năng chat riêng với client khác

```
        private void PrivateChat_Load(object sender, EventArgs e)
        {
            Text = "Private with Server";
            wp.URL = Application.StartupPath + @"\Sounds\ding.mp3";
            wp.controls.play();
        }


        private void PrivateChat_FormClosing(object sender, FormClosingEventArgs e)
        {
            pChat.RemovePrivate(this);
        }

        private void textBox1_KeyDown(object sender, KeyEventArgs e)
        {
            if (textBox1.Text != string.Empty)
            {
                if (e.KeyCode == Keys.Enter)
                {
                    e.SuppressKeyPress = true;
                    button1.PerformClick();
                }
            }
        }

        private void button1_Click(object sender, EventArgs e)
        {
            if (textBox1.Text != string.Empty)
            {
```

```
                pChat.SendData("pMessage|" + from + "|" + textBox1.Text);

                richTextBox1.Text += from + " : " + textBox1.Text + "\r\n";
                textBox1.Text = string.Empty;
            }
        }

        private void richTextBox1_LinkClicked(object sender, LinkClickedEventArgs e)
        {
            Process p = Process.Start("Chrome.exe", e.LinkText);
        }
```

Form GroupChat có chức năng gửi tin nhắn chat nhóm

```
    private void button1_Click(object sender, EventArgs e)
        {
            if(txtBox.Text!=string.Empty)
            {
                form.SendData("GroupChat|" + name + "|" + from + " : " + txtBox.Text);
                gchatBox.Text += from + " : " + txtBox.Text+"\r\n";
                txtBox.Text = string.Empty;
            }
        }

        public void setFrom(string s)
        {
            from = s;
        }

        public void setName(string s)
        {
            name = s;
        }

        public string getName()
        {
            return name;
        }

        private void GroupChat_FormClosing(object sender, FormClosingEventArgs e)
        {
            form.SendData("QuitGroupChat|" + name + "|" + from + "|"+" đã thoát khỏi
phòng chat "+name );
            form.RemoveGroup(this);
        }

        private void txtBox_KeyDown(object sender, KeyEventArgs e)
        {
            if(txtBox.Text!=string.Empty)
            {
                if(e.KeyCode==Keys.Enter)
                {
                    e.SuppressKeyPress = true;
                    btnSend.PerformClick();
                }
            }
        }
```

2.3 Đa ngôn ngữ

Ở đây chúng ta dung cách đọc file xml.Phương pháp này thuận tiện ở việc sau này nếu chúng ta có cập nhật thêm các ngôn ngữ mới hoặc chỉnh sửa,chúng ta chỉ cần cập nhật lại ở file xml.Cách này giúp chúng ta tiết kiệm được rất nhiều thời gian so với việc phải cập nhật lại các dòng code Class Language.cs có nhiệm vụ đọc từ file xml lên và tùy theo loại control mà thay đổi ngôn ngữ thích hợp

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace Client
{
    public enum eLanguage
    {
        TiengViet = 0,
        TiengAnh = 1
    }
    public class MultiLang
    {
        private DataSet ds = new DataSet();
        private int flag_language = (int)eLanguage.TiengViet;
        string from = "viet";
        string to = "anh";


        public MultiLang()
        {
            ds.ReadXml(Application.StartupPath+"\\Language\\Language.xml");
        }
        public void SetLanguage(int val)
        {
            this.flag_language = val;
            switch (this.flag_language)
            {
                case (int)eLanguage.TiengViet:
                    from = "viet";
                    to = "anh";
                    break;
                case (int)eLanguage.TiengAnh:
                    from = "anh";
                    to = "viet";
                    break;
            }
        }


        private void ChangeMenu(ToolStripMenuItem item)
        {
            item.Text = this.get_text(ds.Tables[0], to, item.Text, from);
            for (int i = 0; i < item.DropDown.Items.Count; i++)
            {
                ToolStripItem subItem = item.DropDown.Items[i];
                if (item is ToolStripMenuItem)
```

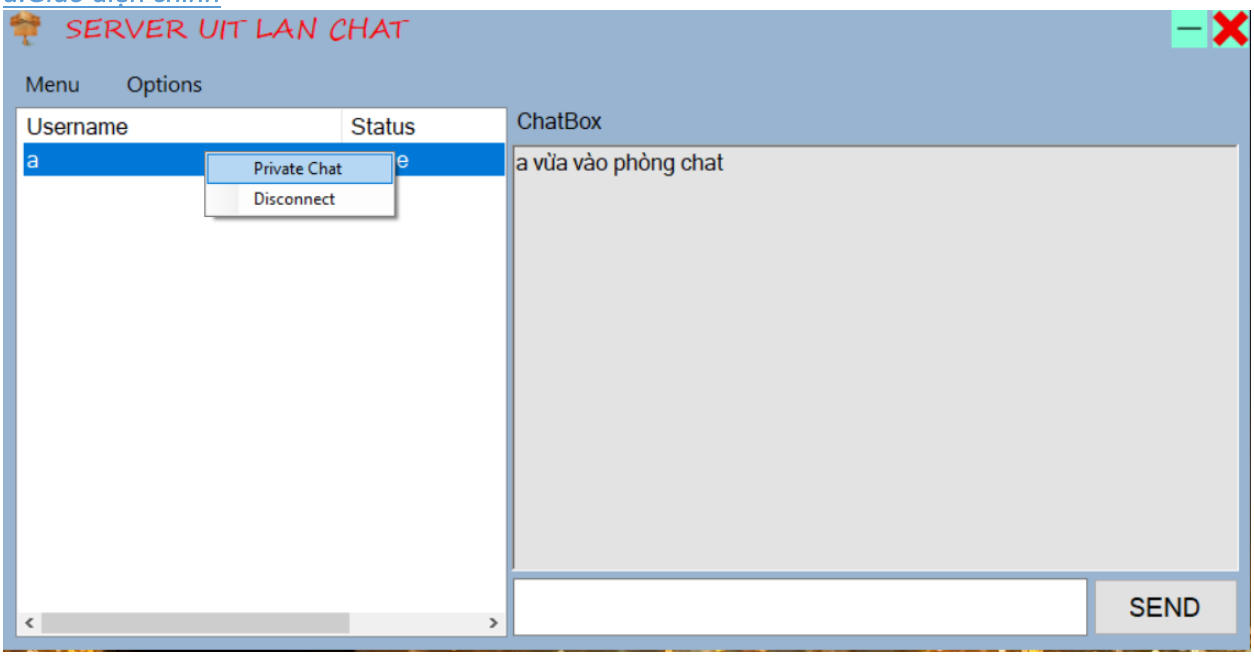
```
                {
                    ChangeMenu(subItem as ToolStripMenuItem);
                }
            }
        }

        private void ChangeButton(Control item)
        {
            if(item is Button)
                item.Text = this.get_text(ds.Tables[0], to, item.Text, from);

        }
        private void ChangeLabel(Control item)
        {
            if(item is Label)
                item.Text = this.get_text(ds.Tables[0], to, item.Text, from);

        }

        private void ChangeToolStrip(Control item)
        {
            if (item is ContextMenuStrip)
            {
                var temp = item as ContextMenuStrip;
                for (int i = 0; i < temp.Items.Count; i++)
                {
                    temp.Items[i].Text = get_text(ds.Tables[0], to, temp.Items[i].Text,
from);
                }
            }
        }


        //Đổi ngôn ngữ
        public void ChangeLanguage(Form frm)
        {

            frm.Text = this.get_text(this.ds.Tables[0], to, frm.Text, from);
            foreach (Control control in frm.Controls)
            {

                switch (control.GetType().ToString())
                {
                    case "System.Windows.Forms.Label":
                    case "System.Windows.Forms.Button":
                        control.Text = this.get_text(this.ds.Tables[0], to, control.Text,
from);
                        break;
                    case "System.Windows.Forms.MenuStrip":
                        MenuStrip menuControl = control as MenuStrip;
                        menuControl.Text = this.get_text(this.ds.Tables[0], to,
menuControl.Text, from);
                        foreach (ToolStripMenuItem menu in menuControl.Items)
                        {
                                ChangeMenu(menu);
                        }

                        break;
```

```
                    case "System.Windows.Forms.ContextMenuStrip":
                        ChangeToolStrip(control);
                        break;
                    default:
                        break;

                }
            }
        }
        private string get_text(DataTable dt, string to, string text, string from)
        {
            string str = text;
            string dkt = to + "='" + text + "'";
            DataRow dataRow = this.getrowbyid(dt, dkt);
            if (dataRow != null)
            {
                str = dataRow[from].ToString();
            }
            return str.Trim();
        }

        public DataRow getrowbyid(DataTable dt, string exp)
        {
            DataRow dataRow;
            try
            {
                dataRow = dt.Select(exp)[0];
            }
            catch
            {
                dataRow = null;
            }
            return dataRow;
        }

    }

}
```

CHƯƠNG 4 : GIAO DIỆN PHẦN MỀM

1.Server

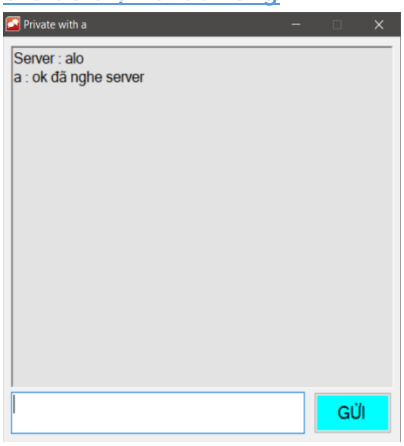
a.Giao diện chính

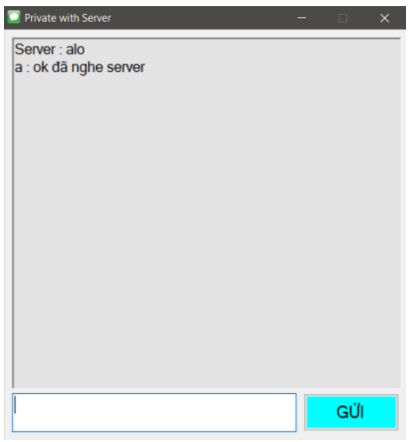


Giao diện chính

Mô tả thành phần giao diện : Gồm 1 listview để hiển thị các client đang online trong server và richtextbox để hiển thị tin nhắn từ client gửi tới(tin nhắn public).1 contextmenustrip gán với listview để thực hiện việc chat riêng tư với client hoặc ngắt kết nối với client đã chỉ định.textbox để type tin nhắn gửi đến tất cả các client

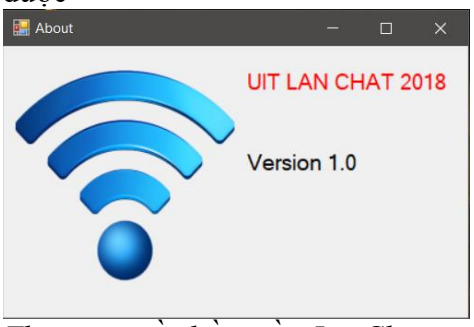
b.Giao diện chat riêng





Giao diện chat riêng với client

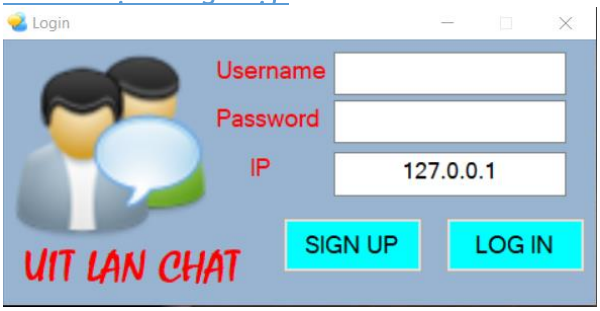
Mô tả thành phần giao diện : gồm textbox để gõ tin nhắn và richtextbox để hiển thị tin nhắn nhận được



Thông tin về phần mềm LanChat

2.Client

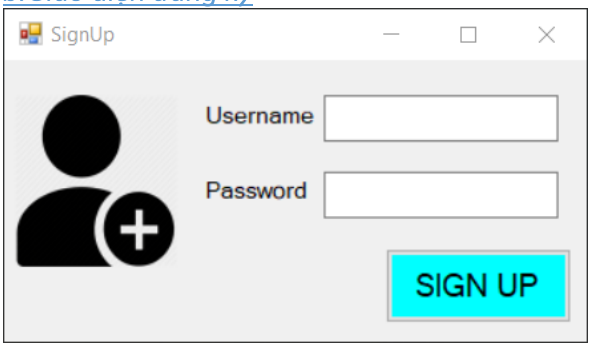
a.Giao diện đăng nhập



Giao diện đăng nhập

Mô tả thành phần giao diện : Gồm các textbox để gõ tên đăng nhập,mật khẩu và địa chỉ của server.2 nút button để đăng nhập và đăng ký

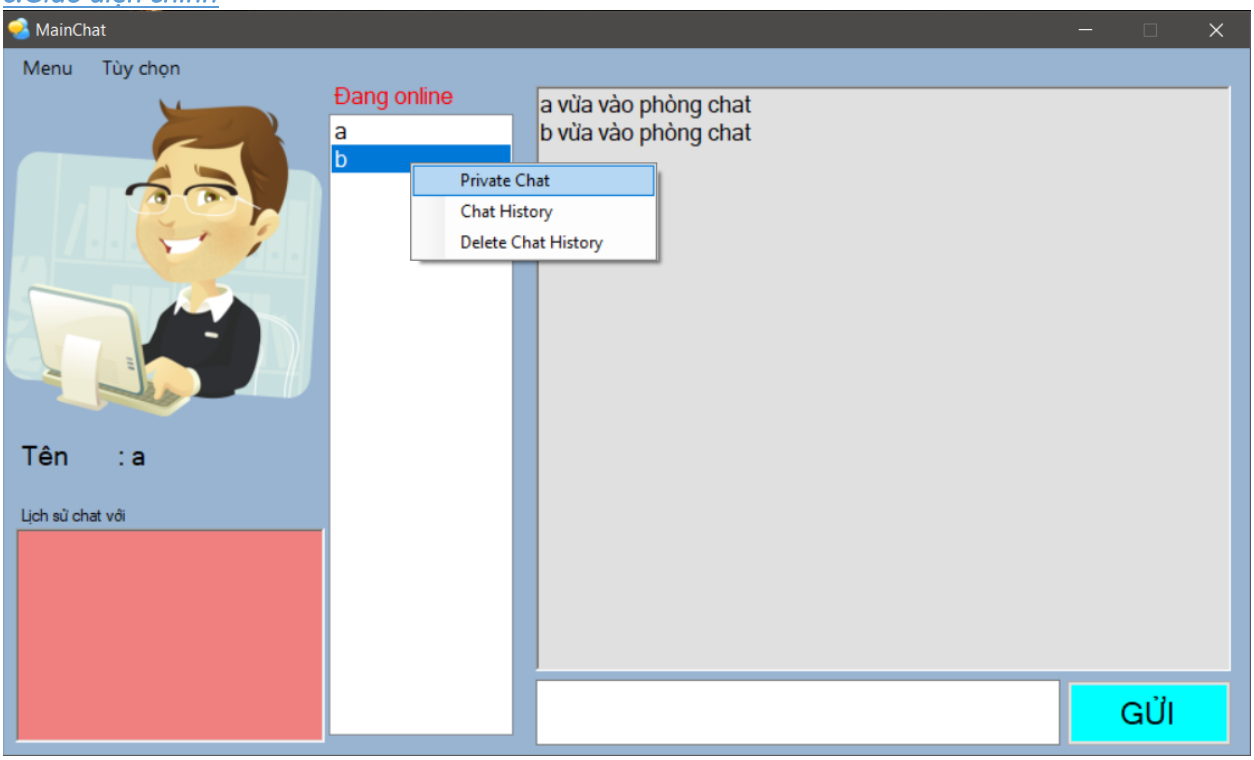
b.Giao diện đăng ký



Giao diện đăng ký tài khoản

Mô tả thành phần : Gồm textbox gõ tên tài khoản muốn đăng ký và nút click đăng ký

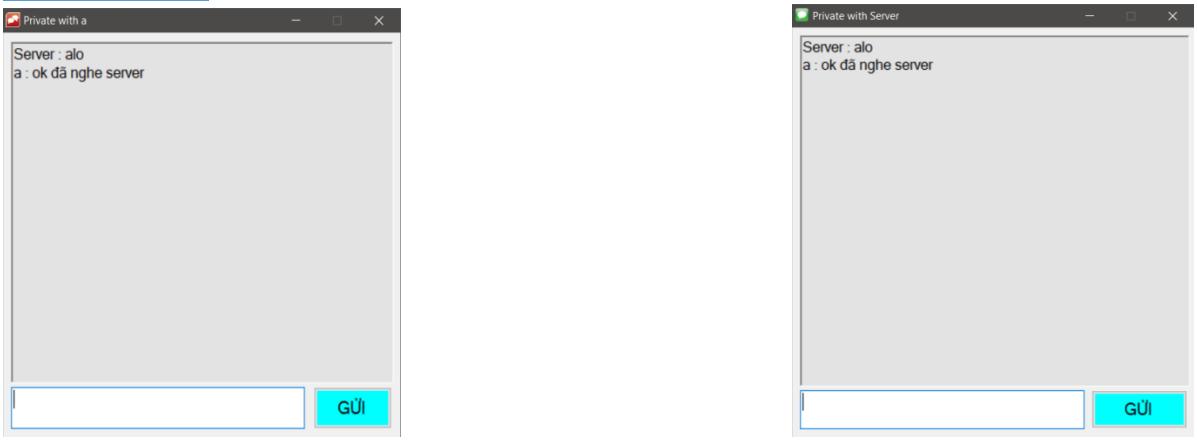
c.Giao diện chính



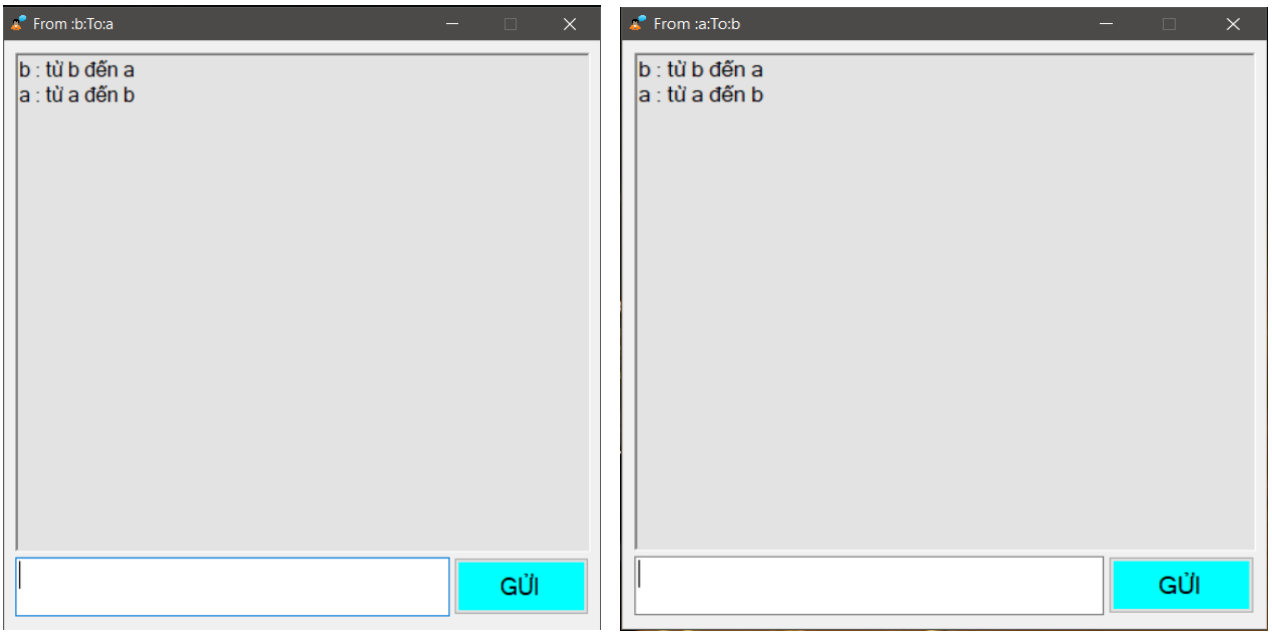
Giao diện chính của người dùng

Mô tả thành phần giao diện : gồm listbox để hiển thị người dùng đang online theo thời gian thực,textbox để gửi tin nhắn public và richtexbox để nhận tin nhắn public.Ngoài ra còn kho richtextbox để hiển thị tin nhắn riêng tư với client khác

d.Giao diện chat

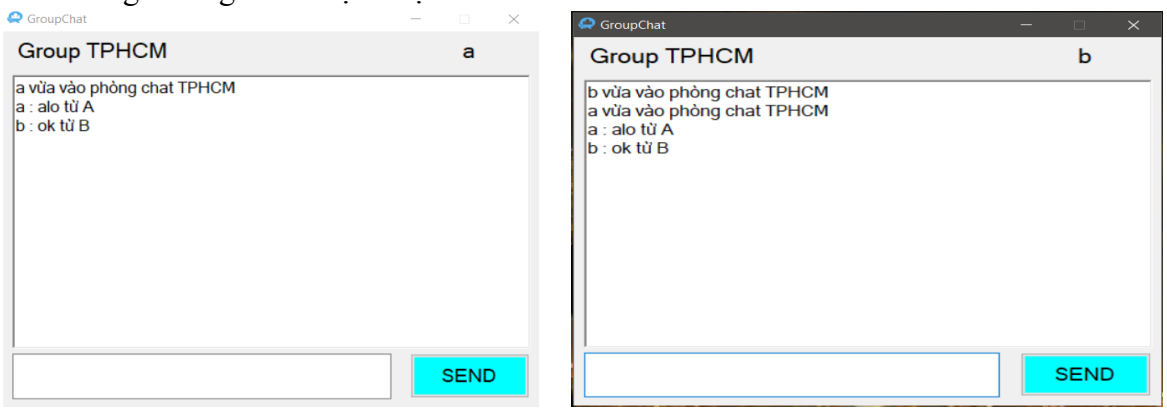


Giao diện chat riêng với server



Giao diện chat riêng tư với client

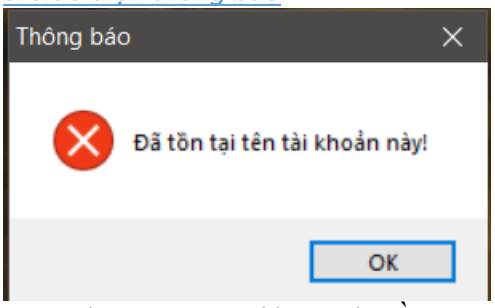
Mô tả thành phần giao diện : Gồm 1 textbox để gõ tin nhắn riêng tư và richtextbox để hiển thị tin nhắn riêng tư đã gửi và nhận được



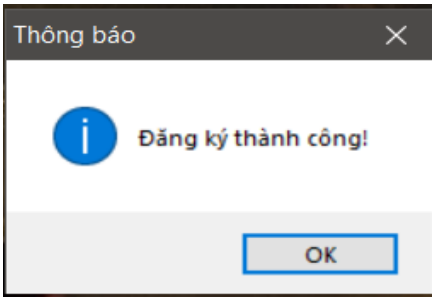
Giao diện chat nhóm

Mô tả thành phần giao diện : Gồm 2 label để thể hiện tên nhóm và tên người dùng,1 textbox để gõ tin nhắn chat nhóm và richtextbox để hiển thị tin nhắn chat nhóm

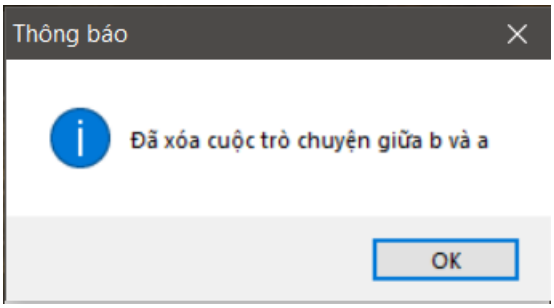
e.Giao diện thông báo



Đăng ký với tên tài khoản đã tồn tại



Thông báo đăng ký tài khoản thành công



Giao diện thông báo đã xóa cuộc trò chuyện riêng tư giữa 2 client

CHƯƠNG 5 :CÀI ĐẶT-KIỂM THỬ

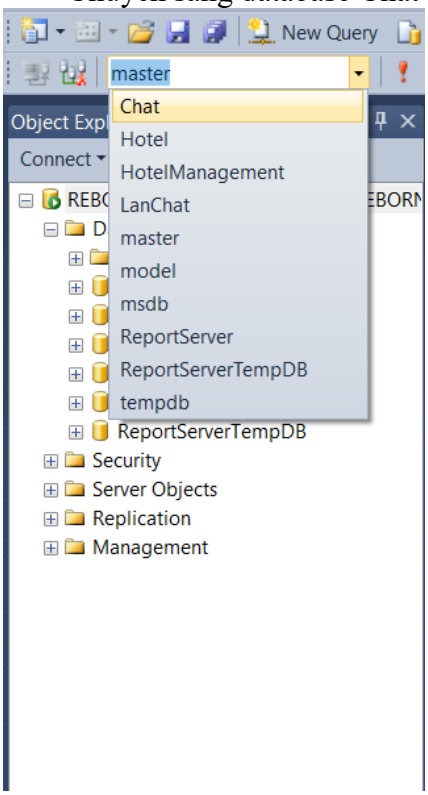
1.Hướng dẫn cài đặt

a.Cài đặt cơ sở dữ liệu

Đầu tiên ta cần cài phần mềm SQL Server (ở đây là phiên bản 2012)

Microsoft® SQL Server® 2012

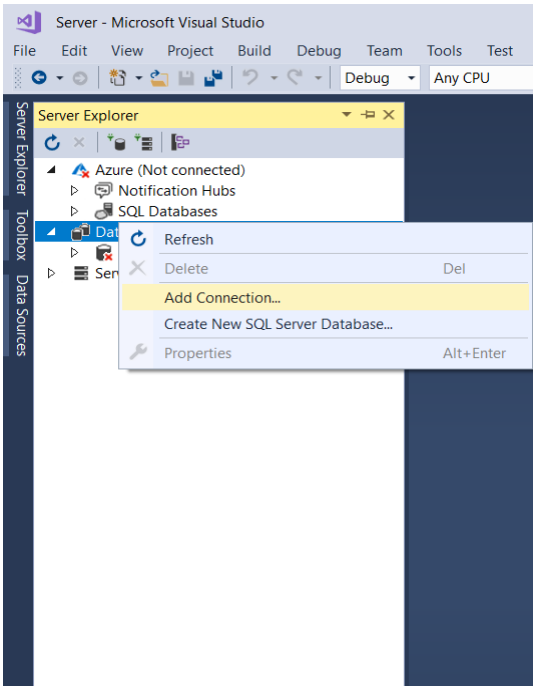
B1 : Chạy phần mềm SQL Server
B2 : Sau đó mở file ChatDatabase.sql
B3 : Chạy dòng lệnh “create database Chat” bằng cách tô đen dòng lệnh và nhấn F5(Excute)
Chuyển sang database Chat vừa mới tạo bằng cách như hình



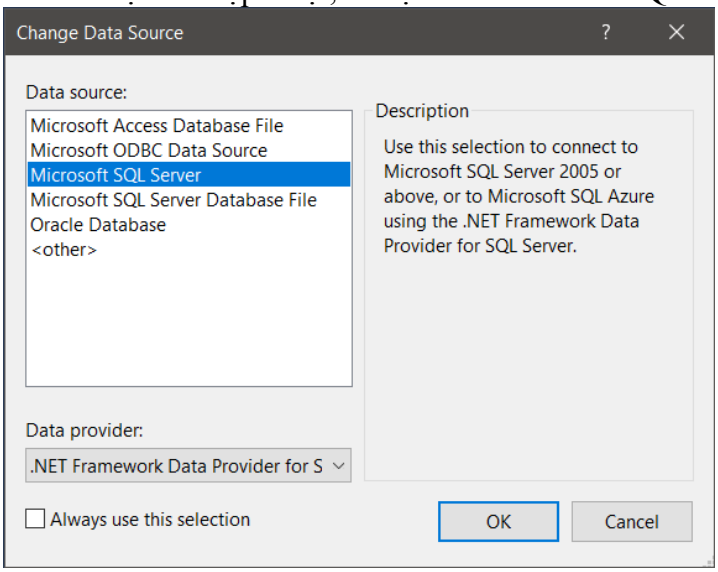
B4 : Tiếp tục thực hiện các dòng lệnh còn lại trong ChatDatabase.sql
B5 : Mở file ChatProc.sql lên và thực hiện các dòng lệnh trong file

b.Cài đặt server

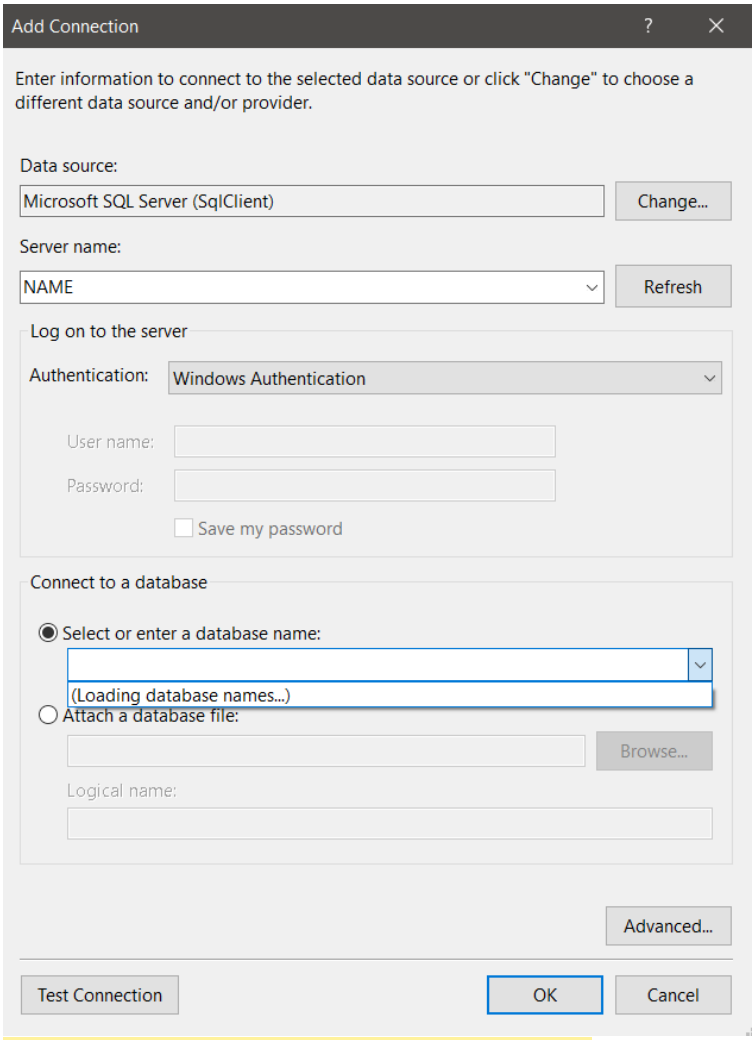
Mở source code
Vào Server Explorer

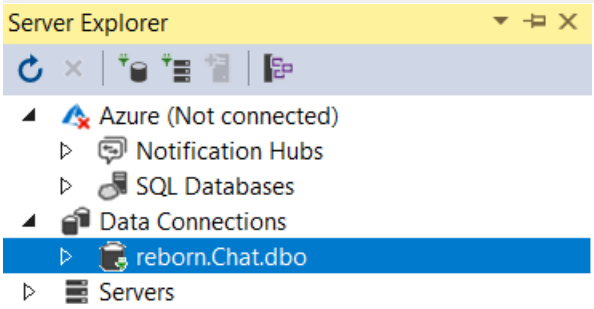


Sau đó hiện lên hộp thoại,ta chọn vào Microsoft SQL Server



Ta gõ đúng tên Server name cài đặt trên máy tính mình,lúc này ở phần Select or enter a database name ta chọn đúng CSDL vừa tạo ở trên và nhấn OK





Ta chọn vào CSDL đã được kết nối trong Data Connection và xem thuộc tính Connection String.Copy toàn bộ chuỗi vào class Server.cs thay đổi source= @"đường link";

2.Cài đặt và kiểm thử

CHƯƠNG 6 : KẾT LUẬN

1.Kết quả đạt được

Đề tài Lan Chat đã giúp thành viên của nhóm đạt được các nội dung sau:
Tìm hiểu về cách lập trình Socket và luồng Thread trên môi trường .NET
Tìm hiểu được cách thức hoạt động,xử lý của 1 phần mềm Chat cơ bản
Ưu điểm :
Nhóm đã thực hiện được đa số các chức năng cơ bản cần có của 1 phần mềm lan chat
-Gửi tin nhắn public
-Gửi tin nhắn riêng tới server/client khác
-Chat nhóm
-Xem lịch sử chat
-Xóa lịch sử chat
Nhược điểm :
Vì điều kiện khách quan,mà nhóm vẫn chưa có các chức năng đặc biệt
-Chưa hỗ trợ gửi file
-Chỉ hỗ trợ chat text thông thường

2.Hướng phát triển

Nếu có điều kiện,nhóm sẽ phát triển đồ án thêm các chức năng
-Hỗ trợ gửi file
-Có khả năng tạo nhóm chat cho người dùng
-Hỗ trợ chức năng voice chat…

Tài liệu tham khảo :
[1] Lập trình mạng căn bản
[2] Chương 8-Đa tiến trình,Giáo trình Lập trình trực quan
[3] https://www.youtube.com
[4] https://codeproject.com