**TRƯỜNG ĐH CÔNG NGHỆ THÔNG TIN – ĐHQG TPHCM**

**KHOA CÔNG NGHỆ PHẦN MỀM**

**----🙣🕮🙡----**

**ĐỒ ÁN P2P “SkyChat”**



Nguyễn Tấn Phát

Dương Thị Thu Thủy

Đặng Thành Quyên

Lớp: IT008.J11

Giáo viên hướng dẫn: Huỳnh Tuấn Anh

Năm học: 2018 - 2019

**MỤC LỤC**

[**DANH SÁCH BẢNG BIỂU** 3](#_Toc533184317)

[**DANH SÁCH HÌNH ẢNH** 3](#_Toc533184318)

[**PHẦN 1/ MỞ ĐẦU:** 4](#_Toc533184319)

[1.1/ Lý do chọn đề tài: 4](#_Toc533184320)

[1.2/ Mục đích của đề tài: 4](#_Toc533184321)

[1.3/ Đối tượng và phạm vi nghiên cứu: 4](#_Toc533184322)

[1.3.1/ Đối tượng nghiên cứu: 4](#_Toc533184323)

[1.3.2/ Phạm vi nghiên cứu: 4](#_Toc533184324)

[**PHẦN 2/ KIẾN THỨC ỨNG DỤNG:** 4](#_Toc533184325)

[2.1/ Sơ lược về lập trình socket 4](#_Toc533184326)

[2.1.1/ Một số khái niệm cơ bản: 4](#_Toc533184327)

[2.1.2/ Lớp IPAddress 5](#_Toc533184328)

[2.1.3/ Lớp IPEndpoint 6](#_Toc533184329)

[2.1.4/ Lớp UDP (UDP client) 7](#_Toc533184330)

[2.1.5/ Giao thức TCP 8](#_Toc533184331)

[2.2/ Lập trình với Thread trong P2P 8](#_Toc533184332)

[2.3/ Mô hình P2P 9](#_Toc533184333)

[**PHẦN 3/ PHÂN TÍCH PHẦN MỀM** 10](#_Toc533184334)

[3.1/ Phân tích nhu cầu thực tiễn: 10](#_Toc533184335)

[3.2/ Yêu cầu đề ra: 10](#_Toc533184336)

[**PHẦN 4/ THIẾT KẾ PHẦN MỀM:** 11](#_Toc533184337)

[4.1/ Các lớp xử lý 11](#_Toc533184338)

[4.2/ Một số quy tắc và hàm xử lý cơ bản: 13](#_Toc533184339)

[4.2.1/ Quy tắc gửi và nhận dữ liệu trong mạng 13](#_Toc533184340)

[4.2.2/ Hàm xử lý cơ bản: 14](#_Toc533184341)

[4.3/ Giao diện 37](#_Toc533184342)

[4.3.1/ Màn hình: 37](#_Toc533184343)

[4.3.2/ Các tính năng 39](#_Toc533184344)

[**PHẦN 5/ CÀI ĐẶT – THỬ NGHIỆM** 42](#_Toc533184345)

[5.1/ Yêu cầu hệ thống: 42](#_Toc533184346)

[5.2/ Hướng dẫn cài đặt: 42](#_Toc533184347)

[5.3/ Hướng dẫn gỡ cài đặt: 44](#_Toc533184348)

[**PHẦN 6/ KẾT LUẬN:** 45](#_Toc533184349)

[6.1 Kết quả đạt được: 45](#_Toc533184350)

[6.2 Hướng phát triển 45](#_Toc533184351)

[**TÀI LIỆU THAM KHẢO** 46](#_Toc533184352)

DANH SÁCH BẢNG BIỂU

[Bảng 2-1: Một số thành phần của lớp IPAddress 5](#_Toc533183014)

[Bảng 2-2: Các thành viên của lớp IPEndPoint 6](#_Toc533183015)

[Bảng 2-3: Một số thành phần của lớp UdpClient. 7](#_Toc533183016)

[Bảng 4-1: Source code xử lý dữ liệu từ byte[] -> Packet 13](#_Toc533183017)

[Bảng 4-2:Source code Gửi dữ liệu trong Chat Form 16](#_Toc533183018)

[Bảng 4-3: Source code Gửi dữ liệu trong Chat Private 21](#_Toc533183019)

[Bảng 4-4: Source code Nhận dữ liệu trong Chat Form 26](#_Toc533183020)

[Bảng 4-5: Source code Nhận dữ liệu Chat Private 34](#_Toc533183021)

[Bảng 4-6: Source code Thread 34](#_Toc533183022)

[Bảng 4-7: Source code tạo Chat Private 35](#_Toc533183023)

DANH SÁCH HÌNH ẢNH

[Hình 2-1: Mô hình P2P. 7](#_Toc533183551)

[Hình 4-1: Giao diện Đăng nhập. 35](#_Toc533183552)

[Hình 4-2: Giao diện Chat Public. 36](#_Toc533183553)

[Hình 4-3: Giao diện Chat Private. 36](#_Toc533183554)

[Hình 4-4: Giao diện Giới thiệu và trợ giúp. 37](#_Toc533183555)

[Hình 4-5: Tính năng chọn Font. 38](#_Toc533183556)

[Hình 4-6: Tính năng Font & Color 38](#_Toc533183557)

[Hình 4-7: Tính năng gửi file & hình ảnh trong Chat Public. 39](#_Toc533183558)

[Hình 4-8: Tính năng gửi file & hình ảnh trong Chat Private. 39](#_Toc533183559)

[Hình 4-9: Giao diện khi đóng chương trình. 40](#_Toc533183560)

[Hình 4-10: Giao diện khi đóng Chat Private. 40](#_Toc533183561)

[Hình 5-1: Hộp thoại Welcome. 41](#_Toc533183562)

[Hình 5-3: Hộp thoại Select Installation Folder 41](#_Toc533183563)

[Hình 5-2: Hộp thoại Confirm Installation 42](#_Toc533183564)

[Hình 5-4: Hộp thoại Installation Complete 42](#_Toc533183565)

PHẦN 1/ MỞ ĐẦU:

1.1/ Lý do chọn đề tài:

Hiện nay, mạng Lan đã có những tiến bộ vượt bậc và ngày càng phổ biến hơn trong đời sống sinh hoạt nhất là trong các cơ quan, tổ chức, đội, nhóm …Điều này làm cho nhu cầu liên lạc và trao đổi thông tin thông qua mạng Lan với tốc độ nhanh ngày càng lớn hơn. Chính vì vậy, phần mềm SkyChat – Lan Chat sử dụng mô hình P2P được xây dựng để đáp ứng phần nào những nhu cầu cấp thiết đó.

1.2/ Mục đích của đề tài:

Xây dựng phần mềm Chat hoạt động trong mạng Lan bằng mô hình P2P với các chức năng cơ bản như: gửi, nhận tin nhắn, hỗ trợ gửi file, hình ảnh, thả emotion, chat public và chat private.

1.3/ Đối tượng và phạm vi nghiên cứu:

1.3.1/ Đối tượng nghiên cứu:

Tìm hiểu cơ chế hoạt động của mô hình P2P, Socket (UDP, TCP) và Thread trong .NET Framwork từ đó tạo ứng dụng Chat P2P trong mạng Lan.

1.3.2/ Phạm vi nghiên cứu:

Chương trình Chat được xây dựng với khả năng Chat bằng , file hoặc image giữa các User, hỗ trợ chat public và chat giữa 2 User.

PHẦN 2/ KIẾN THỨC ỨNG DỤNG:

2.1/ Sơ lược về lập trình socket

2.1.1/ Một số khái niệm cơ bản:

**-** Trong lập trình, Socket là một API(Application Programming Interface) cung cấp các phương thức để giao tiếp thông qua mạng. Gồm 3 loại: Raw socket, Datagram sockets, Stream sockets.

- Một socket có thể thực hiện 7 thao tác cơ bản sau:

1. Kết nối với một máy ở xa (ví dụ: chuẩn bị gửi và nhận data).

2. Gửi dữ liệu.

3. Nhận dữ liệu.

4. Ngắt liên kết.

5. Gắn cổng.

6. Nghe dữ liệu đến.

7. Chấp nhận liên kết từ accs máy ở xa trên cổng đã được gán.

* Địa chỉ socket gồm có 2 phần:

- Địa chỉ IP (IP Address) là một số nguyên 32 bits/card mạng/máy tính (host).

- Cổng dịch vụ (Port): là một số nguyên 16 bits xác định điểm kết nối với một ứng dụng trên một host.

VD: Mỗi ứng dụng trên máy B sẽ được gán một số hiệu (cổng Port). Khi ứng dụng trên máy A muốn gửi cho ứng dụng nào trên máy B thì chỉ việc điền thêm số hiệu cổng (vào trường RemotePort) vào gói tin cần gửi. Trên máy B, các ứng dụng sẽ kiểm tra giá trị cổng trên mỗi gói tin xem có trùng với số hiệu cổng của mình (đã được gán – chính là giá trị LocalPort) hay không? Nếu bằng thì xử lý, còn trái lại thì không làm gì.

* Như vậy, khi cần trao đổi dữ liệu cho nhau thì hai ứng dụng cần phải

biết thông tin tối thiểu là địa chỉ IP và cổng dịch vụ (Port) của ứng dụng kia.

2.1.2/ Lớp IPAddress

Trong .NET, IP Address là một lớp dùng để mô tả địa chỉ IP. Đây là lớp rất cơ bản được sử dụng khi thao tác (truyền) vào các lớp như IPEndpoint, UDP, TCP, Socket…

|  |  |
| --- | --- |
| **Thành viên Static** | **Mô tả** |
| ***Any*** | Cung cấp một địa chỉ IP (thường là 0.0.0.0) để chỉ ra rằng máy tính phải lắng nghe các hoạt động của máy khác trên tất cả các Card mạng (sử dụng khi xây dựng như một 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. |
| ***AddressFamily*** | Trả về họ địa chỉ của địa chỉ IP hiện hành. Nếu địa chỉ IP ở dạng địa chỉ IPv4 thì kết quả là Internetwork và Internetwork 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. |

Bảng 2-1: Một số thành phần của lớp IPAddress

2.1.3/ Lớp IPEndpoint

Trong mạng, để hai trạm có thể trao đổi thông tin được với nhau thì chúng cần phải biết được địa chỉ IP của nhau và số hiệu cổng mà hai bên dùng để trao đổi thông tin. Lớp IPAddress mới chỉ cung cấp cho ta một vế là địa chỉ IP (IPAddress). Như vậy, lớp IPEndpoint chính là lớp chứa đựng cả IPAddress và Port number.

Đối tượng IPEndpoint sec được dùng sau này để truyền trực tiếp cho các đối tượng UDP, TCP…

|  |  |
| --- | --- |
| **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. |
| ***ToString()*** | Trả về địa chỉ IP và số hiệu cổng thoe khuông dạng địa chỉ:cổng. Ví dụ: “192.168.1.11:14000” |

Bảng 2-2: Các thành viên của lớp IPEndPoint

2.1.4/ Lớp UDP (UDP client)

Giao thức UDP (User Datagram Protocol hay User Define Protocol) là một gioa thức phi kết nối (connectionless) có nghĩa là một bên có thể gửi dữ liệu cho bên kia mà không cần biết là bên đó đã sẵn sàng hay chưa. Nói cách khác là không cần thiết lập kết nối giữa hai bên khi tiến hành trao đổi thông tin. Giao thức này không tin cậy bằng giao thức TCP nhưng tốc độ nhanh và dễ cài đặt. Ngoài ra, với giao thức UDP ta còn có thể gửi các gói tin quảng bá (Broadcast) cho đồng thời nhiều máy.

Trong .NET, lớp **UDPClient** (nằm trong namespace **System.Net.Sockets**) đóng gói các chức năng của giao thức UDP.

|  |  |
| --- | --- |
| **Phương thức khởi tạo** | **Mô tả** |
| ***UdpClient(String, Int32)*** | Tạo một **UdpClient**và thiết lập với một trạm từ xa mặc định. |
| **Phương thức** | **Mô tả** |
| ***Receive(ref IPEndPoint)*** | Nhận dữ liệu (đồng bộ) do máy ở xa gửi. (Đồng bộ có nghĩa là các lệnh ngay sau lệnh Receive đã nhận được dữ liệu về. Nếu chưa nhận được – dù chỉ một chút – thì nó vẫn cứ cờ (blocking)). |
| ***Send(byte[] dgram, int byte)*** | Gửi một UDP Datagram đến remote host. |
| ***Connect()*** | Thiết lập một Default remote host. |

Bảng 2-3: Một số thành phần của lớp UdpClient.

2.1.5/ Giao thức TCP

Để đảm bảo độ tin cậy trong khi gửi và nhận người ta dùng một giao thức khác, gọi là giao thức có kết nối: TCP (Transport Control Protocol).

2.2/ Lập trình với Thread trong P2P

**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(). Nhưng thường thì một chương trình cần làm nhiều công việc hơn vào cùng một lúc. Ví dụ: Trong mô hình P2P, mỗi Peer gồm có hai phương thức chính là Send() và Receive(), hai phương thức này phải được đồng thời thực hiện song song với nhau. Nhưng phương thức Receive() trên mỗi Socket sẽ rơi vào trạng thái chờ đợi khi không có dữ liệu trên kết nối mà nó quản lý và vì vậy khi một kết nối đang được xử lý không có dữ liệu các kết nối khác sẽ bị khóa. Để giải quyết vấn đề này c# cung cấp công cụ lập trình Thread.

2.3/ Mô hình P2P

**A picture containing sky, table

Description automatically generated**

Hình 2-1: Mô hình P2P.

**-** Mạng ngang hàng ([tiếng Anh](https://vi.wikipedia.org/wiki/Ti%E1%BA%BFng_Anh): *peer-to-peer network*), còn gọi là mạng đồng đẳng, là một [mạng máy tính](https://vi.wikipedia.org/wiki/M%E1%BA%A1ng_m%C3%A1y_t%C3%ADnh) trong đó hoạt động của mạng chủ yếu dựa vào khả năng tính toán và [băng thông](https://vi.wikipedia.org/wiki/B%C4%83ng_th%C3%B4ng) của các máy tham gia chứ không tập trung vào một số nhỏ các [máy chủ trung tâm](https://vi.wikipedia.org/w/index.php?title=M%C3%A1y_ch%E1%BB%A7_trung_t%C3%A2m&action=edit&redlink=1) như các mạng thông thường. Mạng đồng đẳng thường được sử dụng để kết nối các máy thông qua một lượng kết nối dạng [ad hoc](https://vi.wikipedia.org/w/index.php?title=Ad_hoc&action=edit&redlink=1). Mạng đồng đẳng có nhiều ứng dụng. Ứng dụng thường xuyên gặp nhất là chia sẻ tệp tin, tất cả các dạng như âm thanh, hình ảnh, dữ liệu,... hoặc để truyền dữ liệu thời gian thực như điện thoại [VoIP](https://vi.wikipedia.org/wiki/VoIP). Một mạng đồng đẳng đúng nghĩa không có khái niệm [máy chủ](https://vi.wikipedia.org/wiki/M%C3%A1y_ch%E1%BB%A7) và [máy khách](https://vi.wikipedia.org/w/index.php?title=M%C3%A1y_kh%C3%A1ch&action=edit&redlink=1), nói cách khác, tất cả các máy tham gia đều bình đẳng và được gọi là peer, là một nút mạng đóng vai trò đồng thời là máy khách và máy chủ đối với các máy khác trong mạng.

**-** Ưu điểm:

+ Tất cả các máy khi tham gia mạng lưới đều có thể đóng góp thông tin bao gồm băng thông, dữ liệu và cả khả năng tính toán.

+ Nhờ tình chất phân tán giúp các mạng khác vẫn hoạt động tốt khi một máy trong mạng lưới gặp sự cố.

+ Máy tính trong hệ thống đóng vai trò như máy chủ và máy khách.

+ Tránh tắc nghẽn mạng, quá tải, sử dụng dễ dàng và được tích hợp tốt.

**-** Nhược điểm:

+ Độ an toàn và bảo mật kém, phụ thuộc vào mức truy nhập được chia sẻ.

+ Có thể dẫn tới sự biến mất của một dịch vụ trong khoảng thời gian nhất định.

**-** Quá trình trao đổi dữ liệu bao gồm: các Peer yêu cầu dịch vụ từ các bên khác và cung cấp dịch vụ ngược lại cho các bên khác.

PHẦN 3/ PHÂN TÍCH PHẦN MỀM

3.1/ Phân tích nhu cầu thực tiễn:

Mạng Lan phát triển rất mạnh cả trong trường học, các cơ quan tổ chức và ở các hộ gia đình. Chính điều đó kéo theo nhu cầu trao đổi thông tin trong mạng Lan cũng phát triển theo. Chính vì vậy, một chương trình Chat phục vụ cho nhu cầu liên lạc, trao đổi thông tin trong mạng Lan là rất cần thiêt.

- Hơn nữa ngày càng có nhiều nhu cầu trao đổi thông tin một cách nhanh chóng và thuận tiện cho việc cài đặt đang gia tăng không ngừng. Do đó, với mô hình chat Lan truyền thống (Server – Client) – mô hình yêu cầu phải có Server trung gian giữa các Client đang dần trở nên chậm chạp mà mô hình P2P lại đáp ứng được điều đó.

3.2/ Yêu cầu đề ra:

Xây dựng chương trình Chat P2P hoạt động trong mạng Lan sử dụng Socket và Thread đòi hỏi các tính chất, chức năng sau:

- Mỗi máy vừa đóng vai trò vừa là Server vừa là Client.

- Chat giữa các máy trong mạng với nhau (Chat public): các máy có thể giao tiếp, thông tin với nhau khi kết nối vào một mạng Broadcast. Trong đó, mỗi người tham gia sẽ có một username và được lưu vào danh sách bạn bè của những người tham gia khác.

- Chat giữa hai người với nhau (Chat private): hai người gửi thông điệp qua lại cho nhau.

- Một User có khả năng chọn một người bạn trong danh sách bạn bè để chat private và có thể gửi tin nhắn với các định dạng text, file, image, emotion…

PHẦN 4/ THIẾT KẾ PHẦN MỀM:

4.1/ Các lớp xử lý

**\* Lớp Package**

Thuộc tính:

* myIP (string)
* myName (string)
* myFont (Font )
* myColor (Color )
* myType (TypePacket )
* myPort (string)

Phương thức:

* Packet()
* Packet(string IP, string Name, string Mess, Font font, Color color, TypePacket type, string Port)
* Packet(Packet packet)
* Packet(SerializationInfo info, StreamingContext context)
* GetObjectData(SerializationInfo info, StreamingContext context)

\* Enum: enum TypePacket

NONE,

REPLY,

SEND\_INFO\_USER\_1,

SEND\_INFO\_USER\_2,

MESSAGE,

REQUIRE\_CHAT,

CLOSING,

SEND\_FILE,

SEND\_IMAGE,

OUT\_CHAT,

SEND\_PORT

**\* Lớp Message**

\* Thuộc tính:

* pack (Packet )
* who (Type )

\* Phương thức:

* Message(Packet pck, Type whoIs)

\* Enum: enum Type

SENDER,

RECEIVER,

CLOSER

**\* Lớp SendType**

\* Enum: enum SendType

MESSAGE,

FILE,

IMAGE,

NONE

**\* List:**

List<string> lstOnline : chứa Name và IP.

List<PrivateChat> lstFormPrivChat: chứa danh sách form PrivateChat

List<Object> ChatLog: lưu trữ các tin nhắn

4.2/ Một số quy tắc và hàm xử lý cơ bản:

4.2.1/ Quy tắc gửi và nhận dữ liệu trong mạng

* **Chat Public**

**Bước 1:** Bên Gửi gửi tín hiệu (packet) cho bên nhận.

1. Gửi packet chứa tin nhắn nếu chat thường.
2. Gửi packet chứa tín hiệu ‘sendFile” nếu gửi File.
3. Gửi packet chứa tín hiệu “sendImage” nếu gửi Ảnh.

**Bước 2:** Bên Nhận xử lý tín hiệu (packet) từ bên gửi.

1. Nhận packet chứa tin nhắn, đưa tin nhắn lên web browser.
2. Nhận packet chứa tín hiệu “sendFile”, bật tín hiệu nhận File lên.
3. Gửi packet chứa tín hiệu “sendImage”, bật tín hiệu nhận Image lên.

**Bước 3:** Nếu gửi media (File hoặc Image), bên Gửi gửi tiếp data của File (Image) dạng mảng byte.

**Bước 4:** Nếu nhận media (File hoặc Image), bên Nhận nhận và giải mã mảng byte được gửi đến theo luồng tín hiệu được bật sau đó đưa thông tin lên web browser.

* **Chat Private**

**Bước 1:** Bên Gửi tạo một kết nối mới (Socket) đến bên nhận, nếu kết nối thành công, tiến hành chat(Bước 3). Ngược lại, báo lỗi và tắt Form.

**Bước 2:** Bên Nhận dùng cổng nghe đã được cài đặt khi tạo Form để chấp nhận kết nối từ bên gửi.

**Bước 3:** Bên Gửi gửi packet tin nhắn hoặc media theo luồng kết nối ở Bước 1.

**Bước 4:** Bên Nhận xử lý tín hiệu(packet) từ bên gửi

1. Nhận packet chứa tin nhắn, đưa tin nhắn lên web browser.

2. Nhận packet chứa tín hiệu “sendFile”, bật tín hiệu nhận File lên.

3. Gửi packet chứa tín hiệu “sendImage”, bật tín hiệu nhận Image lên.

**Bước 5:** Nếu gửi media (File hoặc Image), bên Gửi gửi tiếp data của File(Image) dạng mảng byte.

**Bước 6:** Nếu nhận media (File hoặc Image), bên Nhận nhận và giải mã mảng byte được gửi đến theo luồng tín hiệu được bật sau đó đưa thông tin lên web browser.

4.2.2/ Hàm xử lý cơ bản:

**\* Hàm xử lý dữ liệu**

|  |
| --- |
| private Packet DeSerialize(byte[] data)  {  Package.Packet myPacket = new Packet();  MemoryStream str = new MemoryStream(data);  BinaryFormatter bformat = new BinaryFormatter();  myPacket = (Packet)bformat.Deserialize(str);  return myPacket;  } |

Bảng 4-1: Source code xử lý dữ liệu từ byte[] -> Packet

* Chức năng: Giải mã dữ liệu truyền đến sang kiểu dữ liệu dành cho tin nhắn (Packet).

**\* Hàm gửi dữ liệu**

* **Chat Public**

|  |
| --- |
| private void btnSend\_Click(object sender, EventArgs e)  {  if (!string.IsNullOrEmpty(txtsend.Text))  {  if (Status == SendType.MESSAGE)  {  byte[] data = SendPacket();  sendingClient.Send(data, data.Length);  txtsend.Text = "";  }  else if (Status == SendType.FILE)  {  FileStream fs = null;  bool bSendOk = true;  try  {  FileInfo fi = new FileInfo(txtsend.Text);  byte[] buf = new byte[64\*1024];  fs = File.OpenRead(txtsend.Text);  long filesize = fi.Length;  byte[] buf1 = SendTypeFile(fi.Name);  sendingClient.Send(buf1, buf1.Length);  //Send size cua file  int nr = fs.Read(buf, 0, (int)filesize);  sendingClient.Send(buf, nr);  }  catch (Exception exx)  {  bSendOk = false;  MessageBox.Show(exx.Message, "File Sending Error", MessageBoxButtons.OK, MessageBoxIcon.Error);  }  if (fs != null)  {  try { fs.Close(); }  catch (Exception) { }  }  if (bSendOk)  {  MessageBox.Show("Gửi Thành công !", this.Text, MessageBoxButtons.OK, MessageBoxIcon.Information);  Packet fpck = new Packet();  fpck.MyName = userName;  fpck.MyMessage = "<a href='" + txtsend.Text.Replace(":", "(~\*)") + "'>" + "<b>" + Path.GetFileName(txtsend.Text) + "</b></a>";  Message fmess = new Message(fpck, Type.SENDER);  ChatLog.Add(fmess);  RefreshWeb();  }  Status = SendType.MESSAGE;  }  else if(Status == SendType.IMAGE)  {  FileStream fs = null;  bool bSendOk = true;  try  {  FileInfo fi = new FileInfo(txtsend.Text);  byte[] buf = new byte[64 \* 1024];  fs = File.OpenRead(txtsend.Text);  long filesize = fi.Length;  byte[] buf1 = SendTypeImage(fi.Name);  sendingClient.Send(buf1, buf1.Length);  //Send size cua file  int nr = fs.Read(buf, 0, (int)filesize);  sendingClient.Send(buf, nr);  }  catch (Exception exx)  {  bSendOk = false;  MessageBox.Show(exx.Message, "Gửi ảnh lỗi!", MessageBoxButtons.OK, MessageBoxIcon.Error);  }  if (fs != null)  {  try { fs.Close(); }  catch (Exception) { }  }  if (bSendOk)  {  MessageBox.Show("Gửi Thành công !", this.Text, MessageBoxButtons.OK, MessageBoxIcon.Information);  Packet fpck = new Packet();  fpck.MyName = userName;  fpck.MyMessage = "<a href='" + txtsend.Text.Replace(":", "(~\*)") + "'>" + "<img src='" + txtsend.Text + "' style='max-width:300px'/><br/>" + "<b>" + Path.GetFileName(txtsend.Text) + "</b></a>";  Message fmess = new Message(fpck, Type.SENDER);  ChatLog.Add(fmess);  RefreshWeb();  }  Status = SendType.MESSAGE;  }  txtsend.Focus();  txtsend.Text = "";  }  } |

Bảng 4-2:Source code Gửi dữ liệu trong Chat Form

* **Chat Private:**

|  |
| --- |
| private void btnSend\_Click(object sender, EventArgs e)  {  try  {  #region Gui Message  if (Status == SendType.MESSAGE)  {  Socket sck = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);  sck.NoDelay = true;  byte[] buf = new byte[1024];  buf = SendPacket();  sck.Connect(new IPEndPoint(IPAddress.Parse(friendIP), int.Parse(friendPort)));  sck.Send(buf);  sck.Close();  }  #endregion  #region Gui File  else if (Status == SendType.FILE)  {  FileStream fs = null;  Socket socket1 = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);  Socket socket = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);  socket.NoDelay = true;  bool bSendOk = true;  string extension = "";  try  {  FileInfo fi = new FileInfo(txtsend.Text);  ulong fileSize = (ulong)fi.Length;  extension = fi.Extension;  byte[] buf = new byte[32 \* 1024];  MemoryStream ms = new MemoryStream(buf);  BinaryWriter bw = new BinaryWriter(ms);  bw.Write(fileSize);  bw.Close();  ms.Close();  fs = File.OpenRead(txtsend.Text);  socket1.Connect(new IPEndPoint(IPAddress.Parse(friendIP), int.Parse(friendPort)));  byte[] buf2 = new byte[1024];  buf2 = SendPacket4(fi.Name, TypePacket.SEND\_FILE);  socket1.Send(buf2);  socket1.Close();  //Send size cua file  socket.Connect(new IPEndPoint(IPAddress.Parse(friendIP), int.Parse(friendPort)));  int ns = socket.Send(buf, sizeof(ulong), SocketFlags.None);  ulong pos = 0;  while (pos < fileSize)  {  int nr = fs.Read(buf, 0, buf.Length);  if (nr <= 0)  {  break;  }  pos += (ulong)nr;  ns = socket.Send(buf, nr, SocketFlags.None);  }  }  catch (Exception exx)  {  bSendOk = false;  MessageBox.Show(exx.Message, "Gửi file lỗi!", MessageBoxButtons.OK, MessageBoxIcon.Error);  }  if (fs != null)  {  try { fs.Close(); }  catch (Exception) { }  }  socket.Close();  if (bSendOk)  {  MessageBox.Show("Gửi Thành công !", this.Text, MessageBoxButtons.OK, MessageBoxIcon.Information);  Packet fpck = new Packet();  fpck.MyMessage = "<a href='" + txtsend.Text.Replace(":", "(~\*)") + "'>" + ((extension == ".jpg" || extension == ".png" || extension == ".PNG") ? "<img src='" + txtsend.Text + "' style='max-width:300px'/><br/>" : "") + "<b>" + Path.GetFileName(txtsend.Text) + "</b></a>";  Message fmess = new Message(fpck, Type.SENDER);  ChatLog.Add(fmess);  }  Status = SendType.MESSAGE;  }  #endregion  #region Gui Anh  else if (Status == SendType.IMAGE)  {  FileStream fs = null;  Socket socket1 = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);  Socket socket = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);  socket.NoDelay = true;  bool bSendOk = true;  string extension = "";  try  {  FileInfo fi = new FileInfo(txtsend.Text);  ulong fileSize = (ulong)fi.Length;  extension = fi.Extension;  byte[] buf = new byte[32 \* 1024];  MemoryStream ms = new MemoryStream(buf);  BinaryWriter bw = new BinaryWriter(ms);  bw.Write(fileSize);  bw.Close();  ms.Close();  fs = File.OpenRead(txtsend.Text);  socket1.Connect(new IPEndPoint(IPAddress.Parse(friendIP), int.Parse(friendPort)));  byte[] buf2 = new byte[1024];  buf2 = SendPacket4(fi.Name, TypePacket.SEND\_IMAGE);  socket1.Send(buf2);  socket1.Close();  //Send size cua file  socket.Connect(new IPEndPoint(IPAddress.Parse(friendIP), int.Parse(friendPort)));  int ns = socket.Send(buf, sizeof(ulong), SocketFlags.None);  ulong pos = 0;  while (pos < fileSize)  {  int nr = fs.Read(buf, 0, buf.Length);  if (nr <= 0)  {  break;  }  pos += (ulong)nr;  ns = socket.Send(buf, nr, SocketFlags.None);  }  }  catch (Exception exx)  {  bSendOk = false;  MessageBox.Show(exx.Message, "Gửi ảnh bị lỗi!", MessageBoxButtons.OK, MessageBoxIcon.Error);  }  if (fs != null)  {  try { fs.Close(); }  catch (Exception) { }  }  socket.Close();  if (bSendOk)  {  MessageBox.Show("Gửi Thành công !", this.Text, MessageBoxButtons.OK, MessageBoxIcon.Information);  Packet fpck = new Packet();  fpck.MyMessage = "<a href='" + txtsend.Text.Replace(":", "(~\*)") + "'>" + ((extension == ".jpg" || extension == ".png" || extension == ".PNG") ? "<img src='" + txtsend.Text + "' style='max-width:300px'/><br/>" : "") + "<b>" + Path.GetFileName(txtsend.Text) + "</b></a>";  Message fmess = new Message(fpck, Type.SENDER);  ChatLog.Add(fmess);  }  Status = SendType.MESSAGE;  }  #endregion  }  catch (Exception ex) { };  RefreshWeb();  txtsend.Text = "";  txtsend.Focus();  } |

Bảng 4-3: Source code Gửi dữ liệu trong Chat Private

* Chức năng: Xử lý dữ liệu gửi được nhập vào từ người dùng và truyền đi đến người nhận.

**\* Hàm nhận dữ liệu**

* **Chat Public:**

|  |
| --- |
| private void Receiver()  {    IPEndPoint endPoint = new IPEndPoint(IPAddress.Any, port);  AddMessage messageDelegate = MessageReceived;  while (true)  {  byte[] data = receivingClient.Receive(ref endPoint);  #region Nhan File  if (curType == TypePacket.SEND\_FILE)  {  if (curIP == GetLocalIP())  {  curType = TypePacket.NONE;  }  else  {  this.Invoke((MethodInvoker)(() =>  {  SaveFileDialog saveFileDlog1 = new SaveFileDialog();  saveFileDlog1.InitialDirectory = System.Environment.GetFolderPath(Environment.SpecialFolder.MyPictures);  saveFileDlog1.FileName = saveFileName;  saveFileName = System.Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments) + "\\" + saveFileDlog1.FileName;  }));  bool bTranferOk = true;  FileStream fs = null;  try  {  fs = File.Create(saveFileName);  fs.Write(data, 0, data.Length);  }  catch (Exception e)  {  bTranferOk = false;  MessageBox.Show(e.Message, "Nhận file bị lỗi!", MessageBoxButtons.OK, MessageBoxIcon.Error);  }  if (fs != null)  {  try { fs.Close(); }  catch (Exception) { }  }  if (!bTranferOk)  {  try { File.Delete(saveFileName); }  catch (Exception) { }  }  else  {  string[] extension;  extension = saveFileName.Split('.');  this.Invoke((MethodInvoker)(() =>  {  Packet rpck = new Packet();  rpck.MyName = curSender;  rpck.MyMessage = "<a href='" + saveFileName.Replace(":", "(~\*)") + "'>" + (('.' + extension[1] == ".jpg" || '.' + extension[1] == ".png" || extension[1] == ".PNG") ? "<img src='" + saveFileName + "' style='max-width:300px'/><br/>" : "") + "<b>" + Path.GetFileName(saveFileName) + "</b></a>";  Message fmess = new Message(rpck, Type.RECEIVER);  ChatLog.Add(fmess);  RefreshWeb();  }));  }  curType = TypePacket.MESSAGE;  curIP = "";  curSender = "";  }  }  #endregion  #region Xu ly loi  else if (curType == TypePacket.NONE)  {  curType = TypePacket.MESSAGE;  }  #endregion  #region Nhan Image  else if (curType == TypePacket.SEND\_IMAGE)  {  if (curIP == GetLocalIP())  {  curType = TypePacket.NONE;  }  else  {  this.Invoke((MethodInvoker)(() =>  {  SaveFileDialog saveFileDlog1 = new SaveFileDialog();  saveFileDlog1.InitialDirectory = System.Environment.GetFolderPath(Environment.SpecialFolder.MyPictures);  saveFileDlog1.FileName = saveFileName;  saveFileName = System.Environment.GetFolderPath(Environment.SpecialFolder.MyPictures) + "\\" + saveFileDlog1.FileName;  }));  bool bTranferOk = true;  FileStream fs = null;  try  {  fs = File.Create(saveFileName);  fs.Write(data, 0, data.Length);  }  catch (Exception e)  {  bTranferOk = false;  MessageBox.Show(e.Message, "Nhận Image bị lỗi!", MessageBoxButtons.OK, MessageBoxIcon.Error);  }  if (fs != null)  {  try { fs.Close(); }  catch (Exception) { }  }  if (!bTranferOk)  {  try { File.Delete(saveFileName); }  catch (Exception) { }  }  else  {  this.Invoke((MethodInvoker)(() =>  {  Packet rpck = new Packet();  rpck.MyName = curSender;  rpck.MyMessage = "<a href='" + saveFileName.Replace(":", "(~\*)") + "'>" + "<img src='" + saveFileName + "' style='max-width:300px'/><br/>" + "<b>" + Path.GetFileName(saveFileName) + "</b></a>";  Message fmess = new Message(rpck, Type.RECEIVER);  ChatLog.Add(fmess);  RefreshWeb();  }));  }  curType = TypePacket.NONE;  curIP = "";  curSender = "";  }  }  #endregion  #region Nhan Message binh thuong  else  {  Packet receivePacket = DeSerialize(data);  if (receivePacket.MyType == TypePacket.SEND\_INFO\_USER\_1 && receivePacket.MyIP != meIP)  {  this.Invoke((MethodInvoker)(() =>  {  messageDelegate = MessageName;  Invoke(messageDelegate, receivePacket);  }));  }  else if (receivePacket.MyType == TypePacket.SEND\_INFO\_USER\_2 && receivePacket.MyIP != meIP)  {  this.Invoke((MethodInvoker)(() =>  {  bool ck = true;  for (int i = 0; i < lstOnline.Count; i++)  {  string[] ck\_lstOnline = lstOnline[i].Split(new char[] { ' ' }, StringSplitOptions.RemoveEmptyEntries);  if (ck\_lstOnline[1] == receivePacket.MyIP)  ck = false;  }  if (ck)  {  lstFriends.Items.Add(receivePacket.MyName);  lstOnline.Add(receivePacket.MyName + " " + receivePacket.MyIP);  }  }));  }  else if (receivePacket.MyType == TypePacket.MESSAGE)  {  if (receivePacket.MyMessage == "file#")  {  curType = TypePacket.SEND\_FILE;  curSender = receivePacket.MyName;  curIP = receivePacket.MyIP;  saveFileName = receivePacket.MyPort;  }  else if(receivePacket.MyMessage == "image#")  {  curType = TypePacket.SEND\_IMAGE;  curSender = receivePacket.MyName;  curIP = receivePacket.MyIP;  saveFileName = receivePacket.MyPort;  }  else if(receivePacket.MyMessage != "file#" && receivePacket.MyMessage != "image#" )  {  messageDelegate = MessageReceived;  Invoke(messageDelegate, receivePacket);  }  }  else if (receivePacket.MyType == TypePacket.REQUIRE\_CHAT)  {  messageDelegate = AcceptRequireChat;  Invoke(messageDelegate, receivePacket);  }  else if (receivePacket.MyType == TypePacket.OUT\_CHAT)  {  messageDelegate = ClosingMethod;  Invoke(messageDelegate, receivePacket);  }  }  #endregion  }  } |

Bảng 4-4: Source code Nhận dữ liệu trong Chat Form

* **Chat Private:**

|  |
| --- |
| private void ThreadListen()  {  listenSocket.Bind(new IPEndPoint(IPAddress.Any, int.Parse(mePort)));  listenSocket.Listen(4);  while (true)  {    try  {  socket = listenSocket.Accept(); // Chấp nhận kết nối  socket.NoDelay = true;  } catch(Exception) {}  try  {  #region Nhan File  if (isSendFile)  {  this.Invoke((MethodInvoker)(() =>  {  SaveFileDialog saveFileDlog1 = new SaveFileDialog();  saveFileDlog1.FileName = saveFileName;  if (saveFileDlog1.ShowDialog() == DialogResult.OK)  {  saveFileName = saveFileDlog1.FileName;  }  }));  bool bTranferOk = true;  FileStream fs = null;  try  {  fs = File.Create(saveFileName);  byte[] buf = new byte[32 \* 1024];  int nr = socket.Receive(buf, sizeof(ulong), SocketFlags.None);  MemoryStream ms = new MemoryStream(buf);  BinaryReader br = new BinaryReader(ms);  ulong fileSize = br.ReadUInt64();  br.Close();  ms.Close();  ulong pos = 0;  while (pos < fileSize)  {  nr = socket.Receive(buf);  if (nr <= 0)  {  throw new Exception("File bị rỗng!");  }  pos += (ulong)nr;  fs.Write(buf, 0, nr);  }  }  catch (Exception e)  {  bTranferOk = false;  MessageBox.Show(e.Message, "Nhận file bị lỗi!", MessageBoxButtons.OK, MessageBoxIcon.Error);  }  if (fs != null)  {  try { fs.Close(); }  catch (Exception) { }  }  socket.Close();  if (!bTranferOk)  {  try { File.Delete(saveFileName); }  catch (Exception) { }  }  else  {  string[] extension;  extension = saveFileName.Split('.');  MessageBox.Show("Đã nhận !", this.Text, MessageBoxButtons.OK, MessageBoxIcon.Information);  Packet rpck = new Packet();  rpck.MyMessage = "<a href='" + saveFileName.Replace(":", "(~\*)") + "'>" + (('.' + extension[1] == ".jpg" || '.' + extension[1] == ".png" || extension[1] == ".PNG") ? "<img src='" + saveFileName + "' style='max-width:300px'/><br/>" : "") + "<b>" + Path.GetFileName(saveFileName) + "</b></a>";  Message fmess = new Message(rpck, Type.RECEIVER);  ChatLog.Add(fmess);  }  isSendFile = false;  }  #endregion  #region Nhan Image  else if (isSendImage)  {  this.Invoke((MethodInvoker)(() =>  {  SaveFileDialog saveFileDlog1 = new SaveFileDialog();  saveFileDlog1.InitialDirectory = System.Environment.GetFolderPath(Environment.SpecialFolder.MyPictures);  saveFileDlog1.FileName = saveFileName;  saveFileName = System.Environment.GetFolderPath(Environment.SpecialFolder.MyPictures) + "\\"+saveFileDlog1.FileName;  }));  bool bTranferOk = true;  FileStream fs = null;  try  {  fs = File.Create(saveFileName);  byte[] buf = new byte[32 \* 1024];  int nr = socket.Receive(buf, sizeof(ulong), SocketFlags.None);  MemoryStream ms = new MemoryStream(buf);  BinaryReader br = new BinaryReader(ms);  ulong fileSize = br.ReadUInt64();  br.Close();  ms.Close();  ulong pos = 0;  while (pos < fileSize)  {  nr = socket.Receive(buf);  if (nr <= 0)  {  throw new Exception("Không nhận được ảnh!");  }  pos += (ulong)nr;  fs.Write(buf, 0, nr);  }  }  catch (Exception e)  {  bTranferOk = false;  MessageBox.Show(e.Message, "Nhận Image bị lỗi!", MessageBoxButtons.OK, MessageBoxIcon.Error);  }  if (fs != null)  {  try { fs.Close(); }  catch (Exception) { }  }  socket.Close();  if (!bTranferOk)  {  try { File.Delete(saveFileName); }  catch (Exception) { }  }  else  {  string[] extension;  extension = saveFileName.Split('.');  Packet rpck = new Packet();  rpck.MyMessage = "<a href='" + saveFileName.Replace(":", "(~\*)") + "'>" + (('.' + extension[1] == ".jpg" || '.' + extension[1] == ".png" || extension[1] == ".PNG") ? "<img src='" + saveFileName + "' style='max-width:300px'/><br/>" : "") + "<b>" + Path.GetFileName(saveFileName) + "</b></a>";  Message fmess = new Message(rpck, Type.RECEIVER);  ChatLog.Add(fmess);  }  isSendImage = false;  }  #endregion  #region Nhan Message  else // Nhan message binh thuong  {  try  {  byte[] buf = new byte[1024];  socket.Receive(buf);  Packet recievePack = DeSerialize(buf);  if (recievePack.MyType == TypePacket.SEND\_PORT)  {  friendPort = recievePack.MyPort;  textBox2.Text = friendPort;  }  // Gan co bat dau gui file  else if (recievePack.MyType == TypePacket.SEND\_FILE)  {  if (MessageBox.Show(this.Text + " đang gửi một File cho bạn. Bạn có muốn nhận nó không ?", "Nhận File", MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)  { ///  isSendFile = true;  saveFileName = recievePack.MyMessage;  }  }  else if (recievePack.MyType == TypePacket.SEND\_IMAGE)  {  isSendImage = true;  saveFileName = recievePack.MyMessage;  }  // Nhan tin hieu de dong Form  else if (recievePack.MyType == TypePacket.CLOSING)  {  // Dong san socket nghe cua minh de nao muon tat thi tat binh thuong.  socket.Close();  dispose = true;  // Gui RELY  Socket repsck = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp); ;  repsck.Connect(new IPEndPoint(IPAddress.Parse(friendIP), int.Parse(friendPort)));  byte[] buf2 = new byte[1024];  buf2 = Sendpacketclose(TypePacket.REPLY);  repsck.Send(buf2);  repsck.Close();  Message clsmess = new Message(recievePack, Type.CLOSER);  ChatLog.Add(clsmess);  Invoke((MethodInvoker)(() =>  {  txtsend.Enabled = false;  txtsend.ReadOnly = true;  btnSend.Enabled = false;  RefreshWeb();  }));  return;  }  else if (recievePack.MyType == TypePacket.REPLY)  {  listenSocket.Close();  socket.Close();  this.Close();  return;  }  // Nhan mess binh thuong  else  {  string AddEmoji = recievePack.MyMessage;  AddEmoji = AddEmoji.Replace(":)", EmojiList[":)"]);  AddEmoji = AddEmoji.Replace(":(", EmojiList[":("]);  AddEmoji = AddEmoji.Replace(":-s", EmojiList[":-s"]);  AddEmoji = AddEmoji.Replace("x-)", EmojiList["x-)"]);  AddEmoji = AddEmoji.Replace("(y)", EmojiList["(y)"]);  AddEmoji = AddEmoji.Replace(":o", EmojiList[":o"]);  AddEmoji = AddEmoji.Replace(";-(", EmojiList[";-("]);  AddEmoji = AddEmoji.Replace("(p)", EmojiList["(p)"]);  AddEmoji = AddEmoji.Replace(":-t", EmojiList[":-t"]);  recievePack.MyMessage = AddEmoji;  Message mes = new Message(recievePack, Type.RECEIVER);  ChatLog.Add(mes);  }  }  catch(Exception e1)  {  continue;  }  }  #endregion    }  catch (Exception et)  {  MessageBox.Show(et.ToString());  }  socket.Close();  this.Invoke((MethodInvoker)(() =>  {  RefreshWeb();  }));  }  } |

Bảng 4-5: Source code Nhận dữ liệu Chat Private

* Chức năng: Xử lý dữ liệu nhận được và hiện nội dung chat của người gửi lên webrowser.

**\* Hàm thread**

|  |
| --- |
| Thread thread = new Thread(ThreadListen);  thread.Name = "ThreadListen";  thread.IsBackground = true;  thread.Start(); |

Bảng 4-6: Source code Thread

* Chức năng: Tạo Thread mới để thực hiện hàm ThreadListen.

**\* Hàm tạo Chat Private**

|  |
| --- |
| private void lstFriends\_MouseDoubleClick(object sender, MouseEventArgs e)  {  try  {  int num = 0;  string[] Details;  if(lstFriends.SelectedItem != null)  {  num = lstFriends.IndexFromPoint(e.Location);  Details = lstOnline[num].Split(new char[] { ' ' }, StringSplitOptions.RemoveEmptyEntries);  // Gửi "Hi" đến cho ChatForm bên kia  Random rd = new Random();  Packet hipck = new Packet();  hipck.MyMessage = "Xin chào!";  hipck.MyType = TypePacket.REQUIRE\_CHAT;  hipck.MyName = userName;  this.myPort = hipck.MyPort = rd.Next(10000, 60000) + "";  hipck.MyIP = GetLocalIP();  MemoryStream str = new MemoryStream();  BinaryFormatter bformat = new BinaryFormatter();  bformat.Serialize(str, hipck);  byte[] data = new byte[1024];  data = str.ToArray();  InitializePrivateChat(Details[0], Details[1],"0",hipck);  InitializeRequirer(Details[1], data);  }  }  catch (Exception ex)  {  MessageBox.Show("Không tìm thấy tên nào bạn chọn, vui lòng chọn đúng tên!");  }  } |

Bảng 4-7: Source code tạo Chat Private

4.3/ Giao diện

4.3.1/ Màn hình:

* **Màn hình đăng nhập:**

**A screenshot of a cell phone

Description automatically generated**

Hình 4-1: Giao diện Đăng nhập.

⮎ Chức năng: Dùng để đăng nhập vào chương trình.

* **Màn hình chính (Chat public):**

A screenshot of a cell phone

Description automatically generated

Hình 4-2: Giao diện Chat Public.

⮎ Chức năng: Chứa tất cả các chức năng chính của chương trình.

* **Màn hình Chat Private:**

**A screenshot of a cell phone

Description automatically generated**

Hình 4-3: Giao diện Chat Private.

⮎ Chức năng: Khi người dùng chọn tên bạn bè trong danh sách bạn bè, màn hình này sẽ xuất hiện và cho phép hai người dùng này chat riêng với nhau.

* **Màn hình Giới thiệu và trợ giúp:**

A screenshot of a social media post

Description automatically generated

Hình 4-4: Giao diện Giới thiệu và trợ giúp.

⮎ Chức năng: Giao diện gồm 2 TabPage:

+ Help: chứa một số câu hỏi thường gặp và hướng dẫn người dùng sử dụng phần mềm.

+ About: giới thiệu phần mềm, tính năng, phiên bản, nhóm tác giả và thông tin liên hệ để người dùng liên lạc khi cần giúp đỡ.

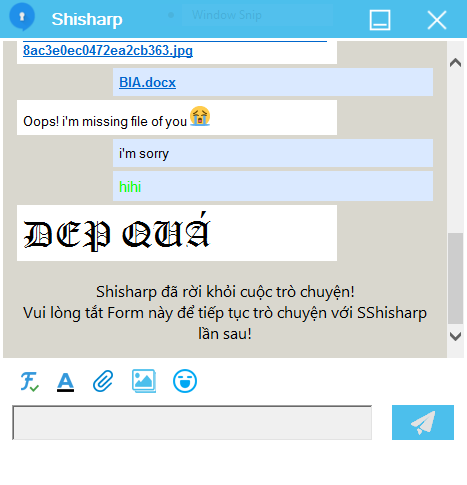
4.3.2/ Các tính năng

* **Font & Color:**

**A screenshot of a social media post

Description automatically generated**

Hình 4-5: Tính năng chọn Font.

****

Hình 4-6: Tính năng Font & Color

* **Các tính năng gửi:**

A screenshot of a cell phone

Description automatically generated

Hình 4-7: Tính năng gửi file & hình ảnh trong Chat Public.

**A screenshot of a cell phone

Description automatically generated**

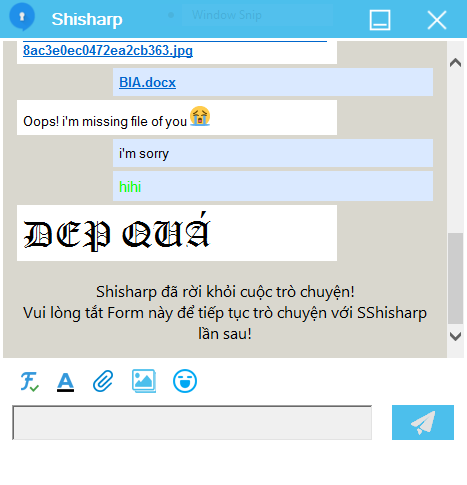
Hình 4-8: Tính năng gửi file & hình ảnh trong Chat Private.

**\* Closed:**

**A screenshot of a social media post

Description automatically generated**

Hình 4-9: Giao diện khi đóng chương trình.

****

Hình 4-10: Giao diện khi đóng Chat Private.

PHẦN 5/ CÀI ĐẶT – THỬ NGHIỆM

5.1/ Yêu cầu hệ thống:

⮚ Máy tính phải cài đặt .Net Framework 4.5 trở lên.

⮚ Máy tính phải có kết nối mạng LAN.

5.2/ Hướng dẫn cài đặt:

* Bước 1: Nhấp đúp chuột vào tập tin setup.exe để tiến hành cài đặt, màn hình Welcome hiện ra, nhấn Next để tiếp tục cài đặt.

**A screenshot of a cell phone

Description automatically generated**

Hình 5-1: Hộp thoại Welcome.

* Bước 2: Màn hình Select Installation Folder hiện ra, chọn “Browse…” và tìm đường dẫn muốn lưu chương trình. Nhấn Next để tiếp tục.

**A screenshot of a cell phone

Description automatically generated**

Hình 5-3: Hộp thoại Select Installation Folder

* Bước 3: Màn hình Confirm Installation hiện lên, nhấn Next.

**A screenshot of a cell phone

Description automatically generated**

Hình 5-2: Hộp thoại Confirm Installation

* Bước 4: Khi đến màn hình Installation Complete, nhấn Close để hoàn thành việc cài đặt.

**A screenshot of a cell phone

Description automatically generated**

Hình 5-4: Hộp thoại Installation Complete

5.3/ Hướng dẫn gỡ cài đặt:

* Bước 1: Nhấn Start -> Tìm “UninstallSkyChat”
* Bước 2: Một hộp thoại hiện ra, nhấn OK và chờ để hoàn tất việc gỡ cài đặt.

PHẦN 6/ KẾT LUẬN:

6.1 Kết quả đạt được:

Đề tài “SkyChat” đã thực hiện được các nội dung sau:

⮚ Tìm hiểu được phương thức lập trình Socket và lập trình đa luồng trên môi trường .NET.

⮚ Xây dựng được một chương trình Chat bằng mô hình P2P đơn giản cài

đặt trên mạng Lan với các tính năng như: gửi tin nhắn, file, hình ảnh, emotion trực tiếp giữa các Users, lưu danh sách bạn bè, cho phép nhiều người chat công khai với nhau hoặc chat riêng giữa hai người…

⮚ Tuy nhiên, sử dụng UDP cho chat public có nguy cơ mất tin nhắn do giao thức UDP xảy ra mất gói.

6.2 Hướng phát triển

Về cơ bản, nghiên cứu đã đạt được những yêu cầu đề ra. Tuy nhiên nếu có thêm thời gian, đồ án sẽ cố gắng phát triển thêm các chức năng sau:

⮚ Hỗ trợ voice chat, video call giữa các Users trong mạng Lan.

⮚ Hỗ trợ gửi và xem tin nhắn Offline.

⮚ Hỗ trợ tìm kiếm và nghe nhạc trực tuyến.

TÀI LIỆU THAM KHẢO

# [1] <https://www.codeproject.com/Articles/568476/Creating-an-MSI-Setup-Package-for-Csharp-Windows>

[2] <https://stackoverflow.com/>

[3] <https://docs.microsoft.com/en-us/dotnet/api/system.net.sockets.socket?view=netframework-4.7.2>

-------------Hết-------------