ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO ĐỒ ÁN MÔN HỌC

MẠNG MÁY TÍNH

ĐỀ TÀI: Điều khiển máy tính thông qua ứng dụng Desktop

Giảng viên lý thuyết: Thầy Đỗ Hoàng Cường
Giảng viên thực hành: Cô Huỳnh Thuỵ Bảo Trân
Lớp: 22TNT1TN
Thành viên thực hiện:

- 22120025 – Nguyễn Long Bảo
- 22120044 – Nguyễn Cao Cường
- 22120049 – Tạ Chí Thành Danh

THÀNH PHỐ HỒ CHÍ MINH, THÁNG 12 NĂM 2023

Mục lục

Danh sách hình vẽ

Danh sách bảng

1 Giới thiệu

1.1 Tổng quan

Ứng dụng cho phép một quản trị viên hoặc người dùng (sử dụng ứng dụng với vai trò là **Admin** hay máy Client) điều khiển một máy tính từ xa (máy Server) trong cùng mạng LAN với máy Server. Ứng dụng sử dụng **socket**, giao thức **TCP** ở tầng Transport, và được lập trình bằng ngôn ngữ C++. Dưới đây là các chức năng của ứng dụng:

- Kết nối đến máy Server qua địa chỉ IP: Cho phép thiết lập kết nối tới máy chủ thông qua việc nhập địa chỉ IP.
- Đăng nhập với vai trò admin hoặc user bình thường: Cung cấp tính năng đăng nhập với vai trò quản trị hoặc người dùng thông thường, đảm bảo quyền truy cập phù hợp.
- Ngắt kết nối với một địa chỉ IP: Cho phép ngắt kết nối tới máy tính có địa chỉ IP cụ thể trong cùng mạng LAN.
- Chia sẻ màn hình hiện tại: Hiển thị và chia sẻ màn hình hiện tại của máy tính Server.
- Gửi tín hiệu chuột đến máy bị điều khiển: Điều khiển các thao tác chuột từ xa đến máy tính được điều khiển.
- Gửi tín hiệu bàn phím đến máy bị điều khiển: Cho phép điều khiển bàn phím từ xa trừ các tổ hợp phím đặc biệt như Ctrl + Alt + Del.

1.2 Về Socket

Socket là là một giao diện lập trình ứng dụng (API) cho việc giao tiếp mạng giữa các ứng dụng mạng. Socket cung cấp cơ chế gửi và nhận dữ liệu, thiết lập kết nối và quản lý các thông tin liên quan đến mạng như địa chỉ IP và Port number.

Ứng dụng sử dụng socket hướng kết nối: dựa trên giao thức TCP, 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. Đồng thời, mỗi thông điệp gửi phải có xác nhận trả về và các gói tin chuyển đi tuần tự.

1.3 Công cụ

1.3.1 wxWidgets

wxWidgets là một thư viện mạnh mẽ và linh hoạt cho phát triển giao diện người dùng đa nền tảng bằng ngôn ngữ lập trình C++. Trong đồ án này, wxWidgets được sử dụng để thiết kế giao diện cho ứng dụng, đồng thời hỗ trợ việc xử lí các sự kiện từ phía Client và Server trong quá trình gửi và nhận các thông tin qua socket.

Thư viện này cung cấp các thành phần giao diện đồ hoạ như cửa sổ, nút bấm, hộp thoại và menu, giúp tạo ra giao diện người dùng trực quan và dễ sử dụng. Đặc biệt,

wxWidgets cho phép triển khai ứng dụng trên nhiều hệ điều hành khác nhau một cách dễ dàng mà không cần phải viết lại mã nguồn.

Việc sử dụng wxWidgets giúp nhóm tập trung vào việc thiết kế giao diện và chức năng điều khiển từ xa mà không cần quá lo lắng về sự không tương thích giữa các nền tảng hệ điều hành. Thư viện cũng cung cấp tài liệu phong phú và có cộng đồng hỗ trợ đông đảo, giúp việc học và phát triển ứng dụng trở nên dễ dàng hơn.

1.3.2 Asio

Thư viện Asio là một thư viện mã nguồn mở mạnh mẽ cho ngôn ngữ lập trình C++, được sử dụng để thực hiện việc lập trình mạng và giao tiếp qua mạng một cách linh hoạt và hiệu quả.

Asio cung cấp các công cụ và lớp trừu tượng để tạo, quản lý và tương tác với các kết nối mạng, bao gồm cả việc xử lý socket, thiết lập kết nối, gửi và nhận dữ liệu trên mạng. Đặc biệt, nó hỗ trợ cả các giao thức đồng bộ (ví dụ như TCP) và không đồng bộ (ví dụ như UDP), cho phép việc truyền dữ liệu một cách đáng tin cậy và nhanh chóng.

Asio được thiết kế để linh hoạt và dễ sử dụng, với cách tiếp cận dựa trên bất đồng bộ (asynchronous), cho phép thực hiện các hoạt động mạng mà không cần chờ đợi kết quả trả về từ mỗi yêu cầu riêng biệt. Điều này làm cho việc xây dựng ứng dụng mạng với hiệu suất cao trở nên dễ dàng hơn, đặc biệt trong việc xử lý hàng loạt yêu cầu từ nhiều nguồn khác nhau.

Trong đồ án này, nhóm tập trung sử dụng chủ yếu với chức năng thiết lập kết nối TCP của nó. Sự linh hoạt và hiệu suất của Asio trong việc xử lý giao tiếp mạng bất đồng bộ đã hỗ trợ nhóm trong việc xây dựng và quản lý việc truyền dữ liệu đáng tin cậy giữa các thiết bị qua mạng.

1.3.3 Windows API

Windows API (Application Programming Interface) là bộ công cụ mạnh mẽ của hệ điều hành Windows, cung cấp chức năng và dịch vụ cần thiết cho việc tương tác ứng dụng Windows với hệ thống và phần cứng.

Trong quá trình phát triển ứng dụng Remote Desktop Control, nhóm đã tận dụng tính linh hoạt của Windows API để thực hiện các thao tác điều khiển từ xa. Chúng tôi đã sử dụng Windows API để gửi các yêu cầu tương ứng với các thao tác như nhấn phím, di chuyển con chuột, và thực hiện các thao tác click chuột trên máy tính được điều khiển.

Hơn nữa, thông qua Windows API, chúng tôi đã có thể truy xuất và lấy thông tin cơ bản của máy tính được kết nối, như địa chỉ IP và địa chỉ MAC, giúp quản lý và xác định máy tính từ xa một cách chính xác.

Bằng cách tích hợp các tính năng linh hoạt của Windows API vào ứng dụng Remote Desktop Control, chúng tôi đã tạo ra một trải nghiệm điều khiển từ xa hiệu quả và ổn định, mang lại sự linh hoạt và tiện ích cho người dùng.

1.4 Các link

1.4.1 Link video demo

Chèn link demo vào đây

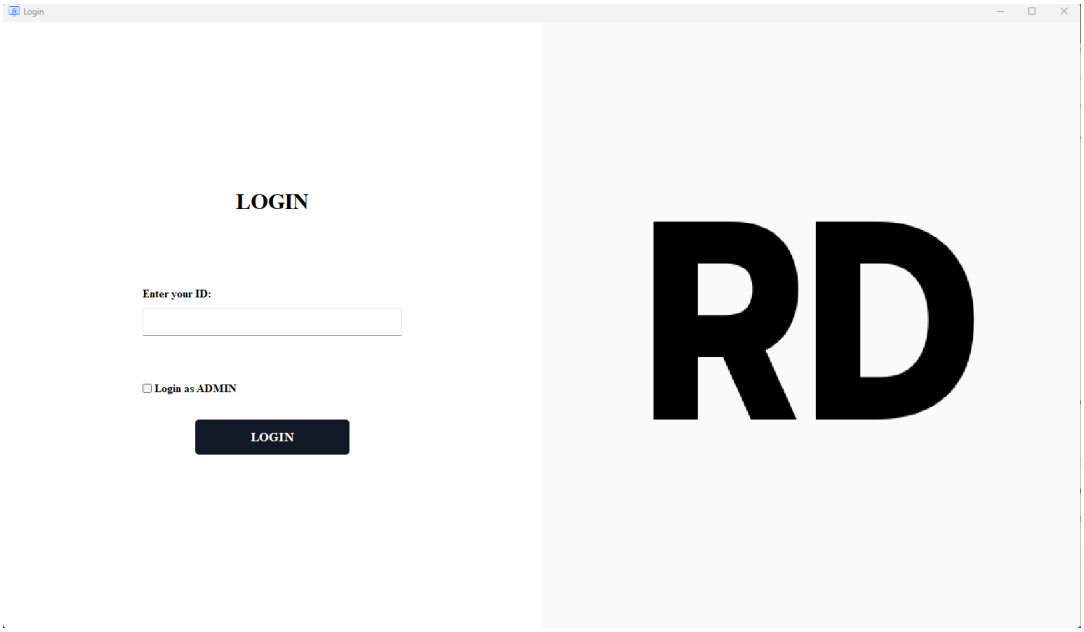
1.4.2 Link mã nguồn

Còn đây là link mã nguồn

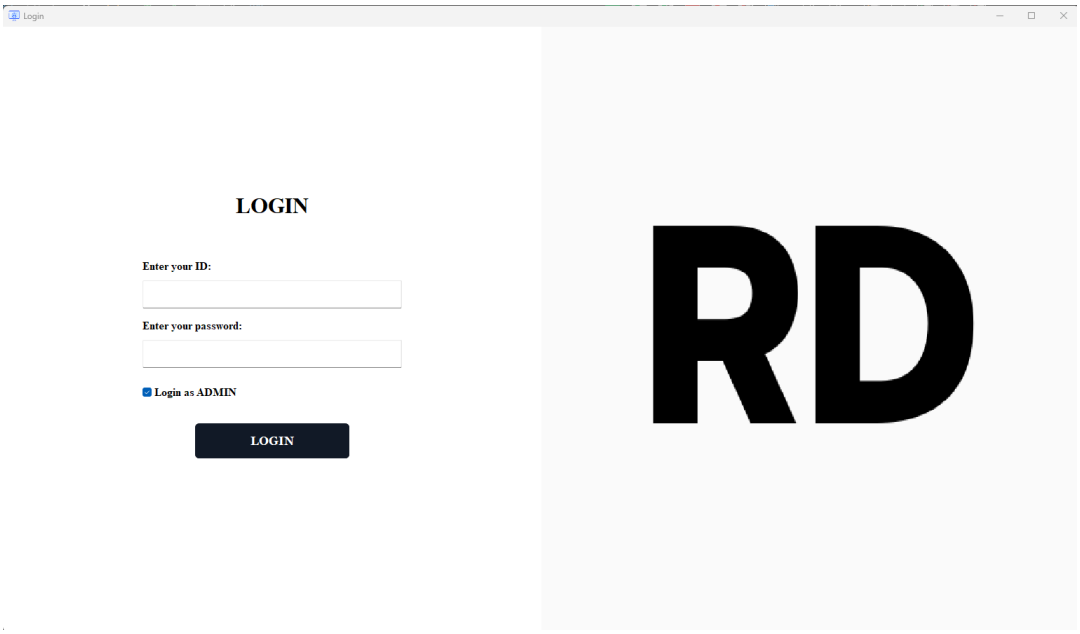
2 Các chức năng của phần mềm

2.1 Giao diện của ứng dụng khi khởi động

Khi mở ứng dụng, giao diện ban đầu của ứng dụng là màn hình đăng nhập, ở đây ta có thể đăng nhập với 2 vai trò đó là User bình thường (Server) hoặc Admin (Client). Mặc định thì ứng dụng sẽ hiển thị ô đăng nhập với vai trò là User, để đăng nhập với vai trò Admin thì ta tích vào ô "**Login as ADMIN**". (Hình 3 và Hình 2)



Hình 1: Màn hình đăng nhập của User



Hình 2: Màn hình đăng nhập của Admin

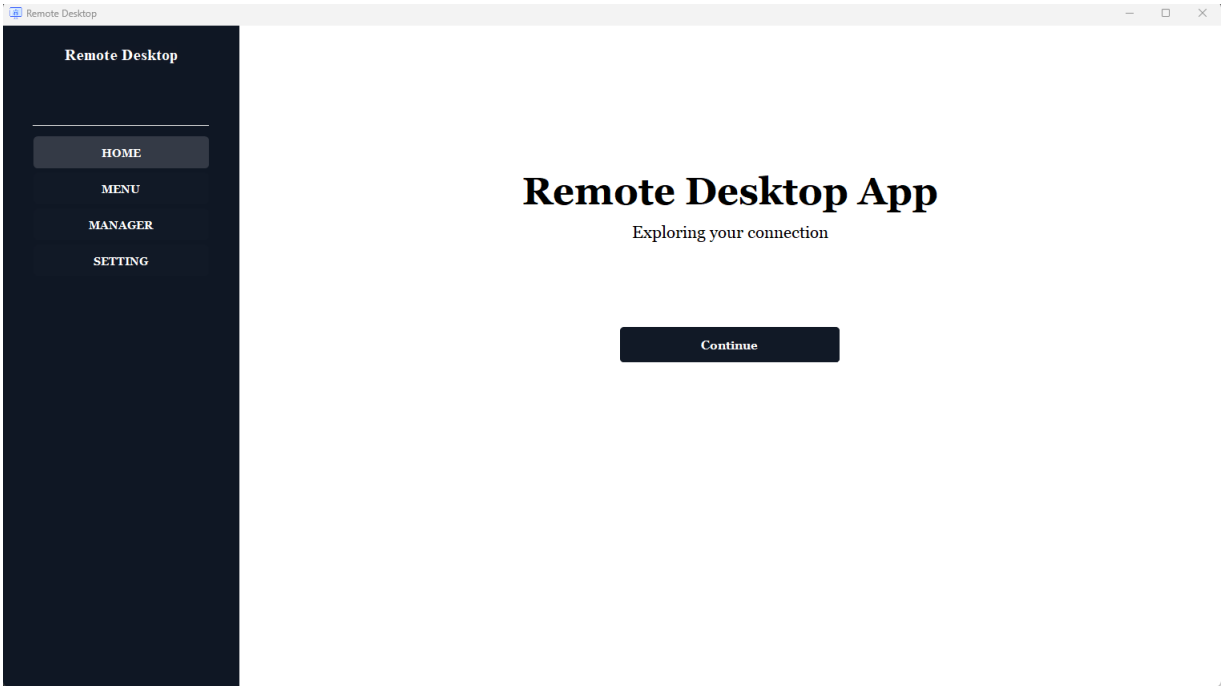
Để đăng nhập được với vai trò là User, ta cần nhập tên đăng nhập là một chuỗi ký

tự ASCII với độ dài từ 4 đến 10 ký tự. Còn đối với Admin thì tên đăng nhập và mật khẩu mặc định là `admin`.

2.2 Giao diện của ứng dụng sau khi đăng nhập

2.2.1 Giới thiệu

Sau khi đăng nhập xong, ứng dụng chuyển sang giao diện chính, đây là nơi người dùng sử dụng các chức năng chính của ứng dụng. Màn hình chính sau khi đăng nhập xong như sau:



Hình 3: Màn hình chính của ứng dụng

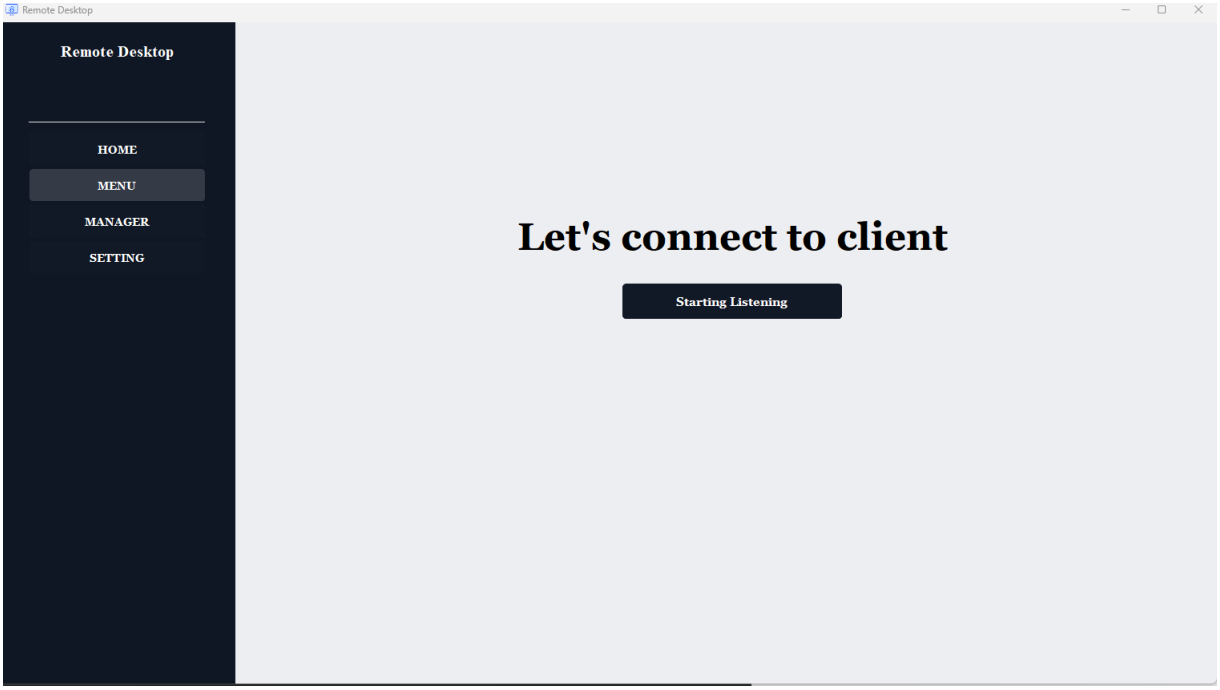
Ở thanh điều hướng, ta có 4 nút chức năng đó là nút **Home**, **Menu**, **Manager**, **Setting**. Nội dung của màn hình **Home** là màn hình chính sau khi đăng nhập vừa trình bày ở trên, còn giao diện của các màn hình tương ứng với các nút sẽ được mô tả ở những mục tiếp theo.

Khi người dùng ấn nút Menu trên cửa sổ chính của ứng dụng, ở khu vực hiển thị bên phải sẽ thay đổi tuỳ theo việc người dùng đang đăng nhập với vai trò là **Admin** (tức **Client**) hay **User thông thường** (tức **Server**).

2.3 Màn hình Menu của Server

2.3.1 Mô tả

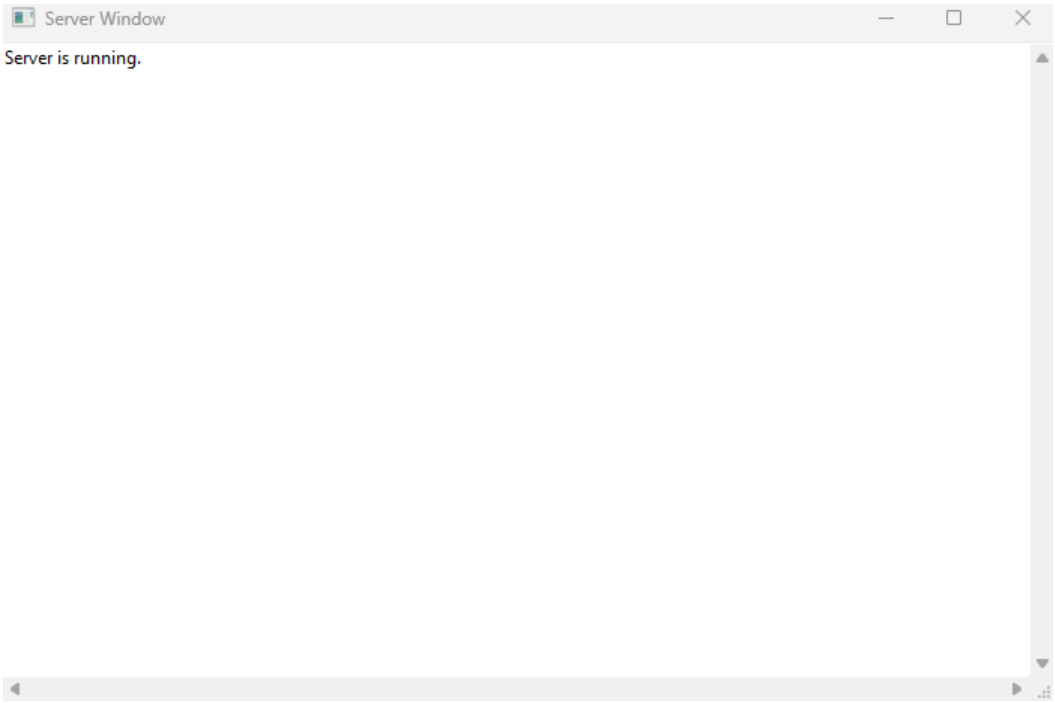
Dưới đây là nội dung hiển thị của màn hình Menu tương ứng với vai trò **User** (**Server**). Vì **Server** chỉ có vai trò gửi nhận và thực hiện các yêu cầu mà **Client** gửi đến, do đó ở màn hình Menu ứng dụng chỉ hỗ trợ 1 chức năng đó là khởi động **Server** để chờ kết nối đến từ **Client** (nút **Start Listening**. (Hình 4)



Hình 4: Màn hình Menu ứng với vai trò User thông thường (Server)

2.3.2 Chờ kết nối từ Client (Nút Start Listening)

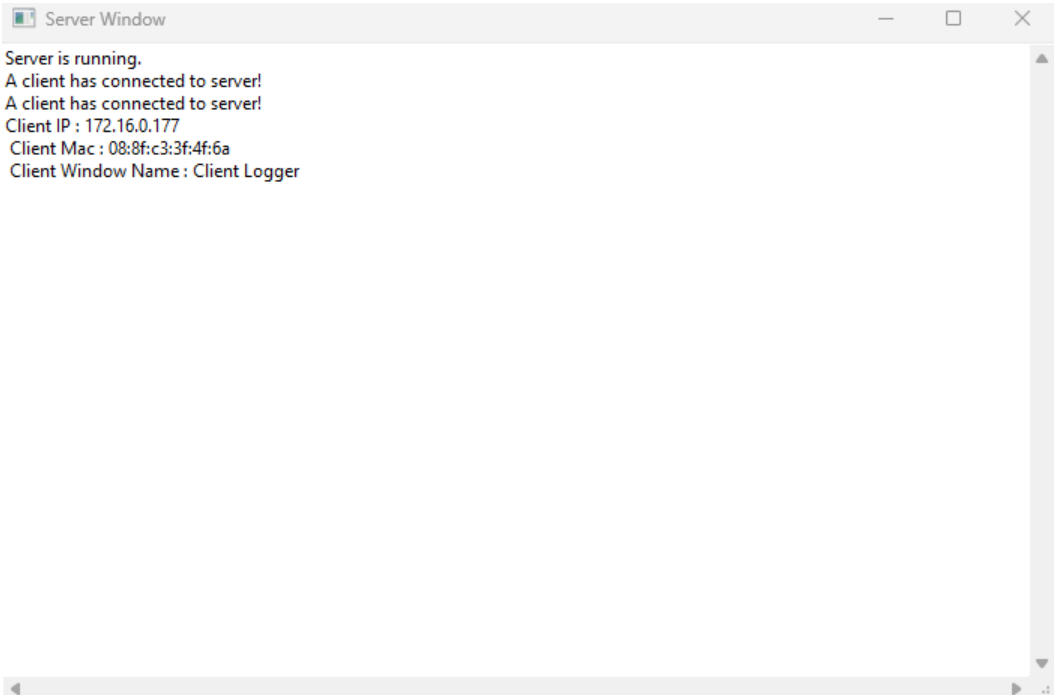
Khi ta ấn vào nút **Start Listening**, một cửa sổ với tiêu đề là **Server Window** hiện lên. Đây là cửa sổ text hiển thị tình trạng của **Server**, ở trạng thái ban đầu cửa sổ sẽ hiển thị mỗi dòng chữ "Server is running". (Hình 5)



Hình 5: Cửa sổ Server Window lúc đầu

Nếu có **Client** kết nối đến **Server**, khi đó cửa sổ **Server Window** sẽ thông báo là Client đã kết nối đến Server thông qua dòng chữ "A client has connected to server!" và

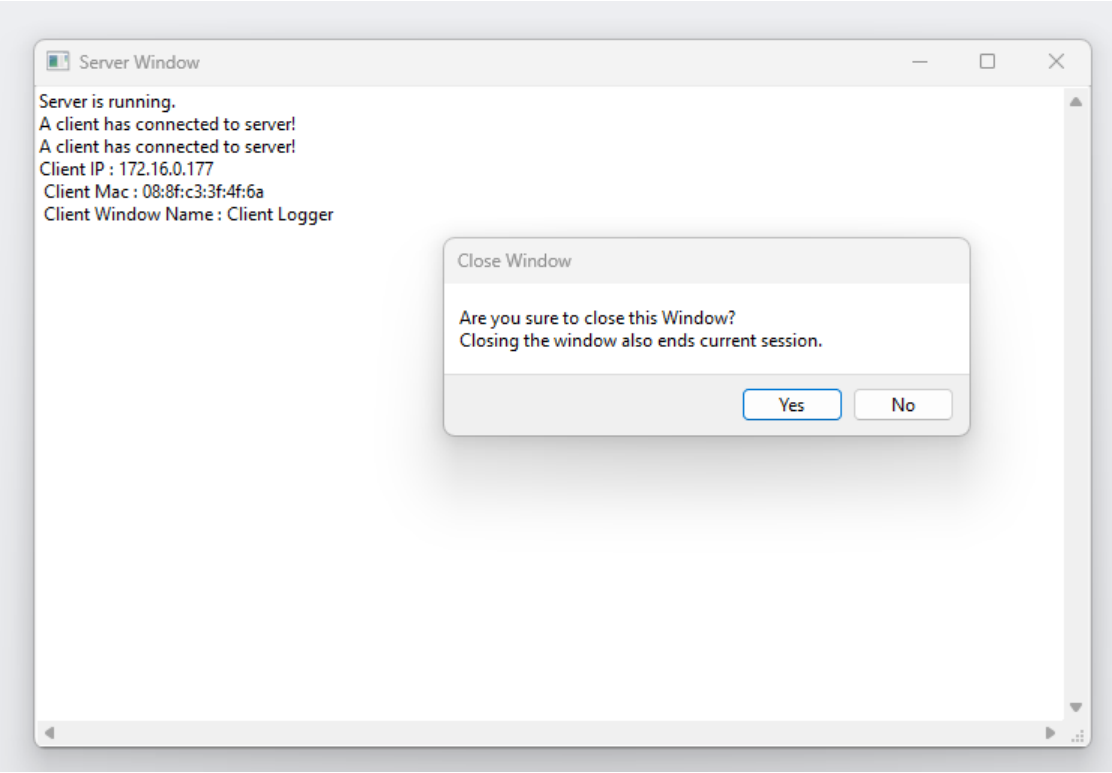
theo sau là những thông tin cơ bản về địa chỉ IP, địa chỉ MAC và tên cửa sổ gửi tín hiệu kết nối của Client. Ở trong hình bên dưới, Server hiển thị 2 dòng chữ đã kết nối vì bên phía Client thiết lập 2 kết nối đến Server, một cái dùng để nhận hình ảnh màn hình từ phía Server và cái còn lại dùng để gửi tín hiệu điều khiển đến máy Server. (Hình 6)



Hình 6: Cửa sổ Server Window khi có kết nối đến từ Client

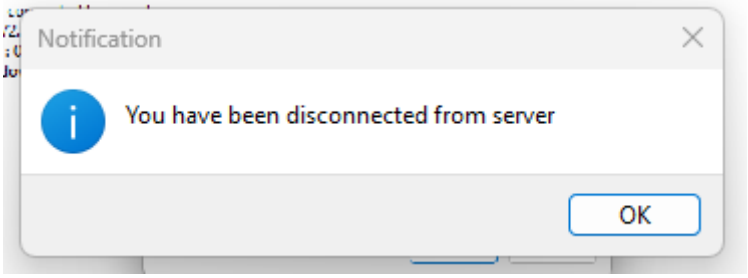
2.3.3 Tắt Server

Để tắt Server, ta chỉ cần nhấn vào nút **X** ở góc phải trên của cửa sổ, khi ta nhấn vào nút này thì một hộp thoại sẽ hiển thị để yêu cầu ta xác nhận về việc có đóng cửa sổ "Server Window" và đồng thời tắt Server luôn hay không (Hình 7).



Hình 7: Hộp thoại đóng/tắt Server

Nếu ta chọn **No**, Server vẫn tiếp tục hoạt động bình thường còn nếu ta chọn **Yes**, Server sẽ gửi thông báo đến Client về việc ngắt kết nối và yêu cầu Client ngắt toàn bộ các kết nối đã thiết lập với Server, sau đó Server với tắt hẳn. Sau khi Server gửi yêu cầu ngắt kết nối đến Client thành công, cửa sổ Client sẽ ngắt các kết nối mà nó đã thiết lập với Server và một thông báo sẽ hiện ra ở bên máy của Client như sau (Hình 8):

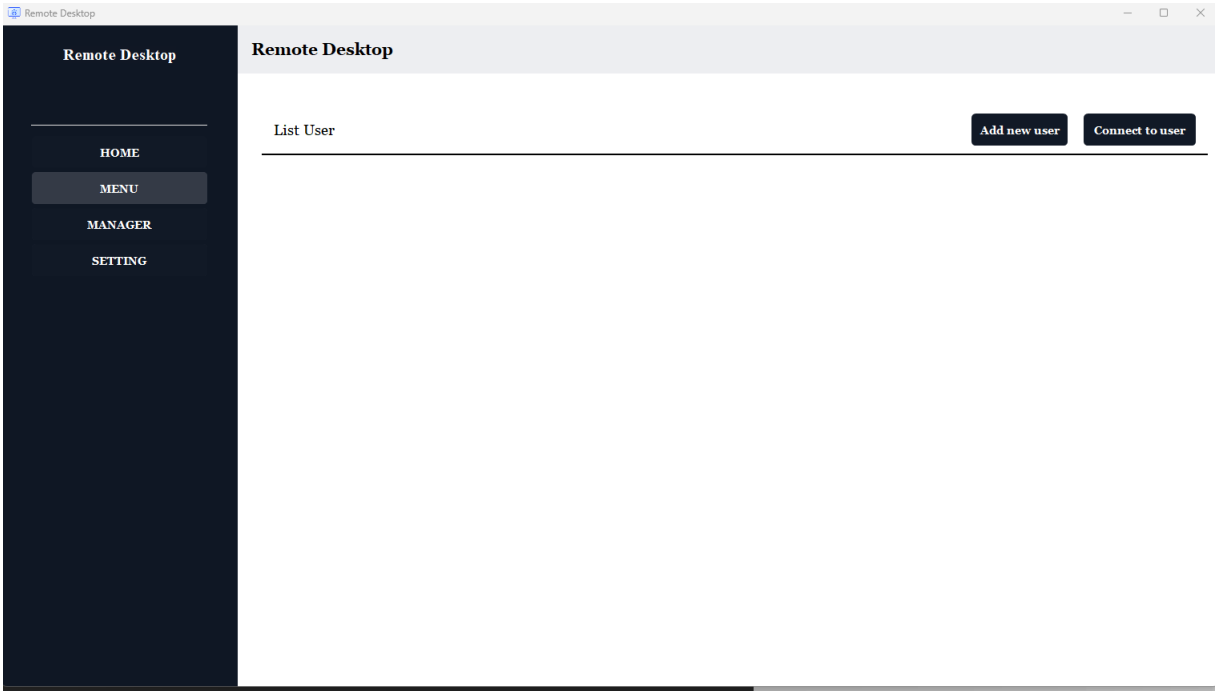


Hình 8: Hộp thoại đóng/tắt Server

2.4 Màn hình Menu của Client

2.4.1 Mô tả

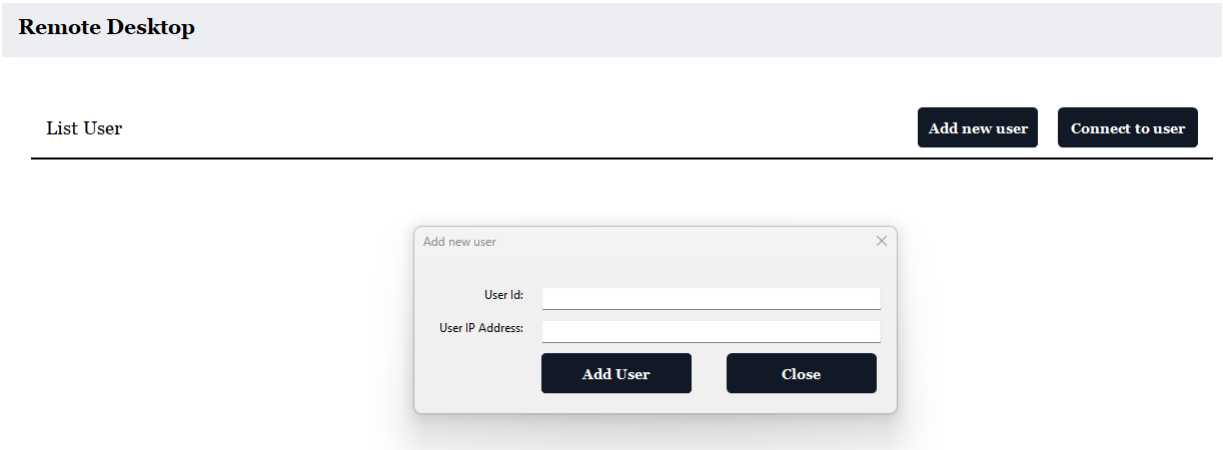
Dưới đây là nội dung hiển thị của màn hình Menu tương ứng với vai trò **Admin** (**Client**). Ở màn hình menu của Client, ứng dụng hỗ trợ 2 chức năng cơ bản: Thêm một người dùng mới vào danh sách kết nối (nút **Add new user**), kết nối trực tiếp đến một người dùng nào đó (**Connect to user**). (Hình 9)



Hình 9: Màn hình chính của ứng dụng

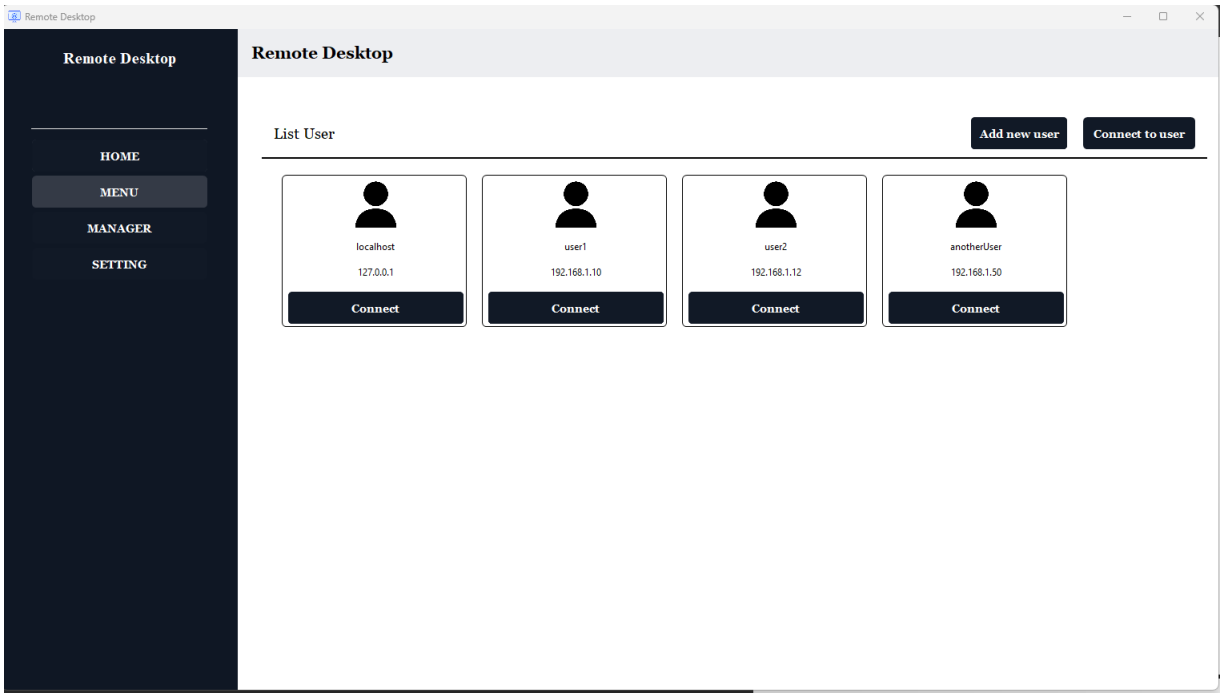
2.4.2 Thêm một người dùng mới vào danh sách các kết nối (nút Add new user)

Khi ta sử dụng chức năng **Add new user**, hộp thoại **Add new user** xuất hiện với 2 trường thông tin cần nhập đó là "User id" và "User IP Address". Trường "User id" yêu cầu ta nhập tên nhận dạng, trường này có kiểu dữ liệu là `std::string`, mục đích của việc điền thông tin này là để tạo lưu lại tên cụ thể trên danh sách các User nằm ở bên dưới để cho ta dễ nhận biết tên User ta muốn kết nối. Ở trường "User IP Address", ta nhập địa chỉ IP của máy cần được kết nối (Hình 10).



Hình 10: Hộp thoại Add new user

Sau khi nhập xong, ta ấn nút "Add user" ở góc trái để thêm một người dùng mới vào danh sách kết nối bên dưới. Sau khi thêm một vài người dùng vào danh sách kết nối thì màn hình Menu sẽ có hiển thị như sau. (Hình 11)

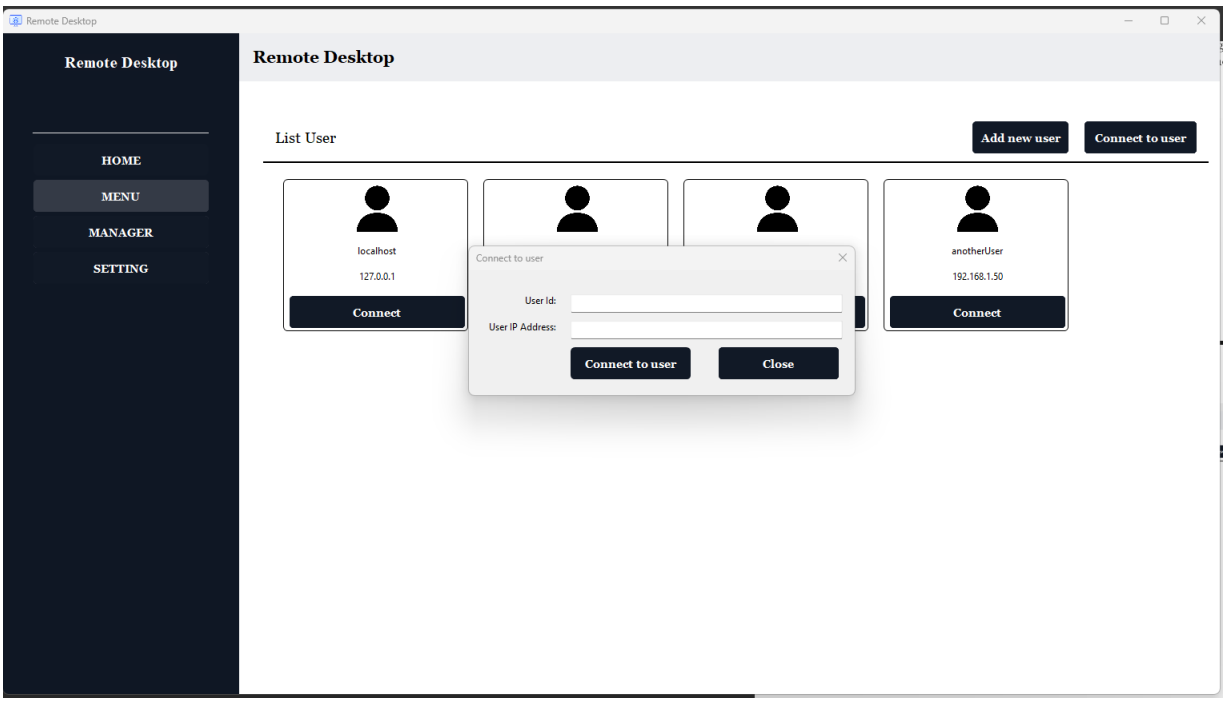


Hình 11: Danh sách các user sau khi nhập

2.4.3 Kết nối đến Server (nút Connect/Connect to user)

Để kết nối đến server, ta có 2 cách để thực hiện: Sử dụng nút "Connect to user" ngay bên cạnh "Add new User" hoặc nút "Connect" ngay bên dưới user tương ứng trong danh sách User trong hình 11.

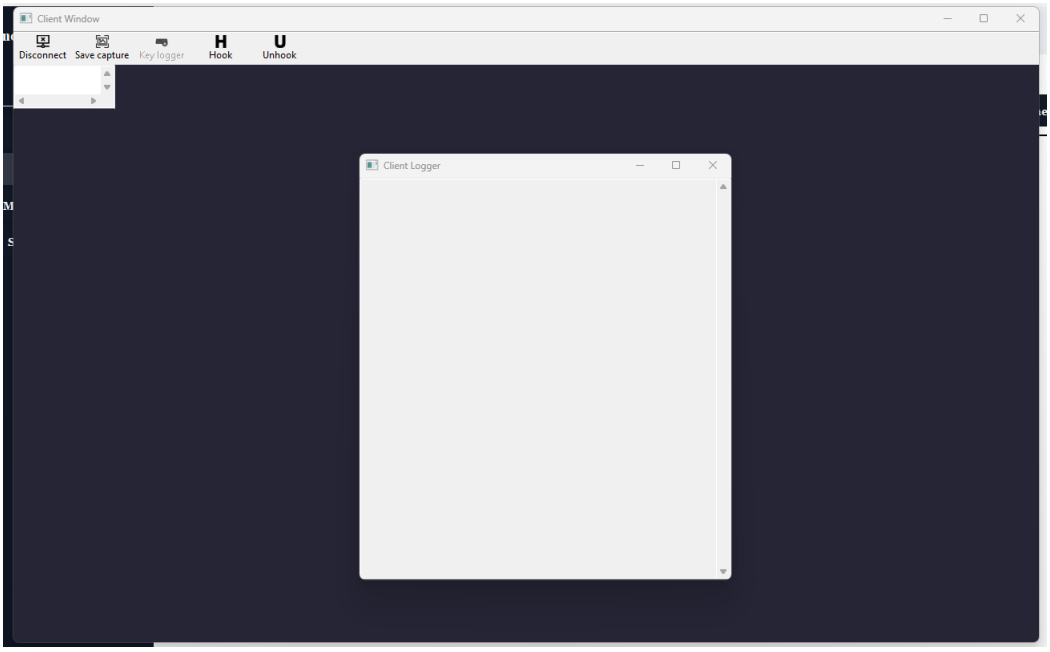
Nếu ta sử dụng lựa chọn "Connect to user", hộp thoại hiện lên với cấu trúc tương tự hộp thoại "Add new user", chỉ khác là ở góc trái hiển thị lựa chọn "Connect to user". (Hình 12)



Hình 12: Hộp thoại Connect to user

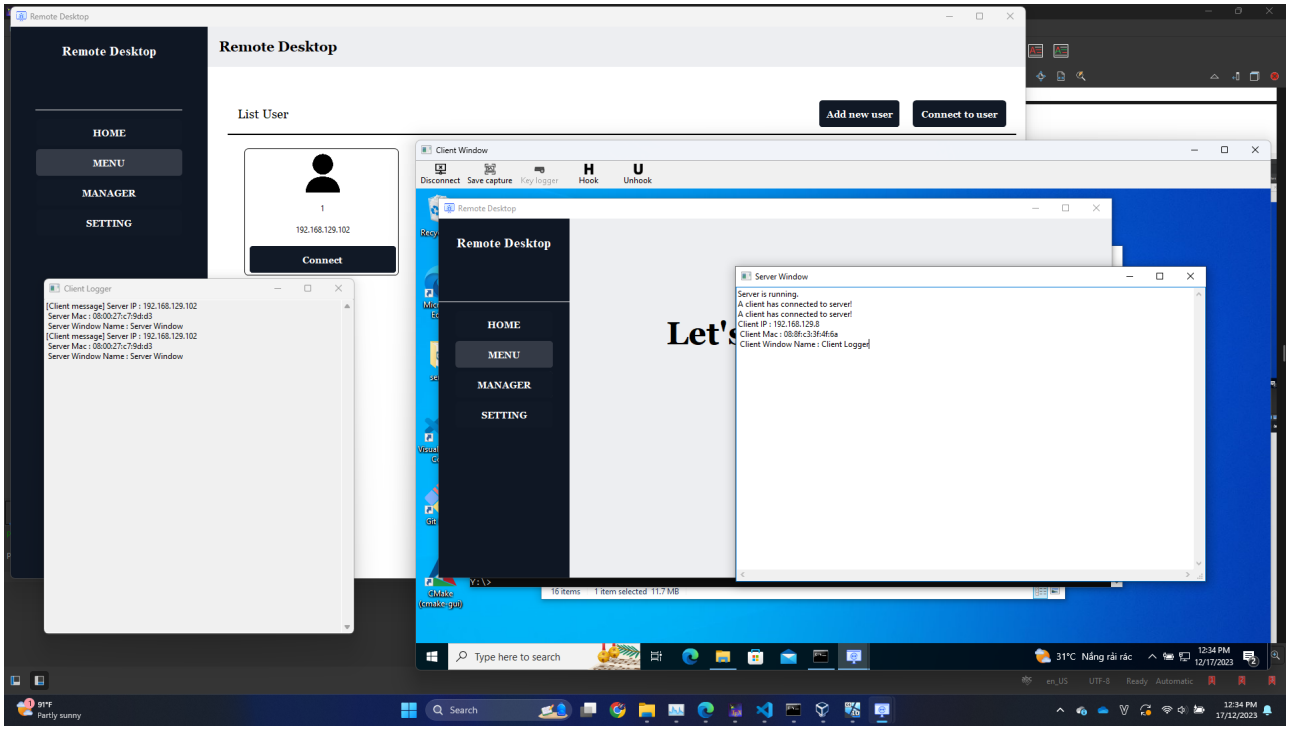
Nếu ta sử dụng lựa chọn "Connect" bên dưới user cần kết nối trong danh sách User, ta chỉ nhấn vào nút "Connect" tương ứng.

Sau khi nhập địa chỉ IP thích hợp và nhấn kết nối, sẽ có hai cửa sổ hiện ra với tiêu đề tương ứng là "Client Logger" và "Client Window". Nếu kết nối thất bại, 2 cửa sổ này sẽ không có thông tin gì trên đó, cửa sổ Logger sẽ không ghi thông tin gì và cửa sổ "Client Window" sẽ không hiển thị màn hình của máy Server (Hình 13). Khi gặp trường hợp trên, ta tắt cửa sổ "Client Window" để đóng kết nối hiện tại và quay lại cửa sổ Menu sử dụng lệnh "Connect" để thiết lập kết nối mới.



Hình 13: Cửa sổ Client Window khi kết nối không thành công

Nếu kết nối thành công, cửa sổ "Client Window" sẽ hiển thị màn hình của máy được kết nối, trong khi cửa sổ "Client Logger" sẽ cung cấp thông tin cơ bản về máy chủ được kết nối (Hình 14). Thông tin này bao gồm địa chỉ IP, địa chỉ MAC và tên cửa sổ của máy chủ. Ở đây, thông tin về máy chủ xuất hiện hai lần vì client thiết lập hai kết nối tới server: một kết nối được sử dụng để nhận hình ảnh và một kết nối khác được dùng để truyền tín hiệu điều khiển.



Hình 14: Cửa sổ Client Window khi đã thiết lập được kết nối đến một Server

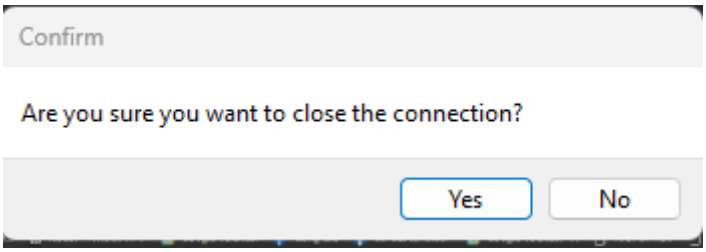
Một số lưu ý khi kết nối:

- Kết nối 2 ứng dụng trên 2 máy khác nhau trong cùng mạng: người dùng phía Client cần biết trước IP của máy Server. IP của máy Server có thể lấy ra bằng lệnh `ipconfig` của **Command Prompt**.
- Kết nối 2 ứng dụng trên cùng 1 máy: ngoài cách trên, còn có thể nhập địa chỉ loopback **127.0.0.1**. Đây là địa chỉ IP đặc biệt được dùng để tham chiếu đến chính máy tính đang sử dụng. Khi gửi dữ liệu đến địa chỉ loopback, dữ liệu sẽ được gửi và xử lý trên cùng một máy tính mà không cần thông qua mạng vật lý.
- Mỗi Client **chỉ điều khiển** được 1 Server ở tại một thời điểm. Điều này có nghĩa là, khi muốn điều khiển máy khác, Client phải ngắt kết nối với Server hiện tại để có thể thiết lập kết nối mới với máy khác.

2.4.4 Ngắt kết nối với Server (nút Disconnect)

Để sử dụng chức năng này, trước hết phía người dùng phía Client cần nhấn nút "Disconnect" trên thanh Toolbar của Client Window. Một hộp thoại lúc này sẽ hiện ra, yêu cầu ta xác nhận về việc có ngắt kết nối đến Server và đồng thời tắt Client Window

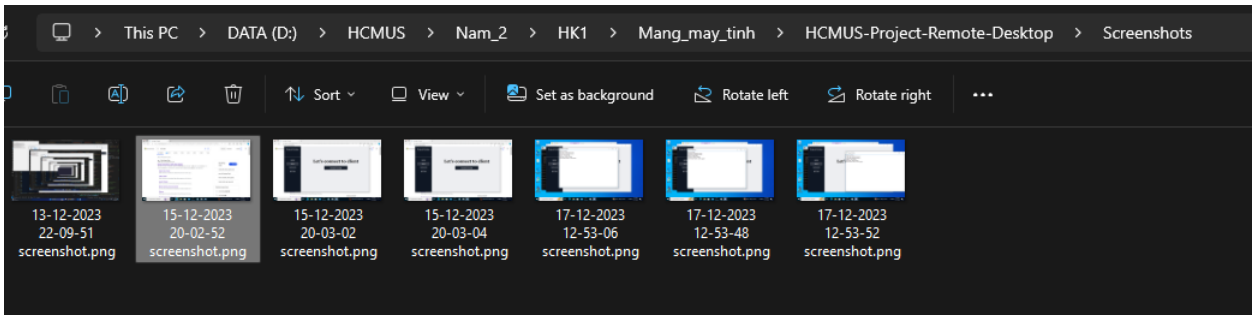
hay không (Hình 15). Nếu xác nhận ngắt kết nối, cửa sổ Client Window và Client Logger sẽ được đóng đồng thời kết nối từ Client đến Server sẽ không còn.



Hình 15: Thông báo ở máy Client sau khi Server đóng

2.4.5 Chụp màn hình Server (nút Save Capture)

Để sử dụng chức năng này, trước hết người dùng phía Client cần nhấn nút "Save Capture". Sau khi ấn xong, hình ảnh sẽ được lưu vào thư mục "CURRENTFOLDER/ Screenshots" với "CURRENTFOLDER" là thư mục đang đứng để gọi ứng dụng Remote Desktop Control (Hình 16). Hình ảnh được lưu với cú pháp `dd-mm-yyyy hh-mm-ss screenshot.png`.



Hình 16: Thông báo ở máy Client sau khi Server đóng

2.5 Màn hình Manager

chưa code xong nên chưa mô tả

2.6 Cửa sổ Setting

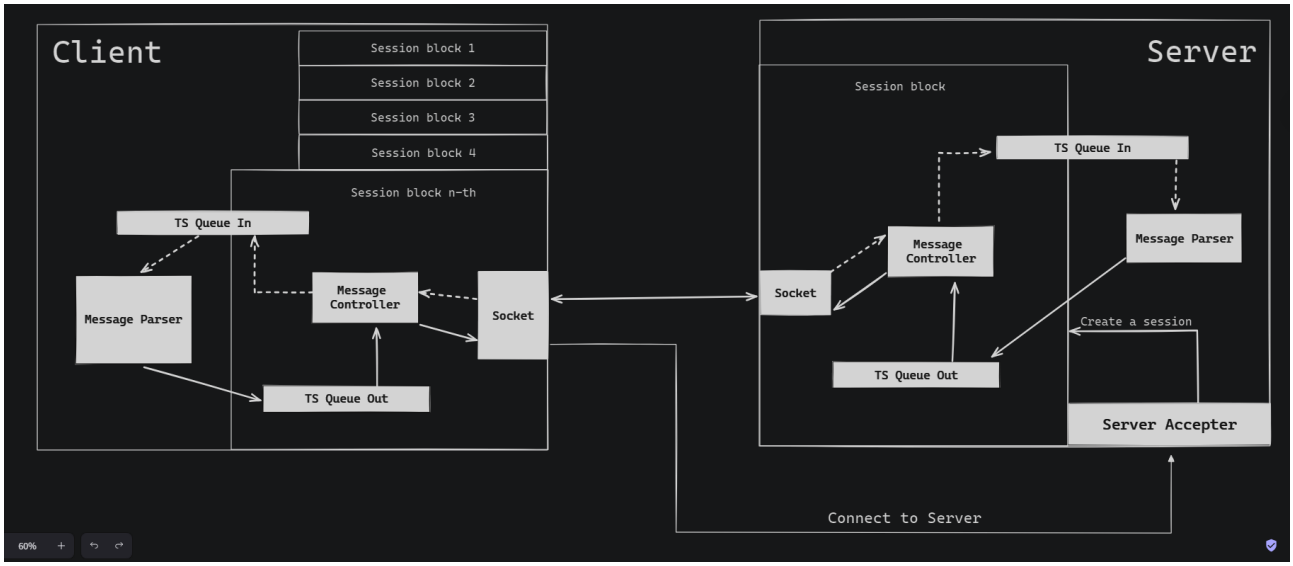
Chưa code nên chưa mô tả

3 Kiến trúc ứng dụng:

Tại phần này, chúng tôi sẽ trình bày về các thành phần chính của ứng dụng, mô tả cách mà các thành phần trong ứng dụng như Network, View, Model... có thể giao tiếp với nhau và các giao thức được định nghĩa trong phần Network.

4 Kiến trúc Network:

Ở phần này chúng tôi sẽ giới thiệu các thành phần cơ bản của Network.



Hình 17: Networking Architechture Diagram

Trên đây là mô hình cơ bản của kiến trúc network của chúng tôi và luồng di chuyển của message. Lưu ý: ở đây với mũi tên vẽ liền đại diện cho luồng di chuyển message *ra ngoài*, còn mũi tên nét đứt đại diện cho message được gởi *vào*.

4.1 Message:

Message là dữ liệu cơ bản được truyền đi nhằm thực hiện giao tiếp giữa hai ứng dụng với nhau.

4.2 ThreadSafe Queue:

Lưu ý: ThreadSafe Queue được sử dụng cho mọi hàng đợi được tạo ra trong phần Network, vì thế để thuận tiện, kể từ bây giờ chúng tôi sẽ gọi là *Hàng đợi* (*Queue*) và ngầm hiểu đó là đối tượng *ThreadSafe Queue*.

4.3 Session:

Session là thành phần chính của kiến trúc mạng của chúng tôi. Thành phần này được thiết kế như là nơi thực hiện việc giao tiếp trực tiếp với đối tượng bên ngoài thông qua socket. Mỗi session sẽ chỉ tạo với mỗi kết nối được thiết lập giữa *Client* và *Server* và được dùng để giao tiếp giữa chính hai đối tượng đó. Điều này giúp việc quãn lý bộ nhớ và các trạng thái một cách dễ dàng và tự động, giảm độ phức tạp trong việc xử lý gởi nhận trong quá trình giao tiếp.

Session chứa các đối tượng chính như là:

- **Socket** - Endpoint cho kết nối. Trong kiến trúc mạng của chúng tôi, chỉ session mới chứa socket để thực hiện giao tiếp.

- **IncommingQueue** - Hàng đợi message cho dữ liệu được nhận vào, được thiết kế là một đối tượng std::shared_ptr và được tham chiếu bởi cả *Session* và đối tượng chứa nó (*Client* hoặc *Server*). Khi *Session* nhận được message từ bên ngoài, đối tượng này sẽ kiểm tra tính hợp lệ của message (theo phong cách đã được định nghĩa), sau đó sẽ được đưa vào xử lý bởi *Client* và *server*. Điều này giúp làm tránh các xung đột và những bất cập trong việc quãn lý khi một server có thể xử lý nhiều *Session*. Khi đó, các đối tượng như *Server* hay *Client* có thể dễ dàng xử lý trong một hàng đợi duy nhất và gởi đi message mới cho đúng *Session*.

- **OutcommingQueue** - Hàng đợi message cho dữ liệu gởi đi. Khác với *IncommingQueue*, *OutcommingQueue* chỉ được tham chiếu bởi *Session*. Trong khi kết nối được thiết lập, *Session* sẽ luôn kiểm tra *OutcommingQueue* còn dữ liệu không, nếu có sẽ gởi tất cả những dữ liệu bên trong và đợi cho đến khi có message mới được thêm vào.

4.4 IClient và IServer:

IClient và *IServer* là hai abstract class tạo ra để quãn lý, phân phối và xử lý các event và message.
Trong *IClient* và *IServer* đều được thiết kế các hàm để kết nối, chấp nhận kết nối, gởi và nhận các message từ *Session* và quãn lý các *Session* này. *Client* và *Server* chứa một hàng đợi gọi là *IncommingQueue* như hình trên đã thể hiện. Chúng có nhiệm vụ nhận các message từ *Session* và phân phối và xử lý dựa trên HeadID của message mà phân phối cho các handler phù hợp.
Trong cách implementation của mình, chúng tôi chỉ cho các class này tạo ra 2 *Session* mỗi kết nối, một cho việc truyền dữ liệu ảnh từ *Server* sang *Client*, một dành cho việc truyền các event điều khiển từ *Client* sang *Server*. Việc này giúp dữ liệu ở *Session* không bị tràn khi mà kích thước dữ liệu ảnh sẽ khá lớn.

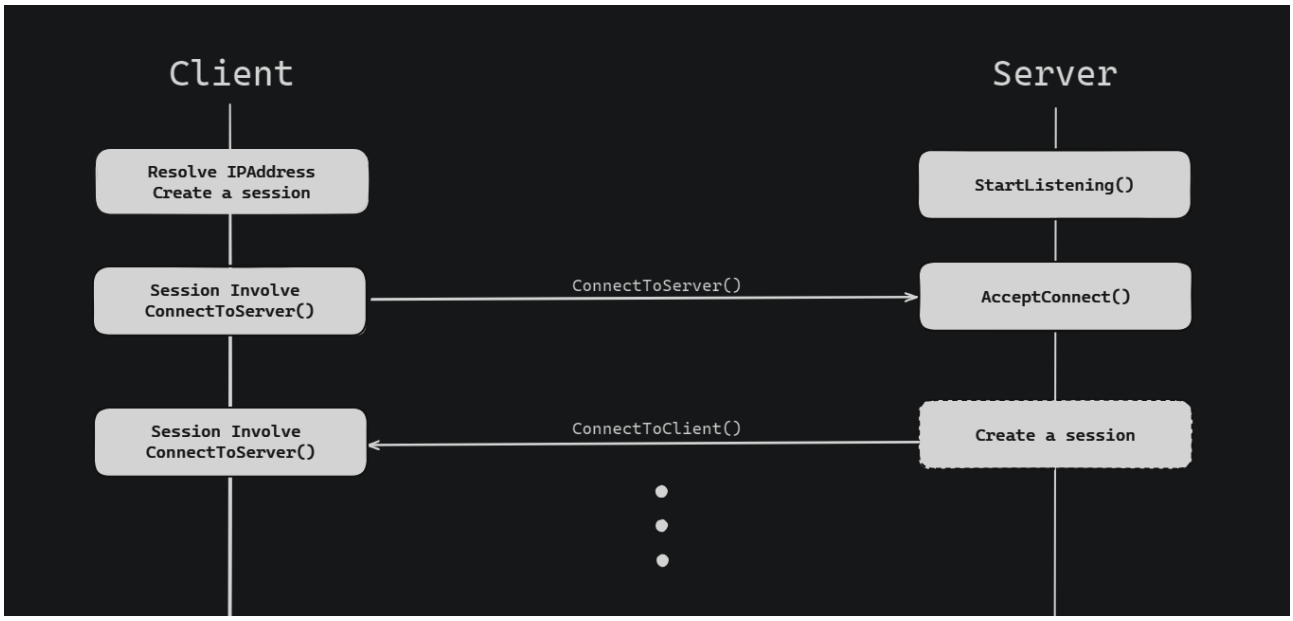
Một vài điểm khác biệt giữa Client và Server ở đây:

- **IServer:** Là đối tượng được phía **Client** kết nối đến và điều khiển từ xa. *IServer* chứa các đối tượng acceptor của thư viện ASIO, nhằm *accept* các kết nối từ *Client*. Để tránh các xung đột không cần thiết, chúng tôi đã thiết kế mỗi một Server chỉ tạo duy nhất một kết nối với một Client.

- **IClient:** Là đối tượng được dùng để điều khiển máy từ xa hay cụ thể ở đây chính là *Server*.

4.5 Giao tiếp giữa các thành phần:

4.5.1 Thiết lập kết nối:



Hình 18: Establish connection

4.5.2 Nhận và gởi:

Sơ đồ 17 cũng đã mô tả quá trình giao tiếp giữa các thành phần trong mạng với nhau.

- Ở quá trình **gởi**: Sau khi message được tạo bởi *Client* hoặc *Server*, message sẽ được kiểm tra đưa vào đúng *Session* cần gởi. Message khi vào *Session* sẽ được đưa vào *OutcommingQueue* để chờ các gói message phía trước nó được gởi. Khi đến lượt, message sẽ được gởi thông qua hai giai đoạn: *WriteHeader()* gởi Header và *WriteBody()* gởi Body bằng *Socket*. Việc này sẽ làm giảm sức ép lên băng thông và vẫn đảm bảo nhận đủ dữ liệu khi mà kích thước của Header là cố định còn kích thước Body đã được lưu trữ ở Header.

- Ở quá trình **Nhận**: Khi nhận, *socket* của *Session* bên nhận sẽ là nơi xử lý đầu tiên, lặp lại đúng như thứ tự gởi ở bên gởi. *ReadHeader()* và *ReadBody()*. Khi quá trình nhận hoàn thành, message mới sẽ được tạo và sẽ được đẩy vào *IncommingQueue* để *Client* hoặc *Server* để xử lý.

5 Demo

Video demo của chương trình được đăng tải tại đây.

Lời cảm ơn

Trong quá trình thực hiện đồ án này, những kiến thức về Mạng máy tính được giảng dạy bởi thầy Đỗ Hoàng Cường đã giúp ích chúng em rất nhiều trong việc giải quyết các vấn đề. Ngoài ra, source code TelePC từ thầy đã giúp nhóm có thể dễ dàng hơn khi xử lý các yêu cầu của đồ án. Nhóm chúng em xin cảm ơn thầy vì những kiến thức cũng như những chia sẻ này ạ.
Bên cạnh đó, những kỹ năng lập trình socket được các giáo viên hướng dẫn thực hành là thầy Lê Hà Minh và thầy Nguyễn Thanh Quân truyền đạt đã góp phần không nhỏ trong việc hoàn thành đồ án này.
Ngoài ra, nhóm cũng xin chân thành cảm ơn những người bạn đã đóng góp ý kiến, góp phần hoàn thiện đồ án.

Thành phố Hồ Chí Minh, tháng 6 năm 2022