**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN**

**KHOA CÔNG NGHỆ THÔNG TIN**

---o0o---

**Ảnh có chứa biểu tượng, Phông chữ, Nhãn hiệu, Đồ họa

Mô tả được tạo tự động**

**NHÓM 8**

**MẠNG MÁY TÍNH**

**EMAIL CONTROLLING**

**Giáo viên hướng dẫn:** **Đỗ Hoàng Cường**

**Sinh viên thực hiện:23122003 - Nguyễn Văn Linh**

**23122022 - Trần Hoàng Gia Bảo**

**23122026 - Trần Chấn Hiệp**

MỤC LỤC

[**I. Giới thiệu 2**](#_Toc1837094624)

[1. Mục đích của ứng dụng 2](#_Toc949544557)

[2. Phạm vi ứng dụng 2](#_Toc974011376)

[**II. Kiến trúc hệ thống 2**](#_Toc380080959)

[1. Tổng quan về kiến trúc Client – Server 2](#_Toc891473615)

[2. Giao tiếp giữa Client và Server 2](#_Toc703019237)

[3. Các thư viện ngoài được sử dụng 3](#_Toc1625985873)

[**III. Phân tích chức năng 3**](#_Toc475617105)

[1. Chức năng của client 3](#_Toc1060217249)

[**1.1. Giao diện của người dùng (GUI) 3**](#_Toc52477104)

[**1.2. Xử lý yêu cầu từ gmail 3**](#_Toc905267029)

[**1.3. Truyền dữ liệu đến server 6**](#_Toc1417680363)

[**1.4. Gửi kết quả về gmail người dùng 7**](#_Toc1734483048)

[**1.5. Các hàm khác 7**](#_Toc988000223)

[2. Chức năng của server 10](#_Toc609201105)

[2.1. Nhận yêu cầu từ Client 10](#_Toc153231178)

[2.2. Truy xuất và xử lý dữ liệu 11](#_Toc1117755907)

[2.3. Các hàm khác 12](#_Toc544434944)

[**IV. Kết luận 12**](#_Toc1718280407)

[1. Tổng kết 12](#_Toc188852297)

[2. Đề xuất cải tiến trong tương lai 12](#_Toc927656210)

# Giới thiệu

## Mục đích của ứng dụng

* Sử dụng kiến thức đã học để làm ra giao tiếp giữa client-server. Đồng thời kết hợp kết nối từ gmail để nhận yêu cầu từ người dùng, hay gọi là điều khiển máy tính bằng gmail.

## Phạm vi ứng dụng

* Trong cùng một mạng nội bộ mới thực hiện được ứng dụng, sử dụng thông qua IP của máy tính server và điều khiển bằng gmail.

# Kiến trúc hệ thống

## Tổng quan về kiến trúc Client – Server

* Sử dụng socket là chủ yếu để thực hiện kết nối giữa client và server và bao gồm các thư viện ngoài như là : Eagetmail, opencv, curl,..

## Các thư viện được sử dụng

* **#include <iostream>**
* **#include <winsock2.h>**
* **#include <string>**
* **#include <cstring>**
* **#include <thread>**
* **#include <chrono>**
* **#include <cstdlib>**
* **#include <Ws2tcpip.h>**
* **#include <fstream>**
* **#include <wbemidl.h>**
* **#include <comdef.h>**
* **#include <tlhelp32.h>**
* **#include <vector>**
* **#include <windows.h>**
* **#include <opencv2/opencv.hpp>**
* **#include <mutex>**
* **#include <atomic>**
* **#include <iomanip>**
* **#include <ctime>**
* **#include <direct.h>**
* **#include <locale>**
* **#include <codecvt>**
* **#include <mutex>**

# Phân tích chức năng

## Chức năng của client

### Giao diện của người dùng (GUI)

* Dùng để đăng nhập vào client bằng giao diện với đồ họa bắt mắt, dễ nhìn,..
* Sử dụng winform có sẵn trong Microsoft visual studio.
* Các tính năng mà giao diện cung cấp :
  + Nhập địa chỉ gmail của client
  + Nhập địa chỉ gmail của user cho phép gửi đến gmail của client
  + Nhập địa chỉ ip của server
  + Khởi động client

### Xử lý yêu cầu từ gmail

* **Hàm base64\_encode() :**
* Ý nghĩa:
* Dùng để mã hóa một chuỗi nhị phân hoặc văn bản (std::string) sang định dạng Base64. Base64 là một phương pháp mã hóa dữ liệu phổ biến, thường được dùng để truyền dữ liệu nhị phân (như file) dưới dạng chuỗi văn bản ASCII trong các hệ thống truyền tải (email, HTTP, v.v.)

* **Hàm \_getCurrentPath() :**
* Ý nghĩa:
* Lấy đường dẫn thư mục hiện tại của tệp thực thi trả ra độ dài đường dẫn thư mục.
* Thành phần:
* Lấy đường dẫn đầy đủ của tệp thực thi:
* GetModuleFileName: Lưu đường dẫn đầy đủ (bao gồm cả tên tệp) vào lpPath.
* Trả về độ dài đường dẫn thư mục

* **Hàm SaveEmailToTxt()** :
* Ý nghĩa:
* Dùng để phân tích một tệp email định dạng .eml và lưu các thông tin (như người gửi, tiêu đề, nội dung) vào một tệp văn bản dưới dạng UTF-8.
* Thành phần:
* Nhận vào tham số LPCTSTR lpszFile: Đường dẫn đến tệp .eml cần phân tích.
* Dùng thư viện EAGetMail để trích xuất mail. Tóm tắt quá trình: Tạo đối tượng IMail -> Tải tệp .eml -> Trích xuất thông tin email -> Lưu thông tin vào tệp
* Trà ra const char\* outputFilePath: Đường dẫn tệp văn bản nơi lưu thông tin sau khi phân tích.

* **Hàm readFile() :**
* Ý nghĩa:
* Dùng để đọc toàn bộ nội dung của một tệp và trả về dưới dạng chuỗi (std::string). Đây là cách tiện lợi để xử lý dữ liệu tệp, đặc biệt với tệp nhị phân hoặc văn bản
* Thành phần:
* Nhận const std::string& filePath: Chuỗi chứa đường dẫn đến tệp cần đọc.
* Tóm tắt quá trình: Mở tệp ở chế độ nhị phân -> Đọc toàn bộ nội dung tệp -> Trả về chuỗi nội dung
* Trà về nội dung file dưới dạng chuỗi
* **Hàm payload\_source\_()** :
* Ý nghĩa:
* Sử dụng để truyền dữ liệu từ một chuỗi (payload) vào một bộ đệm (ptr) theo cách tuần tự, thường áp dụng trong các thư viện truyền dữ liệu như libcurl để xử lý dữ liệu khi gửi yêu cầu HTTP.
* Thành phần:
* Nhận tham số: void\* ptr: Bộ đệm đích, nơi dữ liệu từ payload sẽ được sao chép; size\_t size: Kích thước của mỗi phần tử trong bộ đệm, size\_t nmemb: Số lượng phần tử trong bộ đệm, void\* userp: Một con trỏ đến dữ liệu nguồn (ở đây là payload\_text).
* Tóm tắt: Kiểm tra điều kiện kết thúc -> Tính toán kích thước dữ liệu cần sao chép -> Sao chép dữ liệu -> Cập nhật con trỏ dữ liệu
* Trả về số byte đã sao chép

* **Hàm checkaddress()** :
* Ý nghĩa:
* Kiểm tra xem một chuỗi địa chỉ (a) có tồn tại trong file user.txt tại dòng có số thứ tự bằng với giá trị của dòng cuối cùng hay không
* Thành phần:
* Nhận vào string a
* Mở file và lấy dòng cuối cùng -> Duyệt lại file để kiểm tra điều kiện
* Trả ra true nếu thỏa yêu cầu, false ngược lại

* **Hàm run():**
  + Ý nghĩa:
    - Thực hiện chức năng kết nối với máy chủ IMAP (ở đây là Gmail) để tải email và xử lý nó
  + Thành phần:
    - Thiết lập môi trường: Khởi tạo môi trường COM để làm việc với các đối tượng COM và tạo thư mục “inbox” để lưu trữ email xuống
    - Kết nối máy chủ IMAP: sử dụng thư viện EaGetmail để cấu hình với Gmail và thiết lập thông tin đăng nhập và các thông số liên quan(SSL, cổng)
    - Tải email cuối cùng: lấy danh sách email từ máy chủ và tải email cuối cùng về và lưu thành tệp .eml trong “inbox”
    - Xử lý email: phân tích tệp .eml và lưu nội dung vào tệp output.txt và đọc tệp output.txt và thực hiện một số kiểm tra với tệp
    - Dọn dẹp tài nguyên: Đóng kết nối và giải phóng môi trường COM

### Truyền dữ liệu đến server

* **Hàm client\_run():**
  + Ý nghĩa: dùng để chạy chương trình và khởi tạo kết nối từ client đến server
  + Công việc khởi tạo:
    - Khởi tạo Winsock: Dùng WSAStartup trong thực tế dùng để tạo một socket TCP/IP với địa chỉ máy chủ là 127.0.0.1(local host) với cổng kết nối là 8080
    - Dùng hàm connect để thực thi kết nối với server nếu thất bại dọn dẹp socket và Winsock sau đó thoát
  + Giao tiếp với Server:
    - Dùng vòng lặp để gửi tin nhắn đến server và xử lý phản hồi
    - Dựa trên phản hồi của server, thực hiện các hành động cụ thể:
      * send\_imp: client sẽ nhận được hình ảnh từ server
      * recording: client sẽ nhận được video đã được record lại từ server
      * list\_app: gửi tin nhắn từ tệp output.txt sau đó client sẽ nhận và lưu danh sách vào tệp list\_app.txt
      * list\_server: tương tự như list\_app, nhưng lưu vào danh sách list\_service.txt
      * fetch: Gửi tên tệp cần tải từ server sau đó nhận tệp từ server vào thư dir
    - Đóng socket và giải phóng winsock
* **Hàm receive\_txt():**
  + Ý nghĩa: Hàm này nhận một tệp văn bản từ máy chủ (server) thông qua một socket và lưu nó vào một tệp cục bộ.
  + Tham số:
    - SOCKET serverSocket: Socket kết nối đến máy chủ, được sử dụng để nhận dữ liệu.
    - const char\* outputFilePath: Đường dẫn nơi tệp văn bản sẽ được lưu.
  + Chức năng:
    - Mở tệp cục bộ: dùng ofstream để mở tệp đường dẫn trong chế độ ghi nhị phân
    - Nhận dữ liệu qua socket: Dùng hàm rev để nhận dữ liệu từ socket vào bộ đệm buffer, dữ liệu nhận được ghi vào tệp mở
    - Vòng lặp dừng lại khi recv trả về giá trị <= 0

### Gửi kết quả về gmail người dùng

* **Hàm sendEmai() :**
* Ý nghĩa:
* Gửi email với nội dung và tệp đính kèm, được cấu hình thông qua libcurl
* Thành phần:
* Tóm tắt quá trình: Đọc thông tin từ file output.txt -> Kiểm tra tính hợp lệ của email -> xử lí theo từng yêu cầu của user để xác định loại tệp -> tạo email và gửi email bằng thư viện libcurl : sử dụng SMTP
* Trả về true nếu email gửi thành công, false nếu ngược lại

### Các hàm khác

* **Hàm setup\_console() :** 
  + Ý nghĩa:
    - Hàm này sẽ đảm bảo rằng console có thể hiển thị và xử lý văn bản UTF-8, cụ thể là tiếng Việt, mà không gặp vấn đề về mã hóa hay ký tự không hiển thị đúng.
    - Điều này hữu ích khi làm việc với văn bản đa ngôn ngữ hoặc cần hiển thị ký tự Unicode trong console.
  + Thành phần:
    - system("chcp 65001");:
      * Lệnh này thay đổi mã code page của console thành 65001, tức là UTF-8.
      * Điều này đảm bảo rằng console có thể hiển thị các ký tự UTF-8 (bao gồm tiếng Việt) một cách chính xác.
    - std::setlocale(LC\_ALL, "vi\_VN.UTF-8");:
      * Hàm setlocale thiết lập cài đặt vùng (locale) cho chương trình.
      * Tham số LC\_ALL chỉ định thay đổi toàn bộ các khía cạnh liên quan đến locale (ngôn ngữ, định dạng số, ký tự, v.v.).
      * "vi\_VN.UTF-8" là chuỗi xác định locale tiếng Việt với mã hóa UTF-8, giúp chương trình xử lý chuỗi văn bản bằng tiếng Việt đúng cách.
* **Hàm wstringToString() :** 
  + Ý nghĩa:
    - Chuyển đổi từ std::wstring (Unicode dạng rộng) sang std::string (UTF-8).
  + Thành phần:
    - Sử dụng std::wstring\_convert với bộ chuyển mã std::codecvt\_utf8<wchar\_t> để thực hiện chuyển đổi.
    - Hàm to\_bytes chuyển các ký tự dạng rộng (wide characters) thành một chuỗi UTF-8.
* **Hàm stringToWstring() :** tương tự hàm wstringToString () nhưng thực hiện ngược lại là chuyển từ std::string thành std::wstring.
* **Hàm extractFileName() :**
  + Ý nghĩa:
    - Trích xuất tên file từ một đường dẫn đầy đủ.
  + Thành phần:
    - Nhận tham số const string& filePath: Đường dẫn đầy đủ đến file.
    - Trả ra string: Tên file được trích xuất (không bao gồm đường dẫn.
* **Hàm trim() :**
  + Ý nghĩa:
    - Dùng để xóa các kí tự như khoảng cách của một chuỗi dạng wstring
  + Thành phần:
    - Nhận vào một chuỗi wstring đường dẫn file và trả về đường dẫn chính xác sau khi xóa các kí tự thừa.
* **Hàm receiveFile() :**
  + Ý nghĩa:
    - Dùng để nhận dữ liệu từ server thông qua socket (giao thức TCP) và lưu file với tên được nhận vào máy client.
  + Thành phần:
    - Nhận vào socket serverSocket và string đường dẫn.
    - Tóm tắt quá trình: Nhận độ dài tên file -> Nhận tên file -> Xử lý tên file -> Mở file để ghi -> Nhận kích thước file -> Nhận và ghi dữ liệu
    - Lưu file đã nhận vào một tệp chỉ định trong máy client

## Chức năng của server

## Nhận yêu cầu từ Client

* Hàm main():
  + Ý nghĩa: dùng để khởi chạy server và thiết lập một server TCP đơn giản với nhiều bước như là: khởi tạo winsock, tạo socket server, gán địa chỉ và cổng cho socket,...
  + Thành phần
    - Khởi tạo Winsock:
      * Khởi tạo Winsock để sử dụng các hàm liên quan đến socket trên hệ điều hành Windows.
      * Hàm WSAstartup được gọi với tham số MAKEWORD(2,2) để yêu cầu sử dụng WINSOCK phiên bản 2.2.
      * Nếu khởi tạo thất bại, chương trình sẽ in lỗi và thoát.
    - Tạo socket server:
      * Tạo socket TCP sử dụng giao thức IPv4 và kiểu SOCK\_STREAM
      * Nếu tạo socket thất bại, chương trình in lỗi, dọn dẹp Winsock bằng WSACleanup(), và thoát
    - Gán địa chỉ và cổng cho socket:
      * Thiết lập địa chỉ cho cổng socket server với ipv4, chuyển cổng 8080 sang định dạng mạng
      * Hàm bind kết nối socket địa chỉ và cổng
      * Nếu bind thất bại, chương trình in lỗi, đóng socket, dọn dẹp winsock, và thoát.
    - Lắng nghe kết nối:
      * Đặt socket vào chế độ lắng nghe, cho phép chấp nhận kết nối từ tối đa 3 client đồng thời
      * Nếu listen thất bại, chương trình in lỗi, đóng socket, dọn dẹp Winsock, và thoát
    - Chấp nhận và xử lý kết nối từ client:
      * Chấp nhận kết nối: Hàm accept chờ kết nối từ một client. Khi có client kết nối, một socket mới được tạo để giao tiếp với client đó. Địa chỉ client được lưu vào biến client( có kiểu struct sockaddr\_in).
      * Tạo luồng xử lí yêu cầu từ client thông qua hàm handleClient. Luồng được tách rời để hoạt động độc lập.
    - Dọn dẹp:
      * Khi server tắt, socket server được đóng bằng closesocket
      * Winsock được dọn dẹp bằng WSAcleanup để giải phóng tài nguyên hệ thống

## Truy xuất và xử lý dữ liệu

* **Hàm handleClient():**
  + Ý nghĩa: chịu trách nhiệm xử lý các yêu cầu từ client sau khi server chấp nhận kết nối.
  + Thành phần:
    - Nhận yêu cầu từ client:
      * Nhận dữ liệu từ client thông qua socket
      * Dữ liệu được lưu vào buffer với kích thước tối đa 1024 byte
      * Nếu không nhận được dữ liệu hay xảy ra lỗi, hàm sẽ thoát
      * Dữ liệu được in ra console để server có thể quan sát yêu cầu của client
    - Xử lý từng lệnh từ client:
      * Lệnh shutdown: tắt máy tính server và gửi phản hồi về choa client
      * Lệnh screenshot: Chụp màn hình server và lưu thành file sccreenshot.bmp và gửi thông báo “send\_imp” cho client. Gửi file ảnh qua socket bằng hàm send\_imp.
      * Lệnh turn\_on\_webcam và turn\_off\_webcam: mở camera và bắt đầu truyền stream video và tắt camera.
      * Lệnh recording: nhận thời gian ghi màn hình từ client, ghi màn hình trong thời gian chỉ định, gửi kích thước video đến client và cuối cùng sẽ gửi video qua socket bằng sendVideo.
      * Lệnh fetch: nhận đường dẫn file từ client và gửi file được chỉ định về client qua socket bằng sendFile.
      * List\_app: Gửi danh sách ứng dụng đang chạy về client, nhận yêu cầu "start" hoặc "stop" từ client để khởi động hoặc tắt ứng dụng chỉ định và phản hồi kết quả thực hiện về client.
      * List\_service: Gửi danh sách dịch vụ hệ thống về client, nhận yêu cầu "start" hoặc "stop" từ client để quản lý dịch vụ chỉ định và phản hồi kết quả thực hiện về client.
    - Chức năng tổng thể:
      * Xử lý các yêu cầu điều khiển từ xa từ client như:
        + Tắt máy tính (shutdown).
        + Chụp màn hình (screenshot).
        + Quản lý webcam (turn\_on\_webcam, turn\_off\_webcam).
        + Ghi màn hình (recording).
        + Gửi file (fetch).
        + Quản lý ứng dụng (list\_app).
        + Quản lý dịch vụ hệ thống (list\_service).
      * Giao tiếp hai chiều giữa server và client, với phản hồi trạng thái về client.

## Các hàm khác

* **Hàm Savebitmap():**
  + Ý nghĩa: lưu ảnh chụp màn hình dưới dạng file BMP
  + Thành phần:
    - Cấu hình thông tin của file ảnh BMP qua BITMAPINFOHEADER (thông tin tiêu đề ảnh).
    - Dữ liệu bitmap được trích xuất từ device context bằng GetDIBits
    - Ghi thông tin file và dữ liệu ảnh vào file BMP
* **Hàm CaptureScreen():**
  + Ý nghĩa: chụp màn hình và lưu thành file ảnh BMP
  + Thành phần:
    - Lấy kích thước màn hình qua GetSystemMetrics.
    - Tạo HDC: hscreenDC, hMemoryDC
    - Tạo Hbitmap
    - Lưu file ảnh thành file BMP bằng cách gọi SaveBitmap
    - Giải phóng tài nguyên
* **Hàm send\_imp():**
  + Ý nghĩa: Gửi file ảnh BMP tới client qua socket
  + Thành phần:
    - Mở file BMP ở chế độ nhị phân và đọc toàn bộ dữ liệu vào bộ nhớ.
    - Lấy kích thước file ảnh và gửi kích thước này tới client
    - Gửi dữ liệu ảnh qua socket tới client bằng send
    - Giải phóng bộ nhớ được cấp phát và thông báo gửi thành công.
* **Hàm sendVideo():**
  + Ý nghĩa: Gửi video từ server tới client qua socket
  + Thành phần:
    - Mở file ở chế độ nhị phân
    - Đọc dữ liệu theo từng khối(CHUNK\_SIZE byte) và gửi tới client
    - Kiểm tra lỗi khi gửi dữ liệu: Nếu gửi thành công, tiếp tục. Nếu lỗi, thông báo và thoát vòng lặp.
    - Nếu có dữ liệu cuối cùng chưa gửi hết, gửi nốt phần còn lại
    - Đóng file và thông báo gửi thành công
* Các hàm setup\_console(), hàm wstringToString(), hàm stringToWstring() và hàm trim() tương tự bên client cho phần xử lí file.
* **Hàm sendData():**
  + Ý nghĩa:
    - Dủng để gửi dữ liệu an toàn từ server qua client
  + Thành phần:
    - Nhận vào biến socket socket, một const char\* buffer, và một int length
    - Tóm tắt quá trình: Sau khi kết nối với client, **Biến totalBytesSent**: Theo dõi tổng số byte đã gửi thành công -> lần lượt gửi từng phẩn của file cho dến khi hoàn thành
    - Trả về tổng số byte đã gửi
* **Hàm sendFile():**
  + Ý nghĩa:
    - Dùng để gửi file từ server đến client
  + Thành phần :
    - Nhận vào tham số socket clientSocket và string filePath.
    - Tóm tắt quá trình: Xử lí đường dẫn để mở file -> Trích xuất tên file -> Gửi tên file và kích thước file -> Gửi nội dung file
* **Hàm ListExecutableAppsRecursive():**
  + Ý nghĩa:
    - Dùng để liệt kê tất cả các tệp thực thi và các thư mục con của chúng
  + Thành phần:
    - Nhận vào string directoryPath và một vector<string> appList
    - Tóm tắt quá trình: Tạo đường dẫn tìm kiếm (từ directoryPath)-> Mở thư mục để duyệt -> Duyệt qua tất cả tệp và thư mục -> Lưu các đường dẫn thực thi (tệp .exe) vào vector -> Đóng handle sau khi xử lí xong.
* **Hàm List\_App():**
  + Ý nghĩa:
    - liệt kê tất cả các tệp thực thi (.exe) từ hai ổ đĩa D: và C: rồi lưu danh sách này vào tệp list\_app.txt
* **Hàm List\_Services():**
  + Ý nghĩa:
    - Dùng để liệt kê tất cả các dịch vụ chạy trên hệ thống Windows và ghi danh sách này vào tệp list\_service.txt
  + Thành phần:
    - Tóm tắt quá trình: Khởi tạo COM và thiết lập bảo mật COM ->Kết nối VMI->Truy vấn danh sách dịch vụ->Ghi dịch vụ vào file->Dọn dẹp
    - File list\_service.txt sẽ chứa các dịch vụ hiện có trên hệ thống
* **Hàm change():**
  + Ý nghĩa:
    - Chuyển đổi một chuỗi ký tự (char\* buffer) thành một chuỗi kiểu std::string, đồng thời loại bỏ các ký tự xuống dòng hoặc kết thúc chuỗi không mong muốn ('\r' hoặc NULL)
* **Hàm LauchApp():**
  + Ý nghĩa:
    - Khởi chạy một ứng dụng hoặc mở một tệp được chỉ định thông qua tên ứng dụng hoặc đường dẫn.
  + Thành phần:
    - Nhận vào chuỗi string appName.
    - Tóm tắt quy trình: chuyển string thành wstring->Gọi ShellExecuteW để mở ứng dụng
* **Hàm stop\_app():**
  + Ý nghĩa:
    - Dừng một ứng dụng hoặc tiến trình đang chạy trên hệ thống Windows
  + Thành phần:
    - Nhận vào chuỗi string -> Tạo chuỗi lệnh taskkill->Đóng tiến trình bằng system
* **Hàm ConvertToWideString():**
  + Ý nghĩa:
    - Chuyển đổi một chuỗi ký tự đa byte (std::string) sang chuỗi ký tự rộng (std::wstring)
* **Hàm ManageService():**
  + Ý nghĩa:
    - Quản lý các dịch vụ Windows bằng cách **khởi động** hoặc **dừng** chúng thông qua tên dịch vụ và hành động được cung cấp
  + Thành phần:
    - Nhận vào biến string serviceName và string action.
    - Tóm tắt quá trình: Mở trình quản lí dịch vụ->Mở dịch vụ mục tiêu->Xử lí đóng hay mở tiến trình->Đóng handle
    - Trả về true nếu thành công và false nếu ngược lại
* **Hàm send\_txt():**
  + Ý nghĩa:
    - Dùng để gửi nội dung từ các file văn bản (list\_app.txt hoặc list\_service.txt) tới client
  + Thành phần:
    - Tóm tắt quá trình: Mở file và kiểm tra file txt->Đọc và gửi dữ liệu tới client
* **Hàm CanWriteToDirectory()**
  + Ý nghĩa: Kiểm tra quyền ghi (write permission) vào một thư mục cụ thể.
  + Thành phần:
    - Tạo thử một file tạm (test\_write\_permission.txt) trong đường dẫn được cung cấp.
    - Nếu không thể tạo file, in ra thông báo và trả về false.
    - Nếu việc tạo file thành công, đóng file và trả về true.
    - Nếu có lỗi xảy ra (exception), in ra thông báo lỗi và trả về false
* **Hàm CloseCamera()**
  + Ý nghĩa: Đóng camera một cách an toàn.
  + Thành phần:
    - Dùng lock\_guard để đảm bảo quyền truy cập độc quyền vào biến toàn cục cameraMutex.
    - Kiểm tra trạng thái của camera:
      * Nếu camera chưa được mở, in thông báo và thoát.
    - Nếu camera đang hoạt động:
      * Cập nhật cờ shouldStopCamera để dừng các luồng liên quan.
      * Giải phóng tài nguyên camera (cap.release()).
      * Đóng cửa sổ hiển thị (destroyAllWindows()).
      * Cập nhật trạng thái isCameraOpen thành false và in thông báo.
* **Hàm CameraStream()**
  + Ý nghĩa: Luồng xử lý hiển thị video từ camera.
  + Thành phần:
    - Liên tục kiểm tra trạng thái camera bằng cờ shouldStopCamera.
    - Trong mỗi vòng lặp:
      * Khóa cameraMutex để truy cập an toàn vào camera.
      * Kiểm tra xem camera đã mở chưa và lấy từng frame từ camera.
      * Nếu frame hợp lệ, hiển thị nó trên cửa sổ.
    - Ngắt luồng khi:
      * Nhấn phím ESC (keycode 27) hoặc cờ shouldStopCamera bật.
    - Gọi hàm CloseCamera để giải phóng tài nguyên sau khi luồng dừng.
* **Hàm OpenCamera() :**
  + Ý nghĩa: Mở và khởi tạo camera để sử dụng.
  + Thành phần:
    - Dùng lock\_guard để đảm bảo truy cập đồng thời an toàn qua cameraMutex.
    - Kiểm tra trạng thái camera:
      * Nếu camera đã mở, in thông báo và thoát.
    - Mở camera mặc định (cap.open(0)).
    - Kiểm tra xem camera có mở thành công không:
      * Nếu không, in thông báo lỗi.
      * Nếu thành công, cập nhật trạng thái isCameraOpen và in thông báo.

# Kết luận

## Tổng kết

* Nhóm đã hoàn thành được các yêu cầu mà đề án đưa ra và thêm một vài chức năng để có thể cải thiện hơn
* Công việc của mỗi thành viên trong nhóm được phân công hợp lí và đã hoàn thành được tốt nhiệm vụ được nhận
* Đã có quay video demo sản phầm trên youtube

## Đề xuất cải tiến trong tương lai

* Xử lí mượt mà hơn về kết nối giữa client-server
* Thêm một số chức năng khác tiện ích hơn
* Sử dụng được khi đang ở xa
* Cải thiện đồ họa khi đăng nhập