BÁO CÁO ĐỒ ÁN MÔN HỌC MẠNG MÁY TÍNH

Điều khiển máy tính từ xa bằng email

THÀNH VIÊN NHÓM:

Trần Minh Trọng - 23120100

Hàn Vũ Phương Uyên - 23120106

Phạm Hương Trà - 23120177

fit@hcmus

MỤC LỤC

I. GIỚI THIỆU CHƯƠNG TRÌNH 2
II. CÀI ĐẶT CHƯƠNG TRÌNH 2
1. Cài đặt thư viện Curl (libcurl) 2
2. Cài đặt thư viện OpenCV 3
3. Cài đặt thư viện EASendMail 3
A. Chức năng cơ bản 4
1. Khởi tạo kết nối socket 4
2. Nhận và gửi thông điệp qua lại giữa client, server và gmail 6
3. Mở webcam và quay video trong khoảng thời gian cho trước 7
4. Chụp ảnh màn hình 8
5. Liệt kê danh sách các ứng dụng đã cài đặt 8
6. Mở ứng dụng đã cài đặt trong máy 8
7. Copy file từ đường dẫn cho trước 9
8. Xóa file từ đường dẫn cho trước 9
9. Shutdown và restart máy 9
10. Liệt kê tiến trình đang chạy trong máy 10
11. Đóng tiến trình đang chạy trong máy 10
12. Lưu phím trong khoảng thời gian cho trước (keylogger) 10
13. Khóa phím trong khoảng thời gian cho trước 11
B. Chức năng mở rộng: 11
1. Kiểm tra pin và trạng thái pin 11
2. Kiểm tra bộ nhớ ổ đĩa 12
3. Lấy file từ đường dẫn cho trước 12
III. TÀI LIỆU THAM KHẢO 12

I. GIỚI THIỆU CHƯƠNG TRÌNH

Đây là một ứng dụng điều khiển máy tính từ xa thông qua email. Chương trình được lập trình bằng ngôn ngữ C++.

Sau khi cài đặt chương trình vào máy cần điều khiển, người dùng có thể thực hiện nhiều chức năng trên máy tính đó khi gửi lệnh qua email đến email của chương trình.

Các chức năng có thể sử dụng được trình bày trong phần A mục II của báo cáo này.

Mail để điều khiển ứng dụng **thitkhomamruoc7749@gmail.com**.

II. CÀI ĐẶT CHƯƠNG TRÌNH

1. Cài đặt thư viện Curl (libcurl)

Thư viện được sử dụng cho việc nhận email từ Gmail gửi đến client.

1.1. Cài Đặt vcpkg

- Mở Command Prompt.
- Gõ lệnh: *git clone https://github.com/microsoft/vcpkg.git.*
- Di chuyển đến thư mục vcpkg: *cd vcpkg.*
- Tải vcpkg cho Window : *.\bootstrap-vcpkg.bat.*
- Cài đặt vcpkg cho Window: *.\vcpkg integrate project.*

1.2. Cài Đặt Thư Viện libcurl thông qua vcpkg

- Cài đặt libcurl: *.\vcpkg install libcurl.*
- Tích hợp vcpkg với Visual Studio: *.\vcpkg integrate install.*

1.3. Cấu hình Visual Studio

- Trong Visual Studio, click chuột phải vào tên dự án trong Solution Explorer và chọn *Properties*.
- Vào *Configuration Properties* > *C/C++* > *General*.
- Tại mục *Additional Include Directories*, thêm đường dẫn tới thư mục include trong thư mục vcpkg đã giải nén.
- Vào *Configuration Properties* > *Linker* > *General.*
- Tại mục *Additional Library Directories*, thêm đường dẫn tới thư mục *lib* trong thư mục vcpkg đã giải nén.
- Vào *Configuration Properties* > *Linker* > *Input.*
- Tại mục *Additional Dependencies*, thêm *libcurl.lib*.

1.4. Cấu hình môi trường

Thêm đường dẫn tới thư mục bin của libcurl vào PATH.

2. Cài đặt thư viện OpenCV

Thư viện dùng để thực hiện thao tác mở camera/webcam và quay video.

2.1. Tải và Giải Nén OpenCV

- Tải OpenCV tại đây.
- Giải nén thư mục đã tải về.

2.2. Cấu Hình Include và Library Directories

- Trong Visual Studio, click chuột phải vào tên dự án trong Solution Explorer và chọn *Properties*.
- Vào *Configuration Properties* > *C/C++* > *General.*
- Tại mục *Additional Include Directories*, thêm đường dẫn tới thư mục *include* trong thư mục OpenCV đã giải nén.
- Vào *Configuration Properties* > *Linker* > *General.*
- Tại mục *Additional Library Directories*, thêm đường dẫn tới thư mục lib trong thư mục OpenCV đã giải nén.
- Vào *Configuration Properties* > *Linker* > *Input.*
- Tại mục *Additional Dependencies*, thêm các thư viện cần thiết cho OpenCV.

2.3. Cấu Hình Biến Môi Trường (Environment Variables)

Thêm đường dẫn tới thư mục bin của OpenCV vào PATH.

3. Cài đặt thư viện EASendMail

Thư viện dùng để thực hiện việc gửi mail từ client đến gmail server.

3.1. Tải và Giải Nén EASendMail

Folder giải nén từ file zip tải về từ trang chủ thư viện:

- Lib: chứa file thư viện .lib và .dll.
- Include: chứa file header.h.
- Sample: chứa mã mẫu để gửi email.

3.2. Cấu hình Include và Library Directories

- Trong Visual Studio, click chuột phải vào tên dự án trong Solution Explorer và chọn *Properties*.
- Thêm file header vào dự án: copy file *"EASendMailObj.tlh"* và *EASendMailObj.h"* vào thư mục nguồn dự án.

- Link thư viện trong dự án: *Properties* > *Linker* > *Input* > *Additional Dependencies* > *Thêm "EASendMailObj.lib"*.
- Thêm file DLL: Đảm bảo file *EASendMailObj.dll* nằm cùng thư mục với file .exe khi chạy chương trình hoặc nằm trong thư mục hệ thống.

III. CÁC CHỨC NĂNG ĐÃ THỰC HIỆN

A. Chức năng cơ bản

1. Khởi tạo kết nối socket

Các hàm khởi tạo server bao gồm:

```
//SERVER
bool checkDLL();
SOCKET createSocket();
sockaddr_in setAddress();
void createConnection(SOCKET& sock, sockaddr_in& address);
void bindSocket(SOCKET& sock, sockaddr_in& address);
void setListen(SOCKET& sock, int n);
void acptConnection(SOCKET& clientSocket, SOCKET& serverSocket, bool& connect);
```

Quy trình hoạt động: Đảm bảo Winsock được khởi tạo > Khởi tạo socket > Cấu hình địa chỉ server > Liên kết socket với địa chỉ server > Đặt socket ở chế độ sẵn sàng kết nối (listen state) > Chấp nhận kết nối từ client.

1.1. Hàm checkDLL()

- Chức năng: Khởi tạo Winsock để có thể sử dụng các chức năng liên quan đến socket, mục đích để đảm bảo hệ thống đã sẵn sàng để sử dụng socket trước khi thực hiện các thao tác khác.
- Cách hoạt động:
  - Sử dụng hàm WSAStartup() để khởi tạo Winsock với phiên bản 2.2.
  - Nếu WSAStartup() trả về giá trị âm nghĩa là khởi tạo thất bại.

1.2. Hàm createSocket()

- Chức năng: một socket sử dụng giao thức TCP (một kênh liên lạc để lắng nghe kết nối từ client).
- Cách hoạt động: Hàm này tạo socket với các tham số:
  - AF_INET: Sử dụng IPv4.
  - SOCK_STREAM: Sử dụng giao thức TCP (kết nối ổn định).
  - IPPROTO_TCP: Chỉ định giao thức TCP.

- Hàm trả về socket đã được tạo.

1.3. Hàm setAddress()

- Chức năng: Thiết lập địa chỉ và cổng cho server (cấu hình địa chỉ IP và cổng mà server sử dụng để lắng nghe kết nối).
- Cách hoạt động: Sử dụng cấu trúc sockaddr_in để lưu thông tin địa chỉ.
  - sin_family: Thiết lập là AF_INET để chỉ định IPv4.
  - sin_addr.S_un.S_addr: Đặt địa chỉ IP là "127.0.0.1" (localhost).
  - sin_port: Chuyển đổi cổng sang dạng big-endian bằng htons(PORT).

1.4. Hàm bindSocket()

- Chức năng: Liên kết socket với địa chỉ và cổng đã được thiết lập.
- Cách hoạt động:
  - Sử dụng hàm bind() để gán socket với địa chỉ IP và cổng.
  - Nếu bind() thất bại, in ra thông báo lỗi.
  - Nếu thành công, in ra thông báo bind thành công.

1.5. Hàm setListen()

- Chức năng: đặt socket ở chế độ lắng nghe để chờ kết nối từ client, giúp server chấp nhận nhiều kết nối từ client một cách tuần tự.
- Cách hoạt động:
  - Sử dụng hàm listen() với tham số n để chỉ định số lượng kết nối tối đa có thể chờ trong hàng đợi.
  - Nếu listen() thất bại, in ra thông báo lỗi.
  - Nếu thành công, in ra thông báo socket đang ở chế độ lắng nghe.

1.6. Hàm acptConnection()

- Chức năng: Chấp nhận kết nối từ client, cho phép nhận và xử lý kết nối từ client.
- Cách hoạt động:
  - Sử dụng hàm accept() để chấp nhận kết nối từ client.
  - Nếu accept() thất bại, in ra thông báo lỗi.
  - Nếu thành công, in ra thông báo kết nối thành công.
  - Sau đó gọi hàm receiveCommand() để xử lý dữ liệu từ client.
  - Cuối cùng, đóng socket của client sau khi hoàn tất việc xử lý.

Các hàm khởi tạo client:

```
//CLIENT
bool checkDLL();
SOCKET createSocket();
sockaddr_in setAddress();
int createConnection(SOCKET& sock, sockaddr_in& address);
```

Quy trình hoạt động: Đảm bảo Winsock được khởi tạo > Khởi tạo socket > Cấu hình địa chỉ client > Liên kết socket với địa chỉ client > Yêu cầu kết nối từ client.

Các hàm checkDLL(), createSocket(), setAddress() tương tự như trên.

1.7. Hàm createConnection()

- Chức năng: Tạo kết nối từ client đến server (server đang lắng nghe).
- Cách hoạt động:
  - Sử dụng hàm connect() để yêu cầu kết nối đến địa chỉ server đã được cấu hình.
  - Nếu connect() thất bại, in ra thông báo lỗi.
  - Nếu thành công, in ra thông báo kết nối thành công.

2. Nhận và gửi thông điệp qua lại giữa client, server và gmail

2.1. Đọc mail từ email server của Gmail

```
std::string readEmail(const std::string& username, const std::string& appPassword, int index)
```

- Tham số đầu vào: tên email muốn đọc, khóa bảo mật 2 lớp của tài khoản email, thứ tự của email.
- Cách hoạt động:
  - Dùng hàm curl_easy_setopt để thực hiện các thao tác cấu hình và curl_easy_perform để kết nối.
  - Lấy email theo index mong muốn (các hàm trên của thư viện curl).
  - Sau đó tiến hành tách lấy các thông tin trong dữ liệu đã lấy (người gửi, nội dung mail, etc).

2.2. Truyền dữ liệu từ client tới server

- Thao tác:
  - Dùng hàm send() của thư viện winsock để gửi nội dung lệnh qua socket.
  - Server sau khi nhận sẽ tiến hành tách chuỗi thành cấu trúc lệnh để tiến hành thực hiện lệnh.
- Hàm tách chuỗi thành cấu trúc lệnh là hàm:

```
Command getCommandInfoFromEmail(string inp)
```

2.3. Truyền dữ liệu từ server tới client

```
void responseToClient(SOCKET ClientSocket, string response, const char* path)
```

- Truyền dữ liệu dạng text/string:
  - Tham số đầu vào: Gồm địa chỉ IP và cổng của client, cùng với thông báo dạng văn bản (text).
  - Cách hoạt động:
    + Server sẽ gửi thông báo đến client thông qua kết nối socket đã thiết lập.
    + Thông báo sẽ được định dạng với loại dữ liệu là văn bản (TYPE = text).
    + Sau đó, dữ liệu sẽ được mã hóa và gửi dưới dạng bit từ server đến client thông qua giao tiếp socket.

- Truyền dữ liệu dạng file:
  - Tham số đầu vào: Gồm địa chỉ IP và cổng của client, thông báo dạng văn bản (text) và có thêm đường dẫn đến file muốn lấy.
  - Cách hoạt động:
    + Gửi đến thông báo cho client loại dữ liệu sẽ nhận là file (TYPE = file).
    + Gửi kích thước file đến client.
    + Cuối cùng bắt đầu gửi các bit dữ liệu đến client.
    + Các thao tác gửi đều thông qua socket đi từ server về client.

2.4. Phản hồi client đến gmail:

```
void responseToEmail(string response,string userMail, const char* path)
```

- Tham số đầu vào: Chuỗi thông tin phản hồi, địa chỉ email của admin, và đường dẫn đến tệp đính kèm (nếu có).
- Cách thức hoạt động:
  + Khởi tạo COM và tạo đối tượng email.
  + Thiết lập thông tin email: Bao gồm địa chỉ người gửi, người nhận, tiêu đề và nội dung email.
  + Cấu hình SMTP: Chỉ định máy chủ SMTP, tài khoản Gmail, mật khẩu, và giao thức bảo mật (TLS/SSL).
  + Thêm tệp đính xkèm: Nếu đường dẫn tệp được cung cấp, tệp sẽ được thêm vào email.
  + Gửi email: Tiến hành gửi email và kiểm tra kết quả gửi:
    + Nếu thành công: Thông báo gửi email hoàn tất.
    + Nếu thất bại: Hiển thị thông báo lỗi chi tiết.

3. Mở webcam và quay video trong khoảng thời gian cho trước

```
void openWebcam(SOCKET ClientSocket, int n)
```

- Tham số đầu vào: thông số thời gian, cổng và địa chỉ IP của client.

- Cách hoạt động:
  - Thực hiện mở webcam, quay video trong khoảng thời gian n giây (thời gian được yêu cầu).
  - Lưu video vào đường dẫn chỉ định, hiển thị khung hình trong quá trình quay.
  - Gửi phản hồi thành công cùng đường dẫn video đến client thông qua socket.

4. Chụp ảnh màn hình

```
void takeScreenshot(const char* filename, SOCKET ClientSocket)
```

- Tham số đầu vào: đường dẫn và tên file muốn lưu ảnh, soket (cổng và địa chỉ IP) của client.
- Cách hoạt động:
  - Thực hiện thao tác chụp ảnh màn hình (lưu trữ từng bit ảnh) ngay thời điểm server nhận được lệnh và lưu file định dạng .bmp vào đường dẫn trong tham số đầu vào.
  - Hàm con **saveBitmap** dùng để thực hiện thao tác lưu các bit ảnh vào file, hỗ trợ cho hàm takeScreenshot.

5. Liệt kê danh sách các ứng dụng đã cài đặt

```
string ListInstalledApplications(SOCKET ClientSocket)
```

- Tham số đầu vào: địa chỉ port và IP của client.
- Cách hoạt động:
  - Dùng cấu trúc IShellItems và hàm BindToHandler của thư viện shobjidl_core.h) quét qua các ứng dụng có trong máy.
  - Dùng hàm GetDisplayName để lấy tên của ứng dụng rồi chuyển từ định dạng LPWSTR sang dạng string và lưu trữ lại tên ứng dụng.
  - Cuối cùng, trả danh sách ứng dụng (dạng string) về client.

6. Mở ứng dụng đã cài đặt trong máy

```
void openApp(string name, SOCKET ClientSocket)
```

- Tham số đầu vào: Tên ứng dụng cần mở, cổng và địa chỉ IP của client socket.
- Chuỗi lệnh: Bắt đầu với "start " (lệnh trong Windows để mở ứng dụng). Dựa trên tên ứng dụng (name), hàm ánh xạ tên ứng dụng sang lệnh tương ứng như sau:
  - "visual studio" → devenv
  - "google chrome" → chrome
  - "paint" → mspaint
  - "calculator" → calc

- …

(Nếu tên ứng dụng không khớp với danh sách đã định nghĩa, lệnh sẽ mặc định sử dụng trực tiếp tên ứng dụng để thực thi (giả sử đó là một ứng dụng hợp lệ trên hệ thống).

– Kết quả:
  - Mở ứng dụng thành công: Gửi thông báo "Open [name] completed!" đến client thông qua socket ClientSocket.
  - Mở ứng dụng thất bại: Gửi thông báo "Unable to open [name]!" đến client.

7. Copy file từ đường dẫn cho trước

```
bool copyFile(const string& source, const string& destination, SOCKET ClientSocket)
```

– Tham số đầu vào: đường dẫn của nguồn muốn copy, đường dẫn của điểm đến muốn copy, địa chỉ port và IP của client.
– Cách hoạt động:
  - Mở file tại đường dẫn nguồn, nếu mở file thành công thì dùng hàm rdbuff() lưu nội dung của file.
  - Lưu tên file nguồn lại để đặt tên cho file copy.
  - Tạo 1 file tại điểm đến muốn copy ra (bằng ofstream) và đặt tên file đó như tên file nguồn đã lưu.
  - Chép vào file đó nội dung có được từ file nguồn.
  - Nếu không mở được file nguồn: "Error: Source file does not exist!".
  - Nếu đường dẫn điểm đến không hợp lệ/không tạo được file tại đường dẫn được yêu cầu "Error: Destination folder not valid!".
  - Nếu copy file thành công: "Copy file completed!".

8. Xóa file từ đường dẫn cho trước

```
void deleteFile(string filePath, SOCKET ClientSocket)
```

– Tham số đầu vào: đường dẫn của file muốn xóa, địa chỉ cổng và IP của client
– Cách hoạt động :
  - Sử dụng hàm remove() của thư viện mặc định, với đầu vào là đường dẫn đến file.
  - Hàm sẽ thực hiện xóa file có đường dẫn đã cho. Trả về client kết quả của thao tác:
    + Nếu thành công: trả về “Delete file successfully”.
    + Nếu thất bại: trả về "Fail to delete file!".

9. Shutdown và restart máy

```
void shutDownOrRestart(int choice, SOCKET ClientSocket)
```

- Tham số đầu vào: thao tác lựa chọn (1 là shutdown, 2 là restart), địa chỉ port và IP client.
- Cách hoạt động:
  - Sau khi chuyển thông báo về client, hàm dùng lệnh shut down (/s) hoặc lệnh restart (/r) của thư viện mặc định để tắt máy hoặc khởi động lại với khoảng thời gian chờ mong muốn (hiện trong chương trình đang là 10s).
  - Thông báo trả về client có dạng:
    + Shut down: "Shutdown computer successfully!".
    + Restart: "Restart computer successfully!".

10. Liệt kê tiến trình đang chạy trong máy

```
struct Application {
    HWND hwnd;
    wstring windowTitle;
    DWORD processID;
};


void listProcess(SOCKET ClientSocket);
vector<Application> listApplications();
string printListOfProcess(const vector<Application>& app);
```

- Hàm chính (listProcess) nhận tham số đầu vào là cổng và địa chỉ IP của client.
- Cách hoạt động: dùng EnumWindows của thư viện mặc định để liệt kê danh sách các tiến trình đang chạy trong máy tính bị điều khiển. Sau đó lưu thành chuỗi danh sách các tiến trình và gửi nội dung về client.

11. Đóng tiến trình đang chạy trong máy

```
void stopProcess(vector<Application>& apps, int choice, SOCKET ClientSocket)
```

- Tham số đầu vào là các tiến trình đang khởi chạy, thứ tự process người dùng muốn đóng và cổng, IP của client.
- Cách hoạt động: hàm dừng tiến trình đang chạy nhờ vào 2 hàm:
  + closeApplicationByHWND : Trong trường hợp lựa chọn phù hợp, hàm này sẽ đóng một ứng dụng hoặc cửa sổ dựa trên **HWND** (handle của cửa sổ).
  + TerminateProcessByID: Hàm này kết thúc một tiến trình (process) dựa trên **ID của tiến trình** (Process ID).

12. Lưu phím trong khoảng thời gian cho trước (keylogger)

```
void keyLogger(int second, SOCKET ClientSocket);
```

- Tham số đầu vào: thời gian muốn keylog, cổng và địa chỉ IP của client.

- Cách hoạt động:
  - **Ghi lại phím nhập**
    + Hàm sẽ thu thập tất cả các phím mà người dùng nhập trong suốt khoảng thời gian đã chỉ định.
    + Các ký tự đặc biệt (ngoài chữ cái và số) như SHIFT, SPACE, ENTER, v.v., sẽ được chuyển thành dạng chữ tương ứng để dễ dàng ghi nhận.
  - **Lưu trữ dữ liệu:**
    + Dữ liệu được ghi lại sẽ được lưu vào file **keylogger.txt**.
    + Để đảm bảo tính tương thích trên nhiều hệ thống, file này sẽ được lưu tại đường dẫn cố định **C:\Users\Public\keylogger.txt**, vì đa số các máy tính đều có thư mục này, giúp giảm thiểu lỗi khi chương trình chạy trên các máy khác nhau.
  - **Gửi thông tin qua client:**
    + Sau khi hết thời gian nhập liệu, thông tin trong file **keylogger.txt** sẽ được gửi qua client. Cuối cùng, thông tin này sẽ được chuyển tiếp đến email của người quản trị.

13. Khóa phím trong khoảng thời gian cho trước

```
void lockKeyboardAndMouse(int seconds, SOCKET ClientSocket)
```

- Tham số đầu vào: biến giá trị thời gian khóa mong muốn, cổng và địa chỉ IP của client.
- Cách hoạt động: Thực hiện khóa bàn phím và chuột trong một khoảng thời gian nhất định (tính bằng giây) bằng cách sử dụng hàm blockInput(), tạm dừng chương trình, sau đó mở khóa và gửi phản hồi đến client qua socket.

B. Chức năng mở rộng:

1. Kiểm tra pin và trạng thái pin

```
void checkBatteryStatus(SOCKET ClientSocket)
```

- Tham số đầu vào: Cổng và địa chỉ IP của client.
- Cách hoạt động:
  - **Lấy thông tin trạng thái pin**: Hàm sử dụng API GetSystemPowerStatus để lấy thông tin trạng thái nguồn điện của hệ thống và lưu trong cấu trúc SYSTEM_POWER_STATUS.
  - **Kiểm tra nguồn điện**:
    + Nếu máy tính đang sử dụng pin (không cắm sạc): Hàm sẽ gửi thông báo kèm theo mức pin hiện tại (phần trăm).
    + Nếu máy tính đang cắm sạc: Hàm gửi thông báo rằng laptop đang cắm sạc kèm mức pin hiện tại.

2. Kiểm tra bộ nhớ ổ đĩa

```
void checkStorage(SOCKET ClientSocket)
```

- Tham số đầu vào: địa chỉ port và IP của client.
- Cách hoạt động:
  - Sử dụng hàm GetLogicalDriveStrings() để lấy tên các ổ đĩa.
  - Dùng hàm GetDiskFreeSpaceEx() để lấy thông tin về bộ nhớ của từng ổ đĩa, sau đó trả về dạng string thông tin của các ổ đĩa.
  - Nếu không kiểm tra được ổ đĩa: "Cannot check storage info!".

3. Lấy file từ đường dẫn cho trước

```
string getFileNameFromPath(string filePath)
```

- Tham số đầu vào: đường dẫn đến file muốn lấy.
- Cách hoạt động:
  - Gửi đến thông báo cho client loại dữ liệu sẽ nhận là file (TYPE = file). Sau đó gửi kích thước file đến client.
  - Bắt đầu gửi các bit dữ liệu đến client. Các thao tác gửi đều thông qua socket đi từ server về client.
  - Client sẽ phản hồi dựa trên kết quả:
    + Không mở được file cần gửi: “Failed to open file”.
    + Không gửi được kích thước file: "Failed to send file size!”.
    + Không gửi được dữ liệu: "Failed to send file data!”.
    + Gửi file thành công: "File sent successfully”.
    + Số bytes dữ liệu: "The number of bytes sent : (số bytes)”.

III. TÀI LIỆU THAM KHẢO

- Microsoft, *vcpkg*. GitHub. [Online]. Available: https://github.com/microsoft/vcpkg.
- GeeksforGeeks, "Socket Programming in C++," *GeeksforGeeks*, 2020. [Online]. Available: https://www.geeksforgeeks.org/socket-programming-cc/.
- "libcurl - the Multiprotocol File Transfer Library," *curl.se*, 2024. [Online]. Available: https://curl.se..
- Microsoft, "Windows Sockets 2," *Microsoft Learn*. [Online]. Available: https://learn.microsoft.com/en-us/windows/win32/winsock/.
- OpenCV, "cv::VideoCapture Class Reference," *OpenCV Documentation*, ver. 3.4. [Online]. Available: https://docs.opencv.org/3.4/d8/dfe/classcv_1_1VideoCapture.html.

- Microsoft, "Windows API Reference," *Microsoft Learn*. [Online]. Available: https://learn.microsoft.com/en-us/windows/win32/api/.
- Microsoft, "Shell entry point," *Microsoft Learn*. [Online]. Available: https://learn.microsoft.com/en-us/windows/win32/shell/shell-entry.
- Microsoft, "Windows Server Documentation," *Microsoft Learn*. [Online]. Available: https://learn.microsoft.com/en-us/windows-server/.
- Mozilla, "HTML: Hypertext Markup Language," *MDN Web Docs*. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/HTML.
- Mozilla, "CSS: Cascading Style Sheets," *MDN Web Docs*. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/CSS.
- cURL, "cURL Tutorial," *cURL Documentation*. [Online]. Available: https://curl.se/docs/tutorial.html.
- GeeksforGeeks, "Client-Server Architecture - System Design," *GeeksforGeeks*. [Online]. Available: https://www.geeksforgeeks.org/client-server-architecture-system-design/.
- cURL, "Using cURL to Read Email," *Everything cURL*. [Online]. Available: https://everything.curl.dev/usingcurl/reademail.html.