ĐẠI HỌC QUỐC GIA TPHCM

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

AI23 (23TNT1), FIT@HCMUS-VNUHCM

Báo cáo Đồ án

Đề tài: Điều khiển máy tính qua Gmail

Môn học: Mạng máy tính

Sinh viên thực hiện:
Đinh Đức Anh Khoa (23122001)
Nguyễn Đình Hà Dương (23122002)
Đinh Đức Tài (23122013)

Giáo viên hướng dẫn:
Th.S Đỗ Hoàng Cường

Ngày 25 tháng 12 năm 2024

Mục lục

1 Giới thiệu

Đây là bài báo cáo cho **đồ án "Điều khiển máy tính qua Gmail"**, môn Mạng máy tính, lớp 23TNT1, Khoa Công nghệ thông tin, Trường Đại học Khoa học tự nhiên - Đại học Quốc gia TP.HCM.

Đồ án được thực hiện bởi nhóm các thành viên:

- Đinh Đức Anh Khoa (23122001)
- Nguyễn Đình Hà Dương (23122002)
- Đinh Đức Tài (23122013)

Đường dẫn repository Github của đồ án: https://github.com/ductai05/socket [1]

Tóm tắt các chức năng điều khiển máy tính thông qua mail:

- Liệt kê IP các máy Server có trong mạng LAN. Chọn máy Server bị điều khiển.
- Chụp màn hình máy tính. Bật/Tắt webcam.
- Liệt kê các app. Đóng/Mở app.
- Liệt kê các file trong một folder. Lấy file. Xóa file.
- Liệt kê các process. Tắt các process.
- Tắt nguồn máy tính Server.

2 Phân công nhiệm vụ, đánh giá mức độ hoàn thành

Bảng phân công nhiệm vụ cho từng thành viên:

Họ và tên	MSSV	Nhiệm vụ	Mức độ hoàn thành
Đinh Đức Anh Khoa	23122001	- Lập trình các chức năng của Server. - Kết nối các máy Server trong mạng LAN bằng Socket. Test và debug mã nguồn.	Tốt (100%)
Nguyễn Đình Hà Dương	23122002	- Thiết kế UI. - Lập trình các chức năng của Client.	Tốt (100%)
Đinh Đức Tài	23122013	- Viết báo cáo. - Lập trình gửi/nhận mail, kết nối Client-Server	Tốt (100%)

3 Các thư viện và công nghệ

3.1 C++

C++ [2] là một ngôn ngữ lập trình đa năng bậc cao được tạo ra bởi Bjarne Stroustrup như một phần mở rộng của ngôn ngữ lập trình C. Ngôn ngữ đã được mở rộng đáng kể theo thời gian và C++ hiện đại có các tính năng: lập trình tổng quát, lập trình hướng đối tượng, lập trình thủ tục, ngôn ngữ đa mẫu hình tự do có kiểu tĩnh, dữ liệu trừu tượng, và lập trình đa hình, ngoài ra còn có thêm các tính năng, công cụ để thao tác với bộ nhớ cấp thấp.

Phiên bản: Đồ án sử dụng phiên bản C++17

3.2 OpenSSL

OpenSSL [3] là một thư viện phần mềm cho các ứng dụng bảo mật truyền thông qua mạng máy tính, giúp chống nghe trộm và xác thực danh tính của máy chủ hoặc bên giao tiếp. OpenSSL sử dụng các thuật toán mã hóa để bảo vệ dữ liệu khỏi việc bị đánh cắp và các cơ chế xác thực để đảm bảo bạn đang kết nối với đúng máy chủ hoặc client. OpenSSL được sử dụng rộng rãi trong các máy chủ web internet, phục vụ phần lớn tất cả các trang web.

Phiên bản Đồ án sử dụng phiên bản Win64 OpenSSL v3.4.0 Light [4]

3.3 cURL

cURL [5] là một dự án phần mềm máy tính cung cấp thư viện (libcurl) và công cụ dòng lệnh (curl) để truyền dữ liệu bằng nhiều giao thức khác nhau. cURL được phát hành lần đầu tiên vào năm 1997. cURL là chữ viết tắt của "Client URL". Nhà phát triển ban đầu và chính của cURL là Daniel Stenberg, nhà phát triển phần mềm người Thụy Điển.

Phiên bản Đồ án sử dụng cURL có sẵn trên hệ điều hành Windows, phiên bản `>=` **8.9.1**.

3.4 windows.h, winsock2.h

windows.h [6] là một file header mã nguồn mà Microsoft cung cấp để phát triển các chương trình truy cập Windows API (WinAPI) thông qua cú pháp ngôn ngữ C. Nó khai báo các hàm WinAPI, các kiểu dữ liệu liên quan và các macro thông dụng.

winsock2.h [6] là một file header mã nguồn giúp ta sử dụng Windows Sockets API (Winsock). Winsock định nghĩa cách mà phần mềm truy cập các dịch vụ mạng. Winsock xác định giao diện chuẩn giữa một ứng dụng client TCP/IP và các tầng phía dưới trong chồng giao thức TCP/IP.

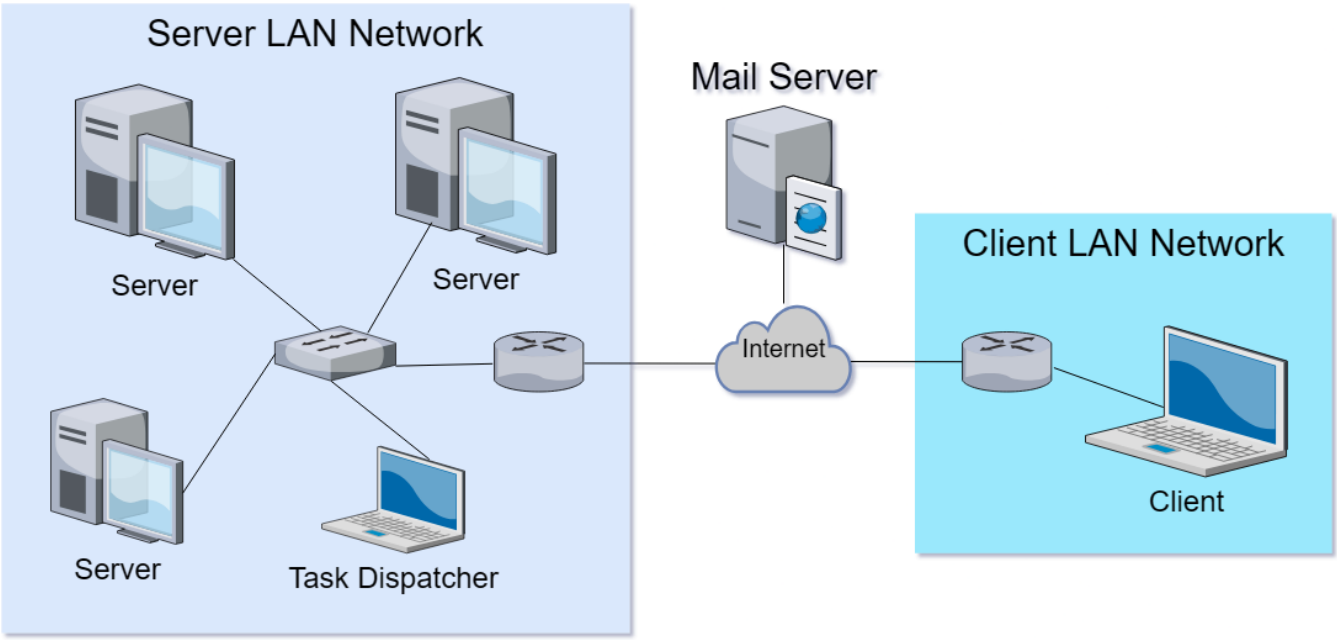
3.5 SDL

SDL [7] (Simple DirectMedia Layer) là một thư viện đa nền tảng được thiết kế để giúp tạo ra các ứng dụng đa phương tiện như trò chơi và chương trình đồ họa. Đồ án sử dụng thư viện `SDL2`, `SDL2_image`, `SDL2_ttf`, `SDL2_mixer` cùng với Visual Studio để biên dịch và xây dựng ứng dụng.

4 Sơ đồ mạng và mô tả mã nguồn

4.1 Tổng quan

Sơ đồ mạng: Các thành phần mạng trong đồ án được minh họa như hình bên dưới.



Hình 1: Sơ đồ kết nối Client-Server thông qua mail

Cách hoạt động: Máy Client gửi lệnh và nghe phản hồi thông qua mail. Ở Server LAN Network, có một Task Dispatcher điều phối lệnh nhận được tới các máy Server để thực hiện nhiệm vụ.

Phương pháp lập trình: Đồ án được lập trình theo phương pháp lập trình hướng thủ tục.

Tổ chức mã nguồn: Mã nguồn (code) được tổ chức thành 2 thư mục chính: Client và Server, bao gồm các file với đuôi *.cpp*, *.h*, *.dll*, ... và các file thư viện khác. Trong đó, Client và Server chia sẻ chung một cách kết nối - gửi và nhận mail thông qua các hàm trong `mail.h`. Tuy nhiên, tùy theo Client và Server, sẽ có một phiên bản `mail.h` với sự bổ sung, chỉnh sửa phù hợp.

- `mail.h`: Định nghĩa các hàm dùng cURL, với phương thức IMAP và SMTP để gửi, nhận mail.
- **Server**: Thiết lập các Server và một Task Dispatcher (máy điều phối nhiệm vụ cho các Server).
  - Gồm: `dispatcher.cpp`, `server.cpp`, `take_screenshot.cpp`, `process.h`, `mail.h`.
- **Client**: Thiết lập Client, gửi yêu cầu cho Server và nhận phản hồi. Cung cấp giao diện cho người dùng tương tác.
  - Gồm: `client.cpp`, `mail.h`, các file *.dll* (thư viện SDL), ảnh...
- **Add-on**: File cài đặt OpenSSL và các file cần thiết khác.

4.2 Chi tiết các thành phần code

4.2.1 Gửi/nhận mail - kết nối client-server

File `mail.h`: Định nghĩa các hàm sử dụng cURL để gửi, nhận email bằng giao thức SMTP, IMAP.

Các hàm chính:

- `sendMail (from, to, subject, body, userPass, fileName)`: Viết nội dung mail, gọi lệnh từ thư viện cURL để gửi mail bằng SMTP.
  - `from`: Địa chỉ mail gửi.
  - `to`: Địa chỉ mail nhận.
  - `subject`: Tiêu đề mail.
  - `body`: Nội dung mail.
  - `userPass`: Mail và mật khẩu (app password[1]), dùng để xác thực thông tin gửi mail.
  - `fileName`: Tên file đính kèm mail.
- `newMail(client, task, numTask, fileContent)`: Thiết lập thông tin cần gửi (nơi gửi, loại yêu cầu, tên file đính kèm, mật khẩu...). Gọi hàm `sendMail` để gửi yêu cầu/phản hồi.
  - `client`: Biến bool xác định Client hay Server gửi yêu cầu.
  - `task`: Loại tác vụ yêu cầu/phản hồi.
  - `numTask`: ID của tác vụ.
  - `fileContent`: Tệp đính kèm (nếu có).
- `readLatestMail(timeLISTEN, isClientLISTEN, MAIL, TASK)`: Đọc thông tin của mail mới nhất.
  - `timeLISTEN`: Thời gian bắt đầu chạy Client/Server.
  - `isClientLISTEN`: Biến bool xác định Client hay Server đang chạy.
  - `MAIL`: Danh sách tiêu đề mail.
  - `TASK`: Danh sách ID mail.
- `autoGetMail(isClientLISTEN)`: Thiết lập một tiến trình liên tục lắng nghe tới mail server để nhận được mail mới nhất.
  - `isClientLISTEN`: Biến bool xác định Client hay Server đang chạy.

[1] App Password: Mật khẩu ứng dụng được tạo riêng cho các ứng dụng của bên thứ ba truy cập tài khoản Google.

Các hàm phụ:

- `getCurrentDateTime()`: Lấy thông tin thời gian hiện tại.
- `base64_encode(in)`: Mã hóa thông tin thành base64.
  - `in`: Chuỗi kí tự base64 cần được giải mã.
- `compareTimeStrings(timeStr1, timeStr2)`: So sánh 2 thời gian dạng chuỗi ký tự, để biết thời gian nào sớm/muộn hơn.
  - `timeStr1`, `timeStr2`: 2 thời gian cần so sánh
- `getID(userPass, isClientLISTEN)`: Dùng cURL và thông qua phương thức IMAP, gửi lệnh tới mail server yêu cầu nhận danh sách các mail có sẵn trong hộp thư.
  - `userPass`: Địa chỉ mail và mật khẩu
  - `isClientLISTEN`: Biến bool xác định là Client hay Server đang gửi yêu cầu
- `readIDMail(orderNow)`: Đọc ID của mail mới nhất có trong hộp thư.
  - `orderNow`: ID của mail hiện tại được trả về.
- `getNewestMail(orderNow, userPass)`: Dùng cURL và thông qua phương thức IMAP, gửi lệnh tới mail server yêu cầu tải xuống mail mới nhất dựa trên ID đã lấy.
  - `orderNow`: ID của mail hiện tại.
  - `userPass`: Địa chỉ mail và mật khẩu
- `extractInfo(subject, responseType, numTask, time)`: Trích xuất các yêu cầu trong mail đã tải. Xác nhận tính hợp lệ của mail.
  - `subject`: Tiêu đề mail.
  - `responseType`: Thể loại mail (request/response).
  - `numTask`: ID của tác vụ.
  - `time`: Thời gian gửi mail.
- `base64_decode(in)`: Giải mã thông tin base64.
  - `in`: Chuỗi ký tự base64 cần được giải mã.
- `saveFile(filename, data)`: Lưu tệp đính kèm mail (nếu có).
  - `filename`: Tên file cần lưu.
  - `data`: Dữ liệu của file đã được giải mã.

File `mail.h` mẫu chỉ cung cấp các mẫu hàm để Server và Client có thể gửi, nhận lệnh thông qua mail. File `mail.h` riêng ở Server và Client có thể được chỉnh sửa cho phù hợp.

4.2.2 Server

File `dispatcher.cpp`: Khởi tạo tiến trình điều phối hoạt động các máy Server trong mạng LAN.

- `sendTask(serverIP, numTask, request)`: Điều phối công việc đến server đang được điều khiển.
  - `serverIP`: IP của server được điều khiển.
  - `numTask`: Mã số của yêu cầu.
  - `request`: Nội dung của yêu cầu.
- `isPortOpen(ip, port)`: Kiểm tra xem một địa chỉ IP nào đó có đang mở cổng kết nối trên một Port nào đó hay không.
  - `ip`: IP cần kiểm tra.
  - `port`: Port cần kiểm tra.
- `checkIP(serversIP, ip, port)`: kiểm tra một địa chỉ IP có phải một máy chủ đang lắng nghe hay không, nếu có thì bỏ vào `serversIP`.
  - `serversIP`: Danh sách IP của các máy chủ.
  - `ip`: IP cần kiểm tra.
  - `port`: Port cần kiểm tra.
- `getServersList(serversIP)`: Lấy danh sách IP của các server bỏ vào `serversIP`.
  - `serversIP`: Danh sách IP của các máy chủ.
- `handleRequest(serversIP, currentIP, numTask, request)`: Xử lý yêu cầu.
  - `serversIP`: Danh sách IP của các máy chủ.
  - `currentIP`: IP của server đang được điều khiển.
  - `numTask`: Mã số của yêu cầu.
  - `request`: Nội dung của yêu cầu.
- `autoGetMail(serversIP)`: Lấy và đọc mail, thực hiện lắng nghe yêu cầu từ người dùng.
  - `serversIP`: Danh sách IP của các máy chủ.

File `server.cpp`: Khởi tạo tiến trình trên một máy Server.

- `start_server()`: Khởi động chương trình và lưu lại đường dẫn của các ứng dụng.
- `createSocket()`: Thiết lập kết nối giao tiếp với chương trình điều phối
- `serveForever()`: Làm cho máy chủ hoạt động vô thời hạn.
- `handleRequest(numTask, request, socket)`: Xử lý các yêu cầu.

  - `numTask`: Mã số của yêu cầu.
  - `request`: Nội dung của yêu cầu.
  - `socket`: Một socket giữa máy chủ và máy điều phối, sử dụng để đóng socket khi cần thiết.

File `take_screenshot.cpp`: Chụp màn hình máy tính.

- `takeScreenshot(Height, Width, targetHeight, targetWidth)`: chụp màn hình và lưu thành "screen.png".
  - `Height`: Chiều cao của màn hình.
  - `Width`: Chiều rộng của màn hình.
  - `targetHeight`: Chiều cao của ảnh đầu ra.
  - `targetWidth`: Chiều rộng của ảnh đầu ra.

File `process.h`: Thực hiện các chức năng của Server.

- `list_apps()`: Tìm các file .exe và viết vào file "uploads/app_list.txt" khi khởi động server.
- `list_files(path)`: In ra các thư mục và file trong thư mục đường dẫn vào file "uploads/-files_list.txt".
  - `path`: Đường dẫn của thư mục cần liệt kê.
- `run_app(path)`: Chạy một chương trình .exe.
  - `path`: Đường dẫn tuyệt đối của chương trình.exe.
- `get_screenshot()`: Thực thi chương trình phụ trợ "take_screenshot.cpp" để chụp màn hình.
- `shut_down()`: Ngưng hoạt động server và tắt nguồn server.
- `camera_switch(cam_status)`: Bật/tắt camera.
  - `cam_status`: 0/1 tương ứng với tắt hoặc bật camera.
- `end_task(task_name)`: Ngừng một chương trình.
  - `task_name`: Tên chương trình dưới dạng <tên.exe>.
- `bool end_task_PID(PID)`: Ngừng một chương trình theo PID.
  - `PID`: PID của chương trình cần dừng.
- `list_running_apps()`: Viết một danh sách các chương trình đang chạy vào file "uploads/running_apps.txt".

4.2.3 Client

File `client.cpp` Tạo tiến trình Client, cung cấp giao diện cho người dùng tương tác.

Các hằng số, cấu trúc dữ liệu:

- Kích thước và thông số màn hình:
  - `SCREEN_WIDTH = 800`: Xác định chiều rộng của cửa sổ app là 1786 pixel.
  - `SCREEN_HEIGHT = 600`: Xác định chiều cao của của sổ app là 868 pixel.
- `enum Page`: Liệt kê các trang trong ứng dụng, bao gồm:
  - `LOGIN`: Trang đăng nhập.
  - `APP`: Trang chính của ứng dụng để giới thiệu đồ án và thành viên nhóm.
  - `REMOTE`: Trang điều kiển dành cho người dùng.
  - `HELP`: Trang trợ giúp, thông tin.
- `struct Sidebar`: Định nghĩa các thuộc tính của thanh bên.
  - `currentPosition`: Vị trí hiện tại của thanh bên.
  - `expanded`: Kiểm tra xem thanh bên có đang mở rộng hay không.
  - `showBackButton`: Kiểm tra xem có hiển thị nút quay lại hay không.

Các hàm chính:

- `runClient(stopClient)`: Khởi chạy giao diện người dùng.
  - `stopClient`: Biến boolen, điều khiển việc dừng chương trình.
- `runListen(stopClient)`: Khởi chạy tiến trình lắng nghe mail phản hồi.
  - `stopClient`: Biến boolen, điều khiển việc dừng chương trình.
- `toggleSidebar(sidebar)`: Thay đổi trạng thái của thanh bên (mở hoặc đóng).
  - `sidebar`: Đối tượng thanh bên.
- `renderText(renderer, font, text, x, y, color, maxWidth)`: Vẽ một đoạn văn bản lên màn hình. Sử dụng thư viện SDL để render văn bản bằng phông chữ TTF (TrueType Font).
  - `renderer`: Đối tượng `SDL_Renderer` dùng để vẽ lên màn hình.
  - `font`: Đối tượng `TTF_Font`, chứa thông tin về phông chữ được dùng để tạo văn bản.
  - `text`: Chuỗi văn bản cần hiển thị.
  - `x, y`: Tọa độ góc trên bên trái để bắt đầu vẽ văn bản.
  - `color`: Màu sắc của văn bản, định nghĩa bằng đối tượng `SDL_Color`.
  - `maxWidth`: Chiều rộng tối đa mà văn bản có thể chiếm.

- `renderSidebar(renderer, sidebar, font)` : Vẽ hình dạng và thông tin của thanh bên khi ở trong giao diện người dùng.
  - `renderer`: Đối tượng `SDL_Renderer` dùng để vẽ lên màn hình.
  - `sidebar`: Đối tượng thanh bên.
  - `font`: Đối tượng `TTF_Font`, chứa thông tin về phông chữ được dùng để tạo văn bản.
- `drawLoginPage(renderer, font, titleFont, username, password, registering, message, backgroundTexture)`: Vẽ trang đăng nhập hoặc đăng ký cho người dùng. Hiển thị các trường nhập liệu cho tên người dùng, mật khẩu và thông điệp (nếu có).
  - `renderer`: Đối tượng `SDL_Renderer` dùng để vẽ lên màn hình.
  - `font, titleFont`: Phông chữ tiêu đề và văn bản thông thường.
  - `username, password`: Thông tin tên người dùng và mật khẩu nhập vào.
  - `registering`: Biến boolean cho biết người dùng có đang ở trang đăng ký hay không.
  - `message`: Thông điệp hiển thị trên màn hình.
  - `backgroundTexture`: Hình nền của trang.
- `handleSidebarAction(itemIndex, running, loggedIn, renderer, font)`: Xử lý các hành động từ thanh bên (sidebar), ví dụ như đăng nhập, đăng xuất.
  - `itemIndex`: Chỉ số mục được chọn trong thanh bên.
  - `running`: Biến boolean cho biết ứng dụng còn đang chạy không.
  - `loggedIn`: Biến boolean cho biết người dùng đã đăng nhập hay chưa.
  - `renderer`: Đối tượng `SDL_Renderer` dùng để vẽ lên màn hình.
  - `font`: Phông chữ để render văn bản.
- `drawAppPage(renderer, font, sidebar, backgroundTexture)`: Vẽ giao diện ứng dụng chính, bao gồm thanh bên và hình nền của trang.
  - `renderer`: Đối tượng `SDL_Renderer` dùng để vẽ lên màn hình.
  - `font`: Đối tượng `TTF_Font`, chứa thông tin về phông chữ được dùng để tạo văn bản.
  - `sidebar`: Đối tượng `Sidebar` dùng để vẽ thanh bên của ứng dụng.
  - `backgroundTexture`: Hình nền trang chính của ứng dụng.
- `drawHelpPage(renderer, font, sidebar, backgroundTexture)`: Vẽ trang trợ giúp của ứng dụng.
  - `renderer`: Đối tượng `SDL_Renderer` dùng để vẽ lên màn hình.
  - `font`: Phông chữ dùng để render văn bản trên trang trợ giúp.
  - `sidebar`: Đối tượng `Sidebar` dùng để vẽ thanh bên trợ giúp.
  - `backgroundTexture`: Hình nền của trang trợ giúp.
- `drawSubmitButton(renderer, font)`: Vẽ nút "Send" trong trang Remote.

  - `renderer`: Đối tượng `SDL_Renderer` dùng để vẽ lên màn hình.
  - `font`: Phông chữ dùng để render văn bản trên nút.

- `drawButton(renderer, rect, text, font, color, textColor)`: Vẽ một nút với văn bản trên giao diện người dùng.
  - `renderer`: Đối tượng `SDL_Renderer` dùng để vẽ lên màn hình.
  - `rect`: Đối tượng `SDL_Rect` định nghĩa hình dạng và kích thước của nút.
  - `text`: Văn bản hiển thị trên nút.
  - `font`: Phông chữ dùng để render văn bản trên nút.
  - `color`: Màu nền của nút.
  - `textColor`: Màu của văn bản hiển thị trên nút.

- `drawRemotePage(renderer, font, sidebar, option1Selected, option2Enabled, option2Title, processID, logMessages, option, selectedOptionText, backgroundTexture, imageTexture, loadImage, visibleLines, logHeight, scrollPosition)`: Vẽ trang điều khiển từ xa (remote), gồm các tùy chọn, thanh cuộn, log.
  - `renderer`: Đối tượng `SDL_Renderer`, dùng để vẽ lên màn hình.
  - `font`: Đối tượng `TTF_Font`, chứa thông tin về phông chữ được dùng để tạo văn bản.
  - `sidebar`: Đối tượng `Sidebar` dùng để vẽ thanh bên.
  - `option1Selected, option2Enabled`: Các biến boolean cho biết các tùy chọn có được chọn hoặc kích hoạt không.
  - `option2Title`: Tiêu đề của tùy chọn 2.
  - `processID`: ID của quá trình hiện tại.
  - `logMessages`: Mảng chứa các thông điệp log hiển thị trên màn hình logConsole.
  - `option`: Các tùy chọn hiện có cho người dùng.
  - `selectedOptionText`: Văn bản của tùy chọn đã chọn để gửi cho server.
  - `backgroundTexture, imageTexture`: Hình nền và ảnh nền của trang.
  - `loadImage`: Biến boolean cho biết có cần tải ảnh không.
  - `visibleLines, logHeight`: Số dòng log hiển thị và chiều cao của phần log.
  - `scrollPosition`: Vị trí hiện thị các câu lệnh trong trang Remote

- `handleEvents(renderer, event, option1Selected, option, selectedOptionText, loadImage, submitButton, processID, logMessages, tenfile, scrollPosition, logHeight, visibleLines, imageTexture, check, font, sidebar, option2Enabled, option2Title, backgroundTexture)`: Xử lý các sự kiện từ người dùng: nhấp chuột, cuộn chuột, cập nhật giao diện tương ứng của trang remote.
  - `renderer`: Đối tượng `SDL_Renderer`, dùng để vẽ lên màn hình.
  - `event`: Đối tượng chứa thông tin sự kiện từ SDL (nhấp chuột, di chuyển chuột, ...).

  - `option1Selected, option2Enabled, check`: Các biến boolean theo dõi trạng thái của các tùy chọn.
  - `option`: Mảng các tùy chọn người dùng có thể chọn.
  - `selectedOptionText`: Văn bản của tùy chọn hiện tại.
  - `loadImage`: Biến boolean xác định có cần tải ảnh hay không.
  - `submitButton`: Hình dạng của nút gửi.
  - `processID`: ID của tiến trình.
  - `logMessages`: Mảng chứa các thông điệp log.
  - `tenfile`: Tên tệp hiện tại.
  - `scrollPosition`: Vị trí của thanh cuộn.
  - `logHeight, visibleLines`: Chiều cao của log và số dòng có thể hiển thị.
  - `imageTexture`: Tải ảnh nếu cần thiết.
  - `font`: Đối tượng `TTF_Font`, chứa thông tin về phông chữ được dùng để tạo văn bản.
  - `sidebar`: Thanh bên của ứng dụng.
  - `option2Title`: Tiêu đề của tùy chọn thứ hai.
  - `backgroundTexture`: Hình nền của trang.

- `handleButtonClick(renderer, event, scrollPosition, visibleLines, totalLines, upButton, downButton, clear, logMessages)`: Xử lý các hành động nhấp vào nút trên giao diện: cuộn lên, cuộn xuống, làm mới trong LogConsole.
  - `renderer`: Đối tượng `SDL_Renderer`, dùng để vẽ lên màn hình.
  - `event`: Sự kiện nhấp chuột hoặc thao tác khác.
  - `scrollPosition`: Vị trí của thanh cuộn.
  - `visibleLines`: Số dòng log có thể hiển thị.
  - `totalLines`: Tổng số dòng log.
  - `upButton, downButton, clear`: Các nút trên giao diện để cuộn lên, cuộn xuống hoặc làm mới log.
  - `logMessages`: Mảng chứa các thông điệp log.

Các hàm phụ:

- `copyToClipboard(text)`: Sao chép văn bản vào bộ nhớ tạm (clipboard).
  - `text`: Văn bản cần sao chép.
- `pasteFromClipboard()`: Dán nội dung từ bộ nhớ tạm (clipboard).
- `validateLogin(username, password)`: Kiểm tra tính hợp lệ của thông tin đăng nhập.
  - `username, password`: Thông tin đăng nhập của người dùng.
- `registerUser(username, password)`: Đăng ký người dùng mới.
  - `username, password`: Thông tin đăng ký của người dùng mới.

5 Hướng dẫn cài đặt, hình ảnh minh họa

5.1 Hướng dẫn cài đặt

- Đồ án được cài đặt trên hệ điều hành Windows 10 (hoặc Windows 11) với thư viện cURL có sẵn. Cần cài đặt thêm OpenSSL.
- Đường dẫn repo Github của đồ án: https://github.com/ductai05/socket [1]

Cài đặt OpenSSL

Để xác thực việc gửi/nhận mail, cần cài đặt OpenSSL và cài biến môi trường

- Tải Win64 OpenSSL v3.4.0 Light (https://slproweb.com/products/Win32OpenSSL.html) hoặc dùng file có sẵn trong repo Github để cài đặt.
- Sau khi cài đặt xong OpenSSL v3.4.0, thêm `C:\Program Files\OpenSSL-Win64\bin` vào PATH của biến môi trường.

Tải và chạy file thực thi

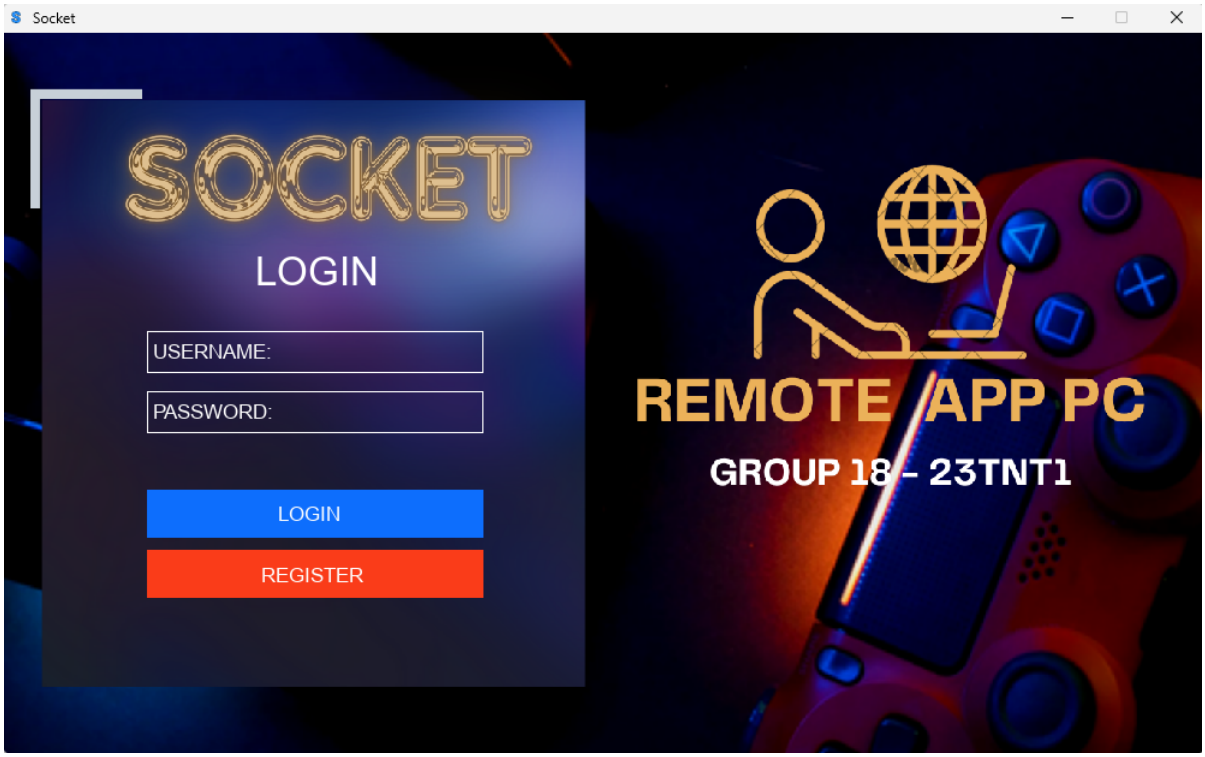
- Tải file nén `release.v1.0.0.zip` gồm các file `take_screenshot.cpp, dispatcher.exe server.exe` và thư mục `client` (chứa `client.exe`) từ phần *releases* của repo Github: https://github.com/ductai05/socket/releases/tag/v1.0.0 [8]
- Thư mục `client` (chứa `client.exe`) được đặt trên máy **Client** (điều khiển). File `server.exe` và `take_screenshot.cpp` được đặt trên các máy **Server** (bị điều khiển) và có thể có nhiều Server. File `dispatcher.exe` được đặt trên máy **Task Dispatcher** (máy điều phối). Các máy Server và máy Task Dispatcher phải được đặt trong **cùng một mạng LAN**.
- Thiết lập Server - Task Dispatcher, Client: các Server (`server.exe`) sẽ được chạy trước, sau đó Task Dispatcher được chạy (`dispatcher.exe`). Điều này giúp Task Dispatcher quét trong mạng LAN xem có những máy Server nào đang hoạt động. Lúc này có thể chạy Client (`client.exe`) để điều khiển các Server.

5.2 Minh hoạ đồ án

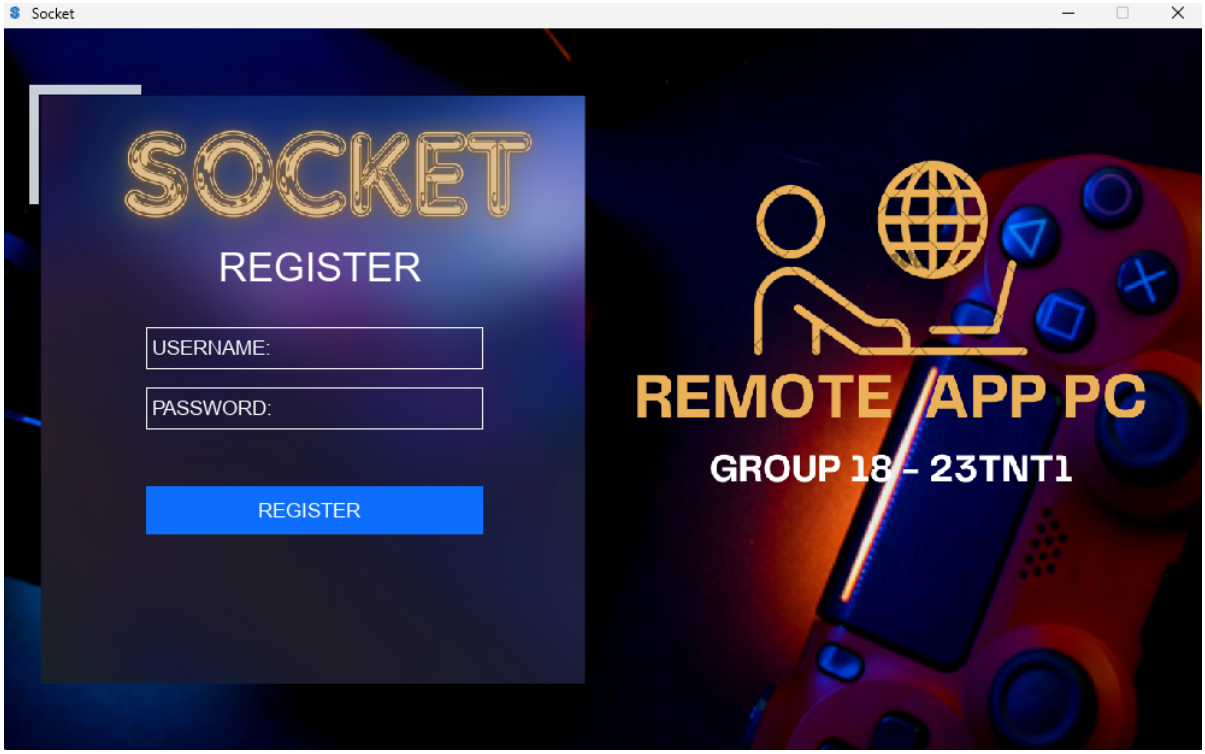
5.2.1 Video thuyết trình

Video Youtube thuyết trình: https://youtu.be/wPzeurOE8So [9]

5.2.2 Hình ảnh minh họa

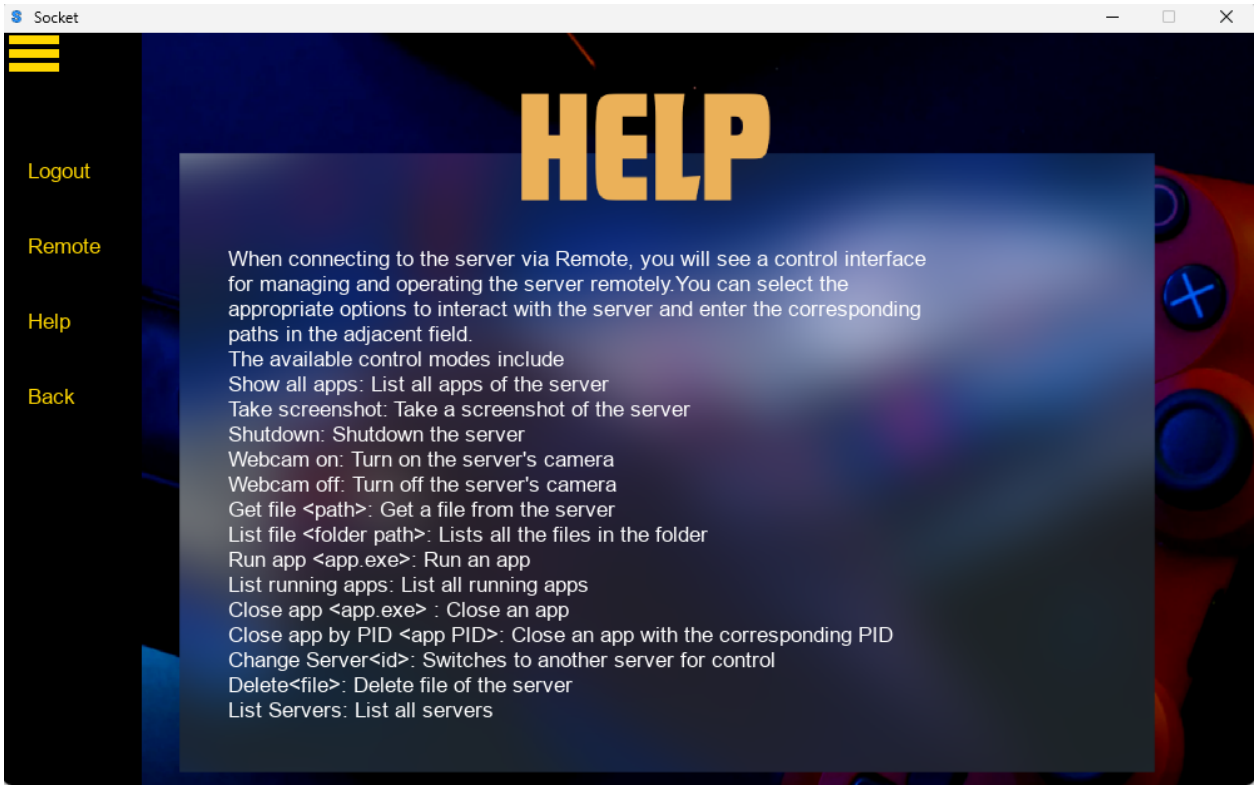


Hình 2: Đăng nhập Client

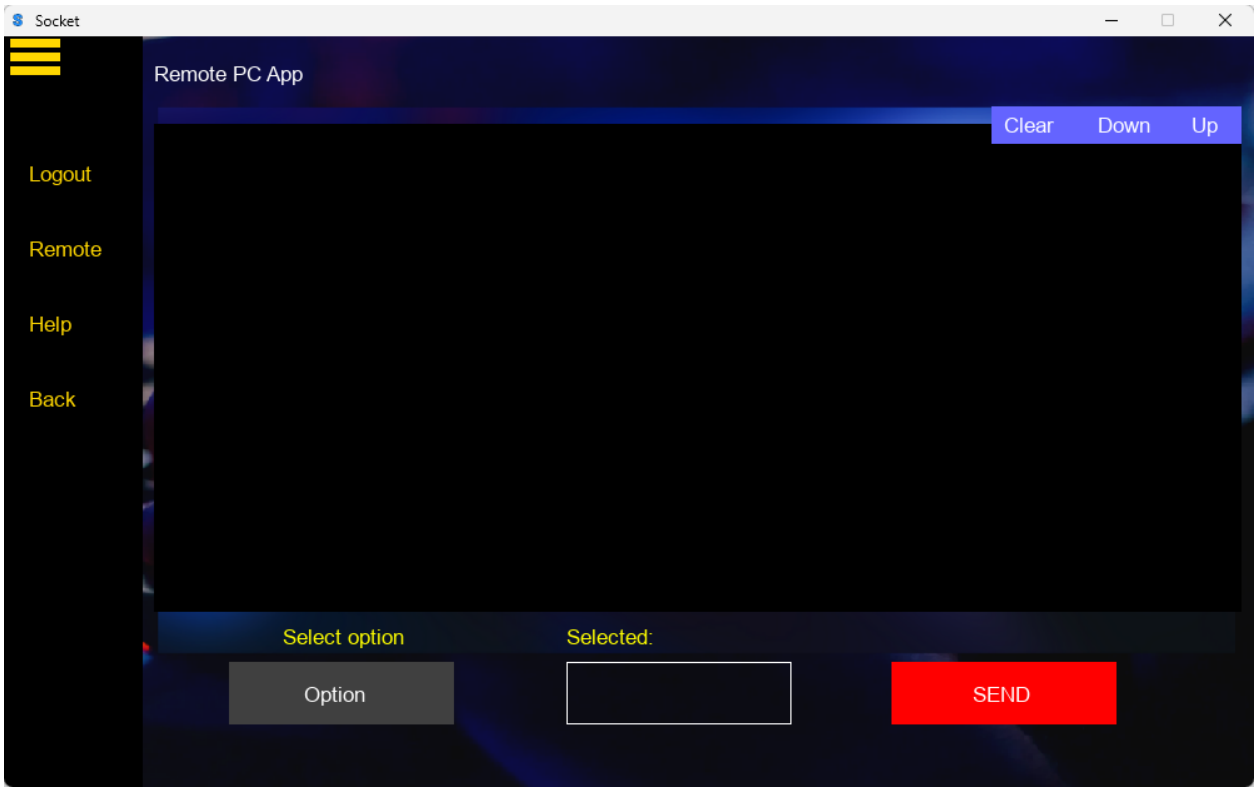


Hình 3: Đăng kí tài khoản Client

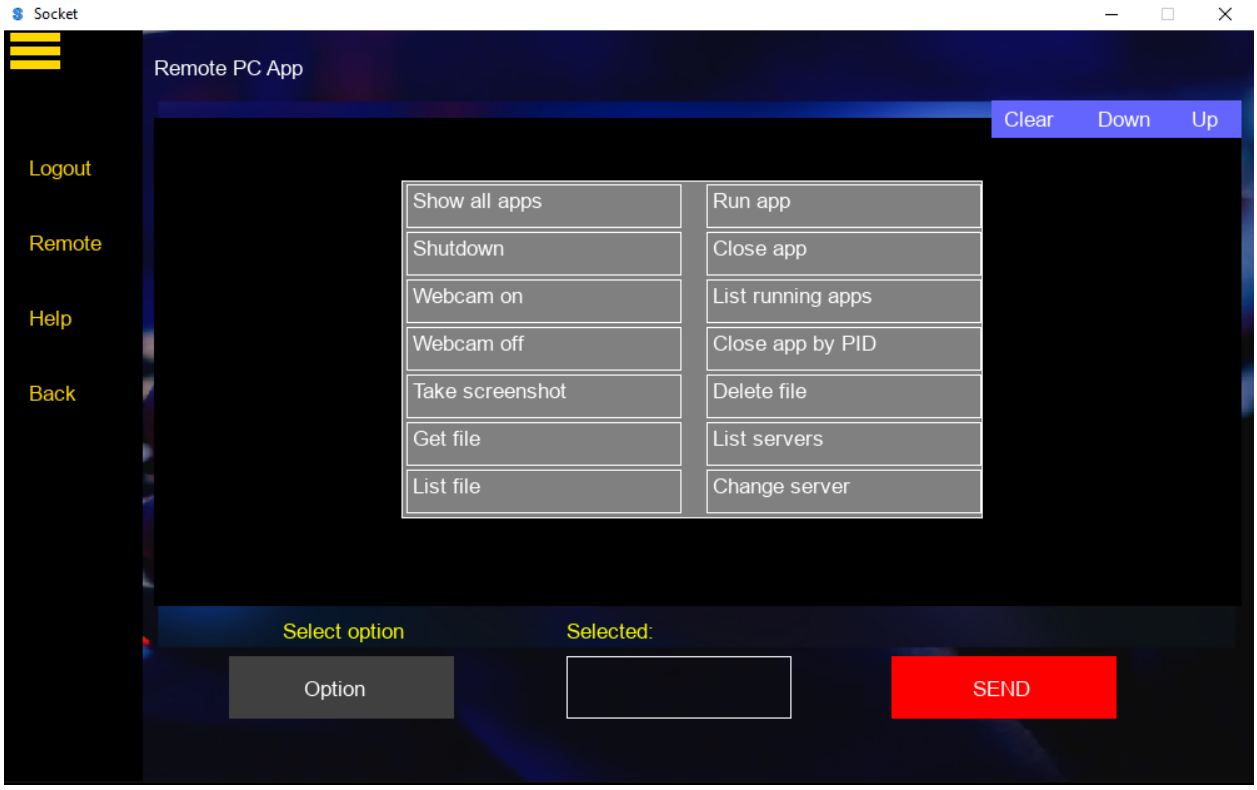
Hình 4: Giao diện giới thiệu app Client



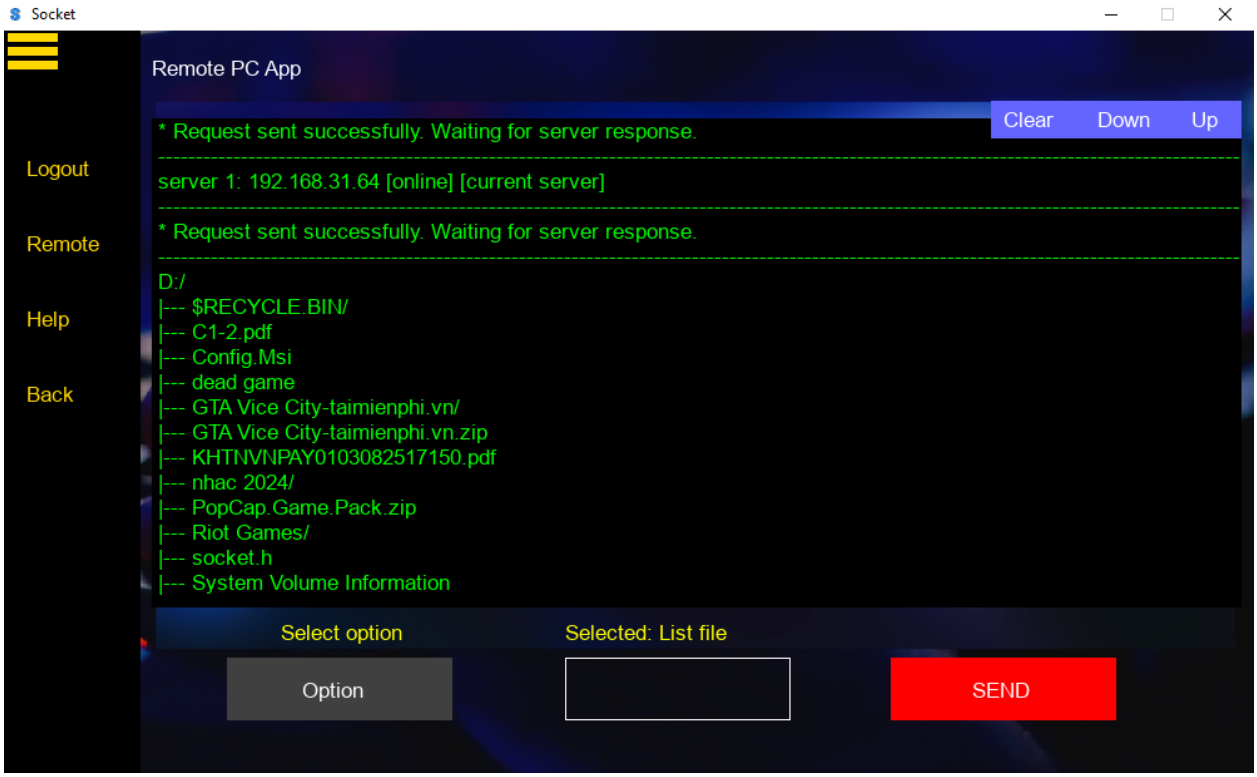
Hình 5: Hướng dẫn sử dụng



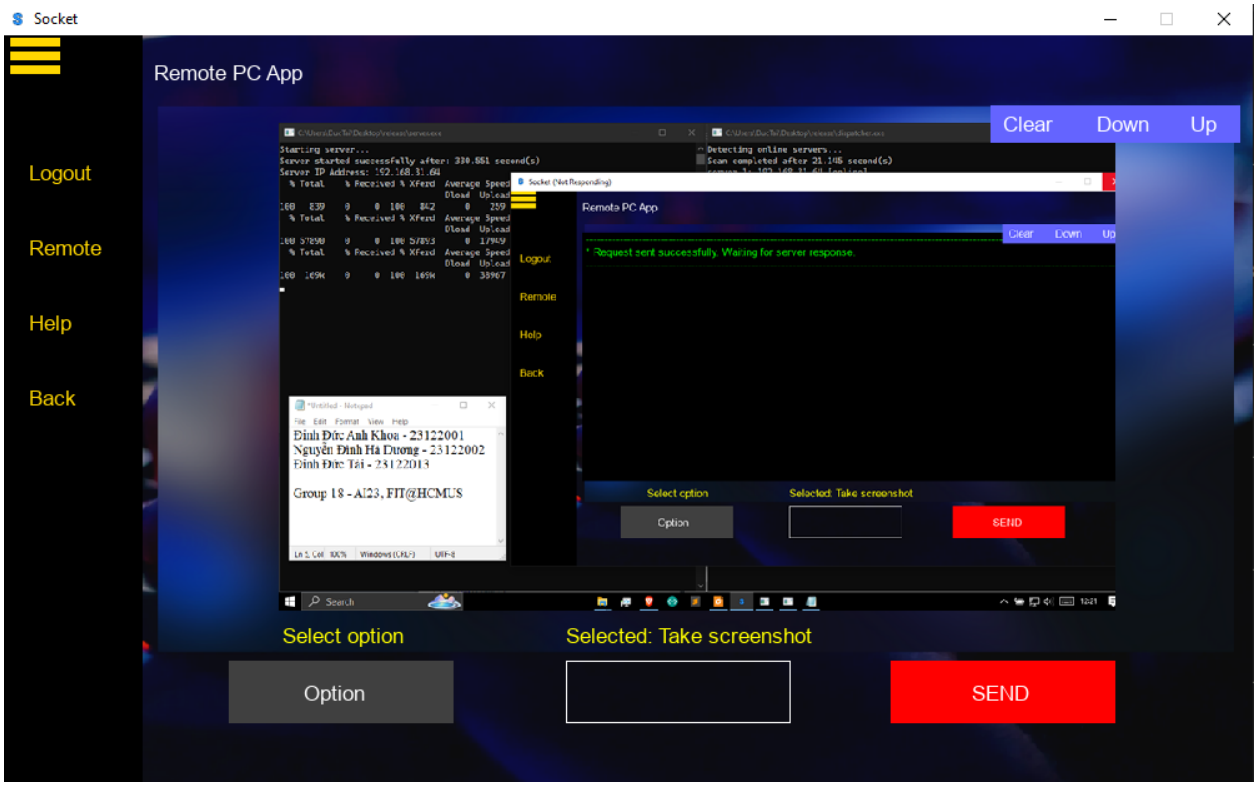
Hình 6: Giao diện điều khiển các Server của app Client



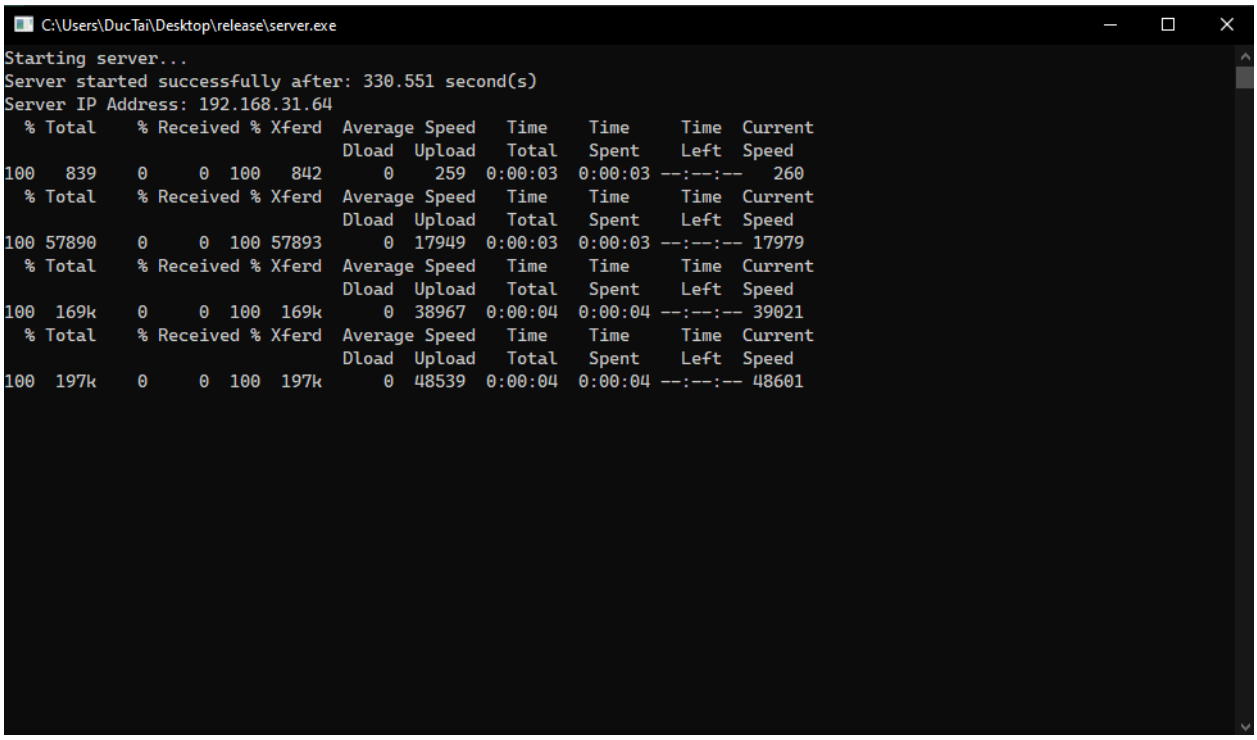
Hình 7: Bảng lựa chọn các chức năng điều khiển



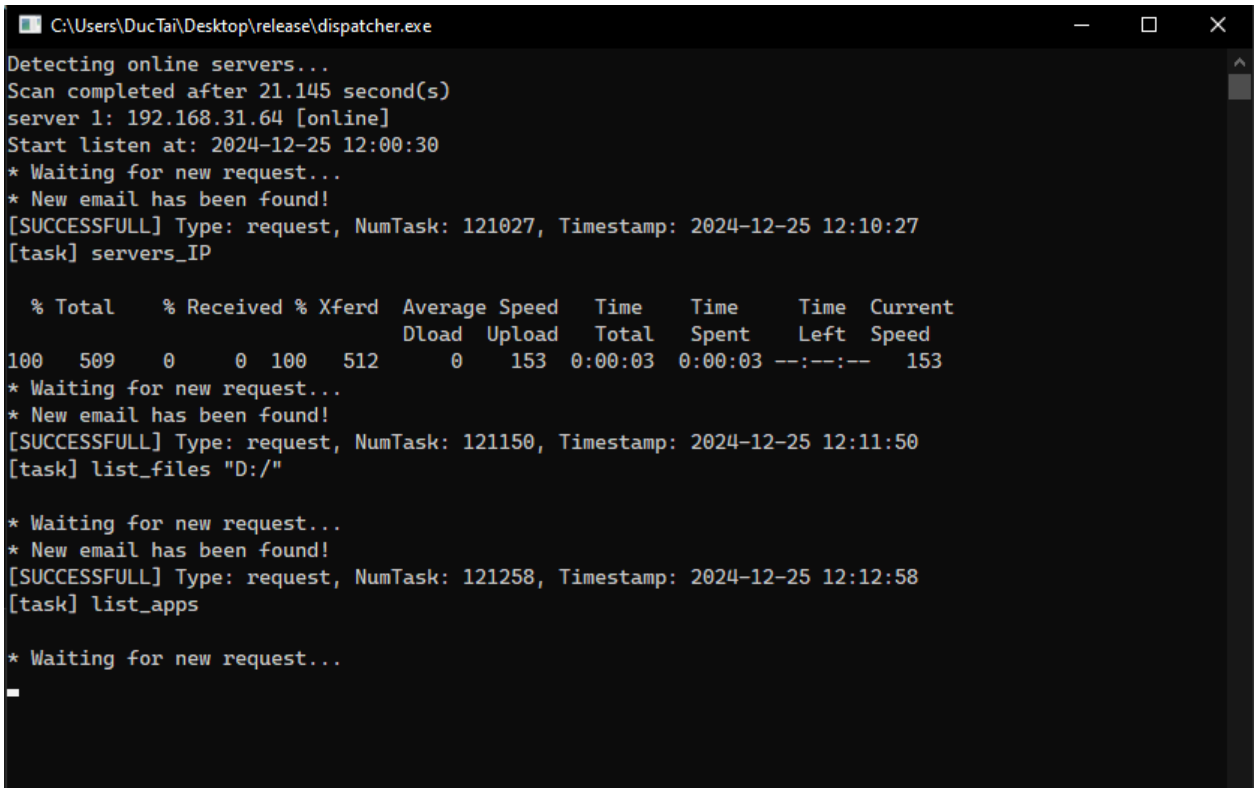
Hình 8: Client hiển thị các thông tin được trả về từ Server



Hình 9: Client hiển thị screenshot lấy từ máy Server



Hình 11: Cửa sổ Console của Server



Hình 10: Cửa sổ Console của Task Dispatcher

Tài liệu

[1] Anh Khoa, Ha Duong, and Duc Tai. Socket Programming: Control Computer via Gmail - CSC10008, AI23@HCMUS. https://github.com/ductai05/socket/, 2024.

[2] cplusplus.com. C++ language. https://cplusplus.com/doc/tutorial/.

[3] OpenSSL Project. openssl/openssl. https://github.com/openssl/openssl.

[4] SLPro Web. Win32 openssl. https://slproweb.com/products/Win32OpenSSL.html.

[5] cURL. command line tool and library for transferring data with urls. https://curl.se/.

[6] Microsoft. windows.h. https://learn.microsoft.com/en-us/windows/win32/api/.

[7] libsdl.org. Simple directmedia layer. https://www.libsdl.org/.

[8] Anh Khoa, Ha Duong, and Duc Tai. Control Computer via Gmail (release). https://github.com/ductai05/socket/releases/tag/v1.0.0, 2024.

[9] Anh Khoa, Ha Duong, and Duc Tai. [Demo Youtube] Control Computer via Gmail. https://www.youtube.com/watch?v=wPzeurOE8So, 2024.