1. GIỚI THIỆU ĐỒ ÁN

- Tên đồ án: **TelePC**
- Chức năng: Giám sát từ xa và kiểm soát máy tính khác trong một mạng máy tính: Xem địa chỉ MAC, theo dõi màn hình máy tính, chụp ảnh màn hình, duyệt thư mục và tập tin máy tính, gửi/nhận files, chỉnh sửa Registry (Windows), quản lý apps/processes, giám sát và bật/tắc thao tác bàn phím từ xa, đăng xuất, tắt máy.
- Môi trường phát triển và công nghệ sử dụng: Phần mềm viết bằng ngôn ngữ Python, thiết kế giao diện với tkinter, giao tiếp giữa các máy qua socket và sử dụng các thư viện hỗ trợ khác để thực hiện các chức năng.

2. THƯ VIỆN HỖ TRỢ

Các thư viện sử dụng trong đồ án bao gồm: socket, tkinter, pickle, struct, json, os, sys, threading, psutil, io, keyboard, pynput, PIL, uuid, re, winreg. Trong đó:

- socket: dùng tạo kết nối
- tkinter: tạo giao diện
- pickle, struct, json: định dạng dữ liệu gửi và nhận
- os, sys, threading, psutil, io: các thao tác hệ thống, thư mục, tập tin
- uuid: lấy các thông số từ máy tính, bao gồm cả địa chỉ MAC
- re: xử lý biểu thức chính quy
- keyboard, pynput: xử lý nhập xuất
- PIL: xử lý hình ảnh
- winreg: xử lý các thao tác lên Registry của Windows

3. NỘI DUNG

Kết nối

- Server sẽ mở một socket để lắng nghe kết nối
- Bên Client, người dùng sẽ nhập địa chỉ IP của Server muốn kết nối. Sau đó, Client sẽ tạo một socket để gửi yêu cầu kết nối đến Server. Server sẽ chấp nhận kết nối và dùng một socket khác để kết nối với Client.
- Sau khi kết nối thành công, Client sẽ được gửi 1 trong 7 yêu cầu đến Server, bao gồm:
    1. Xem địa chỉ MAC (MAC Address)
    2. Lắng nghe phím (Keylogger)
    3. Xem cây thư mục (Directory Tree)
    4. Xem màn hình trực tiếp (Live Screen)
    5. Xem ứng dụng hoặc tiến trình (Application/Process)
    6. Tắt máy hoặc thoát khỏi màn hình desktop (Shutdown/Logout)
    7. Xem Registry (Registry)

Xem MAC Address

- Client gửi một thông điệp "MAC" tới Server.
- Sau khi Server nhận được thông điệp "MAC", nó sẽ dùng phương thức uuid.getnode() của thư viện uuid để lấy địa chỉ MAC → chuyển sang hệ Hexa → gửi cho Client.
- Client nhận dữ liệu và hiển thị thông tin.

Hình 1 Kết quả trả về của chức năng Xem địa chỉ MAC

KeyLogger

- Client gửi một thông điệp "KEYLOG" tới Server
- Sau khi Server nhận được thông điệp "KEYLOG", nó sẽ khởi tạo biến nhớ ishook = 0, islock = 0, cont = "". Server sẽ tạo một thread để lắng nghe phím bằng phương thức Listener của pynput.keyboard. Đồng thời, nó sẽ luôn lắng nghe tiếp xem thông điệp của Client là gì.
    1. Nếu thông điệp tiếp theo của Client là "HOOK" thì Server sẽ kiểm xem ishook mang giá trị gì. Nếu ishook = 0, tức là trạng thái không đang lắng nghe phím thì ta sẽ gán ishook = 1 và thêm vào chuỗi cont các phím lắng nghe được. Nếu ishook = 1, tức là trạng thái đang lắng nghe phím thì ta sẽ gán ishook = 0 và dừng thêm vào chuỗi cont các phím lắng nghe được.
    2. Nếu thông điệp tiếp theo của Client là "PRINT" (tức người dùng Client nhấn "PRINT") thì Server sẽ gửi chuỗi cont lắng nghe được cho Client và gán giá trị thành chuỗi rỗng cont = "".
    3. Nếu thông điệp tiếp theo của Client là "LOCK" thì Server sẽ kiểm xem islock mang giá trị gì. Nếu islock = 0, tức là trạng thái không khoá bàn phím thì ta sẽ gán islock = 1 và khóa bàn phím bằng phương thức keyboard.block_key() của thư viện keyboard. Nếu islock = 1, tức là trạng thái đang khóa bàn phím thì ta sẽ gán islock = 0 và mở khóa bàn phím bằng phương thức keyboard.unblock_key() của thư viện keyboard.
    4. Nếu thông điệp tiếp theo của Client là "QUIT" (tức người dùng Client nhấn "BACK") thì Server sẽ dừng lắng nghe bàn phím và kết thúc chức năng.

Hình 2 Giao diện chức năng KeyLogger

Xem cây thư mục, sao chép và xóa file

- Client gửi thông điệp "DIRECTORY" tới Server để yêu cầu vào gói chức năng này.
- Giao diện: Client khởi tạo giao diện gồm một khung trống bên trái để hiển thị cây thư mục cùng đường dẫn được chọn và một số chỉ dẫn, bên phải là các nút bấm tương ứng với các chức năng.
- Trong khi đó Server sẵn sàng chờ nhận tín hiệu để thực hiện chức năng tương ứng. Giao diện cây thư mục sử dụng widget Treeview của thư viện Tkinter với mỗi file hoặc thư mục là một node và được trỏ đến bởi node thư mục cha. Các tương tác mở node và chọn node ứng với thao tác mở xem các thành phần trong thư mục và chọn đường dẫn tới thư mục đó.

Hình 3 Giao diện chức năng Xem cây thư mục, Sao chép / Xóa file

- Kịch bản các chức năng:
  - Xem cây thư mục:
    - Người dùng click vào nút SHOW trên giao diện và Client sẽ gửi thông điệp "SHOW" đến Server để vào tính năng xem cây thư mục.
    - Khởi tạo cây thư mục ban đầu gồm các ổ đĩa: Server duyệt qua tất cả các tên ổ đĩa (A đến Z) rồi gửi danh sách các ổ đĩa tồn tại (dùng hàm os.path.isdir của thư viện os) cho Client dưới dạng file pickle (hàm showTree phía server). Client nhận danh sách này, gán các node cho các ổ đĩa với parent là rỗng và display lên giao diện.
    - Người dùng tùy ý click đóng mở các thư mục. Mỗi lần mở Client sẽ gửi path thư mục cho Server và Server gửi lại các thành phần trong thư mục (duyệt bằng hàm os.listdir) kèm thông tin thành phần đó là một thư mục hay tập tin (kiểm tra bằng hàm os.path.isdir), Client gán các thành phần này vào các node mới với node cha là thư mục cha và display lên giao diện kèm nút open (nút dấu cộng bên trái tên thư mục), nếu thành phần là tập tin sẽ không có nút open.
    - Nếu thư mục cần mở bị từ chối truy cập bởi hệ điều hành, Server gửi thông điệp "error" cho Client hiện thông báo "Cannot open this directory!"
  - Copy file từ Client sang Server:
    - Người dùng click vào nút "SEND FILE TO FOLDER" sau khi đã chọn thư mục cần copy vào trên cây thư mục của Server (đường dẫn sẽ được

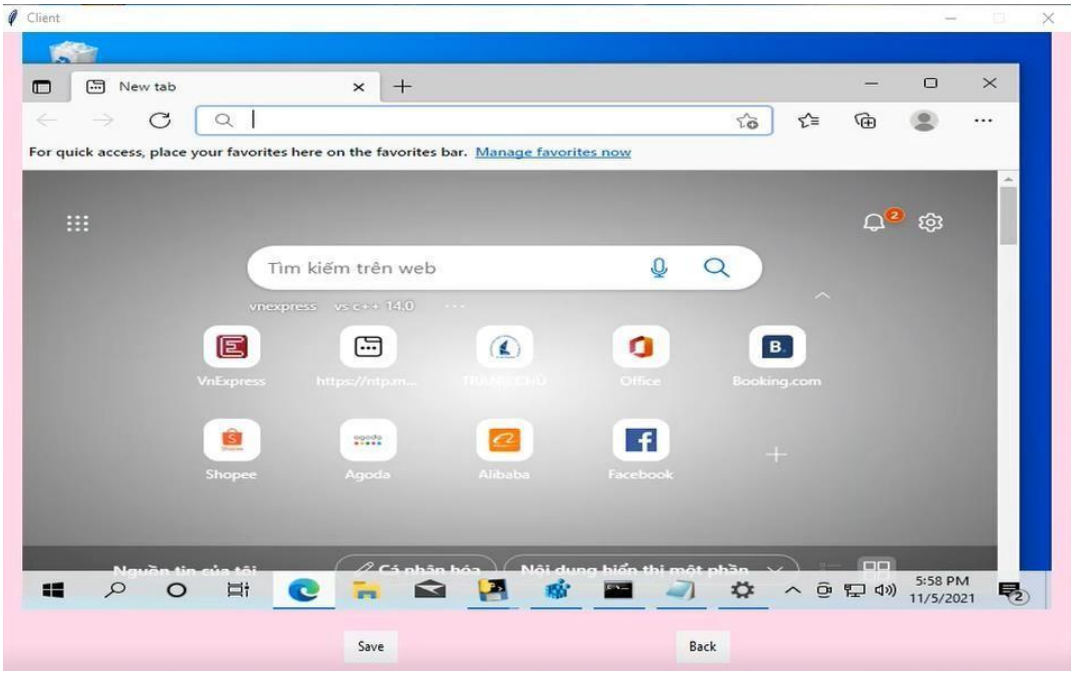
hiển thị ở khung "Selected path" khi chọn), Client gửi thông điệp "COPYTO" đến Server.
        - Client hiện lên cửa sổ chọn file cần gửi. Nếu người dùng không chọn Client sẽ gửi thông điệp "-1" cho Server và kết thúc tính năng này. Nếu có file hợp lệ được chọn, nội dung và các thông tin của file (tên file, kích thước) cũng như địa chỉ cần copy vào ở Server được Client gửi đi.
        - Server nhận các thông tin trên, tạo tập tin tại đường dẫn nhận được, ghi nội dung vào và gửi phản hồi thành công hay không cho Client.
        - Client sẽ hiện thông báo với kết quả tương ứng.
    - Copy file từ Server sang Client (optional):
        - Người dùng click vào nút "COPY THIS FILE" sau khi đã chọn tập tin cần copy trên cây thư mục của Server (tương tự, đường dẫn hiển thị ở khung Selected path), Client gửi thông điệp "COPY" để thông báo vào tính năng.
        - Client hiển thị cửa sổ cho phép chọn thư mục cần copy tập tin vào.
        - Nếu người dùng không chọn hoặc đường dẫn tập tin cần copy của Server không hợp lệ (đường dẫn đến thư mục thay vì file,...), Client sẽ gửi thông điệp "-1" cho Server và kết thúc tính năng.
        - Nếu hợp lệ, Client tiến hành gửi đường dẫn đến tập tin cần copy cho Server và nhận về các thông tin cũng như nội dung tập tin đó.
        - Sau đó Client tạo tập tin tại thư mục được chọn và ghi nội dung nhận được vào rồi hiện thông báo kết quả.
    - Xóa file trên máy Server:
        - Người dùng click vào nút "DELETE" sau khi đã chọn tập tin cần xóa trên cây thư mục của Server. Client gửi thông điệp "DEL" và đường dẫn được chọn.
        - Server nhận đường dẫn, kiểm tra tồn tại bằng hàm os.path.exists và thực hiện xóa bằng hàm os.remove rồi gửi kết quả thành công hay có ngoại lệ xảy ra cho Client.
        - Client nhận kết quả và hiển thị thông báo.
    - Khi click nút "BACK" Client sẽ gửi thông điệp "QUIT" cho Server biết đã kết thúc gói chức năng và thoát ra main menu.
- Flow của gói chức năng:
    - Sau khi nhận thông điệp "DIRECTORY" để vào gói chức năng này, socket của Server chờ một trong các thông điệp "SHOW", "COPYTO", "COPY", "DEL", "QUIT" để đi vào các tính năng cụ thể hoặc thoát về main menu.
    - Dùng vòng lặp while True và biến bool isMod được kiểm tra đầu mỗi lần lặp xem có cần nhận thông điệp mới không.
    - Sau khi vào tính năng xem cây thư mục ("SHOW"), Server sẽ luôn chờ nhận *tín hiệu đường dẫn* cần mở trên cây thư mục, nếu Client không mở node nữa mà chuyển sang tính năng khác thì:
        - *Thông điệp của tính năng* đó sẽ được gửi đi, Server nhận được *thông điệp đến tính năng khác* thay vì *đường dẫn*.

- Hàm sendListDirs ở phía Server: nhận vai trò gửi list các thư mục con của đường dẫn được chọn cho Client. Trong trường hợp trên nó sẽ trả về [False, <thông điệp mới>].
- Khi đó biến isMod được bật lên True nên ở lần lặp while tiếp theo sẽ vào luôn tính năng mới mà không cần chờ nhận thêm thông điệp đi đến tính năng.

○ Với các tính năng khác, việc gửi nhận đã được đóng gói hoàn toàn nên khi hoàn thành biến isMod được gán False để chờ nhận thông điệp đến tính năng khác.

Live Screen / Screen Capture

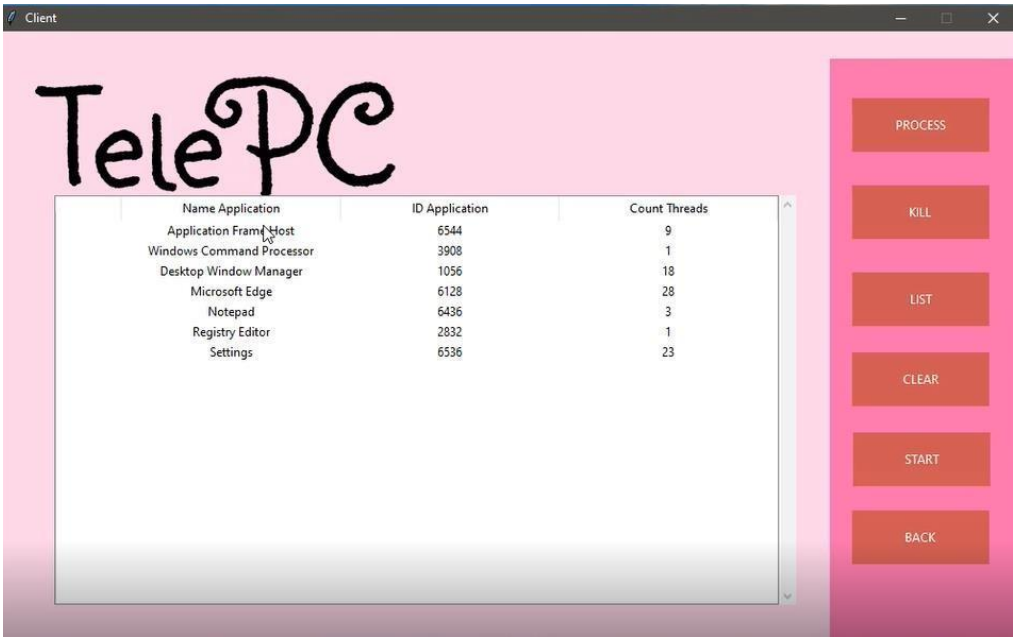
- Client gửi một thông điệp "LIVESCREEN" tới Server.
- Sau khi Server nhận được thông điệp "LIVESCREEN" sẽ thực hiện liên tục quá trình chụp ảnh màn hình hiện tại và gửi đi cho đến khi nhận được tín hiệu kết thúc từ client:
  - Lấy ảnh màn hình bằng phương thức ImageGrab.grab(), sau đó chuyển sang bytes và gửi về Client.
  - Lắng nghe thông điệp kế tiếp từ Client. Nếu thông điệp là "NEXT_FRAME" thì tiếp tục lặp lại quá trình trên. Nếu thông điệp là "STOP_RECEIVING" thì Server thoát khỏi chức năng Live Screen.
- Client liên tục nhận dữ liệu từ Server → thay đổi kích thước ảnh → gắn lên màn hình hiển thị → gửi thông điệp "NEXT_FRAME" để yêu cầu Server tiếp tục gửi hình ảnh.
  1. Nếu người nhấn nút "Save" thì hình ảnh chụp màn hình của Server sẽ được lưu theo đường dẫn mà người dùng chọn bằng phương thức asksaveasfile()
  2. Nếu người dùng nhấn nút "Back" thì sẽ quay về màn hình chính, đồng thời Client sẽ gửi đến Server thông điệp "STOP_RECEIVING".



Hình 4 Giao diện chức năng Live Screen / Screen Capture

Application / Process

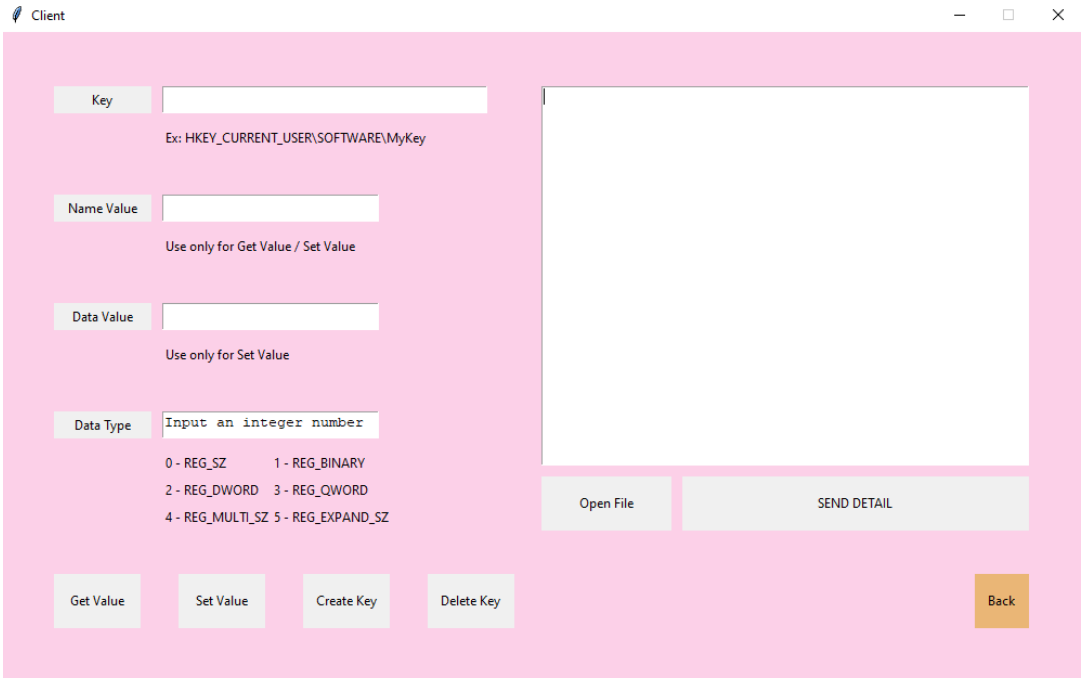
- Client gửi một thông điệp "APP_PRO" tới Server.
- Sau khi Server nhận được thông điệp "APP_PRO", nó sẽ luôn lắng nghe thông điệp từ Client.
    - Nếu thông điệp là "0" (tức người dùng Client nhấn "KILL") thì Server sẽ lắng nghe tiếp giá trị pid (tức id của app/process muốn kill). Sau đó Server sẽ dùng lệnh os.system('taskkill.exe /F /PID' + str(pid)) để kill app/process đó.
    - Nếu thông điệp là "1" (tức người dùng Client nhấn "LIST") thì Server sẽ lắng nghe tiếp xem Client muốn liệt kê Application hay Process. Nếu là Application thì Server sẽ dùng lệnh os.popen('powershell "gps | where {$_.mainWindowTitle} | select Description, ID, @{Name=\'ThreadCount\';Expression ={$_.Threads.Count}}').read().split('\n') để lấy thông tin về Name, ID, Thread counts của app và tách nó ra làm 3 list Name, ID, Thread counts. Còn nếu là Process thì Server sẽ duyệt lần lượt các proc của psutil.process_iter() và lấy các proc.name(), proc.pid, proc.num_threads() gán vào 3 list Name, ID, Thread counts của process. Sau khi có được 3 list, Server sẽ gửi 3 list đó về cho Client bằng pickle và struct.
    - Nếu thông điệp là "2" (tức người dùng Client nhấn "START") thì Server sẽ lắng nghe tiếp tên app/process Client muốn start (pname) → dùng câu lệnh os.system(pname) để start app/process đó.
    - Nếu thông điệp là "QUIT" (tức người dùng Client nhấn "BACK") thì Server sẽ thoát khỏi chức năng App_Process.
- Sau khi thực hiện các tính năng, người dùng có thể nhấn "LIST" để xem kết quả hiển thị trên màn hình.



Hình 5 Giao diện chức năng App/Process Manager

Registry

- Các hàm được đề cập ở phần này đều thuộc thư viện winreg - thư viện được sử dụng chính cho module này với các hàm tương tác với key và value của Registry.
- Client gửi thông điệp "REGISTRY" cho Server để thông báo vào module Registry và hiển thị giao diện như hình bên dưới.



Hình 6 Giao diện chức năng Registry Editor

- Server sau khi nhận thông điệp sẽ chuyển vào module Registry và luôn lắng nghe chờ đợi thông điệp ứng với từng chức năng trong module từ Client. Các thông điệp được Client gửi dưới dạng ID từ 0 đến 4 tương ứng với các chức năng. Các thông tin được đóng gói thành một dictionary theo format {'ID': <ID chức năng>, 'path': <path đến key>, 'name_value': <tên value>, 'value': <data value>, 'v_type' = <kiểu value>} và được gửi đi với format json
    - ID = 0 (Send Detail) chức năng chỉnh sửa registry thông qua file *.reg hoặc file văn bản thuần (Send Detail).
        - Người dùng có thể nhập trực tiếp nội dung cần chỉnh sửa lên khung trống bên phải theo định dạng file .reg hoặc chọn "Open File" để mở lên file
          .reg có sẵn.
        - Chọn "SEND DETAIL", Client cho toàn bộ nội dung này vào key *"path"* trong format dictionary trên và gửi.
        - Server tạo file .reg với nội dung nhận được và chạy nó để cập nhật vào Registry.
        - Server trả về kết quả thành công hay có ngoại lệ xảy ra để Client hiển thị thông báo.
    - ID = 1 (Get Value): chức năng lấy một giá trị của key.

        - Người dùng nhập đường dẫn đến key và tên giá trị cần lấy giá trị rồi click "Get Value".
        - Client gửi các thông tin theo format dictionary như trên.
        - Server dùng hàm OpenKey để mở key, hàm QueryValueEx để lấy giá trị theo tên giá trị truy vấn, sau đó đóng key bằng hàm CloseKey và trả về giá trị lấy được nếu thành công.
    - ID = 2 (Set Value): chức năng set giá trị .
        - Người dùng nhập đường dẫn đến key, tên giá trị, giá trị cần set và kiểu giá trị theo chỉ dẫn trên giao diện rồi click "Set Value".
        - Client gửi theo format dictionary như trên.
        - Server nhận gói tin json, parse thông tin ra tương ứng và gọi các hàm CreateKey, OpenKey để mở key.
        - Tùy vào giá trị của v_type là một trong các giá trị REG_SZ, REG_BINARY, REG_DWORD, REG_QWORD, REG_MULTI_SZ, REG_EXTEND_SZ mà xử lý giá trị cần set cho phù hợp, kết hợp bắt các lỗi giá trị không hợp lệ để đưa về mặc định rồi gọi hàm SetValueEx để nhập giá trị và sau đó CloseKey
        - Trong trường hợp key chứa các values chưa tồn tại, chương trình sẽ mặc định thêm mới key (tương đương Create Key) và đưa các giá trị vào (ngoại trừ các key không được phép truy cập)
        - Server trả về kết quả thành công hay có ngoại lệ xảy ra.
    - ID = 3 (Create Key): chức năng tạo mới một key.
        - Người dùng nhập đường dẫn cho key cần tạo.
        - Client gửi nội dung theo format dictionary như trên.
        - Server nhận, parse thông tin và gọi hàm CreateKey để tạo key.
    - ID = 4 (Delete Key): chức năng xóa key. Tương tự chức năng tạo key. Server gọi hàm DeleteKey để xóa.
- Dựa vào kết quả thực thi ở Server thành công hay không, Server sẽ gửi thông điệp trả về cho Client để Client đưa ra thông báo thao tác đã thành công hay thất bại / không hợp lệ (Key không hợp lệ, giá trị không hợp lệ, truy cập vùng không có quyền điều khiển)
- Khi người dùng click nút "BACK", Client gửi thông điệp "STOP_EDIT_REGISTRY" cho Server để thông báo thoát module và quay lại main menu.

Shutdown và Logout

- Client gửi một thông điệp "SD_LO" tới Server.
- Sau khi Server nhận được thông điệp "SD_LO", nó sẽ lắng nghe tiếp xem thông điệp của Client là gì.
    - Nếu thông điệp tiếp theo của Client là "SHUTDOWN" thì Server sẽ thực hiện câu lệnh os.system('shutdown -s -t 15') của thư viện os để shutdown máy.
    - Nếu thông điệp tiếp theo của Client là "LOGOUT" thì Server sẽ thực hiện câu lệnh os.system('shutdown -l') của thư viện os để logout máy.

4. TÀI LIỆU THAM KHẢO

Python 3.10 Documentation: *https://docs.python.org/*

PIL Documentation: *https://pillow.readthedocs.io/en/stable/*

Pynput Documentation: *https://pynput.readthedocs.io/en/latest/*

Psutil Documentation: *https://psutil.readthedocs.io/en/latest/*

Các thảo luận và hướng dẫn liên quan đến ứng dụng socket trên các trang *https://stackoverflow.com/*, *https://www.geeksforgeeks.org/*

Các repositories có chủ đề liên quan trên *https://github.com/*