**KHOA CÔNG NGHỆ THÔNG TIN**



**4.0**

**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN**

227 Nguyễn Văn Cừ, Phường 4, Quận 5, TP.HCM

Điện thoại: (08) 38.354.266 – Fax: (08) 38.350.096

BÁO CÁO ĐỒ ÁN

MẠNG MÁY TÍNH

**ĐỀ TÀI**

**REMOTE DESKTOP**

🙦🙤

**Giảng viên hướng dẫn: Thầy Nguyễn Thanh Quân**

**Thành viên nhóm:**

|  |  |  |
| --- | --- | --- |
| **STT** | **Họ và tên** | **MSSV** |
| 1 | Lê Tuấn Anh | 22120011 |
| 2 | Trương Tiến Anh | 22120017 |
| 3 | Cao Gia Bảo | 22120020 |

**TP. Hồ Chí Minh, tháng 12 năm 202****3**

Mục lục

[PHẦN MỞ ĐẦU 2](#_Toc155959772)

[I. Giới thiệu 2](#_Toc155959773)

[II. Nội dung nghiên cứu 2](#_Toc155959774)

[III. Công cụ thực hiên 2](#_Toc155959775)

[IV. Phân chia nhiệm vụ 2](#_Toc155959776)

[PHẦN NỘI DUNG 3](#_Toc155959777)

[I. Giao diện 3](#_Toc155959778)

[II. Mô hình hoạt động 6](#_Toc155959780)

[**1.** **Khởi tạo kết nối**: 6](#_Toc155959781)

[**2.** **Keystroke**: 6](#_Toc155959781)

[**3.** **Screenshot**: 6](#_Toc155959781)

[**4.** **Display**: 6](#_Toc155959781)

[**5.** **File Transfer**: 8](#_Toc155959785)

[**6.** **Shutdow**: 6](#_Toc155959781)

[**7. Exit**: 6](#_Toc155959781)

[PHẦN TỔNG KẾT 12](#_Toc155959789)

[I.Video demo: 12](#_Toc155959790)

[II.Tài liệu tham khảo 12](#_Toc155959792)

# PHẦN MỞ ĐẦU

## I. Giới thiệu

Socket là điểm cuối end-point trong liên kết truyền thông hai chiều (two-way communication) biểu diễn kết nối giữa Client – Server. Các lớp Socket được ràng buộc với một cổng port (thể hiện là một con số cụ thể) để các tầng TCP (TCP Layer) có thể định danh ứng dụng mà dữ liệu sẽ được gửi tới.

## II. Nội dung nghiên cứu

Xây dựng chương trình điều khiển PC từ xa. Đồ án **Remote Desktop Control** được thết kế dựa trên khuôn mẫu của ứng dụng Teamviewer. Xây dụng bằng ngôn ngữ Python, và thư viện Socket để tiến hành

## III. Công cụ thực hiên

Visual Studio Code (Viết code)

VMware (Tạo máy ảo)

Qt Dersigner (Thiết kế giao diện)

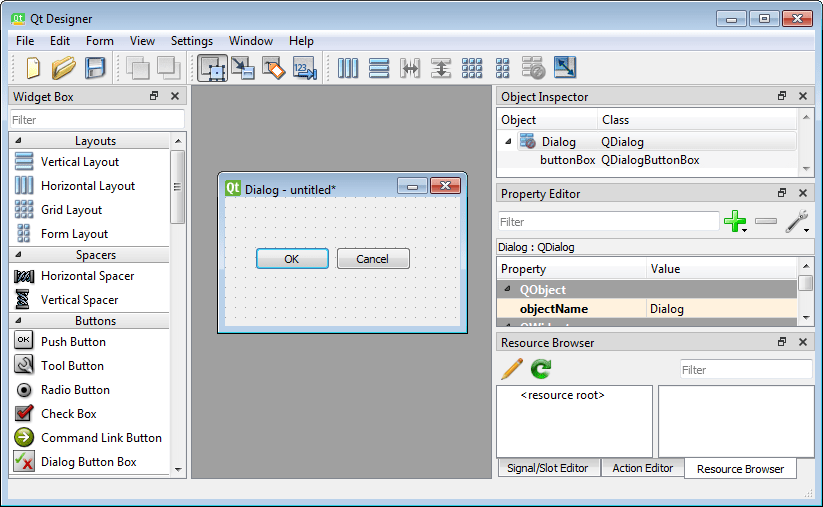
## IV. Phân chia nhiệm vụ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nội dung nghiên cứu** | **Người thực hiện** | **Nhiệm vụ** | **Vấn đề đang gặp** | **Mức độ hoàn thành** |
| Giao diện | **Cao Gia Bảo** | * Thiết kế giao diện cho chương tình | Không có | 100% |
| Chức năng | **Trương Tiến Anh** | * File Transfer * Screenshot * Display | Không có | 100% |
| Chức năng | **Lê Tuấn Anh** | * Keystroke * Exit * Shutdown | Không có | 60% |

**PHẦN NỘI DUNG**

1. **Giao diện**

Công cụ sử dụng: **Qt Dersigner**

**Qt Designer** là một công cụ để nhanh chóng xây dựng giao diện người dùng đồ họa với các widget từ khung Qt GUI . Nó cung cấp cho người dùng một giao diện kéo và thả đơn giản để bố trí các thành phần như nút, trường văn bản, hộp tổ hợp và hơn thế nữa. Đây là ảnh chụp màn hìcủa QtDesigner trên Windows:

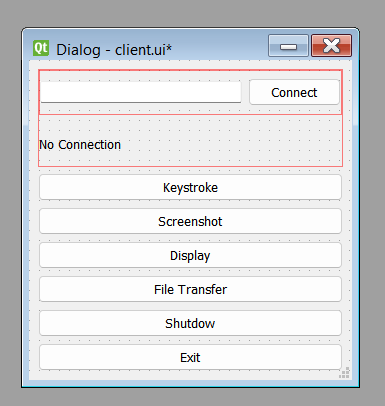
BBước làm:

* + Thiết kế giao diện theo chức năng mong muốn
  + Lưu file dưới dạng namefile.ui
  + Dể Export file.ui sang file.py ta chạy dòng code sau trong terminal

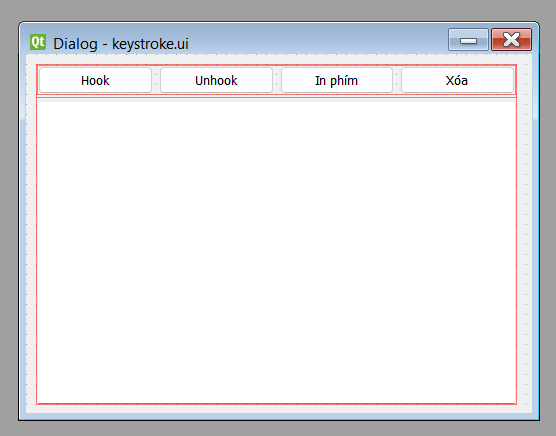
**Pyuic6 –x namefile.ui –o namefile.py**

* + Kết nối các chứ năng đến phần Code của trương trình

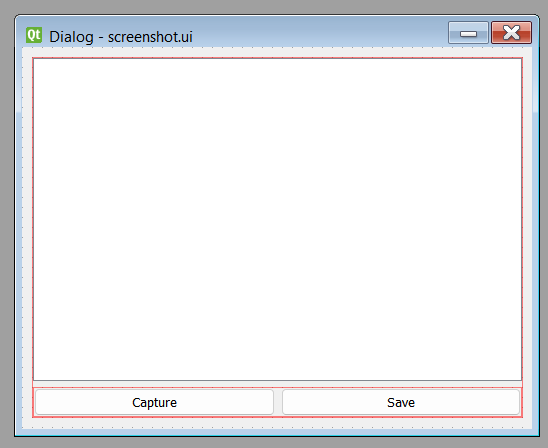
Giao diện chung



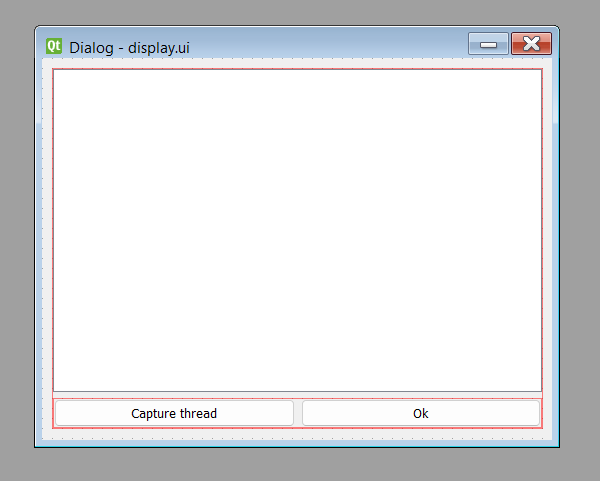
Keystroke



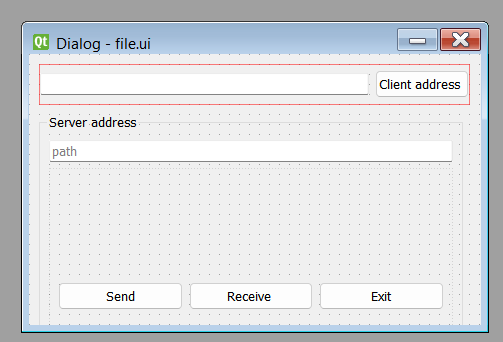
Screenshot



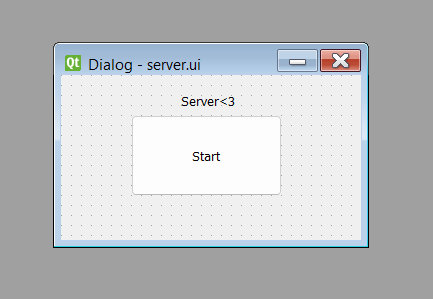
Display



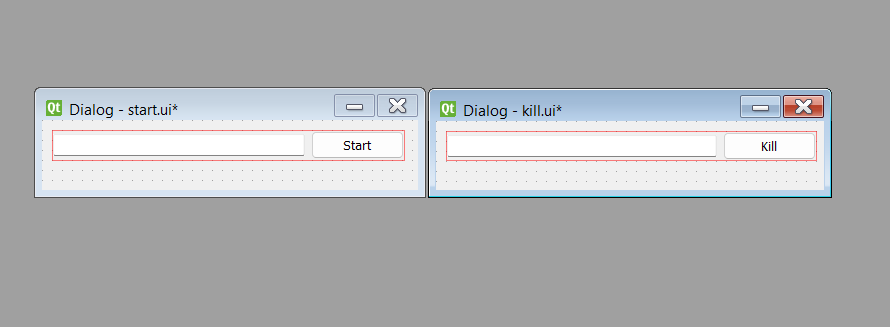
File Transfer



Kết nối Server-Client



Các hộp thoại ứng dụng bổ trợ



## 

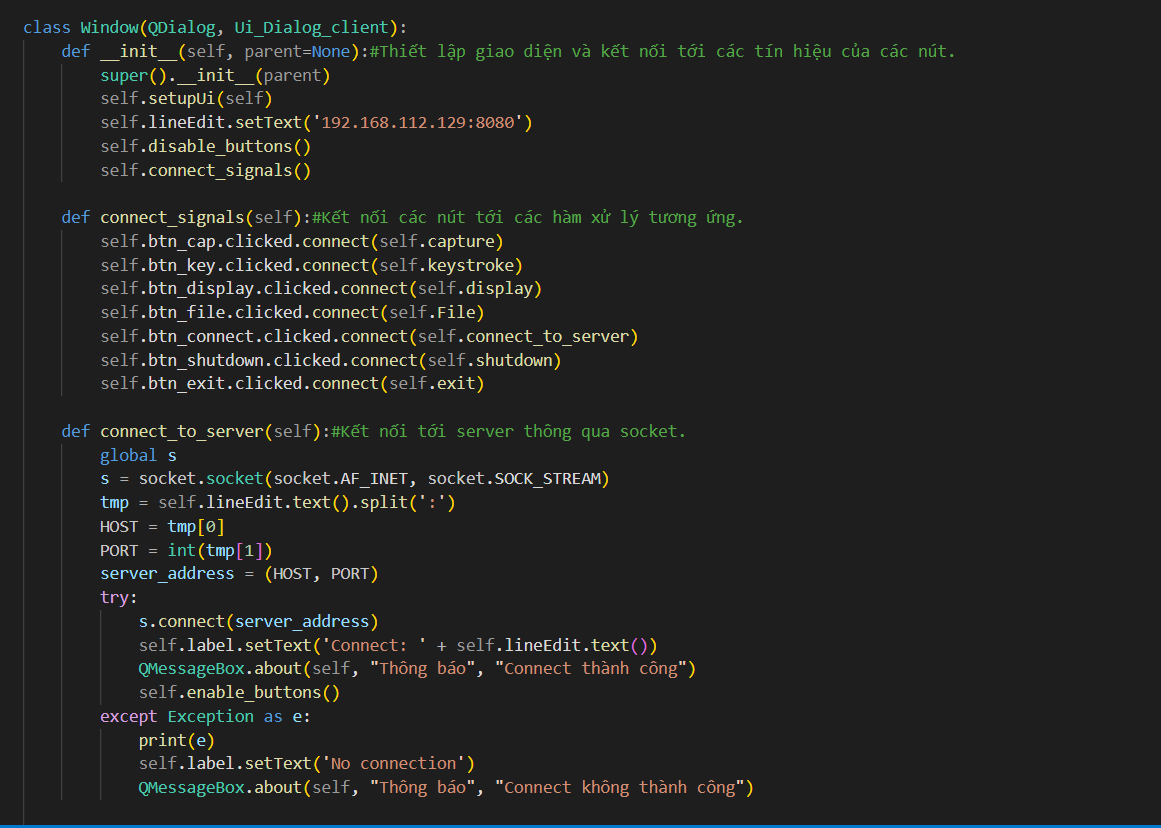
## II. Chức năng

1. **Khởi tạo kết nối:** Khởi tạo phiên kết nối giữa Client-Server

* Phía Client

Khi chạy **Class Window(QDialog, Ui\_Dialog\_client)** để thiết lập giao diện màn hình chính và kết nối của các nút với các hàm tương ứng

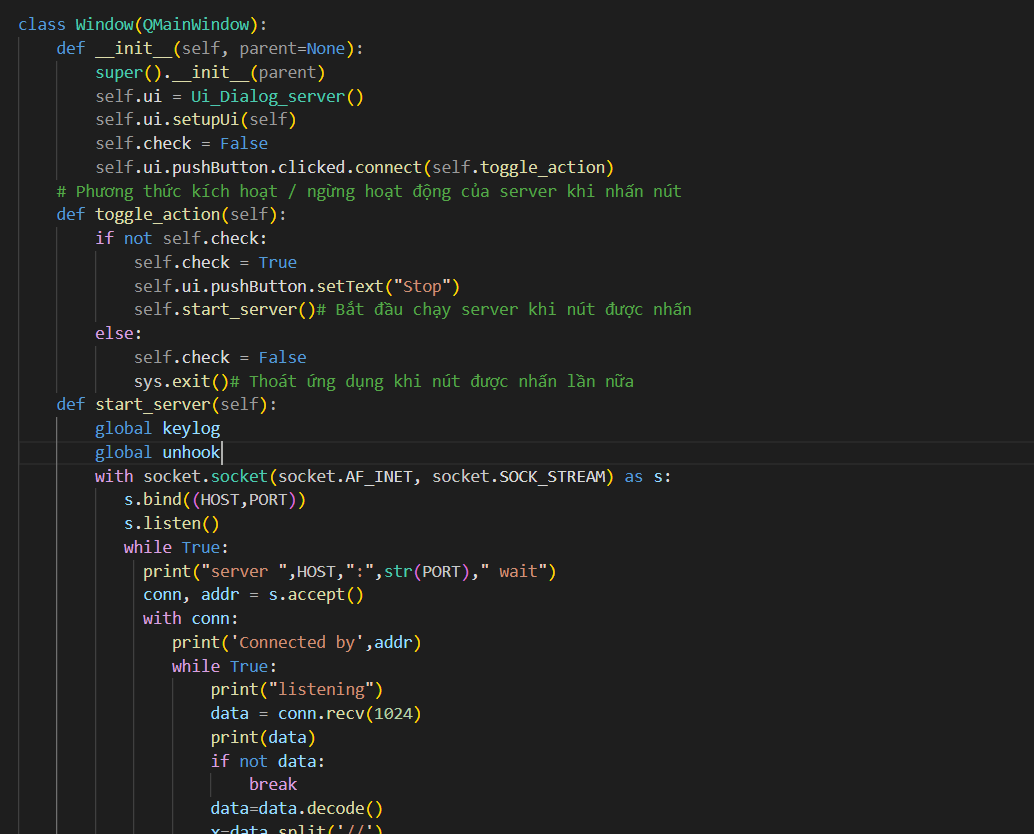
Khi chạy **connect\_to\_server(self)** để khởi tạo kết nối với Server thông qua Socket với Host (IP của server) và Port (Cổng 8080)



* Phía Server

Khi chạy **class Window(QMainWindow)** để thiết lập giao diện màn hình chính và kết nối của các nút với các hàm tương ứng

Khi chạy **start\_server(self)** chấp nhận kết nối từ Client

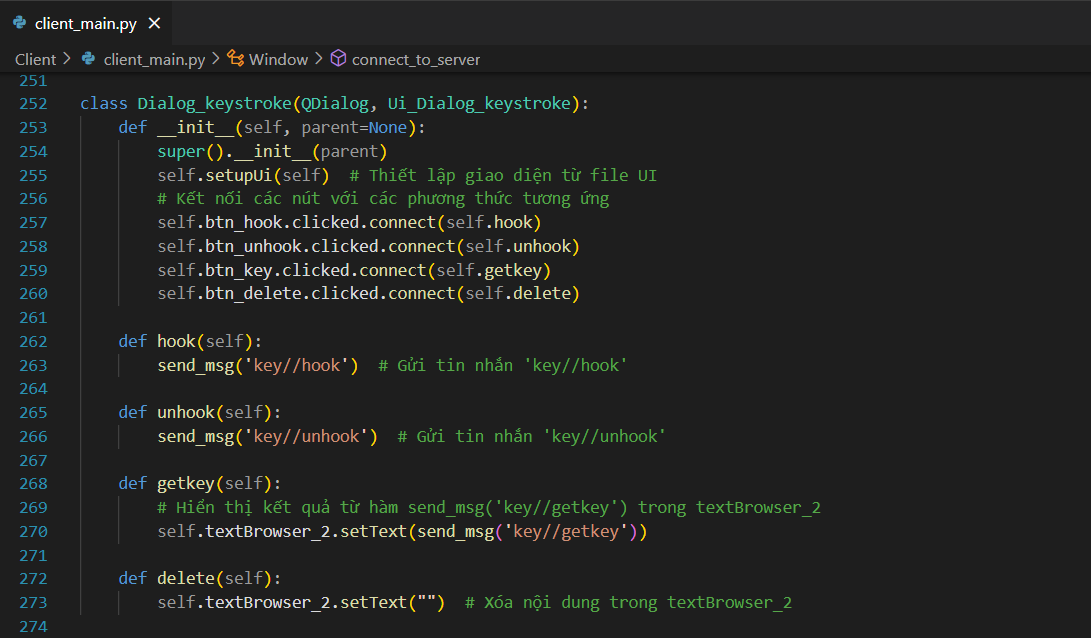


1. **Keystroke**: Bắt bàn phím từ phía máy chủ

* Phía Client

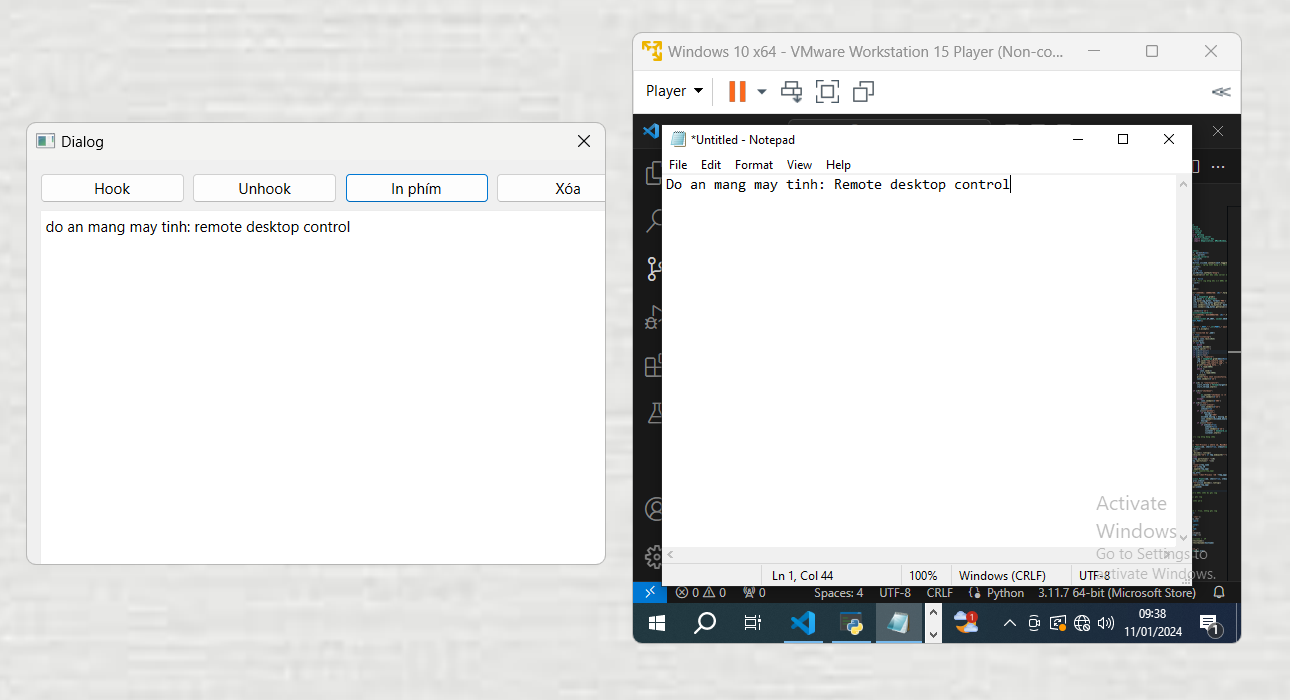
Khi chạy **Class Dialog\_keystoke** để thực hiện chức năng bắt bàn phím từ Server, hàm sẽ khởi tạo giao diện kết nối các nút bấm với các hàm tương ứng

Khi chạy các hàm **hook (bắt thao tác phím từ Server),** **unhook (ngừng bắt phím từ Server),** **getkey (Hiển thị kết quả bắt được),** **delete(xóa nội dung vừa bắt),** lúc này Client sẽ gửi message đến Server bằng hàm **send\_msg()** để tiến hành các sự kiện bắt phím

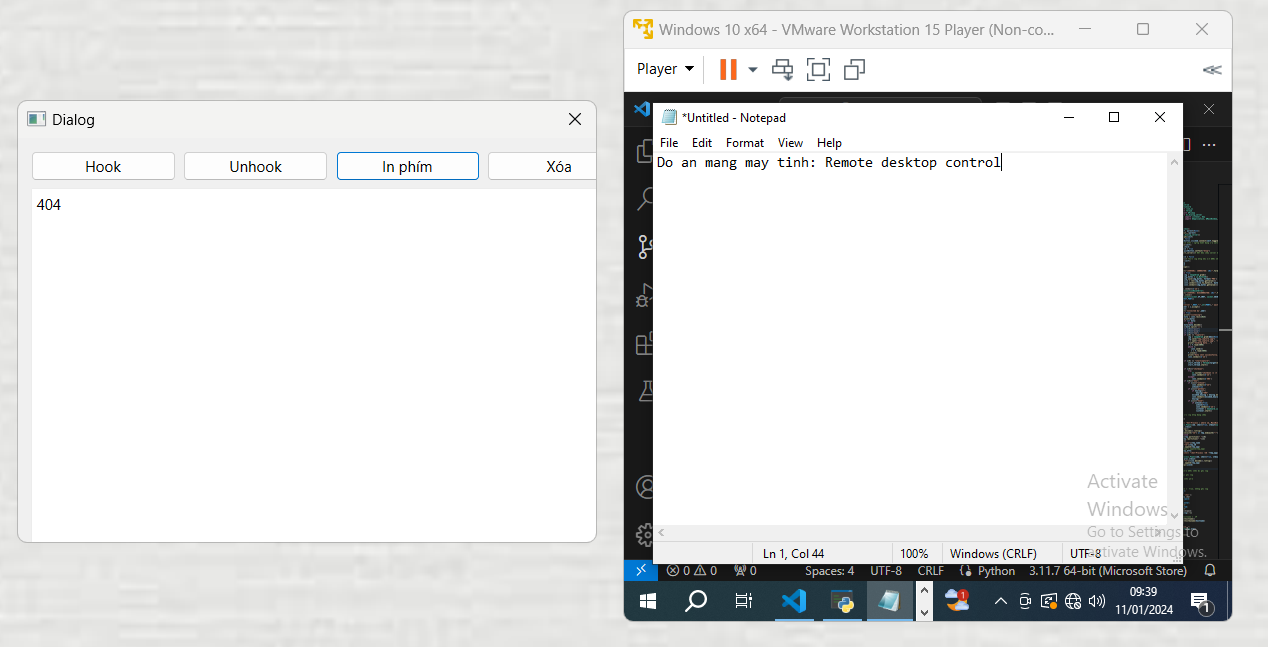


**Hook**: Khi nút ‘**hook’** được bấm Client sẽ gửi thông điệp đến Server về việc bắt phím, khi đó nó sẽ bắt những sự kiện phím từ Server và in ra màn hình khi ta bấm ‘**In phím’**

**Unhook**: Khi bấm ‘**unhook’** Client sẽ ngừng bắt những sự kiện phím từ Server, Nội dung ‘**404’** sẽ được in ra khi bấm nút ‘**In phím’** để thông báo về việc ngừng bắt phím



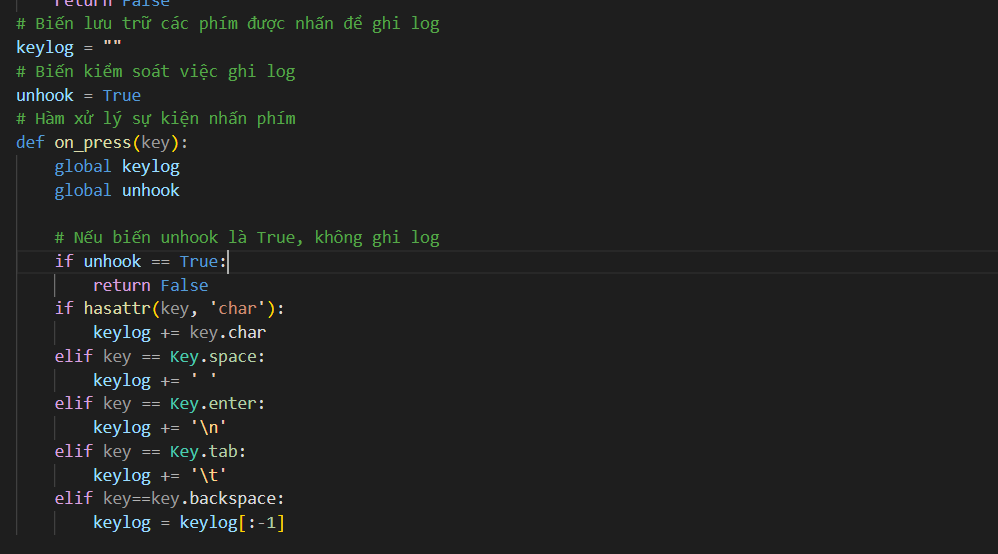
***Hình 1****: Demo bắt phím*



***Hình 2****: Demo ngừng bắt phím*

* Phía Server

Khi nhận được các massage yêu cầu từ Client, Server sẽ tiến hành bắt phím bằng hàm **on\_press(self),** biến **keylog** sử dụng để lưu chuỗi các ký tự mà người dùng đã nhập, biến **unhook** được sử dụng để kiểm doát việc ghi log. Sau khi hoàn tất, nội dụng của biến **keylog** sẽ được gửi về Cient và ỉn ra màn hình

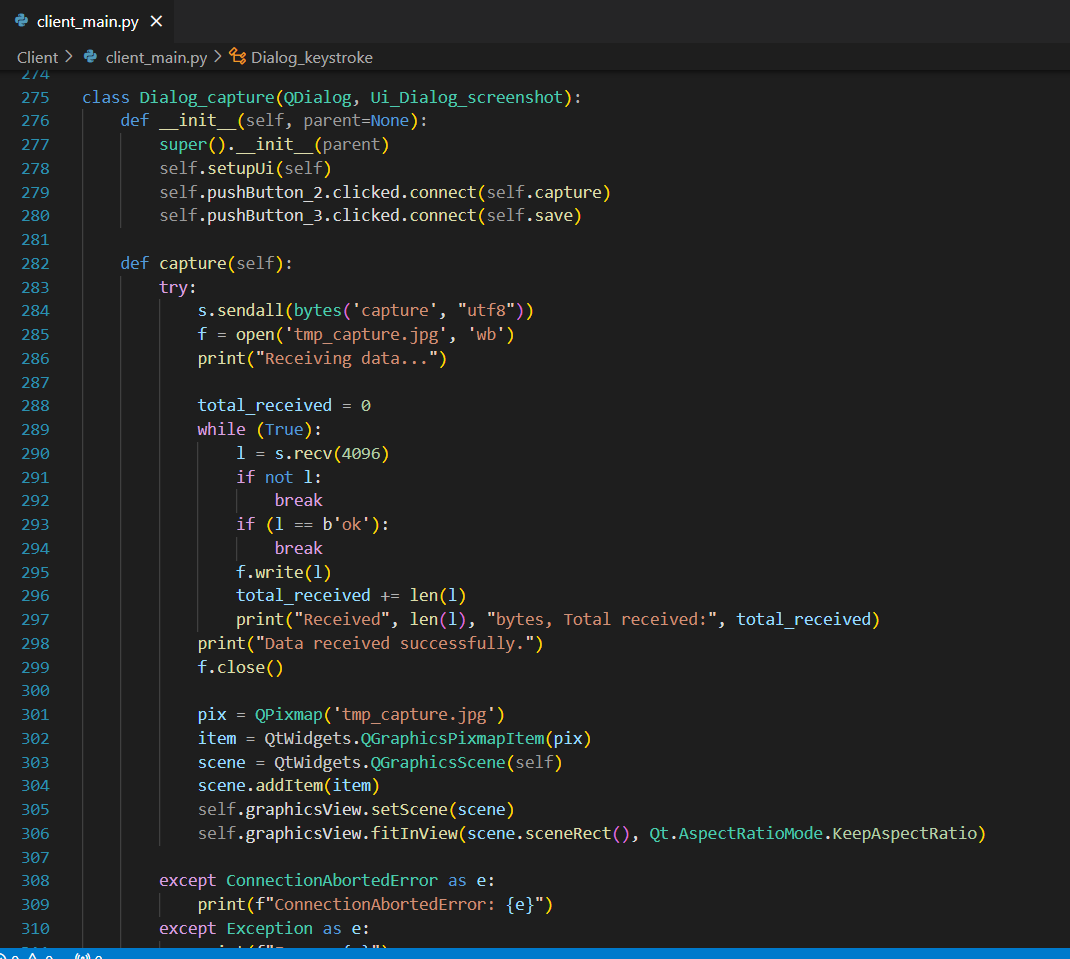
****

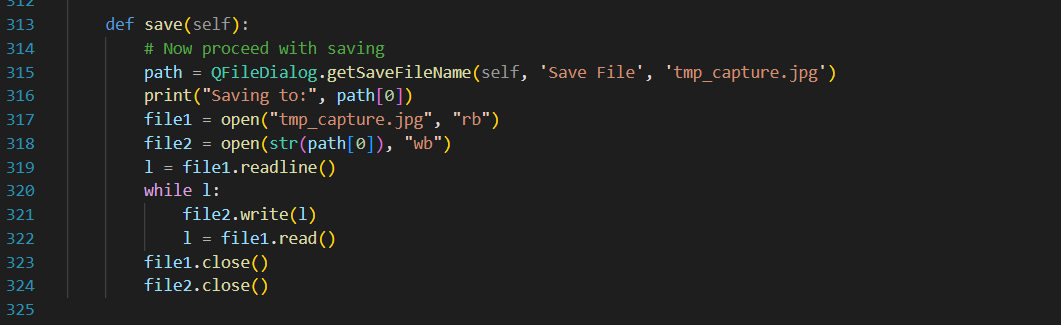
1. **Screenshot**: Chụp màn Server

* Phía Client

**Khi chạy Class Dialog\_capture** hàm sẽ khởi tạo giao diện kết nối các nút bấm với các hàm tương ứng, sử dụng kỹ thuật đọc ghi File nhị phân để hiển thị và lưu ảnh chụp màn hình.

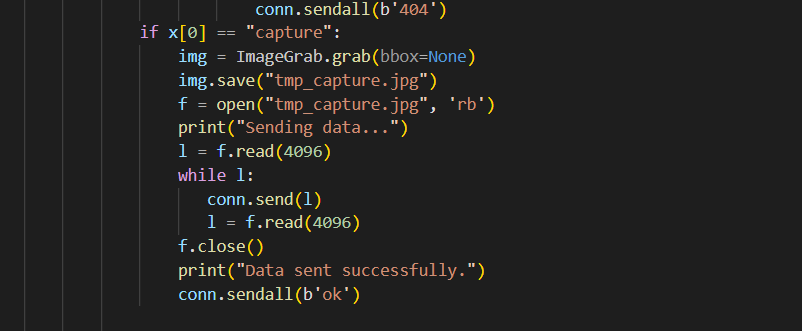
Hàm **def capture(self)** sẽ gửi yêu cầu “**capture**” thông qua socket ‘**s**’ để yêu cầu chụp màn hình Server, nhận dữ liệu hình ảnh từ Server và lưu vào tệp ‘**tmp\_capture.jpg’**, sử dụng thư viện PyQt để hiển thị ành trong một QgraphicsView trên giao diện người dùng. Hàm **def save(self)** có nhiệm vụ lưu ảnh vào thư mục

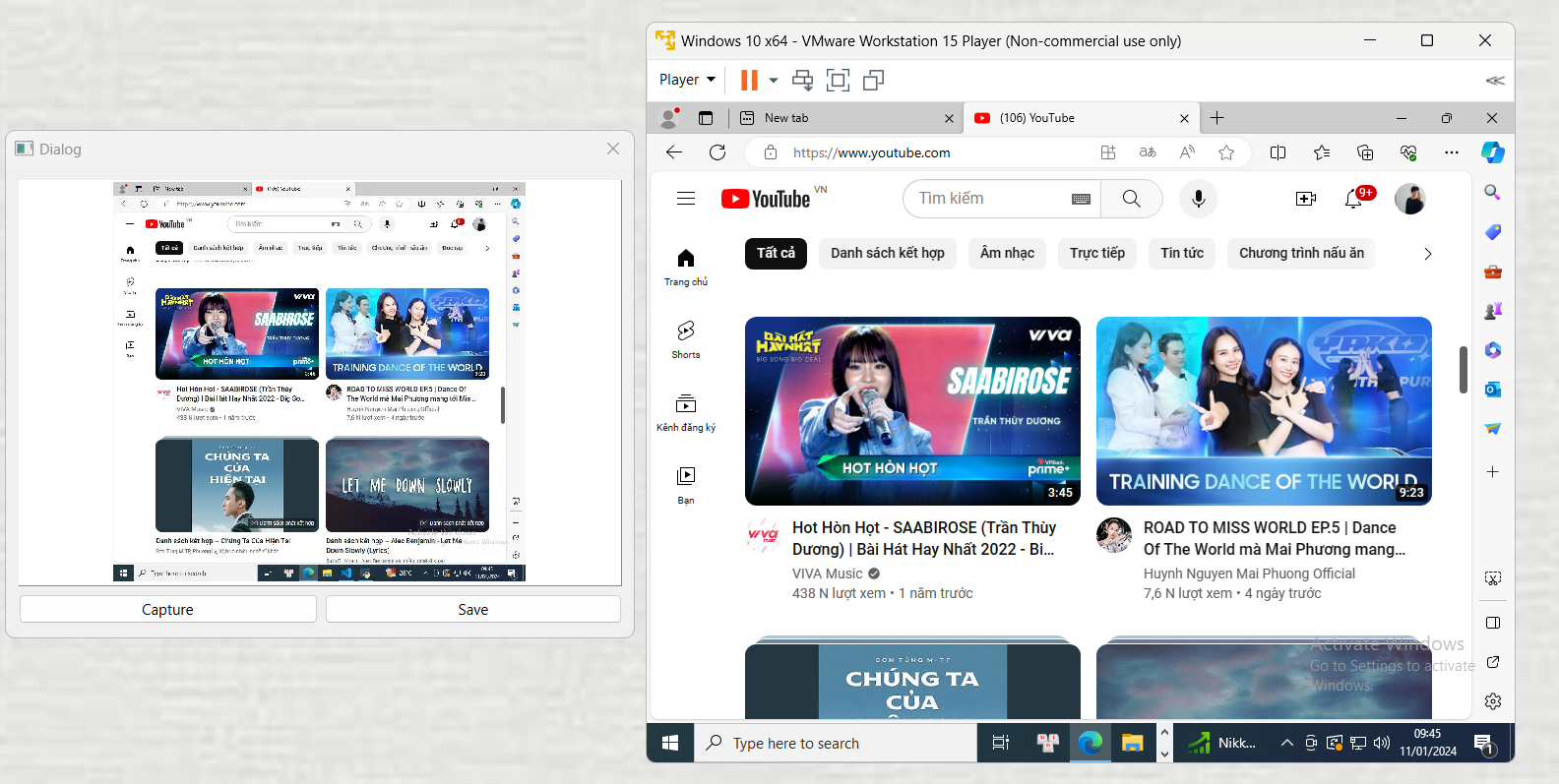




* Phía Server

Khi nhận được thông điệp ‘**capture’** từ Client, Server sẽ chụp màn hình bằng cách sử dụng modul ‘**ImageGrab’** để chụp màn hình, sau đó gửi về cho Client bằng cách đọc từng phần 4069 bytes





***Hình 3****: Demo chụp màn hình*

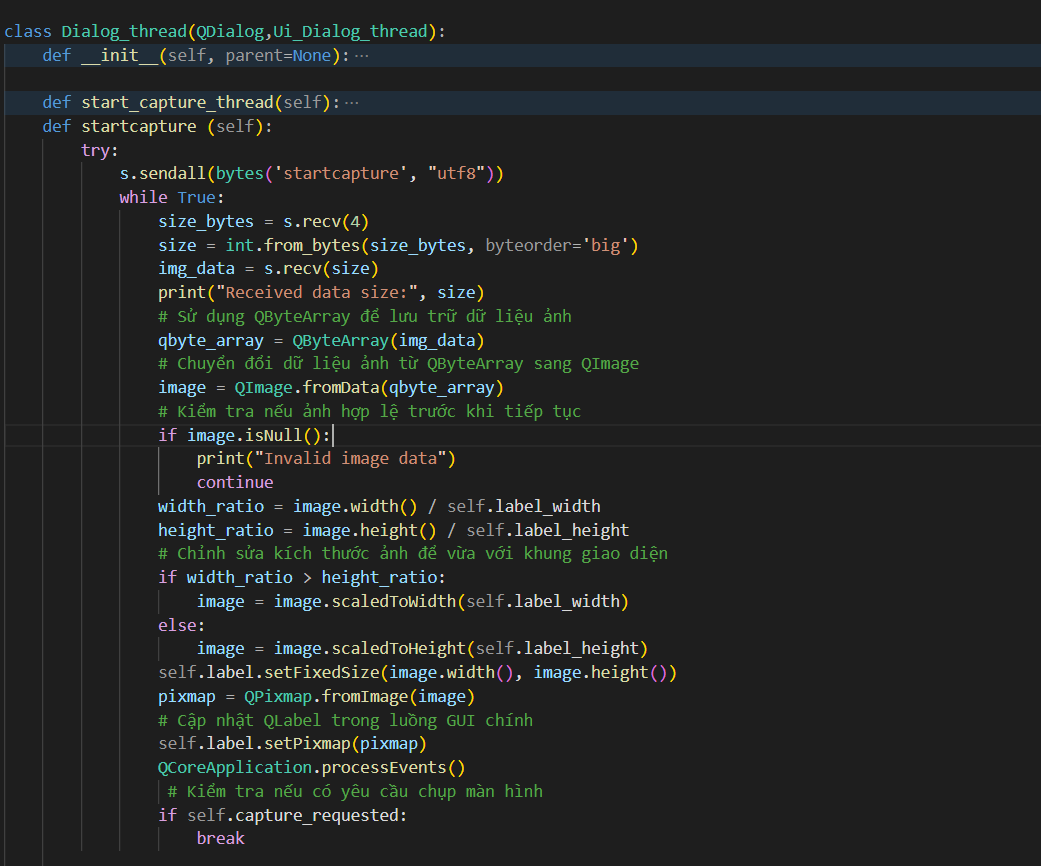
1. **Display**: Hiển thị màn hình Server

* Phía Client

Khi chạy **Class Dialog\_thread(QDialog, Ui\_Dialog\_thread)** hàm sẽ khởi tạo giao diện kết nối các nút bấm với các hàm tương ứng.

Khi chạy hàm **startcapture(self)** Client sẽ gửi thông điệp ‘**startcapture’** qua kết nối socket**(‘s’**) để yêu cầu chụp màn hình, chạy một vòng lặp vô hạn để nhận kích thước và dữ liệu ảnh từ Server, sử dụng ‘**QByteArray’** để lưu dữ liệu ảnh và chuyển đổi nó thành ‘**QImage’**, ảnh nhận về sẽ được điều chỉnh kích thước hợp lệ và hiển thị trên ‘**QLabel’.**

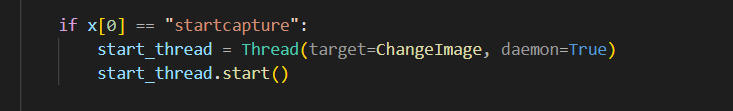
Kết thúc tiến trình bằng biến **capture\_requested** để thoát khỏi vòng lặp và thực hiện các chức năng tiếp theo

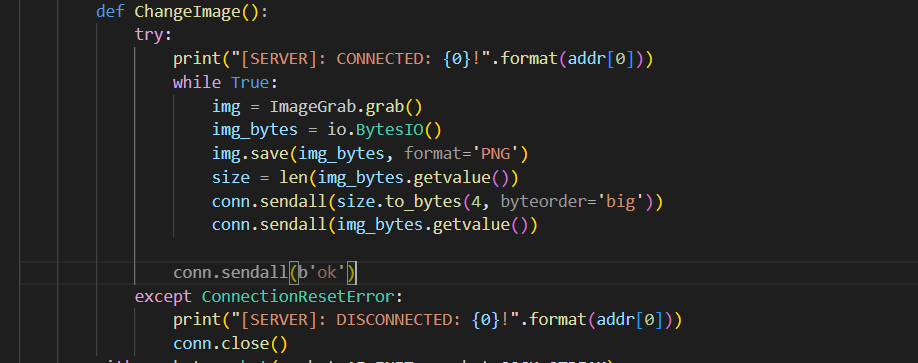


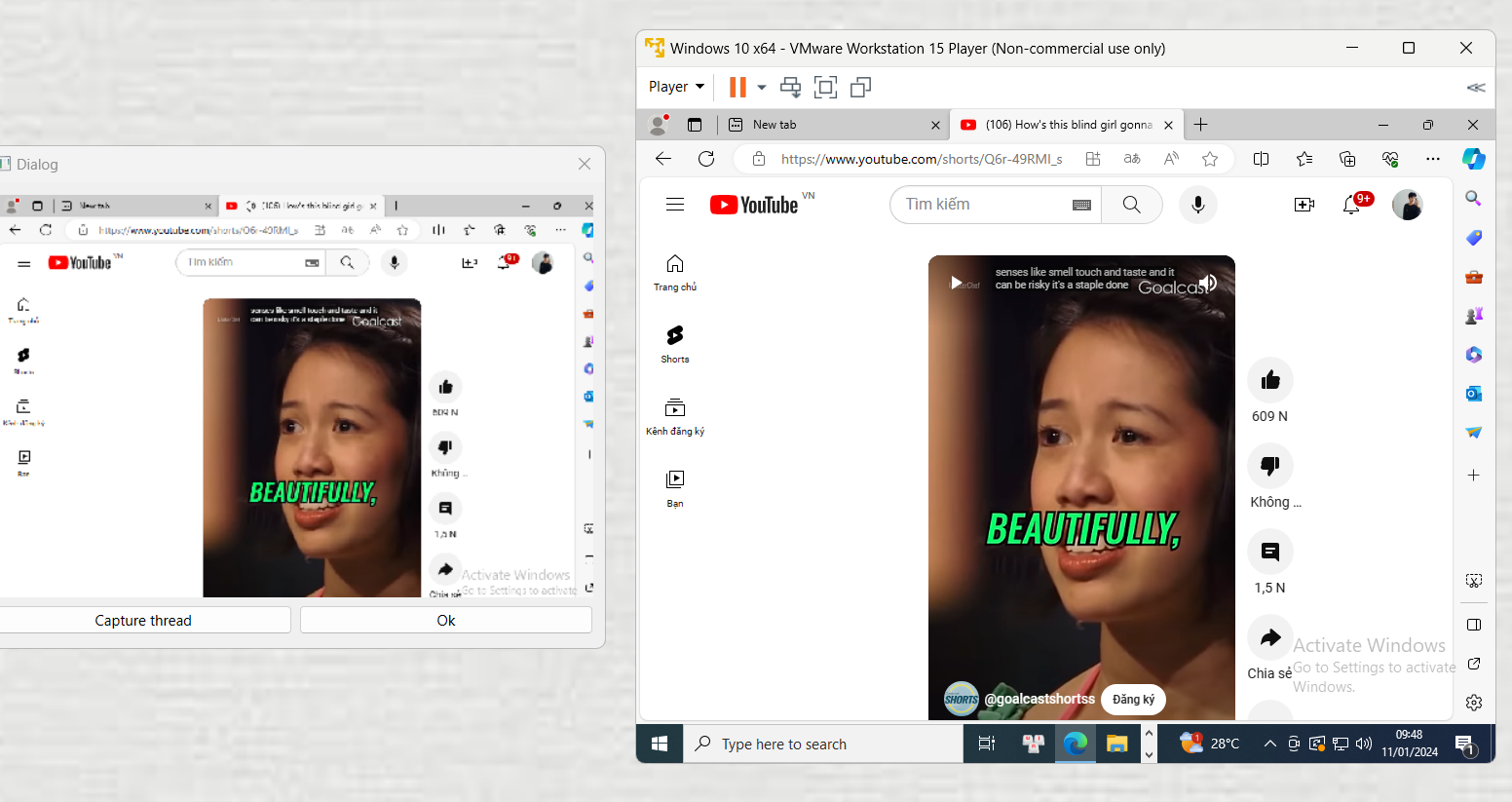
* Phía Server

Khi nhận được thông điệp ‘**startcapture’** từ Client, Server sẽ bắt đầu tiến trình truyền màn hình về cho Client bằng hàm **ChangeImage().**

Sử dụng ‘**ImageGrab.grab()’** để chụp màn hình, tiếp đó sử dụng **‘BytesIO’** để chuyển đổi ảnh thành bytes và lưu vào biến **‘img\_bytes’**, lấy kích thước của dữ liệu ảnh và gửi dữ liệu ảnh dưới dạng bytes về Client







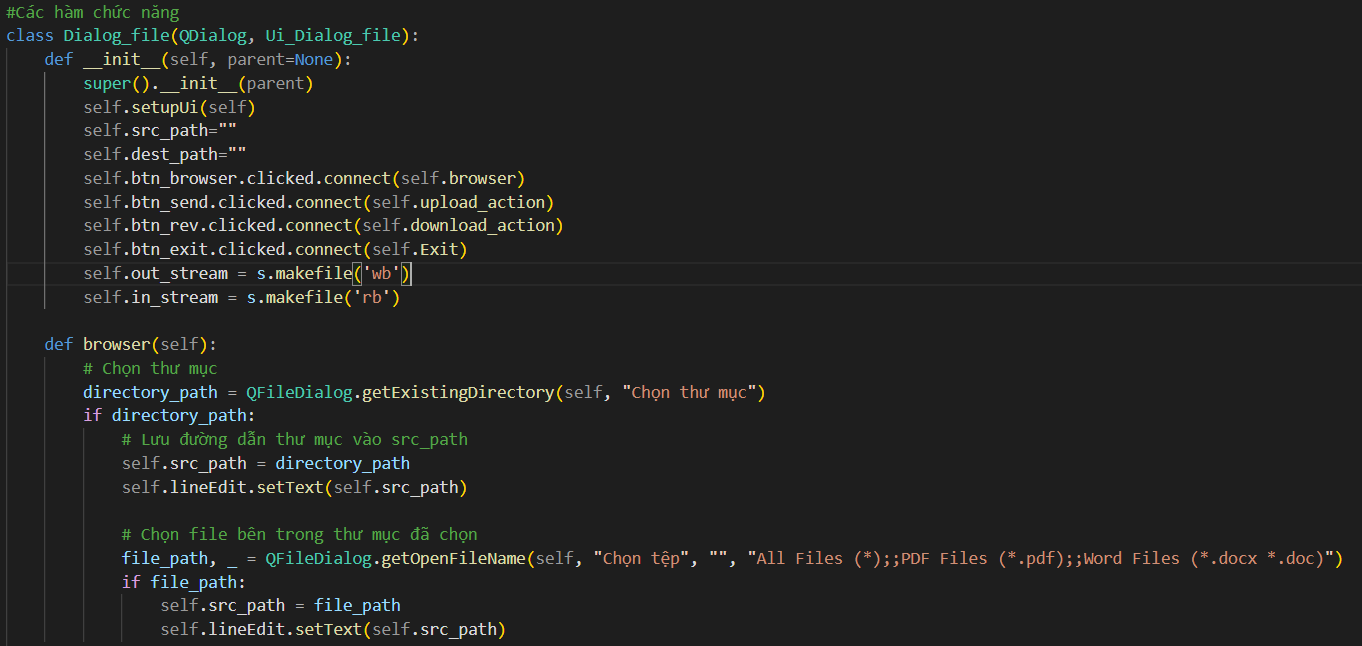
***Hình 4****: Demo hiển thị màn hình*

1. **File Tranfer**

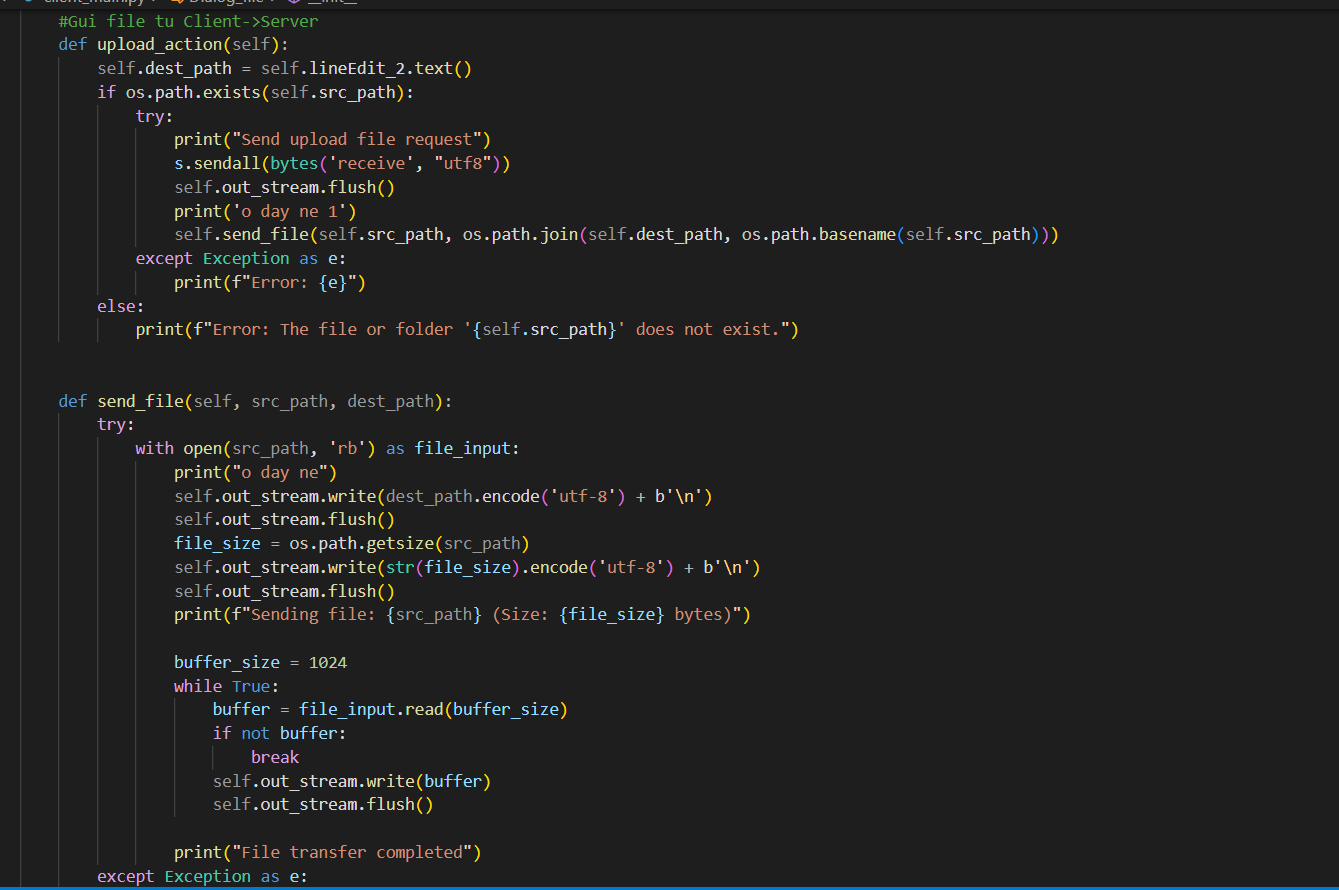
* **Phía Client**

Khi chạy **Class Dialog\_file(QDialog, Ui\_Dialog\_file) )** hàm sẽ khởi tạo giao diện kết nối các nút bấm với các hàm tương ứng.

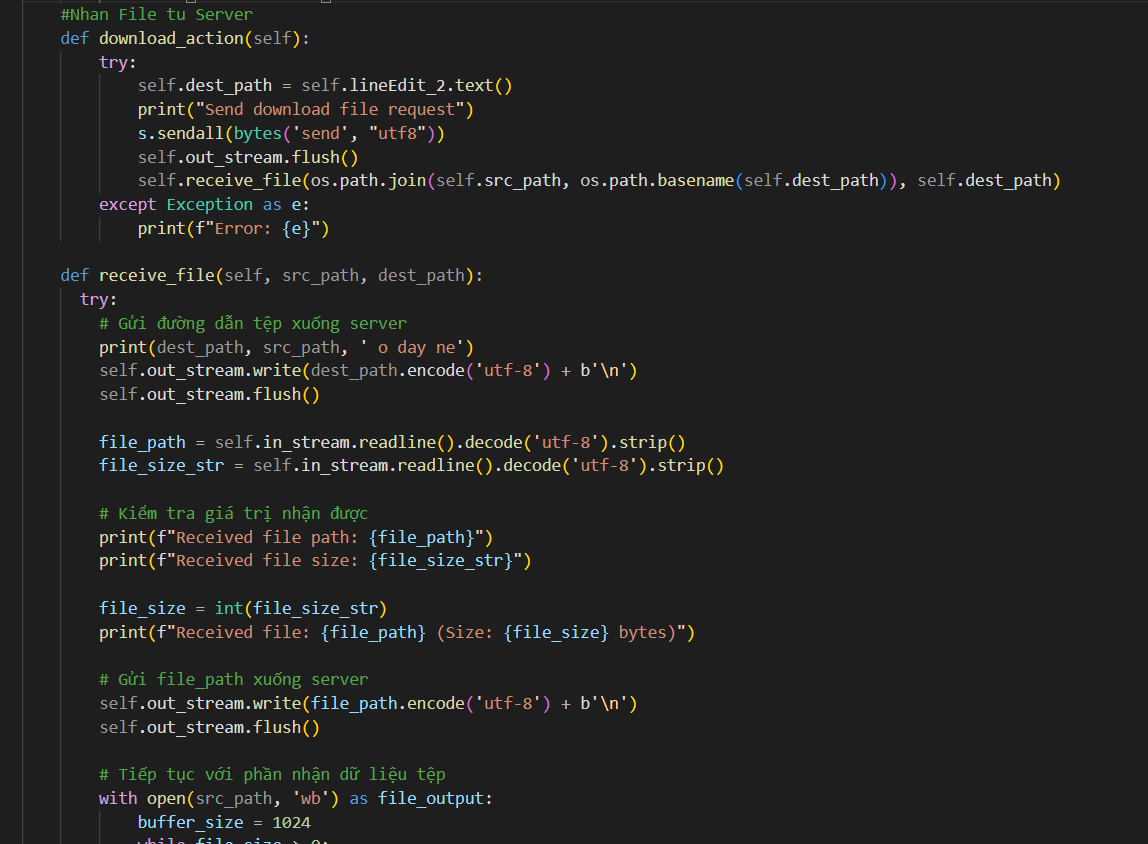
Chọn thư mục/file muốn gửi/nhận bằng hàm **def browser(self)**

****

Chạy hàm **upload\_action(self)** để gửi file từ Client->Server, Client gửi thông điệp ‘**receive’** tới Server thông qua socket**(‘s’),** sau đó gọi hàm **send\_file(self, src\_path, dest\_path)** sẽ gửi đường dẫn và kích thước file đến Server, dùng vòng lặp While đọc và chia nhỏ file thành các buffers và gửi qua kết nối

****

Chạy hàm **def** **download\_action(self)** để nhận file từ Server, Client gửi thông điệp ‘**send’** đến Server thông qua socket(**‘s’**), sau đó gọi hàm **receive\_file(self, src\_path, dest\_path)** gửi đường dẫn đích của file cần tải về từ Server, Client sẽ nhận đường dẫn tệp và kích thước file từ Server và ghi vào đường dẫn nguồn đã chọn

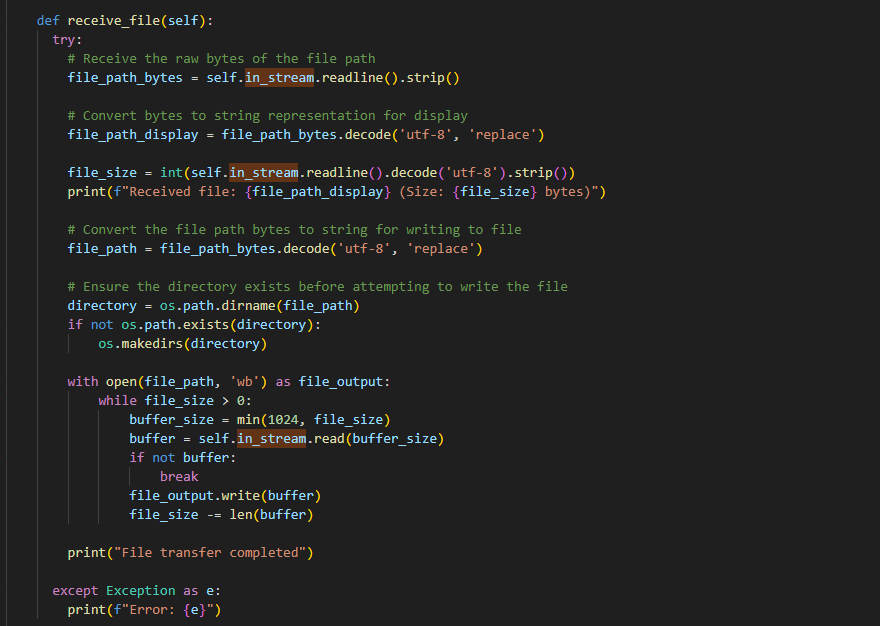
****

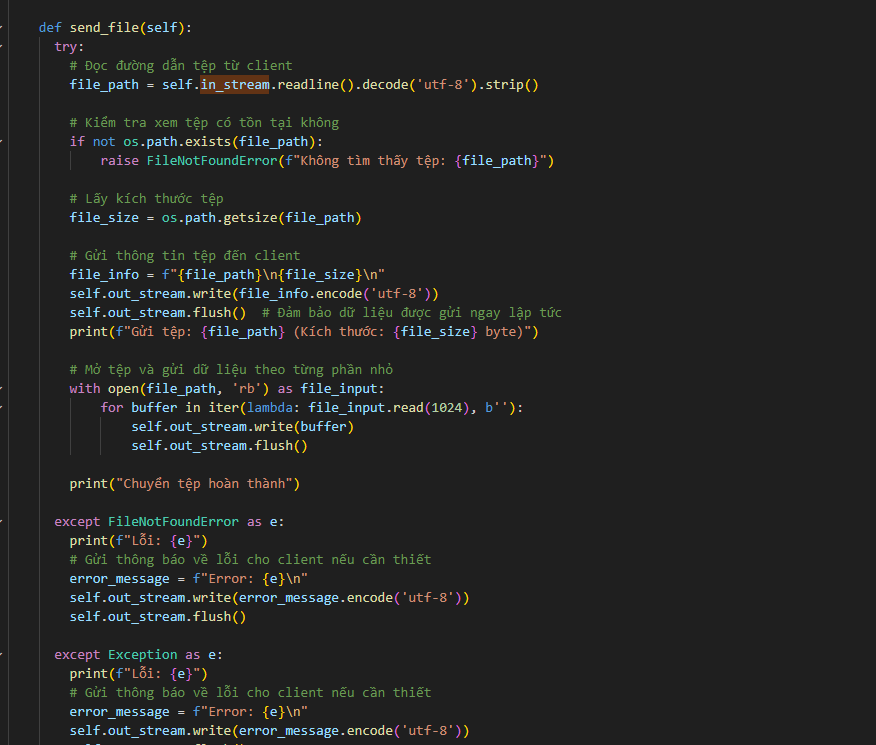
* **Phía Server**

Server nhận thông điệp ‘**send’** / ‘**receive’ và** sẽ chạy các hàm **receive\_file(self) / send\_file(self)** tương ứng.

Khi chạy **receive\_file(self)** Server sẽ nhận đường dẫn đích và dữ liệu file mà Client gửi sau đó chạy vòng while để đọc và ghi dữ liệu xuống

Khi chạy **send\_file(self)** Sever sẽ truy cập file cần gửi để gửi dữ liệu và kích thước file về cho Server theo từng phần nhỏ





1. **Shutdown & Exit**

Client sẽ gửi 1 thông điệp ‘shutdown’ đến Server và shutdown bằng lệnh:

os.system

Client sẽ gửi thông điệp ‘quit’ đến Server và thoát khỏi chương trình đang chạy

**PHẦN TỔNG KẾT**

1. **Video demo:**
2. **Link Youtube:** [**https://youtu.be/I7v1Ly-g9HQ**](https://youtu.be/I7v1Ly-g9HQ)
3. **Link dự phòng:** [**https://drive.google.com/file/d/1K2UmFlB84-v\_hoEwxkQ4KDKu46B5V1ZU/view?usp=sharing**](https://drive.google.com/file/d/1K2UmFlB84-v_hoEwxkQ4KDKu46B5V1ZU/view?usp=sharing)

**II. Tài liệu tham khảo**

1. Công cụ bổ trợ ChatGPT: <https://chat.openai.com/>
2. Video đồ án các năm trước:

<https://youtu.be/9Scgk_6MpVo?si=-WpQDiudBW643O7F>

1. Tài liệu truyền file:

<https://www.geeksforgeeks.org/file-transfer-using-tcp-socket-in-python/>

1. Video hướng dẫn làm giao diện: <https://youtu.be/Abj4N5ckjOc?si=ap3WY0kas3MX-pkX>
2. Video hướng dẫn truyền màn hình:

<https://youtu.be/zXjAsygbyfM?si=hbNLiYkENuB4JVVO>