



MẠNG MÁY TÍNH - CSC10008_21CLC02

PROJECT SOCKET

Đề 1: Web Server

Sinh viên

21127122 - Hồ Thanh Nhân

21127158 - Lê Hoàng Sang

21127740 - Đoàn Nam Thắng

Giảng viên

Lê Ngọc Sơn

Lê Hà Minh

Nguyễn Thanh Quân

Mục lục

1 - Thông tin nhóm	2
2 - Kịch bản giao tiếp của chương trình	2
2.1 Giao thức trao đổi giữa Client và Server	2
2.2 Cấu trúc thông điệp	2
2.3 Kiểu dữ liệu của thông điệp	6
2.4 Các hàm trong file server.py	6
3 - Môi trường lập trình, editor/IDE và các framework hỗ trợ để thực thi ứng dụng	7
4 - Hướng dẫn sử dụng các tính năng chương trình	7
5 - Phân công công việc và mức độ hoàn thành	11
6 - Các nguồn tài liệu tham khảo	11

1 - Thông tin nhóm

- Đề án Socket đề 1: Web Server
- Thành viên nhóm thực hiện đề án:
 - Hồ Thành Nhân - MSSV: 21127122.
 - Lê Hoàng Sang - MSSV: 21127158.
 - Đoàn Nam Thắng - MSSV: 21127740.

2 - Kịch bản giao tiếp của chương trình

2.1 Giao thức trao đổi giữa Client và Server

Client kết nối đến Server thông qua kết nối TCP (*SOCK_STREAM*) - IPv4 (*AF_INET*).

2.2 Cấu trúc thông điệp

- **Thông điệp HTTP:**

- Có hai kiểu HTTP Message: HTTP Request Message (Request) được gửi đi bởi client tới server - khi server nhận được nó biết phải thực hiện nhiệm vụ nào đó, HTTP Response Message (Response) là trả lời từ server về cho client.
- HTTP Message trong phiên bản HTTP/1.1 có các thành phần dữ liệu trình bày trong định dạng text (plain text) mà người có thể đọc hiểu. Trong bản HTTP/2 thì các thành phần đưa vào định dạng nhị phân (binary) là các frame làm cho người không đọc được trực tiếp nữa.

- **HTTP Request Message:**

Dòng đầu tiên: Dòng này chứa thông tin để gửi tới server, dựa vào thông tin này mà server thực thi hành động phù hợp. Dòng này có chứa ba thông tin cách nhau bởi khoảng trắng, ví dụ như: POST /index.html HTTP/1.1

+ **HTTP Method (Phương thức HTTP):** là thành phần thứ nhất (ví dụ trên là **POST**), nó có giá trị như **POST**, **GET**, **PUT**, **DELETE**. Nó cho biết yêu cầu cần được thực hiện trên server đối với một tài nguyên nào đó.

+ **Địa chỉ URL:** là thành phần thứ hai (ví dụ trên là `/html/`). Địa chỉ tài nguyên truy vấn, có thể là URL tương đối - tuyệt đối (kể cả cổng, nếu có cổng thì viết cổng sau ký hiệu `:`).

+ **HTTP Version:** thành phần thứ ba cho biết phiên bản HTTP (thường là HTTP/1.1).

Header của Request: Các header có cấu trúc đó là một chuỗi là tên header tiếp theo là dấu `:` và giá trị cho header. Mỗi header được viết trên một dòng. Có rất nhiều loại header (đã chuẩn hóa hoặc header do bạn tự đặt), ví dụ vài header như:

+ **Host:** là header chỉ ra host (domain, IP) và cổng của server mà Request gửi đến. Nếu không chỉ rõ port thì mặc định là 80 với http và 443 với https.

+ **Accept:** trong Request cho biết kiểu nội dung trả về mà client có thể hiểu. (các kiểu cách nhau bởi , có độ ưu tiên mặc định 1, nếu muốn xác định độ ưu tiên cho kiểu nào thì cho thêm ;q=value). Ví dụ: Accept: text/html, application/xhtml+xml, application/xml;q=0.9, image/webp, */*;q=0.8

+ **User-Agent:** là header của Request cho phép server xác định được ứng dụng nào, hệ điều hành gì gửi Request.

+ **Content-Length:** cho biết độ dài (byte) của phần body (nếu có đính kèm trong request).

+ **Content-Type:** trong Request, header này cho server biết kiểu dữ liệu được đính kèm trong phần body.

Body của Request: Chứa dữ liệu của Request (dữ liệu này có kiểu xác định ở header Content-Type và độ dài Content-Length), không phải mọi Request đều có body, dữ liệu body thường là HTML Form hay JSON.

- HTTP Response Message:

Dòng đầu tiên: còn gọi là dòng trạng thái, nó chứa ba mẩu tin gồm:

+ Phiên bản HTTP, ví dụ HTTP/1.1

+ Mã trả về như 200, 404, 301, 302...

+ Đoạn text ngắn giải thích mã ví dụ OK, Not Found, Moved Permanently...

Ví dụ: HTTP/1.1 200 OK

Header của Response: tương tự như header của Request, một số Header của Response:

+ **Date:** chứa thông tin ngày tháng thông tin (response) được phát sinh.

+ **Set-Cookie:** header này chứa cookie của server gửi cho client. Client lưu lại để gửi lại cho server để phục hồi phiên làm việc.

Một số giá trị thiết lập như:

Max-Age=number thời gian hết hạn cookie

HttpOnly không cho JS truy cập Cookie

+ Có những header giống header của Request như: Content-Type, Content-Length...

Body của Response: Chứa dữ liệu của Response (dữ liệu này có kiểu xác định ở header Content-Type và độ dài Content-Length), không phải mọi Response đều có body.

- Phần **Header** được kết thúc bằng 1 dòng trống (chuỗi "\r\n\r\n").

• Cấu trúc gói tin TCP:

- **Header** của một TCP segment có thể nằm trong khoảng từ 20 bytes đến 60 bytes. Trong đó 40 bytes là phần Options, phần Options không bắt buộc. Nếu không có Options thì Header của TCP segment là 20 bytes, nếu có Options thì có thể lên đến cao nhất là 60 bytes.

Source Port (16 bits)				Destination Port (16 bits)							
Sequence Number (32 bits)											
Acknowledgement Number (32 bits)											
Header Length (4 bits)	Reserved (6 bits)	U R G	A C K	P S H	R T	S N	F I N	Window Size (16 bits)			
Checksum (16 bits)				Urgent Pointer (16 bits)							
Options (optional)											
Data											

TCP Segment Structure

- **Source Port:** Số hiệu của cổng tại máy tính gửi.
- **Destination port:** Số hiệu của cổng tại máy tính nhận.
- **Sequence number:** Trường này có 2 nhiệm vụ. Nếu cờ SYN bật thì nó là số thứ tự gói ban đầu và byte đầu tiên được gửi có số thứ tự này cộng thêm 1. Nếu không có cờ SYN thì đây là số thứ tự của byte đầu tiên.
- **Acknowledgement number:** Nếu cờ ACK bật thì giá trị của trường chính là số thứ tự gói tin tiếp theo mà bên nhận cần.
- **Header Length:** Trường có độ dài 4 bits quy định độ dài của phần header (tính theo đơn vị từ 32 bits). Phần header có độ dài tối thiểu là 5 từ (160 bits) và tối đa là 15 từ (480 bits).
- **Reserved:** Dành cho tương lai và có giá trị là 0.
- **Flags (hay Control bits):** Bao gồm 6 cờ:
 - + **URG:** Cờ cho trường Urgent pointer
 - + **ACK:** Cờ cho trường Acknowledgement
 - + **PSH:** Hàm Push
 - + **RST:** Thiết lập lại đường truyền
 - + **SYN:** Đồng bộ lại số thứ tự
 - + **FIN:** Không gửi thêm số liệu
- **Window Size:** Số byte có thể nhận bắt đầu từ giá trị của trường báo nhận (ACK)
- **Checksum:** 16 bits kiểm tra cho cả phần header và dữ liệu
- **Urgent Pointer:** Nếu cờ URG bật thì giá trị trường này chính là số từ 16 bit mà số thứ tự gói tin (sequence number) cần dịch trái.
- **Options:** Đây là trường tùy chọn. Nếu có thì độ dài là bội số của 32 bits.
- **Data:** Trường cuối cùng không thuộc về header. Giá trị của trường này là thông tin dành cho các tầng trên (trong mô hình 7 lớp OSI). Thông tin về giao thức của tầng trên không được chỉ rõ trong phần header mà phụ thuộc vào cổng được chọn.

- Cấu trúc địa chỉ IPv4:

Version (4 bits)	Header Length (4 bits)	Type of Service (8 bits)	Total Length (16 bits)
		Identification (16 bits)	Flags (3 bits) Fragment Offset (13 bits)
Time to Live (8 bits)		Protocol (8 bits)	Header Checksum (16 bits)
Source IP Address (32 bits)			
Destination IP Address (32 bits)			
Options (optional) (0 to 40 bytes)			
Data			

- **Version:** Chỉ định phiên bản của IP, có giá trị 4.
- **Header Length:** Chỉ định chiều dài IPv4 header (đơn vị đo là khối 4 byte).
- **Type of service:** Chỉ định dịch vụ mong muốn khi truyền các gói tin qua router. Trường này có 8 bit, xác định quyền ưu tiên, độ trễ, thông lượng, các đặc tính chỉ định độ tin cậy khác. Trường Service Type gồm TOS (Type of Service) và Precedence. TOS xác định loại dịch vụ, bao gồm: giá trị, độ tin cậy, thông lượng, độ trễ hoặc bảo mật. Precedence xác định mức ưu tiên, sử dụng 8 mức từ 0-7.
- **Total Length:** Chỉ định tổng chiều dài gói tin ipv4 (IPv4 header + IPv4 payload). Kích thước 16 bit, chỉ định rằng gói tin IPv4 có thể dài tối 65,535 byte.
- **Identification:** Định danh gói tin. Kích thước 16 bit. Định danh cho gói tin được lựa chọn bởi nguồn gửi gói tin. Nếu gói tin IPv4 bị phân mảnh, mọi phân mảnh sẽ giữ lại giá trị trường định danh này, mục đích để node đích có thể nhóm lại các mảnh, phục vụ cho việc phục hồi lại gói tin.
- **Flags:** Xác định cờ cho quá trình phân mảnh. Kích thước 3 bit. Có hai cờ: một xác định gói tin bị phân mảnh và cờ kia chỉ định xem có thêm phân mảnh khác nữa tiếp theo phân mảnh hiện thời hay không.
- **Fragment Offset:** Chỉ định vị trí của phân mảnh trong phần payload của gói tin ban đầu. Trường này có kích thước 13 bit.
- **Time to Live:** Chỉ định số lượng link tối đa mà một gói tin IPv4 có thể đi qua trước khi bị hủy bỏ. Trường này dài 8 bit. TTL được sử dụng như một bộ đếm thời gian mà router IPv4 dùng để quyết định độ dài thời gian cần thiết (bằng giây) để chuyển tiếp gói tin IPv4. Router hiện đại chuyển tiếp gói tin chưa đến một giây song luôn phải giảm giá trị trường này ít nhất 1 đơn vị. Khi giá trị TTL trở về 0, gói tin sẽ được hủy đi và thông điệp lỗi được gửi trả lại địa chỉ IPv4 nguồn.
- **Protocol:** Xác định thủ tục lớp cao hơn gói tin sẽ được chuyển tiếp. Trường này gồm 8 bit. Ví dụ một số giá trị: 6 là TCP, 17 là UDP, 1 là ICMP.
- **Header Checksum:** Cung cấp kiểm tra checksum cho IPv4 header. Có kích thước 16 bit. IPv4 payload không bao gồm trong checksum này mà thường chứa Checksum riêng của nó. Các IPv4 node nhận gói tin sẽ kiểm tra IPv4 header Checksum và loại bỏ gói tin nếu không trùng khớp thông tin. Khi router forward một gói tin IPv4, nó phải giảm giá trị trường TTL, do vậy trường Header Checksum được tính toán lại tại mỗi router giữa nguồn và đích.
- **Source IP Address:** Chứa địa chỉ nguồn gửi gói tin IPv4. Kích thước 32 bit.

- **Destination IP Address:** Chứa địa chỉ IPv4 đích. Kích thước 32 bit.
- **Option:** Chứa một hoặc nhiều hơn tùy chọn trong IPv4. Kích thước trường này là một số nguyên lần của 32 bit (4 byte). Nếu các option không dùng hết và làm lẻ khối 32 bit, các giá trị 0 sẽ được thêm vào để đảm bảo IPv4 header là một số nguyên của khối 4 byte, như vậy chiều dài IPv4 Header mới có thể chỉ định được bằng giá trị của trường Internet Header Length.

2.3 Kiểu dữ liệu của thông điệp

- **GET:** Theo sau là "/" hoặc tên filename (ví dụ index.html, css/style.css, ...), cuối cùng là HTTP/1.1, server parse thông tin của request để biết được cần phải load file nào cho request đó. Nếu là "/" thì coi như "index.html". Sau khi parse request để xác định filename, server sẽ gửi reponse với header tương ứng và theo nội dung của file:

- + Nếu filename tồn tại, thì trả về thành công "200 OK" kèm theo nội dung của file.
- + Nếu file không tồn tại, trả về "404 File Not Found".

- **POST:** Server sẽ nhận được thông tin "uname" và "psw" kèm theo trong body của request. Server cần parse thông tin này ra và kiểm tra nếu "uname" là "admin" và "psw" là "123456" thì trả về nội dung của trang "images.html". Nếu không thì trả về "401 Unauthorized". Post có tính bảo mật cao hơn Get.

- **PUT:** ghi đè (thay thế) tài nguyên nào đó bằng dữ liệu trong Request.

- **DELETE:** xóa tài nguyên.

2.4 Các hàm trong file server.py

- **start():** Server bắt đầu lắng nghe, đồng ý các yêu cầu kết nối và tạo ra các luồng cho mỗi kết nối.

- **handle(connect, address):** Server nhận dữ liệu, ở mỗi luồng hàm sẽ thực hiện các yêu cầu GET hoặc POST và báo lỗi 401 Unauthorized nếu có.

- **check_post_body(message):** Kiểm tra tên đăng nhập và mật khẩu có đúng là "admin" và "123456" hay không.

- **get_from_header(message, key=None):** Lấy các dữ liệu (tham số key) từ nội dung header.

- **set_header(response, message_type, message_length):** Tạo ra header, gán các giá trị vào header.

- **read_file(response, Name_file, message_type):** Đọc file dưới dạng nhị phân, trả về 1 message hoàn chỉnh.

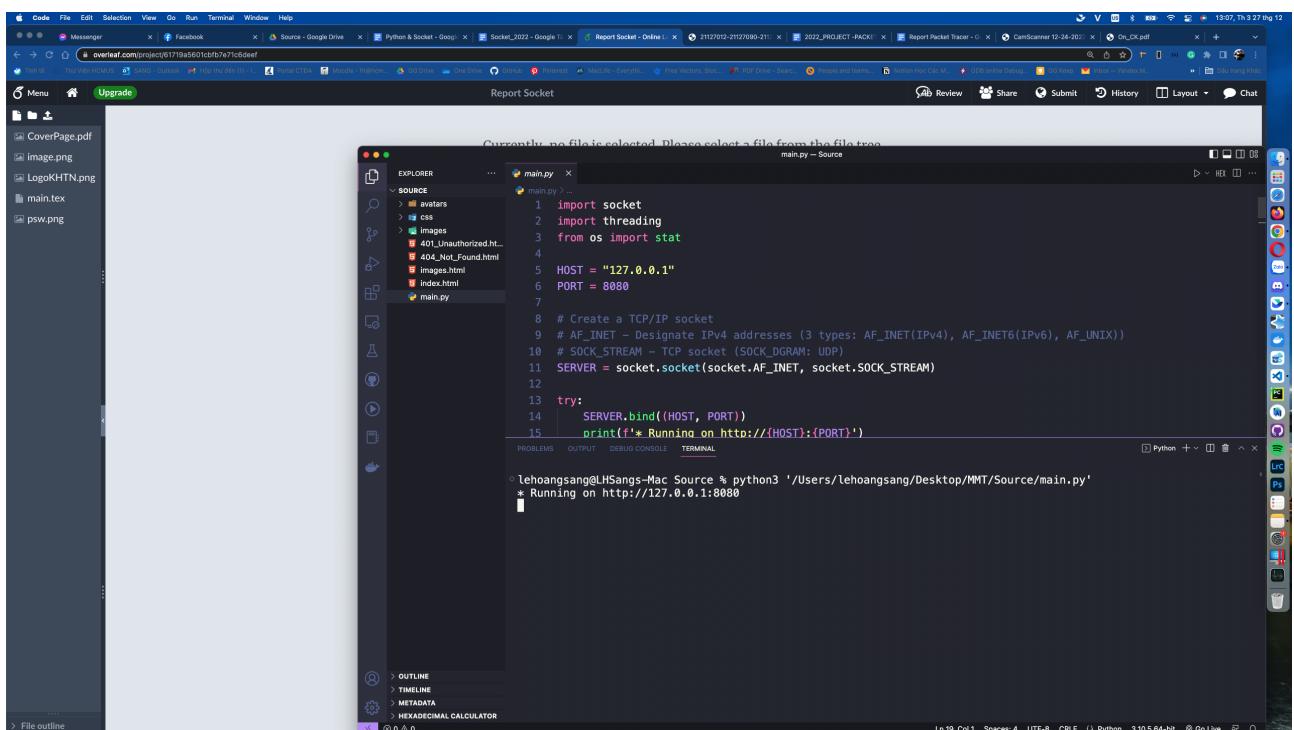
- **content_type_parser(url_ext):** Trả về content-type tương ứng với extension của file.

3 - Môi trường lập trình, editor/IDE và các framework hỗ trợ để thực thi ứng dụng

- Ngôn ngữ lập trình: Python.
- Hệ điều hành: Windows và MacOS.
- Các thư viện sử dụng: Socket, Threading, Os.
- Trình duyệt: Google Chrome.

4 - Hướng dẫn sử dụng các tính năng chương trình

- Compile và run file server.py trên VSCode hoặc Pycharm.
- Click vào địa chỉ http in ra trên terminal, trình duyệt sẽ được mở ra và kết nối với server đang chạy trên máy tính.



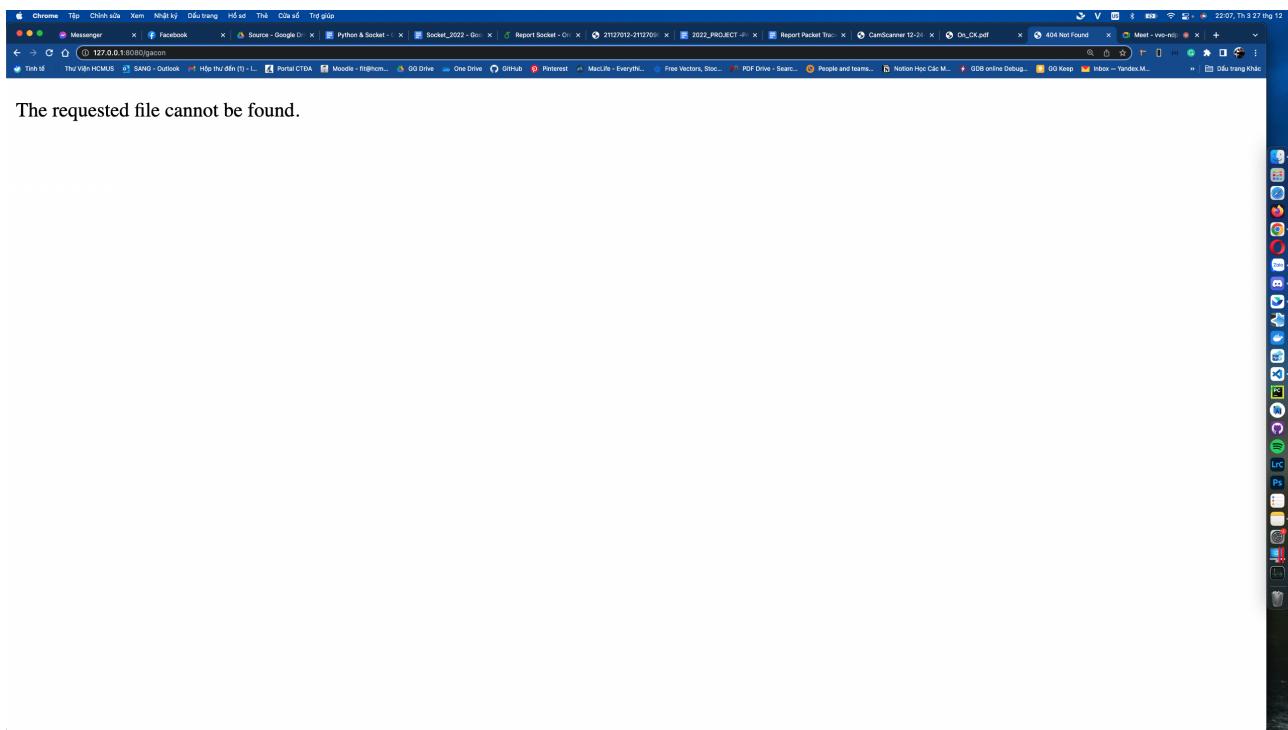
```

import socket
import threading
from os import stat
HOST = "127.0.0.1"
PORT = 8080
# Create a TCP/IP socket
# AF_INET - Designate IPv4 addresses (3 types: AF_INET(IPv4), AF_INET6(IPv6), AF_UNIX)
# SOCK_STREAM - TCP socket (SOCK_DGRAM: UDP)
SERVER = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
try:
    SERVER.bind((HOST, PORT))
    print("* Running on http://{}:{}/".format(HOST, PORT))

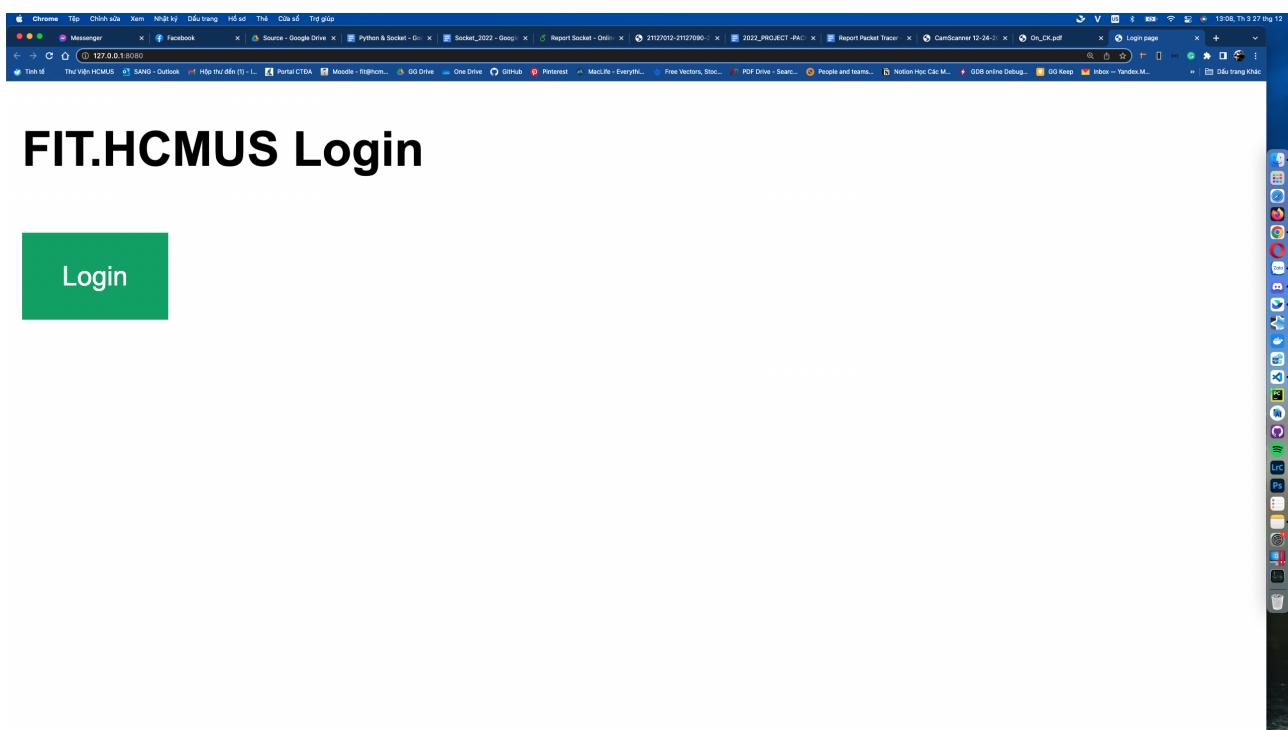
```

lehoangsang@LHSangs-Mac: Source % python3 '/Users/lehoangsang/Desktop/MMT/Source/main.py'
* Running on http://127.0.0.1:8080

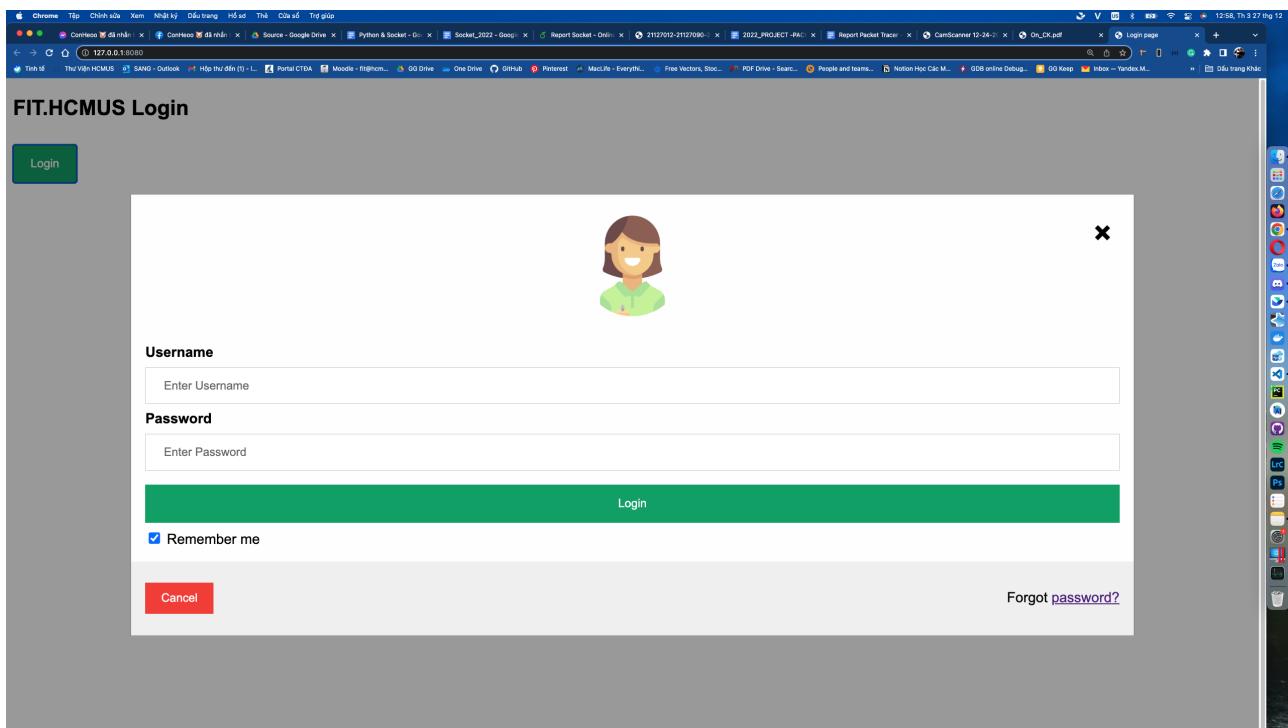
- Trình duyệt tải lên trang web login. Nếu client yêu cầu tải 1 file không hợp lệ, server sẽ trả về trang html 404 Not Found.



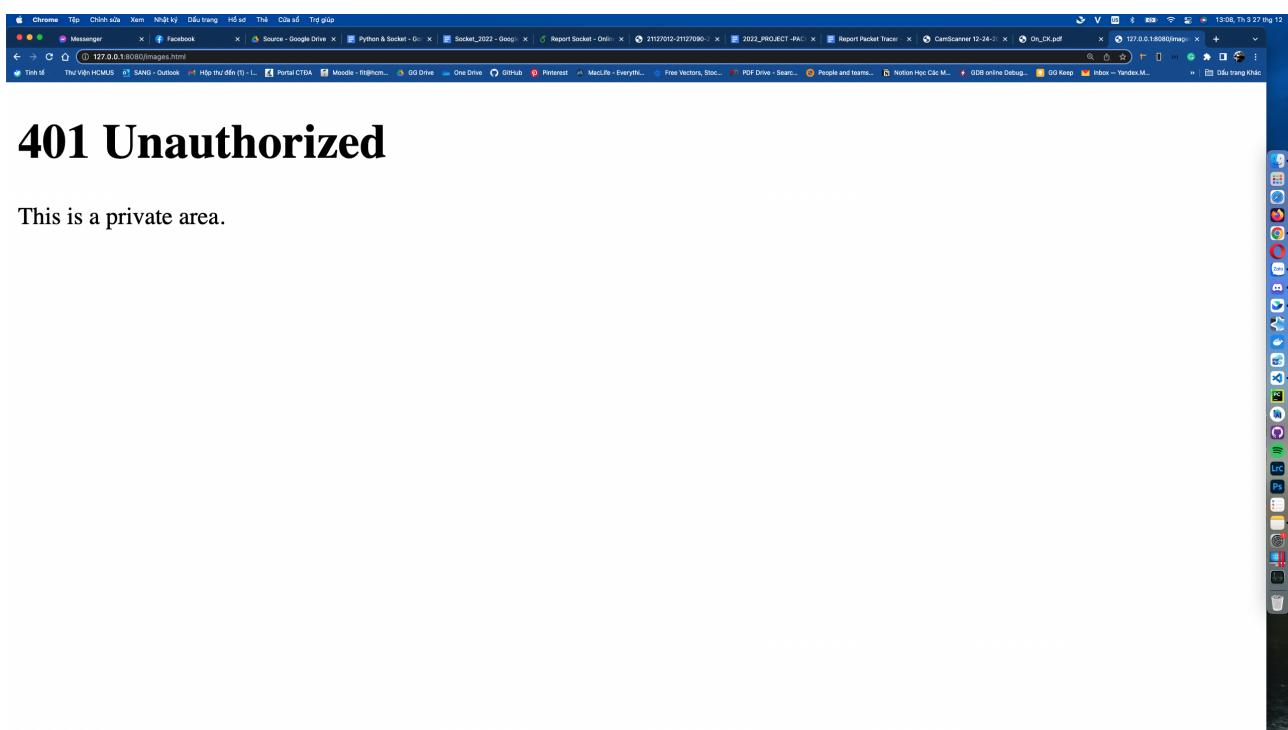
- Nhấn nút login, bảng nhập mật khẩu sẽ hiện lên.



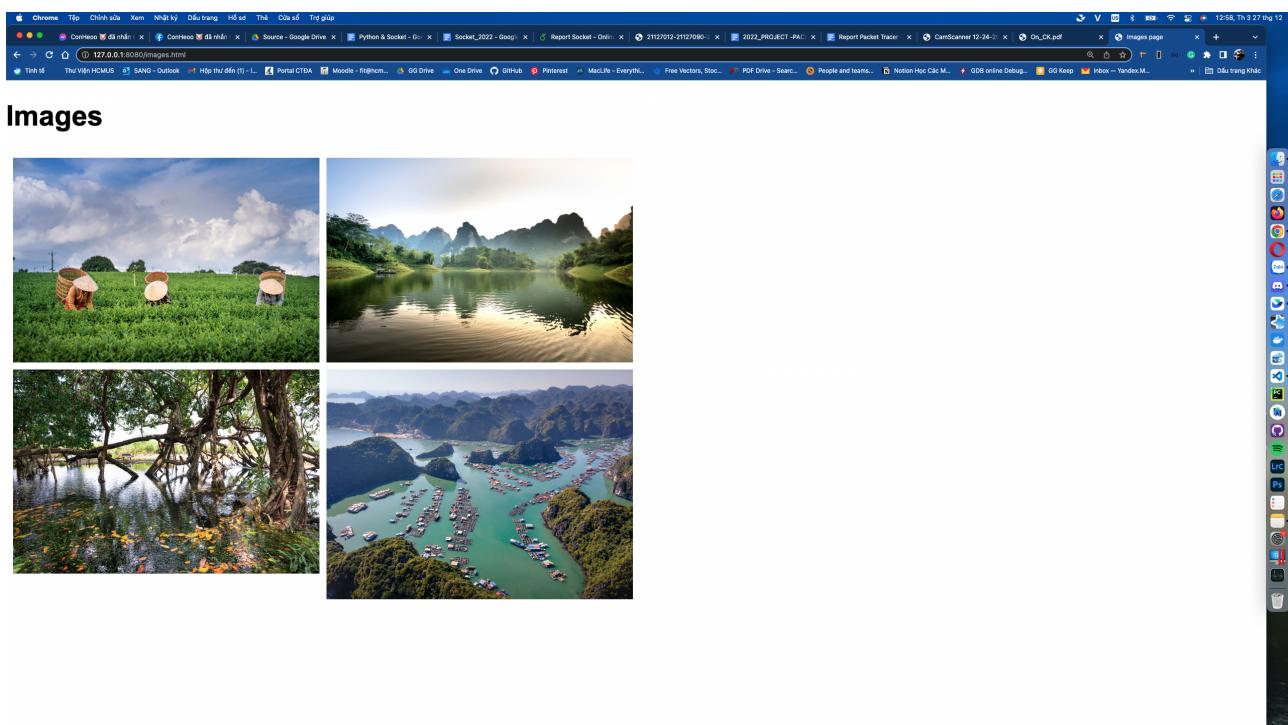
- Nhập tên đăng nhập: admin
- Nhập mật khẩu: 123456



- Nhấn login.
- Nếu nhập sai, page 401 Unauthorized sẽ được load lên.



- Nếu nhập đúng, trình duyệt sẽ tải lên các ảnh.



5 - Phân công công việc và mức độ hoàn thành

	Chức năng	Người thực hiện	Mức độ hoàn thành
1	Kết nối (0.5 điểm)	Đoàn Nam Thắng	100%
2	Quản lý kết nối (0.5 điểm)	Đoàn Nam Thắng	100%
3	Tải được page index.html (3.5 điểm)	Hồ Thành Nhân	100%
4	Dăng nhập (2 điểm)	Lê Hoàng Sang	100%
5	Lỗi page (0.5 điểm)	Đoàn Nam Thắng	100%
6	Multiple requests (1 điểm)	Đoàn Nam Thắng	100%
7	Multiple connection (1 điểm)	Đoàn Nam Thắng	100%
8	Report (1 điểm)	Lê Hoàng Sang	100%

6 - Các nguồn tài liệu tham khảo

- Learn Python Programming - Programiz - <https://www.programiz.com/python-programming>
- Hướng dẫn tạo webserver bằng python - WebServer using TCP Socket in Python - Lê Nguyễn - 2021 - <https://www.youtube.com/@lenguyen2560>
- How to Create Socket Server with Multiple Clients in Python - Digamber - November 25, 2022 - <https://www.positronx.io/create-socket-server-with-multiple-clients-in-python/>
- Socket Programming in Python (Guide) - Nathan Jennings - Feb 21, 2022 - <https://realpython.com/python-sockets/>
- TCP - Wikipedia - <https://vi.wikipedia.org/wiki/TCP>
- Giao thức HTTP và cấu trúc cơ bản của HTTP Message - XuanThuLab - <https://xuanthulab.net/giao-thuc-http-va-cau-truc-co-ban-cua-http-message.html>