ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO ĐỒ ÁN THỰC HÀNH

LẬP TRÌNH SOCKET

Bộ môn: Mạng máy tính

| Sinh viên thực hiện |

Võ Minh Khuê – 21120486

Hoàng Thị Khôn – 21120485

Nguyễn Ngọc Như Huyền – 21120475

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO ĐỒ ÁN THỰC HÀNH

LẬP TRÌNH SOCKET

Bộ môn: Mạng máy tính

| Giáo viên hướng dẫn |

Lý thuyết: Thầy Lê Hà Minh

Thực hành: Thầy Nguyễn Thanh Quân

MỤC LỤC

DANH MỤC HÌNH ẢNH

I. THÔNG TIN THÀNH VIÊN NHÓM

STT	Họ và tên	Mã số sinh viên	Email
1	Võ Minh Khuê	21120486	khuevo098@gmail.com
2	Hoàng Thị Khôn	21120485	hkhon2803@gmail.com
3	Nguyễn Ngọc Như Huyền	21120475	nguyenngocnhuhuyen1307@gmail.com

II. ĐÁNH GIÁ MỨC ĐỘ HOÀN THÀNH

STT	Công việc	Mức độ hoàn thành	Phần chưa hoàn thành
1	Kết nối	100%	
2	Quản lý kết nối	100%	
3	Tải được page index.html	100%	
4	Đăng nhập	100%	
5	Lỗi page	100%	
6	Multiple requests	100%	
7	Multiple connection	100%	
8	Report	100%	

III. PHÂN CÔNG CÔNG VIỆC TRONG ĐỒ ÁN

STT	Công việc	Người thực hiện
1	Kết nối	Hoàng Thị Khôn
2	Quản lý kết nối	Hoàng Thị Khôn
3	Tải được page index.html	Nguyễn Ngọc Như Huyền
4	Đăng nhập	Hoàng Thị Khôn
5	Lỗi page	Võ Minh Khuê
6	Multiple requests	Nguyễn Ngọc Như Huyền
7	Multiple connection	Võ Minh Khuê
8	Report	Cả nhóm

IV. KỊCH BẢN GIAO TIẾP CỦA CHƯƠNG TRÌNH

1. Giao thức trao đổi giữa Client và Server

- Giao thức HTTP (HyperText Transfer Protocol) được sử dụng để Web Server có thể đọc hiểu được thông điệp gửi từ Web Browser và ngược lại.
- Giao thức TCP/IP ở tầng Transport.

2. Xây dựng kịch bản giao tiếp giữa Client và Server

- Đầu tiên để các Client có thể kết nối và gửi nhận dữ liệu với Server chúng ta cần bật Server trước.
- Sau khi bật Server lên, Client (Web Browser) sẽ bắt đầu gửi yêu cầu kết nối cho Server thông qua địa chỉ IP của Server với port là 8080 được đặc tả trong URL.
- Server nhận yêu cầu kết nối của Client, chấp nhận yêu cầu kết nối và tiến hành trao đổi dữ liệu.
- Phản hồi của Server:
    - Nếu Client gửi yêu cầu GET một file tồn tại, Server sẽ trả về nội dung của file đó. Ngược lại, Server sẽ trả về page 404.
    - Nếu Client gửi yêu cầu POST: Server sẽ kiểm tra thông tin đăng nhập ở phần body. Nếu đúng thông tin đăng nhập thì trả về page hình ảnh, ngược lại trả về page 401.
- Server đóng kết nối: Kết nối chỉ đóng khi phía Client đóng kết nối (hàm Receive() trả về 0) hoặc xảy ra lỗi trong quá trình nhận dữ liệu.

3. Kiểu dữ liệu của thông điệp

- Dữ liệu nhận vào và gửi đi đều được lưu vào biến có kiểu String (chuỗi ký tự).

4. Cách thực thi chương trình

❖ **Tạo kết nối**

- Dùng hàm **Create()** khởi tạo socket có **port 8080** và giao thức sử dụng là **TCP** cho Server dùng để lắng nghe các kết nối từ phía Client.
- Sau đó hàm **Listen()** sẽ khởi tạo một hàng đợi để chứa các yêu cầu kết nối từ Client đang chờ được chấp nhận.

- Khi có kết nối từ Client trong hàng đợi, Server sẽ khởi tạo một socket mới dùng riêng cho kết nối này để chấp nhận trao đổi dữ liệu.

❖ **Quản lý nhiều kết nối và nhiều request trong một kết nối**

- Server luôn hoạt động và sẵn sàng chấp nhận kết nối từ phía Client (dùng vòng lặp vô tận).
- Sau khi chấp nhận kết nối thành công, Server sẽ tạo ra luồng tương ứng cho mỗi kết nối từ Client (tạo thread).
- Client được phép gửi nhiều request trong một kết nối bằng cách:
    - Cho phép Server nhận và xử lý các request liên tiếp và chỉ ngưng nhận khi Client đóng kết nối (vòng lặp được thực hiện khi hàm Receive() trả về số bytes nhận được lớn hơn 0).
    - Trong phần response của Server phải kèm theo header **Connection: keep-alive** để Client biết Server vẫn muốn tiếp tục duy trì kết nối sau khi gửi dữ liệu thành công.
- Khi Client ngắt kết nối, hàm Receive() của Server sẽ trả về 0, Server đóng kết nối bằng cách gọi hàm Close() cho socket, thoát thread và tiếp tục chờ những kết nối mới khác.

❖ **Gửi và nhận dữ liệu**

- Client có thể gửi request cho Server với phương thức **GET** hoặc **POST**.
- Request sẽ được lưu vào **buffer kiểu char** và dùng **stringstream** để tách các thông tin cần thiết cho việc xử lý và lưu vào **struct Request** (phương thức, tên file, body).
- Dữ liệu file được đọc từng ký tự và lưu vào biến kiểu string.
- Response được xây dựng theo cấu trúc thông điệp gửi của giao thức HTTP. Dùng **struct Response** và **stringstream** để lưu response.
- Định nghĩa các cấu trúc struct như hình, lưu các thông tin cần thiết cho thông điệp gửi và nhận.

```
18
19  struct info {
20      string uname;
21      string psw;
22  };
23
24  struct Request {
25      string method, path;
26      info user;
27  };
28
29  struct Response {
30      int statusCode;
31      size_t contentLength;
32      string contentType;
33      string statusText;
34      string content;
35  };
```

V. MÔI TRƯỜNG LẬP TRÌNH VÀ CÁC FRAMEWORK HỖ TRỢ CÁC THỰC THI ỨNG DỤNG

1. Môi trường lập trình

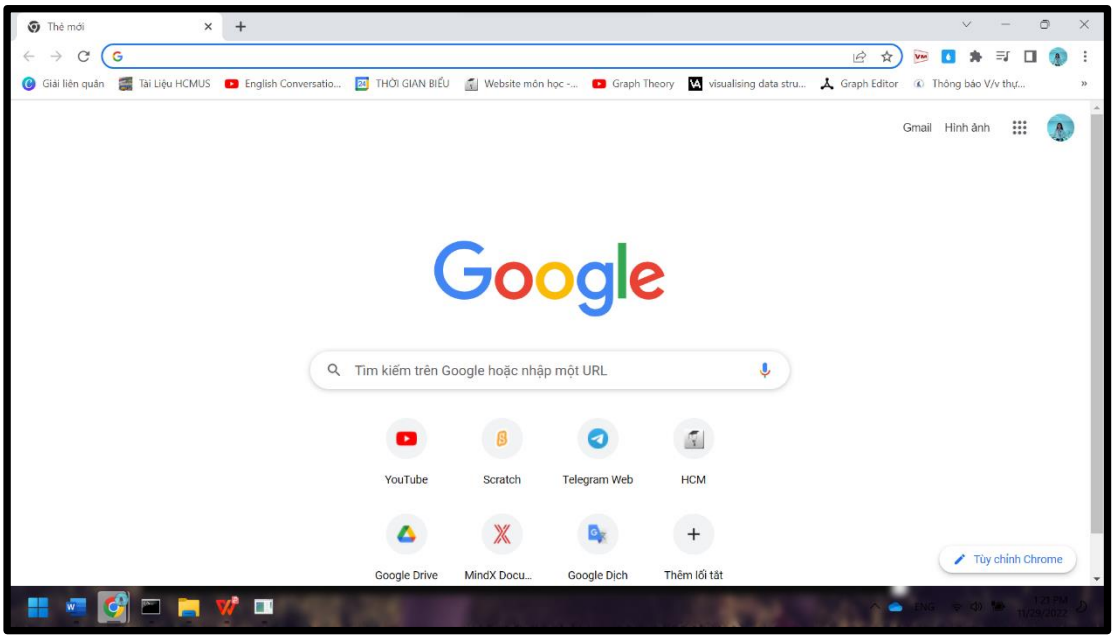
- Web server được viết và chạy trên IDE Visual Studio 2022 hệ điều hành Windows.
- Các file dữ liệu được thực thi ứng dụng được lưu dưới dạng tệp…
- Ngôn ngữ lập trình: C++

2. Các framework hỗ trợ để thực thi ứng dụng

- MFC (The Microsoft Foundation Class): Một thư viện lập trình hướng đối tượng viết bằng ngôn ngữ C++.
- Các thư viện hỗ trợ chương trình stdafx, afxsock, string, fstream, sstream.

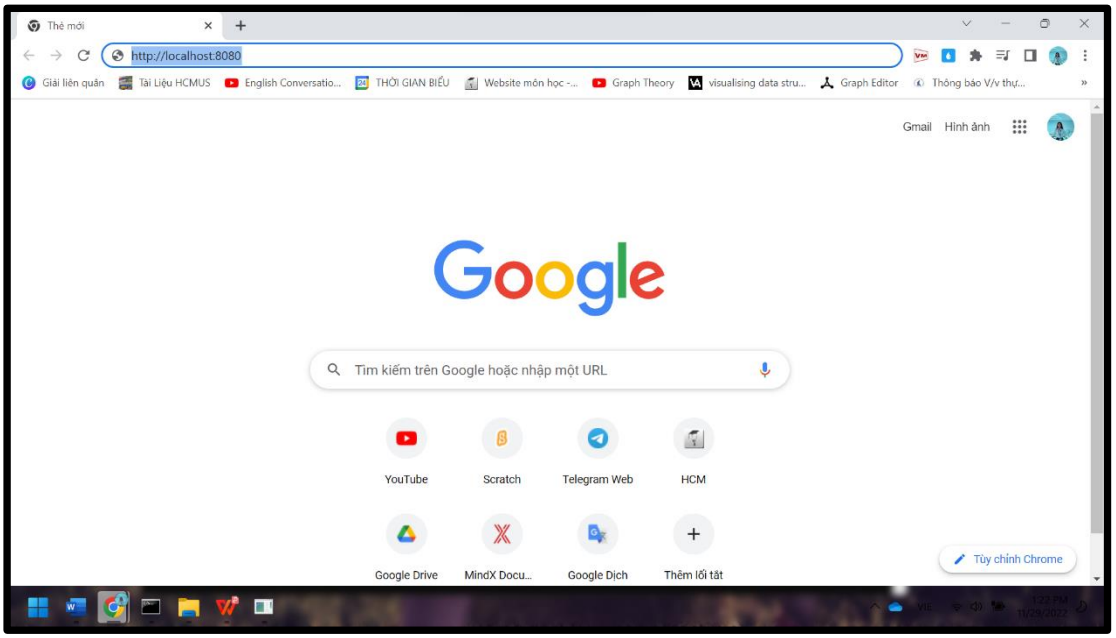
VI. HƯỚNG DẪN SỬ DỤNG CÁC TÍNH NĂNG CỦA CHƯƠNG TRÌNH

- Truy cập vào **Chrome, IE, Firefox, Safari…**

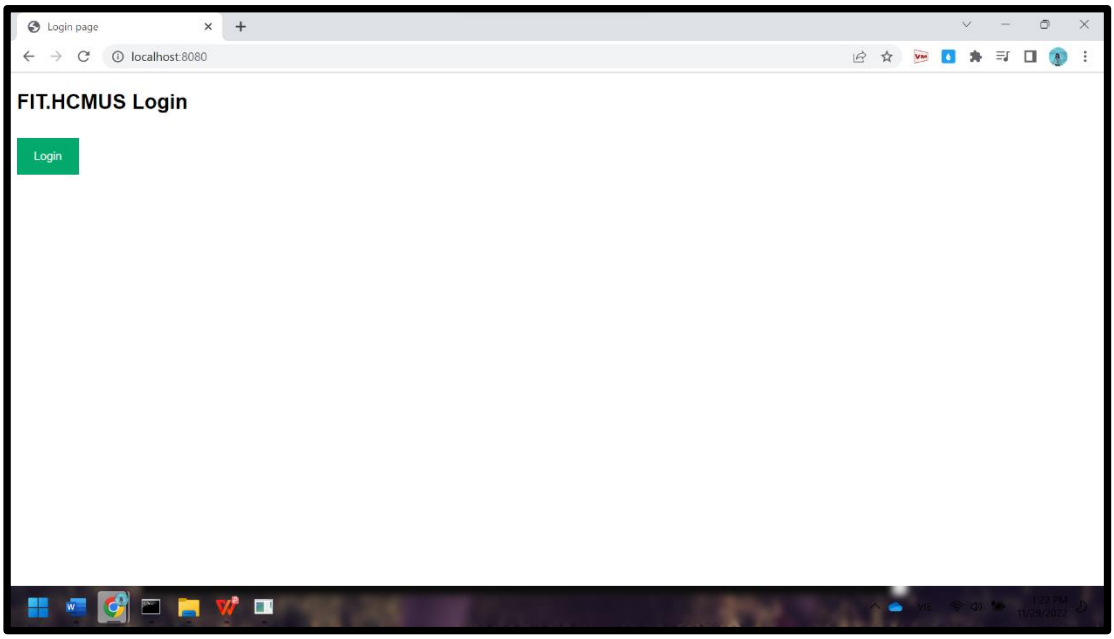


Hình 1: Sau khi truy cập Chrome

- Sau đó truy cập vào trang web HTTP với **port 8080**. Nếu sử dụng một số đường link sau: “http://localhost:8080”, “http://localhost:8080/index.html” thì Server sẽ render lên nội dung trang web.

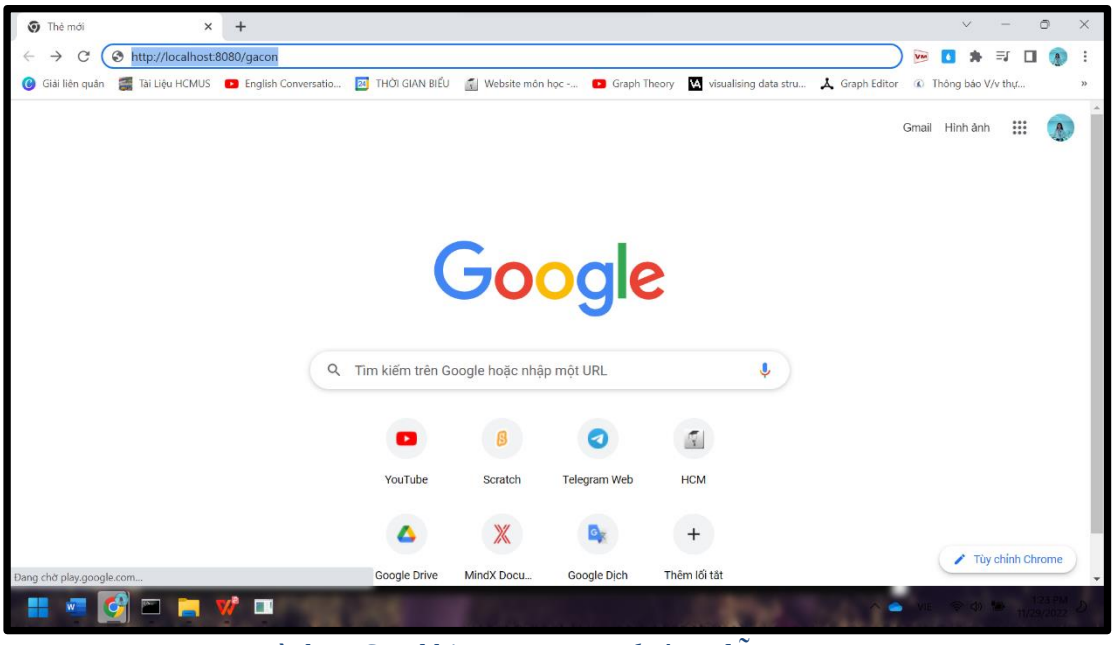


Hình 2: Truy cập vào đúng đường link

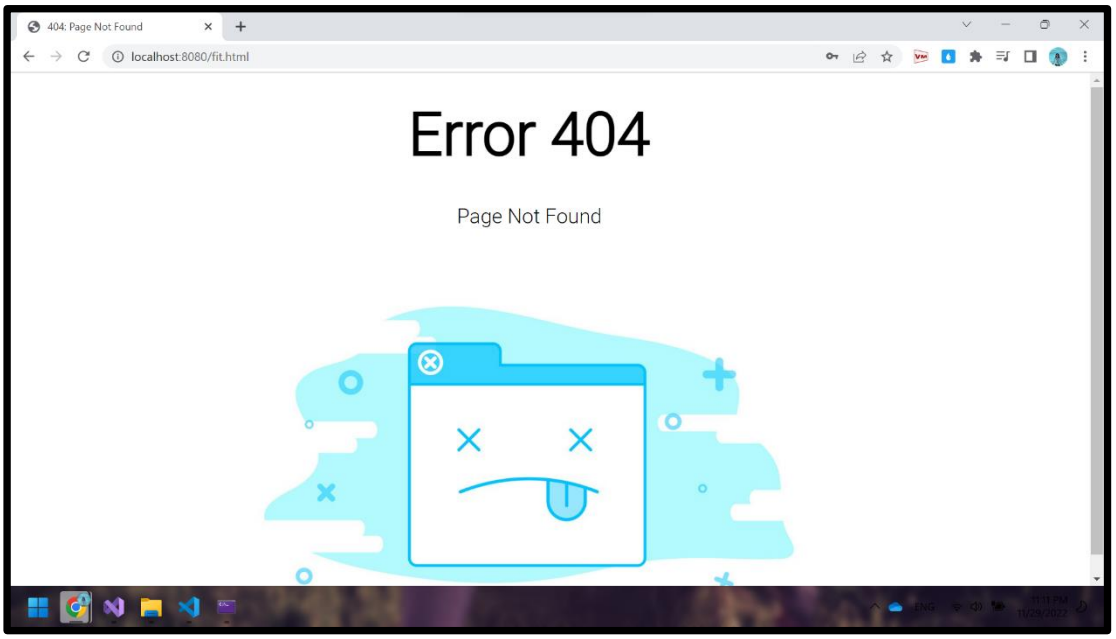


Hình 3: Render khi truy cập đúng đường link

- Nếu nhập địa chỉ web khác như “http://localhost:8080/gacon”, “http://localhost:8080/fit.html” thì Server sẽ render trang lỗi 404.

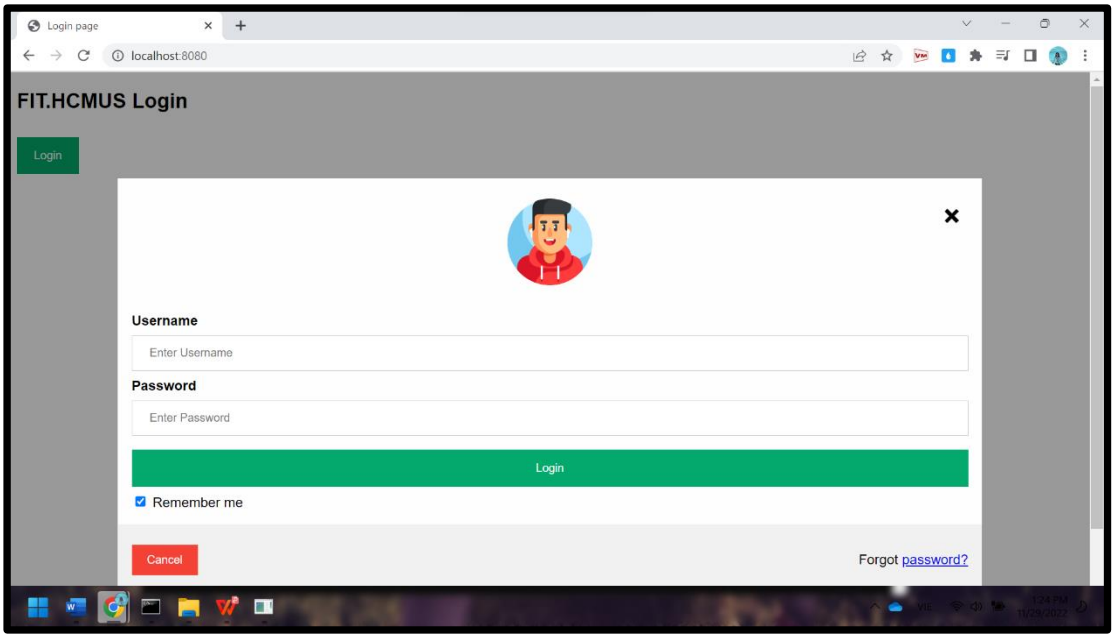


Hình 4: Sau khi truy cập sai đường dẫn



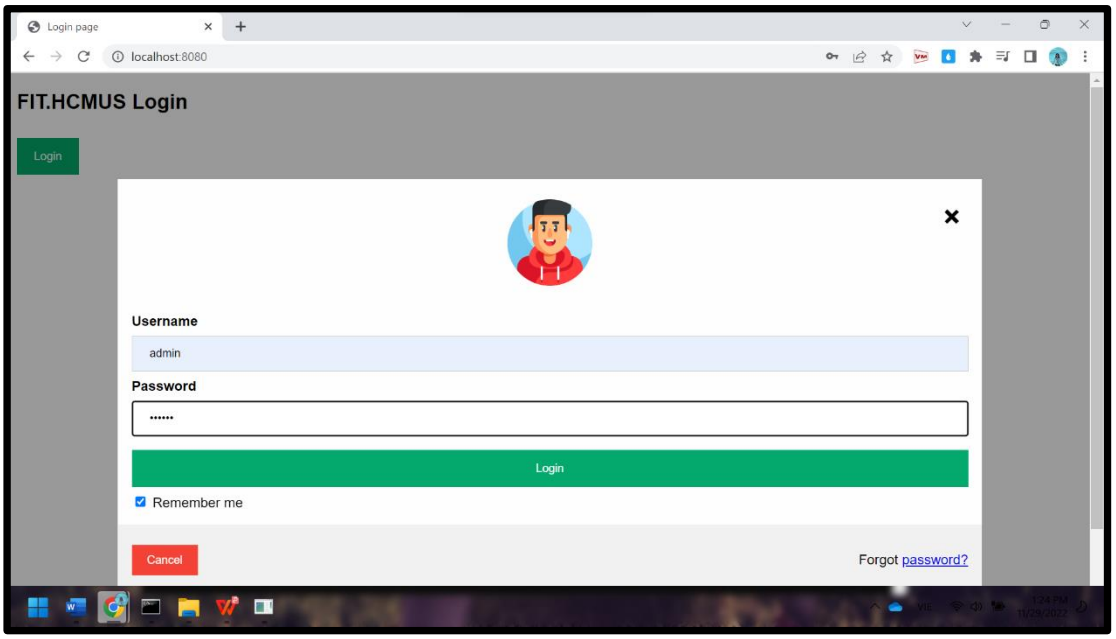
Hình 5: Render khi truy cập sai đường link

- Tiến hành **đăng nhập** thông tin ở phần Login.

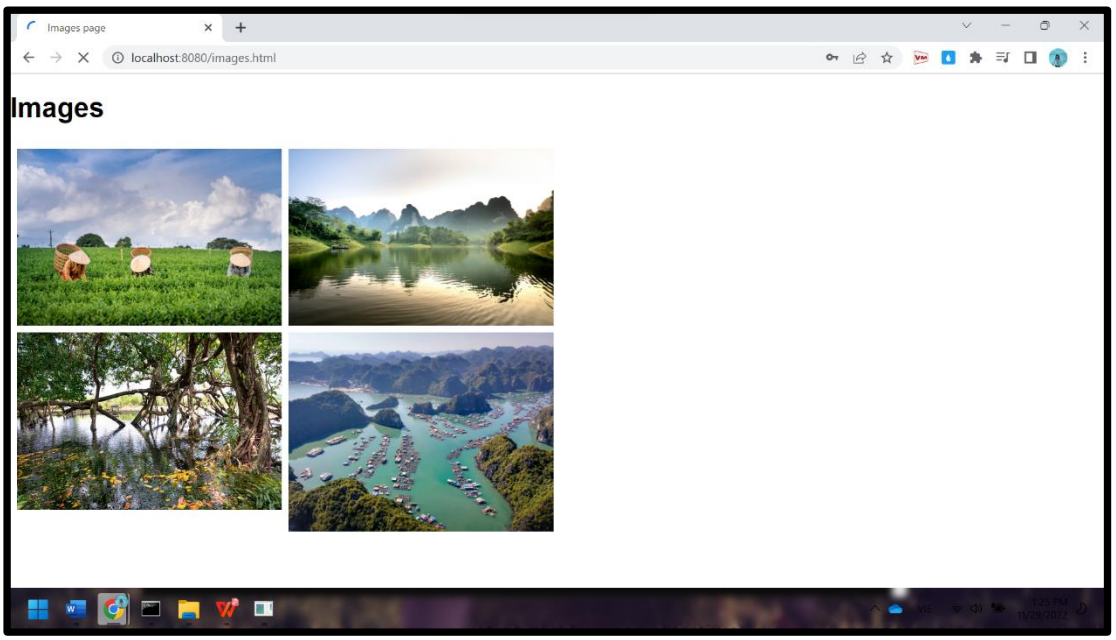


Hình 6: Đăng nhập thông tin

- Nếu đăng nhập đúng với **"uname"** là **"admin"** và **"psw"** là **"123456"** thì Server sẽ request trang web images.html.

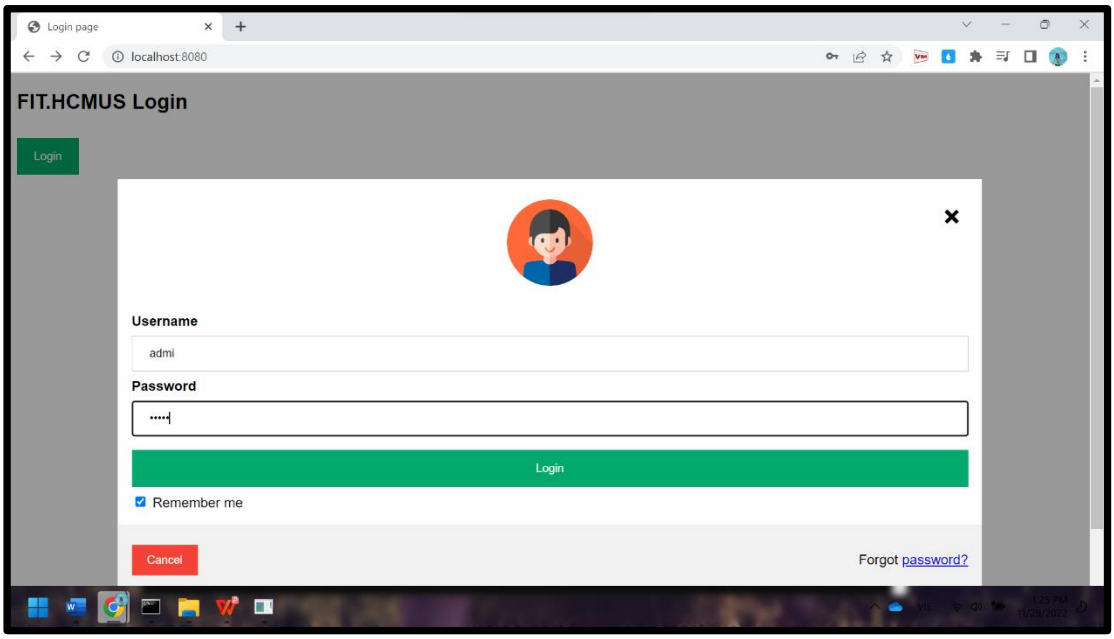


Hình 7: Đăng nhập đúng thông tin

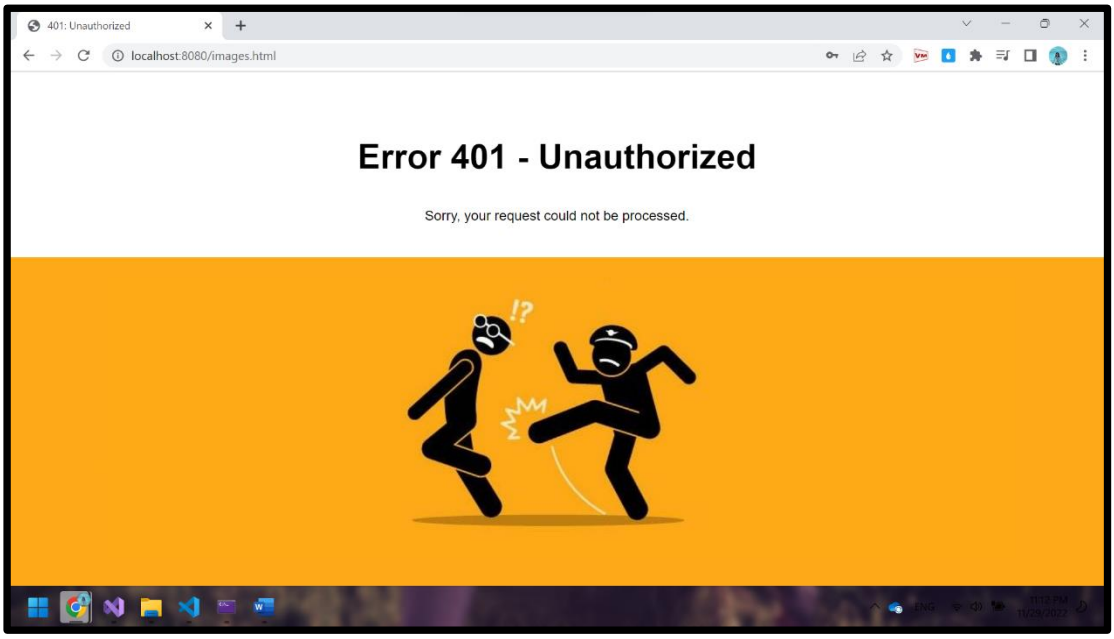


Hình 8: Request sau khi đăng nhập đúng

- Nếu đăng nhập không đúng với **"uname"** là **"admin"** và **"psw"** là **"123456"** thì Server sẽ request trang **lỗi 401.**



Hình 9: Đăng nhập thông tin sai



Hình 10: Request sau khi đăng nhập sai

VII. TÀI LIỆU THAM KHẢO

1. Giáo trình

+ **Mạng máy tính**, Mai Văn Cường - Trần Trung Dũng - Trần Hồng Ngọc - Lê Ngọc Sơn - Lê Giang Thanh - Trương Thị Mỹ Trang - Đào Anh Tuấn, NXB Khoa học & Kĩ Thuật.

+ **Computer Networking**: A top-down approach featuring the Internet, 7th edition, James F. Kurose, Keith W. Ross.

2. Trang web

+ **Các hàm CSocket**

https://learn.microsoft.com/en-us/cpp/mfc/reference/casyncsocket-class?view=msvc-170

+ **Cấu trúc thông điệp HTTP**

https://developer.mozilla.org/en-US/docs/Web/HTTP/Messages

+ **Lưu dữ liệu file**

https://en.cppreference.com/w/cpp/iterator/istreambuf_iterator

+ **Connection**

https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Connection

+ **Thread**

https://learn.microsoft.com/en-us/windows/win32/api/processthreadsapi/nf-processthreadsapi-createthread