ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

Sinh viên thực hiện:	***MSSV:***
Nguyễn Huy Tú	**18120254**
Nguyễn Duy Vũ	**18120264**

ĐỒ ÁN LẬP TRÌNH SOCKET

XÂY DỰNG WEB SERVER ĐƠN GIẢN

Môn học: Mạng Máy Tính

Thành phố Hồ Chí Minh - 2020

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

Nguyễn Huy Tú	**18120254**
Nguyễn Duy Vũ	**18120264**

ĐỒ ÁN LẬP TRÌNH SOCKET

| Đề tài |

XÂY DỰNG WEB SERVER ĐƠN GIẢN

| Giáo viên hướng dẫn |

ThS. Lê Giang Thanh

| Giáo viên thực hành |

ThS. Lê Hà Minh

Môn học: Mạng Máy Tính

Thành phố Hồ Chí Minh – 2020

LỜI CẢM ƠN

Qua thời gian học tập và rèn luyện tại trường đại học Khoa Học Tự Nhiên, Thành phố Hồ Chí Minh, được sự chỉ bảo và giảng dạy nhiệt tình của quý thầy cô khoa Công nghệ Thông tin, vốn kiến thức được tiếp thu trong quá trình học không chỉ là nền tảng cho quá trình nghiên cứu bài báo cáo mà còn là hành trang để nhóm em bước vào đời một cách vững chắc và tự tin.

Với lòng biết ơn sâu sắc nhất, nhóm xin được gửi đến quý thầy cô ở khoa Công nghệ Thông tin trường đại học Khoa Học Tự Nhiên, đặc biệt là ThS. Lê Giang Thanh, người đã tận tình hướng dẫn, giúp đỡ nhóm hoàn thành chuyên đề báo cáo này, lời cảm ơn chân thành nhất. Nếu không có những lời hướng dẫn, dạy bảo của thầy thì bài báo cáo này khó lòng mà hoàn thiện được.

Nhóm cũng muốn gửi lời cảm ơn đến gia đình, bạn bè, và người thân đã luôn ủng hộ và tạo điều kiện tốt nhất để nhóm có thể tập trung nghiên cứu đề tài này.

Trong quá trình thực tập cũng như làm báo cáo, do trình độ lý luận cũng như kinh nghiệm thực tiễn còn hạn chế nên nhóm khó tránh khỏi sai sót. Nhóm rất mong nhận được những ý kiến đóng góp quý báu của quý Thầy Cô và các bạn để nhóm học hỏi được nhiều kinh nghiệm và hoàn thành tốt hơn trong bài báo cáo sau này.

Trân trọng.

TP.HCM, ngày 11 tháng 07 năm 2020

Sinh viên thực hiện

MỤC LỤC

MỤC LỤC HÌNH ẢNH

THÔNG TIN ĐỒ ÁN

Thông tin nhóm

MSSV	Họ tên	Email	Vai trò
18120254	Nguyễn Huy Tú	18120254@student.hcmus.edu.vn	Developer
18120264	Nguyễn Duy Vũ	Vu38988@gmail.com	Developer

Thông tin đồ án

Tên quy trình	Tên
IDE	Visual Studio 2019
Báo cáo	Microsoft Word
Giao diện	Console Prompt
Tên	Web Server đơn giản
Product Owner	Thầy Lê Thanh Giang

Mô tả đồ án

Đồ án này viết một chương trình Web Server đơn giản trên nền console trả về nội dung các trang web được yêu cầu bởi thầy Lê Giang Thanh. Đồ án được viết bằng ngôn ngữ C++, HTML và CSS, không quản lý login session, cookies và không sử dụng thư viện HTTP có sẵn.

BÁO CÁO ĐỒ ÁN

Phân công thành viên

STT	Tên công việc	Thực hiện	Ghi chú
1	Lập trình Server	Nguyễn Huy Tú - 18120254	Các thành viên hỗ trợ nhau trong quá trình thực hiện đồ án
2	Xử lý multi-thread		
3	Xử lý các HTTP Request/Response		
4	Viết báo cáo		
5	Lập trình HTML + CSS	Nguyễn Duy Vũ - 18120264	
6	Kiểm tra đăng nhập		

Tiến trình đồ án

Đánh giá	Mức độ hoàn thành (%)	Phần chưa làm được
Truy cập Web thông qua URL	100%	0%
Hiển thị trang index.html	100%	0%
Hiển thị trang info.html	100%	0%
Hiển thị 404 khi đăng nhập/ truy cập sai	100%	0%
Kiểm tra đăng nhập	100%	0%
Thiết kế giao diện trang web	100%	0%

BÁO CÁO KĨ THUẬT

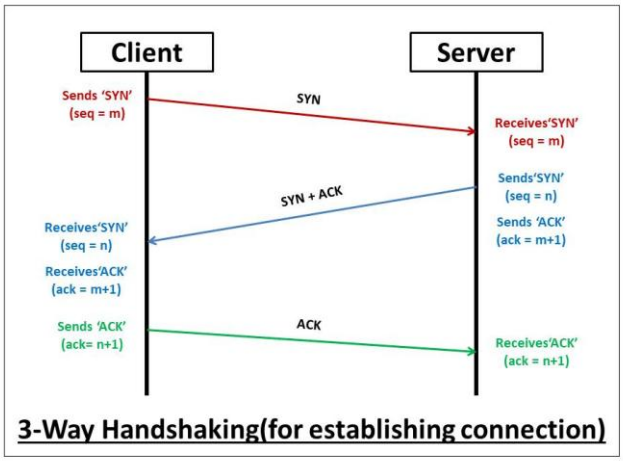
1. Kiến trúc ứng dụng.

Đồ án sử dụng kiến trúc Client-Server. Server được lập trình chạy sẵn, Client là Web Browser. Khi đó, Client sẽ truy cập đến Server thông qua URL là địa chỉ IP và port số hiệu.

2. Giao thức tầng Vận chuyển.

Đồ án sử dụng giao thức TCP nhờ những ưu điểm:

- Cung cấp dịch vụ hai chiều *(full-duplex)*, thực hiện quá trình bắt tay ba bước *(three-way handshake)* trước khi truyền dữ liệu.
- Đảm bảo chất lượng gói tin.
- Có cơ chế kiểm soát tắc nghẽn.



Hình 1. Quá trình bắt tay 3 bước

Một số đặc điểm khi sử dụng giao thức vận chuyển hướng kết nối:

- Tiêu tốn đáng kể nhiều thời gian cho việc thiết lập kết nối.
- TCP kìm hãm tốc độ gửi của bên gửi khi có tình trạng tắc nghẽn.
- Kết nối theo kiểu dòng *(stream-of-bytes).*

3. Các port sử dụng ở Server & Client.

Thông thường, mặc định port sử dụng cho giao thức HTTP (Web) là port 80. Tuy nhiên, có thể có những ứng dụng khác sử dụng port này. Vì thế, để tránh gây xung đột, chúng em chọn port cho Server là port 8080.

```
#define PORT 8080
```

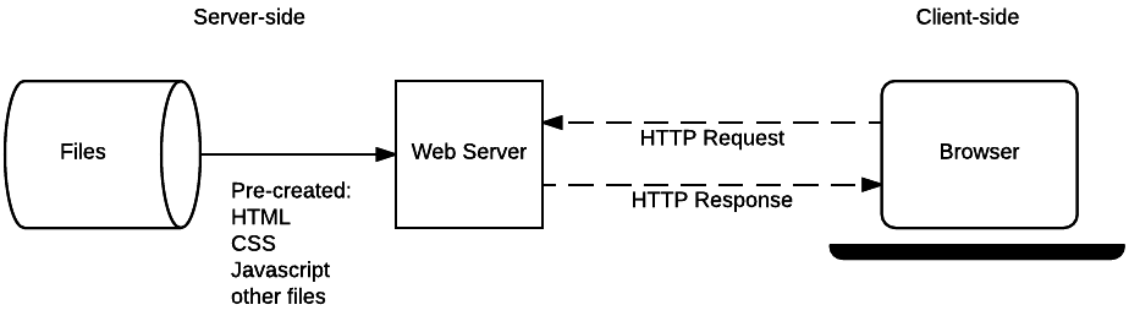
Port cho Client sẽ do Browser tự cung cấp.

4. Giao thức tầng Ứng dụng.

4.1. HTTP Stateless.

Đồ án sử dụng giao thức HTTP, hoạt động theo kiểu Stateless. Cách thức hoạt động:

- HTTP hoạt động trên giao thức TCP/IP port 8080.
- Client mở TCP ở port 8080 kết nối đến Server.
- Client gửi một HTTP Request, Server hồi đáp với một HTTP Response.
- Server đóng kết nối TCP.



Hình 2. Giao tiếp giữa Client & Server

4.2. HTTP Keep-alive.

Mô hình HTTP được sử dụng là HTTP Keep-Alive (kết nối HTTP Persistent): cho phép một kết nối TCP duy nhất vẫn mở cho nhiều HTTP Request/Response.

Quá trình Keep-Alive có 3 thuộc tính để quyết định có đóng kết nối không:

- *tcp_keepalive_time:* thời gian không có tín hiệu, mặc định là 7200 giây.
- *tcp_keepalive_intvl:* thời gian chờ chiều bên kia hồi đáp, mặc định là 75 giây.
- *tcp_keepalive_probles:* số lần thử lại nếu việc giao tiếp gặp lỗi, mặc định là 9.

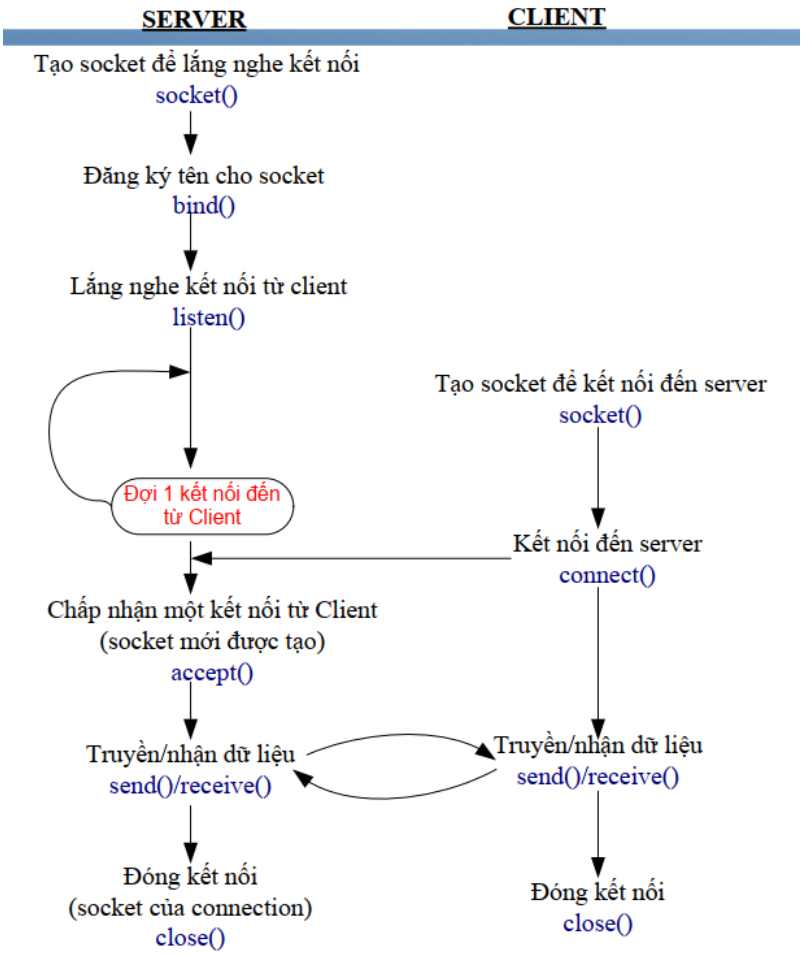
Như vậy thì mặc định, Client socket sẽ mở: tcp_keepalive_time + tcp_keepalive_intvl * tcp_keepalive_probes = 7200 + 75*9 = 2 giờ 11 phút nếu không nhận được RST của Server.

5. Mô hình ứng dụng.

5.1. Mô hình TCP.

Kết nối giữa Web Client và Web Server sẽ trải qua 4 giai đoạn.

- Giai đoạn 1: Server tạo Socket và lắng nghe yêu cầu kết nối PORT
- Giai đoạn 2: Client tạo Socket, yêu cầu thiết lập một kết với Server.
- Giai đoạn 3: Trao đổi thông tin giữ Client và Server.
- Giai đoạn 4: Kết thúc phiên làm việc, Client đóng kết nối. Server tiếp tục hoạt động và lắng nghe các Client khác hoặc sẽ đóng kết nối khi nào người lập trình muốn đóng.



Hình 3. Mô hình TCP

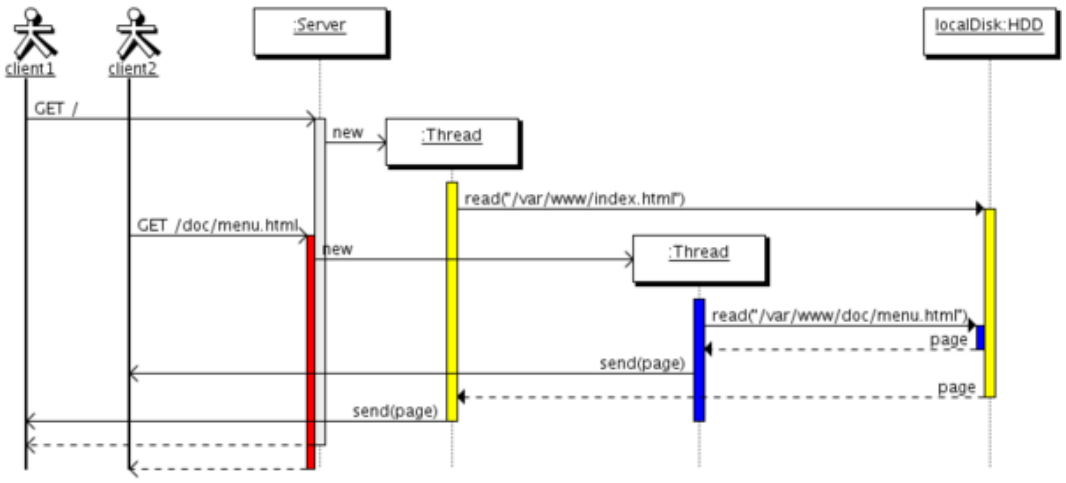
Nhiều hàm trong lập trình socket hoạt động theo cơ chế Blocking (ví dụ như hàm `accept`, hàm `recv`). Khi Server gọi hàm `accept()`, chương trình Server sẽ treo đến khi có một Client gọi hàm `connect()` để kết nối đến. Để giải quyết vấn đề này, chúng em sử dụng cơ chế Multi-threading.

5.2. Multi-threading.

Multi Thread sẽ tạo ra một quá trình bao gồm nhiều Thread được thực thi. Một Thread thực hiện là chuỗi nhỏ nhất của tập lệnh lập trình có thể được quản lý độc lập. Những Thread có thể chạy song song và nó có thể làm tăng hiệu quả của chương trình.

Sau khi tạo ra các Thread con từ Thread cha, ta sẽ join chúng lại để Mainthread đợi các Thread con thực thi xong rồi mới huỷ.

⇨ *Mỗi Client sẽ là 1 Thread.*



Hình 4. Multithread với Client

6. Tóm tắt nội dung đồ án.

6.1. Hàm main.

```
Server s;
if (s.init())
	return 1;
if (s.Bind())
	return 1;
s.Listen();
bool running = true;
while (running) {
	spawnThreads(MAXCLIENT, s);
```

```
        }
        s.close();
        return 0;
```

Đầu tiên, ta khởi tạo đối tượng Server đã được định nghĩa. Sau đó, gọi hàm `Bind()` để kết buộc Server với địa chỉ IP và port cố định. Đưa Server vào trạng thái lắng nghe. Sau đó cho Server vào vòng lặp vô tận để bắt đầu quá trình chạy.

Trong khi Server đang chạy, thì ta tạo ra `MAXCLIENT` thread với `MAXCLIENT` là số client tối đa mà Server lắng nghe được. Vì chỉ mô phỏng Web Server đơn giản nên `MAXCLIENT` được định nghĩa là 20, tức là có thể có cùng lúc 20 Client kết nối tới Server. Mainthread sẽ đợi 20 thread con kết thúc, sau đó tiếp tục vòng lặp với 20 thread mới.

```
#define MAXCLIENT 20
```

Server sẽ đóng kết nối khi nào người quản trị Server muốn tắt Server, dùng tổ hợp Ctrl+C để tắt Server.

6.2. Class Server.

```
class Server {
protected:
    SOCKET listening;
    sockaddr_in hint;
public:
    Server();
    ~Server();

    int initWinSock();
    int initSocket();
    int init();

    int Bind();
    int Listen();
    void accept_sendClient();
    void handleClientRequest(char [MAXBUFLEN], string&, int&);


    void closeSocket();
    void closeWinSock();
    void close();
};
```

Server gồm 2 thuộc tính protected mang kiểu `SOCKET` và `sockaddr_in`:

- Cấu trúc `SOCKET` lưu giữ con số nguyên định danh socket (af, type, protocol). Trong đó:

- af là một con số ID quyết định Socket sử dụng giao thức để kết nối TCP/IP là `AF_INET`.
- type quy định giao thức vận chuyển dữ liệu TCP là `SOCK_DGRAM`.
- protocol: chỉ định rõ lại giao thức. Vì TCP chỉ dùng `SOCK_DGRAM` nên giá trị này không quan trọng.

- Cấu trúc `sockaddr_in` lưu giữ thông tin địa chỉ IP, số port và kiểu mô hình (TCP) của server.

Vì không thể sử dụng public IP (do phải đăng ký), địa chỉ 192.168.1.1 thì đã bị dùng làm Home Gateway (đối với mạng FPT Telecom). Nên chúng em chọn địa chỉ private của máy tính là địa chỉ loopback – `127.0.0.1(INADDR_ANY)` làm địa chỉ IP của Server để demo Web.

Thiết bị khác máy tính muốn truy cập vào trang web thì dùng lệnh `ipconfig` để lấy địa chỉ IPv4 của máy tính thay cho `127.0.0.1`

Các hàm trong class:

`Server();`

- Hàm tạo cho class Server.

`~Server();`

- Hàm huỷ cho class Server.

`int initWinSock();`

- *Chức năng:* Khởi tạo Winsock.
- *Nội dung:* Gọi hàm `WSAStartup()` với giá trị 0x0202, tức là phiên bản 2.2.
- *Trả về:* 0 nếu thành công, 1 nếu thất bại.

`int initSocket();`

- *Chức năng:* Khởi tạo socket lắng nghe cho Server.
- *Nội dung:* Gọi hàm `socket(AF_INET, SOCK_STREAM, 0)` với các tham số phù hợp với mô hình TCP/IP.
- *Trả về:* 0 nếu thành công, 1 nếu thất bại.

`int init();`

- *Chức năng:* Khởi tạo cần thiết trước khi trao đổi dữ liệu Client-Server.
- *Nội dung:* Chứa 2 hàm `initWinSock()` và `initSocket()`.
- *Trả về:* 0 nếu thành công, 1 nếu thất bại.

`int Bind();`

- *Chức năng:* Kết buộc socket lắng nghe đã tạo với địa chỉ IP và port cố định.
- *Nội dung:* Gọi hàm `bind(this->listening, (sockaddr*)&(this->hint), sizeof(this->hint))` để chứa địa chỉ IP và port vào cấu trúc sockaddr_in hint.
- *Trả về:* 0 nếu thành công, 1 nếu thất bại.

`int Listen();`

- *Chức năng:* Đưa socket của server vào trạng thái lắng nghe.
- *Nội dung:*. Trong hàm sẽ gọi `listen(listening, MAXCLIENT)` với `MAXCLIENT` là số Client tối đa kết nối Server trong cùng 1 lúc.
  Các thủ tục tiếp theo sẽ bị block tại đây cho đến khi có Client connect tới Server.
- *Trả về:* 0 nếu thành công, 1 nếu thất bại.

`void accept_sendClient();`

- *Chức năng:* chấp nhận kết nối của Client, xử lý yêu cầu Client, gửi lại thông điệp Client yêu cầu.
- *Nội dung:* Khi có Client kết nối đến Server, Client sẽ chờ Server chấp nhận kết nối bằng hàm `accept(listening, (sockaddr*)&client, &clientSize)`.
  Sau khi hàm `accept` được gọi, một socket mới được trả ra đại diện cho connection giữa Server và Client.
  Tiếp đến, xử lý mong muốn của client thông qua nội dung mà Client gửi bằng hàm `handleClientRequest(buf, output, size)`.
  Cuối cùng, gửi dữ liệu `(output)` cho socket của Client `(clientSocket)` bằng hàm `send(clientSocket, output.c_str(), size, 0)`.

- *Trả về:* không.

`void handleClientRequest(char buf[MAXBUFLEN],string& output,int& size)`

- *Chức năng:* Xử lý HTTP Request (GET/POST). Trả ra Response tương ứng.
- *Truyền vào:* `buf` là nội dung được gửi đến từ client, `output` là nội dung client yêu cầu, `size` là kích thước của `output`.
- *Nội dung:* Phân tích nội dung Client gửi đến là GET hay POST.
  Nếu là HTTP Method GET thì phân tích file html mà Client yêu cầu, sau đó tạo header và body là nội dung file html rồi lưu vào `output`.
  - Nếu người dùng không yêu cầu file html thì mặc định trả ra trang đăng nhập index.html.
  - Nếu người dùng yêu cầu 1 đường dẫn không có thực, trả ra thông báo "404 Not Found".

  Nếu là HTTP Method POST thì phân tích username và password người dùng nhập đúng không:
  - Nếu đúng, trả ra `output` gồm header và body là nội dung file info.html.
  - Nếu sai, trả ra `output` gồm header và body là nội dung file 404.html.
- *Trả về:* không.

`void closeSocket();`

- *Chức năng:* Đóng socket lắng nghe của server.
- *Nội dung:* Chứa hàm `closesocket(this->listening)`.
- *Trả về:* không.

`void closeWinSock();`

- *Chức năng:* Tắt Winsock.
- *Nội dung:* Chức hàm `WSACleanup()`.
- *Trả về:* không.

void close();

- *Chức năng:* "dọn dẹp" sau khi kết thúc chương trình.
- *Nội dung:* Chứa hàm `closeSocket()` và `closeWinSock()`.
- *Trả về:* không.

6.3. HTML cho trang hiển thị.

Code cho trang index.html

```
<!DOCTYPE html>
<html>
<head>
    <style>
        body { font-family: Arial, Helvetica, sans-serif; }
        form { border: 3px solid #f1f1f1; }
        input[type=text], input[type=password] { width: 100%;
            padding: 12px 20px;
            margin: 8px 0;
            display: inline-block;
            border: 1px solid #ccc;
            box-sizing: border-box; }
        button { background-color: #4CAF50;
            color: white;
            padding: 14px 20px;
            margin: 8px 0;
            border: none;
            cursor: pointer;
            width: 100%; }
            button:hover { opacity: 0.8; }
        .container { padding: 16px; }
        h1{ font-size: 2.5em; text-align: center; }
    </style>
</head>
<body>
    <h1>Login Form</h1>
    <form method="post">
        <div class="container">
            <label for="user"><b>Username</b></label>
            <input type="text" placeholder="Enter Username" name="user" required>
            <label for="pass"><b>Password</b></label>
            <input type="password" placeholder="Enter Password" name="pass" required>
            <button type="submit">Login</button>
        </div>
    </form>
</body>
</html>
```

Code cho trang info.html

```
<!doctype html>
<html>
    <head>
        <style>
            body { font-family: Arial, Helvetica, sans-serif; }
            h1 { font-size: 2.5em; text-align: center; }
```

```
            .info { border: 3px solid #f1f1f1; padding: 16px; }
            p{ width: 250px; float: right }
            .member { width: 100%;
                padding: 12px 20px;
                margin: 8px 0;
                display: inline-block;
                border: 10px solid #ccc;
                box-sizing: border-box; }
        </style>
    </head>
    <body>
        <h1>Members</h1>
        <div class="info">
            <div class="member">
                <h2>Huy Tú</h2>
                <img
src="https://i.pinimg.com/originals/c6/44/7f/c6447f42f6a3959e71385ad42e478e3a.jpg"
alt="Huy_Tu" width="150" height="150" />
                <p>
                    Full name: Nguyễn Huy Tú
                    <br />Student ID: 18120254
                    <br />Facebook: <a
href="https://www.facebook.com/threecarbonsminusp">HuyTu Nguyen</a>
                </p>
            </div>
            <div class="member">
                <h2>Duy Vũ</h2>
                <img
src="https://i.pinimg.com/originals/2c/f0/c4/2cf0c4ffb3fd8f41b2d563bf70bd3a35.jpg"
alt="Duy_Vu" width="150" height="150" />
                <p>
                    Full name: Nguyễn Duy Vũ
                    <br />Student ID: 18120264
                    <br />Facebook: <a
href="https://www.facebook.com/nguyenduyvu1807">Nguyễn Duy Vũ</a>
                </p>
            </div>
        </div>
</body>
</html>
```

Code cho trang 404.html

```
<!doctype html>
<html>
    <head>
        <style>
            body{ border: 15px solid #f1f1f1; font-size:xx-large; text-align: center;
}
        </style>
    </head>
    <body>
        <div class="content">
            <h1>404</h1>
            <p>Page not found</p>
        </div>
    </body>
</html>
```

6.4. Cấu trúc source code.

Gồm file *main.cpp* chứa hàm main.

File *Server.h* định nghĩa Header cho Server. *Server.cpp* chứa mã nguồn lập trình Server.

File *Header.h* chứa Header, *Source.cpp* chứa mã nguồn cho phần lập trình Multithread.

Thư mục html chứa 3 file: *index.html, info.html, 404.html.*

7. Những hạn chế của đồ án.

Mô hình Web Server cơ bản nên chưa có những cài đặt để quản lý kiểm soát tắc nghẽn khi có nhiều Client truy cập vào Server. Để tiện cho việc test chương trình, chúng em chỉ đưa ra con số 20 Client truy cập cùng lúc. Sau khi 20 Client disconnect, 20 Client tiếp theo mới có thể truy cập vào, và tiếp tục như thế.

Web Server hiển thị thông tin cơ bản, không yêu cầu sử dụng login session và quản lý cookies. Vì thế chúng em không lập trình theo kiểu HTTP Stateful với database, transaction, … nên không thể bảo mật trường hợp người dùng chưa đăng nhập nhưng biết URL đến trang info thì vẫn có thể hiển thị trang.

Quá trình bắt tay 3 bước tốn khá nhiều thời gian chuẩn bị khi người dùng truy cập và chưa có cách tối ưu.

Giao diện HTML đơn giản, chưa có các pop-ups hướng dẫn nếu người dùng thao tác sai.

Đồ án chỉ sử dụng địa chỉ private của đường mạng, chưa đăng ký đường mạng public với tên miền cụ thể.

HƯỚNG DẪN SỬ DỤNG

Trong folder Release, mở file thực thi Server.exe để bắt đầu khởi động Server.

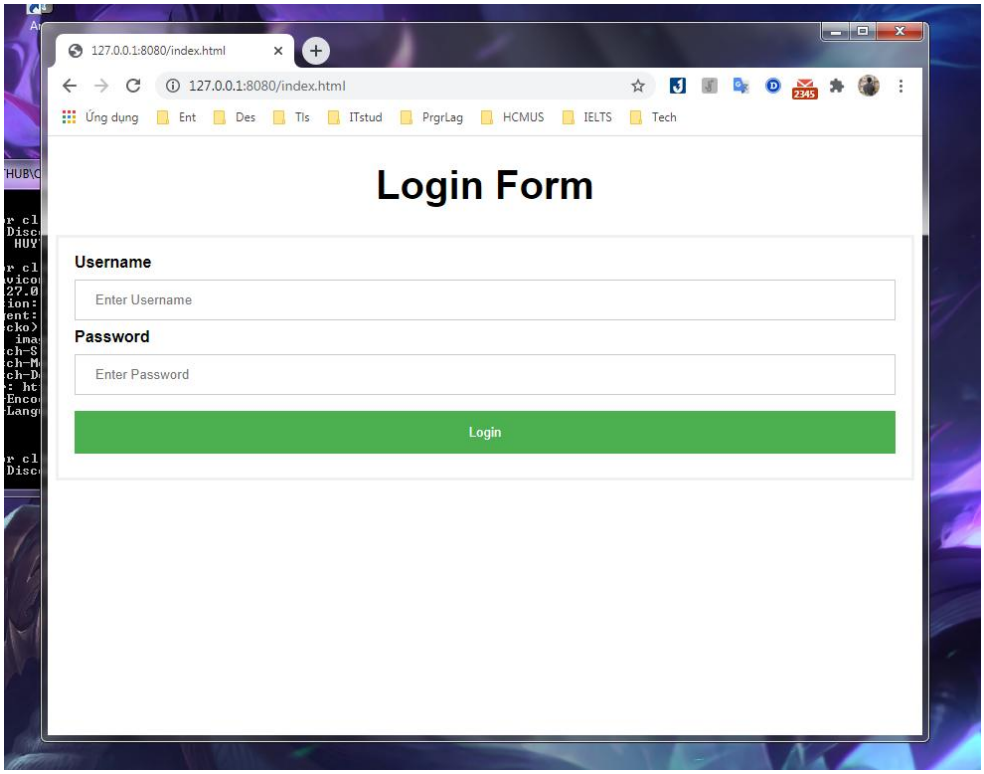
Lưu ý khi sử dụng chương trình:

- *Trong folder Release phải có folder html thì Server mới trả ra trang thông tin được.*
- *Hình ảnh ở trang thông tin được dẫn link từ trang upload hình ảnh trực tuyến Imgur.com. Nếu người dùng không kết nối Internet thì không thể hiển thị hình ảnh.*

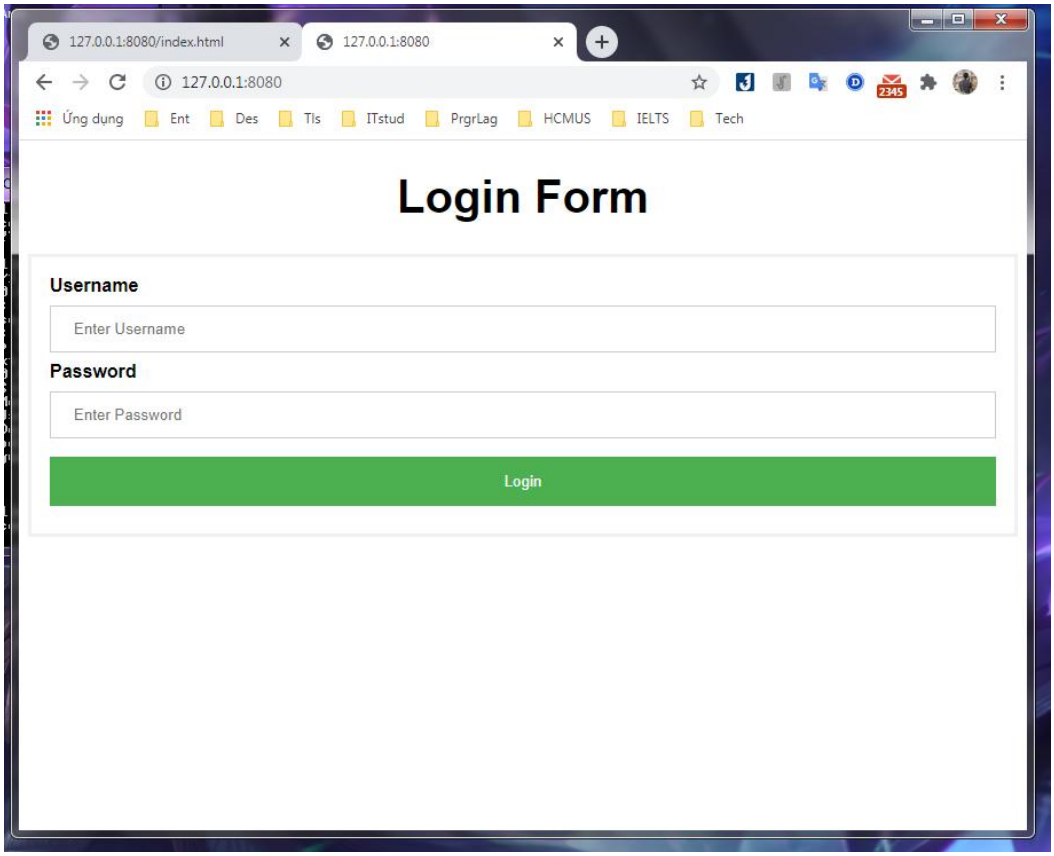
Vì đây là mô hình HTTP Keep-Alive, Server sẽ hoạt động đến khi nào quản trị viên muốn dừng. Nếu muốn dừng Server, nhấn tổ hợp phím CTRL+C để kết thúc chương trình.

Mở trình duyệt bất kỳ *(v.d: Google Chrome)* để bắt đầu truy cập đến Server. Ở thanh địa chỉ URL, người dùng nhập địa chỉ IP và số port của Server theo cú pháp: *protocol://host_name[:port][/path][/file_name]*

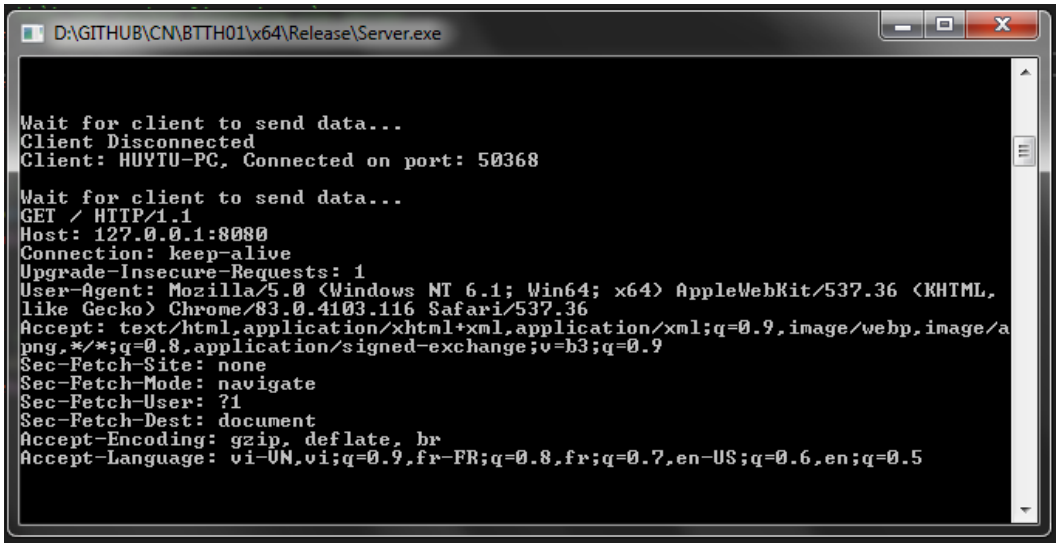
Cụ thể đầy đủ sẽ là: *http://127.0.0.1:8080/index.html*



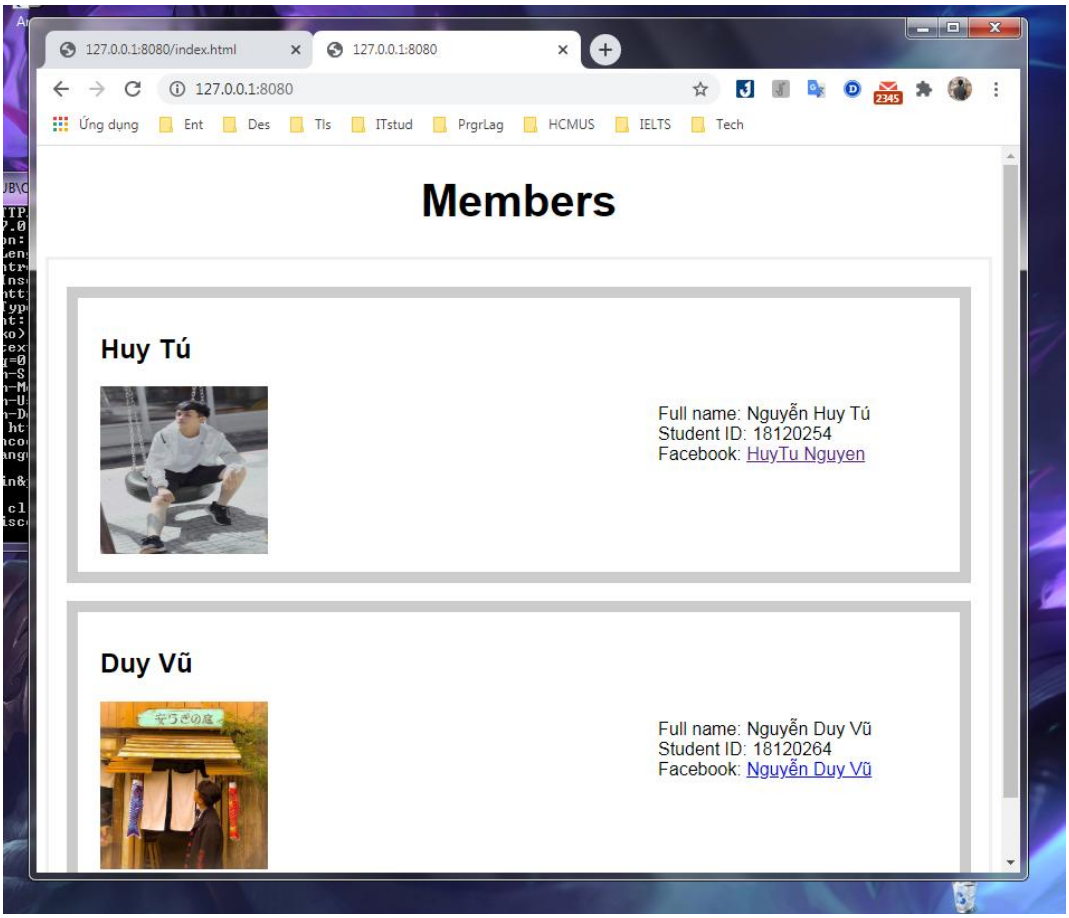
Tuy nhiên, chúng em đã lập trình sao cho khi truy cập vào địa chỉ IP của Server sẽ được điều hướng đến trang đăng nhập nên ta có thể nhập đơn giản là *127.0.0.1:8080* hoặc *localhost:8080*



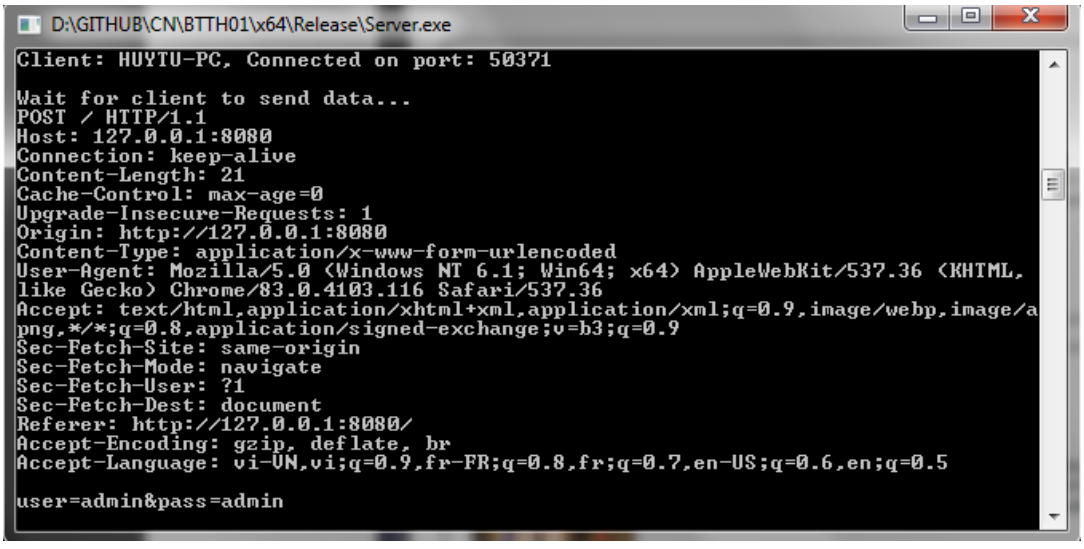
Tại màn hình console, thông điệp và trạng thái của Client gửi đến sẽ được xuất ra khi có Client kết nối đến Server.



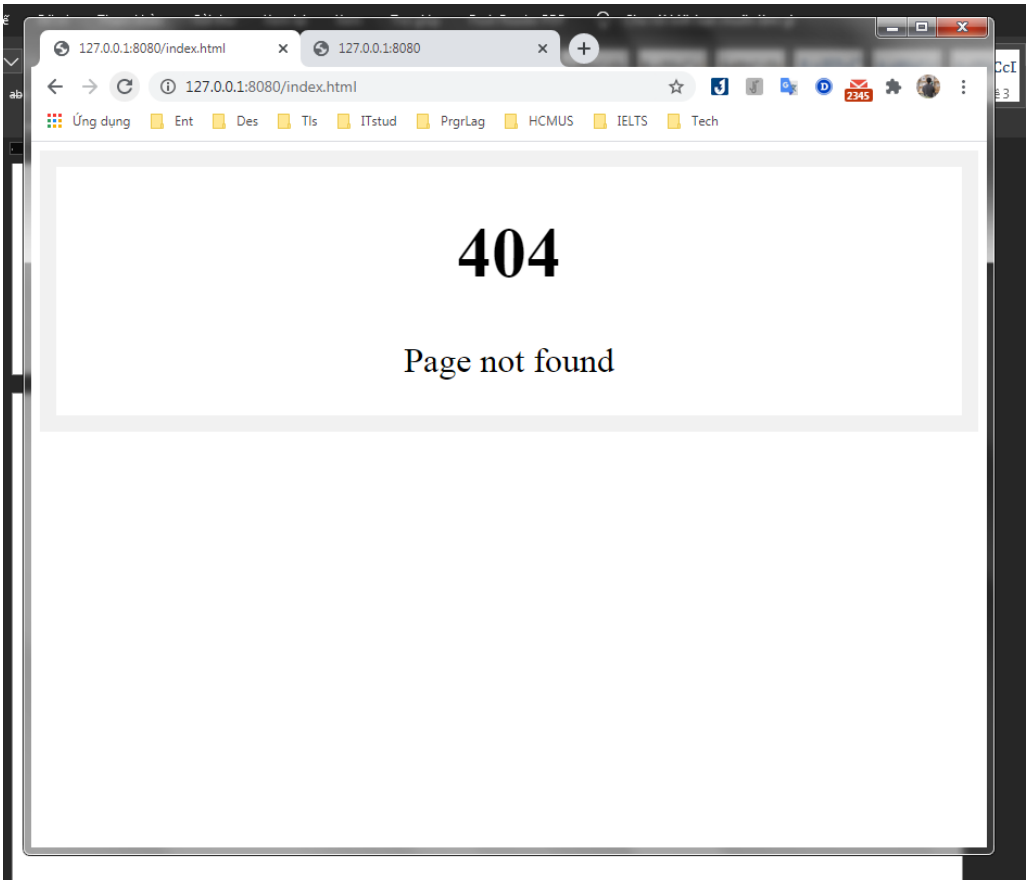
Tại trang đăng nhập được hiển thị, người dùng sẽ nhập username và password vào ô tương ứng. Nếu nhập đúng username là "*admin*", password là "*admin*", người dùng sẽ được điều hướng đến trang thông tin nhóm.



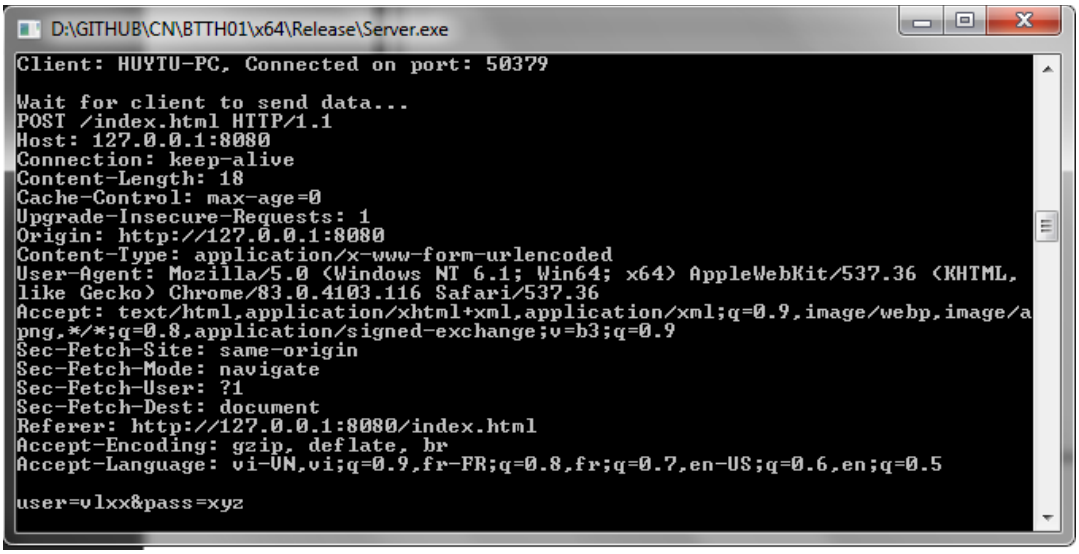
Thông tin HTTP POST Request của Client với tham số user và pass nhập đúng.



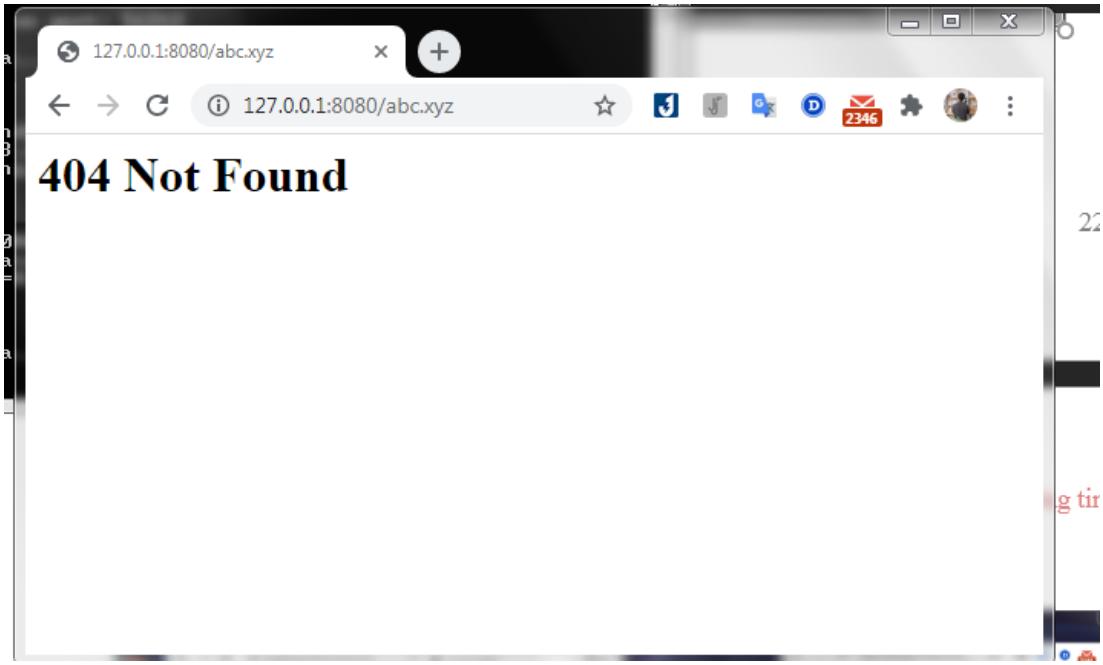
Nếu nhập sai, người dùng sẽ được điều hướng đến trang 404 Page Not Found.



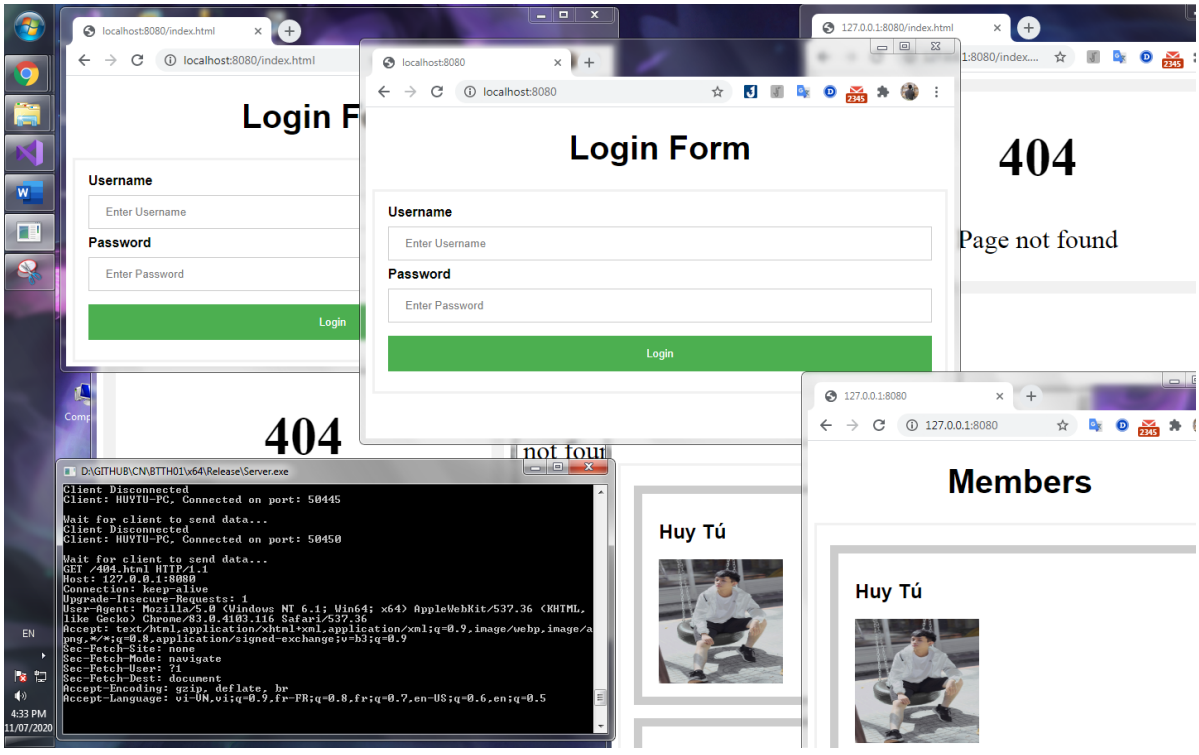
Thông tin HTTP POST Request của Client với tham số user và pass nhập sai.



Nếu người dùng yêu cầu một đường dẫn không tồn tại, ví dụ ở đây là */abc.xyz*, Server sẽ trả về thông báo "404 Not Found".

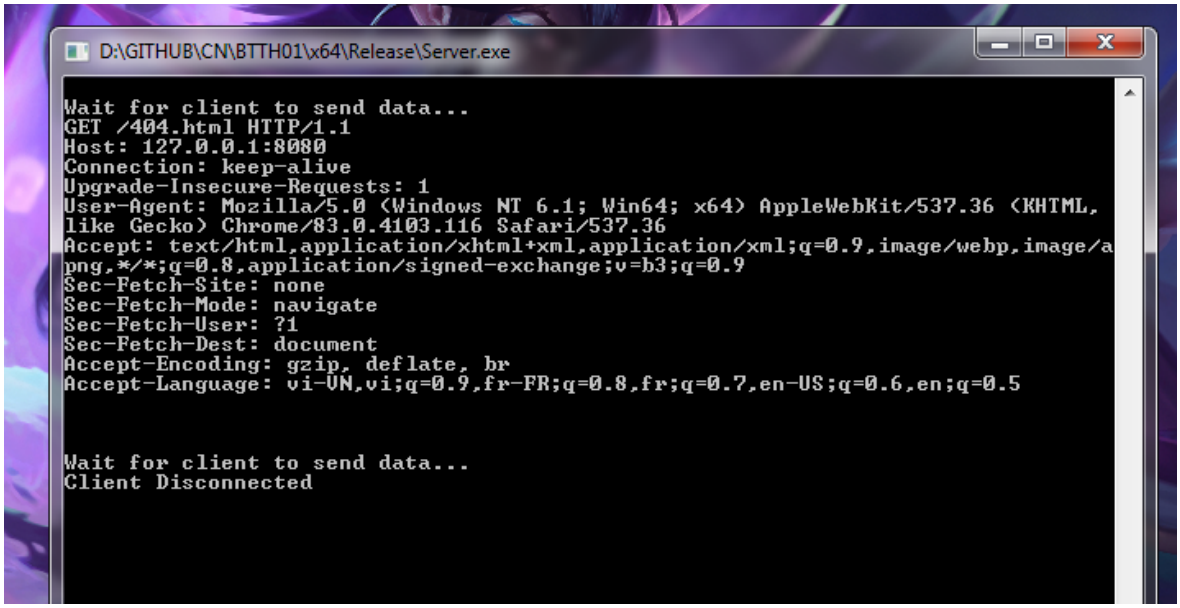


Server được lập trình theo cơ chế Multithread nên có thể có nhiều Client cùng truy cập vào Web và được Server xử lý riêng biệt.



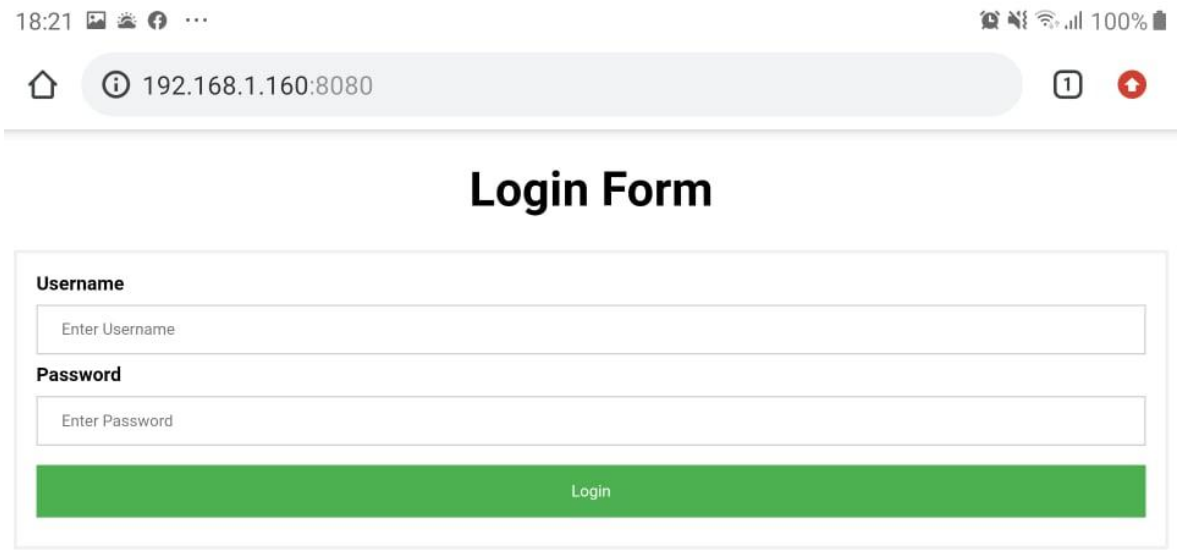
Khi Client ngắt kết nối thì Server vẫn tiếp tục hoạt động cho đến khi người quản trị viên tắt Server.

Màn hình Console xuất ra thông báo khi Client ngắt kết nối.



Các thiết bị khác trong cùng đường mạng muốn truy cập Web thì sử dụng lệnh `ipconfig` để lấy địa chỉ IPv4 của máy tính đang chạy Server thay cho địa chỉ *127.0.0.1*

Ví dụ: địa chỉ IPv4 của máy tính đang chạy Server là 192.168.1.160 thì khi truy cập Web bằng điện thoại, người dùng gõ URL: 192.168.1.160:8080



TÀI LIỆU THAM KHẢO

[1] Đại học Khoa Học Tự Nhiên, ĐHQG TP.HCM, HDTH_DA_Socket.

[2] Mai Văn Cường – Trần Trung Dũng – Trần Hồng Ngọc – Lê Ngọc Sơn – Lê Giang Thanh – Trương Thị Mỹ Dung – Đào Anh Tuấn, NXB Khoa Học và Kỹ Thuật, Giáo trình Mạng Máy Tính.

[2] eXecutive, Lập Trình Mạng Với Thư Viện Winsock Trên VC++, https://argron.wordpress.com/2012/03/13/lap-trinh-mang-voi-thu-vien-winsock-tren-vcpart-1/

[3] Sloan Kelly, Building a Web Server in C++ [VS 2017], https:/www.youtube.com/watch?v=YqEqjODUkWY&t=445s

[4] Nguyễn Quang Hải, NXB Giáo dục Việt Nam, Nhập môn HTML và CSS.

PHỤ LỤC

Thư mục Source chứ source code của chương trình.

Thư mục Release chứa file thực thi Server.exe.