TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN TPHCM

KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO MÔN HỌC
MẠNG MÁY TÍNH

ĐỒ ÁN

LẬP TRÌNH SOCKET
TẠO HTTP SERVER ĐƠN GIẢN

Nhóm sinh viên thực hiện

Nguyễn Đức Minh Trí	18120612
Triệu Trang Tòng	18120602

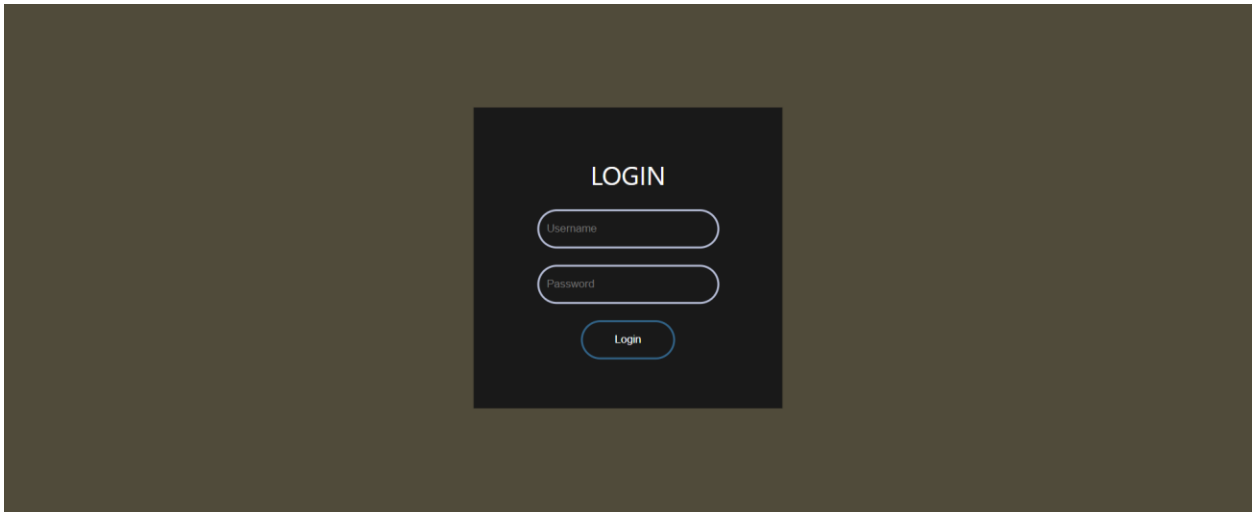
Giáo viên hướng dẫn
Lê Ngọc Sơn

I. Phân công công việc

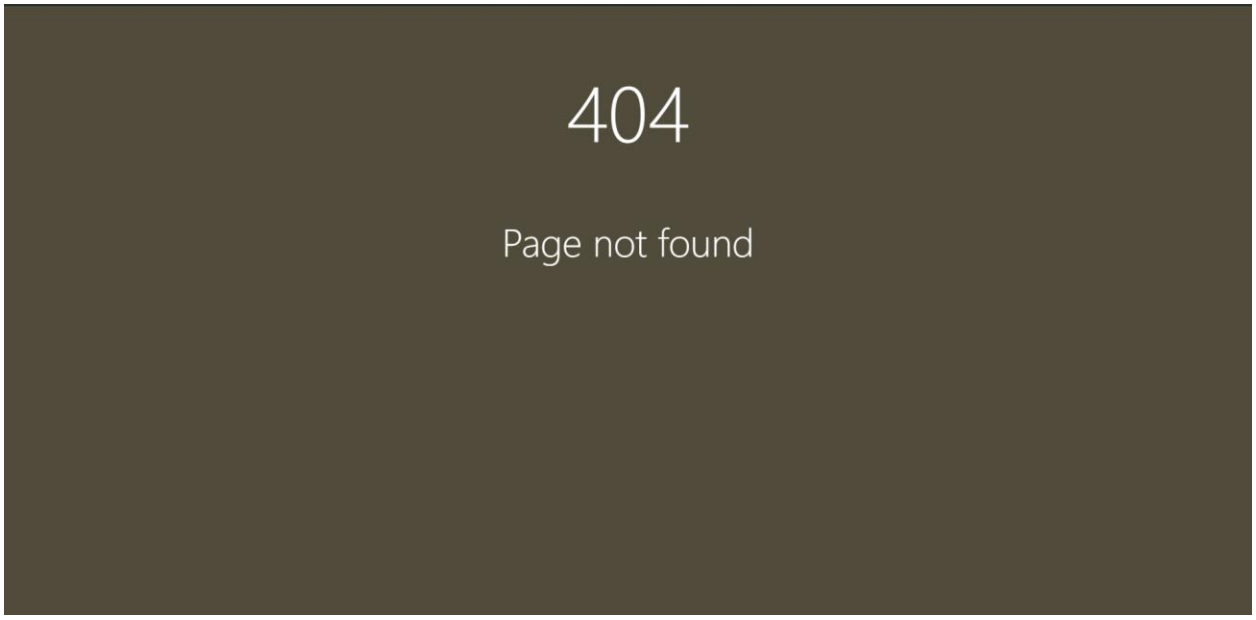
Công việc	Thành viên	Tỉ lệ thoàn thành
Front-end: html, css	Triệu Trang Tòng	100%
Back-end: xử lý yêu cầu	Nguyễn Đức Minh Trí	100%

II. Front-end

Thiết kế các giao diện trang đăng nhập, thông tin thành viên, 404. Giao diện như bên dưới:



Trường nhập: user có name = “username” và id = “username”. Password có name = “password” và id = “password” (trong html)



III. Back-end

Chương trình phía Server viết bằng ngôn ngữ C++. Sử dụng thư viện winsock2 trên Windows.

Các class trong chương trình:

Class	Chức năng
RequestParser	Tách Request, hỗ trợ lấy các thông tin từ Request
SimpleHTTPServer	Cài đặt chung cho một Server HTTP đơn giản
MyHTTPServer	Cài đặt HTTP Server theo yêu cầu của đề bài đưa ra.

Ta cùng đi vào chức năng cụ thể của từng class

1. RequestParser

- Thuộc tính:

```
string m_sMethod; // phương thức HTTP
string m_sURL; // URL request HTTP
string m_sVersion; // phiên bản HTTP
map<std::string, std::string> m_mHeader; // chứa Header + value
string m_sContent; // chưa nội dung HTTP request
```

- Phương thức chính:

```
// đưa dữ liệu của request vào, sau đó request sẽ được xử lý tách
void setRequest(const char* request);

// trả về phiên bản HTTP trong request
std::string getVersion();

// trả về phương thức HTTP: GET POST…
std::string getMethod();

// trả về URL trong request
std::string getURL();

// trả về nội dung của tên đưa vào header
std::string getHeader(std::string headerFieldName);

// trả về nội dung của request
std::string getContent();

// trả về loại nội dung của url. Ví dụ: /index.html -> text/html
// hỗ trợ vài loại nội dung: png, jpg, bmp.
std::string getContentType(std::string url);
```

2. SimpleHTTPServer

- Thuộc tính:

```
int m_nPort; // cổng lắng nghe của Server (mặc đinh HTTP: 80)
bool m_bServerRunning; // trạng thái Server
SOCKET m_listenSocket; // đối tượng SOCKET lắng nghe trên server
```

- Phương thức chính:

```
// Khởi tạo HTTP Server, port mặc định 80
SimpleHTTPServer(int port = 80);

// Hủy HTTP Server
~SimpleHTTPServer();

// Bắt đầu Server
int start();
```

B1. Lấy thông tin IP của Server và gán thông tin port

B2. Khởi tạo SOCKET cho việc kết nối từ client đến server

B3. Gán các thông số cho socket (IP, port = 80, ..)

B4. Lắng nghe trên SOCKET vừa tạo, số lượng kết nối hàng chờ tối đa 0x7fffffff

B5. Chấp nhận kết nối, tạo một SOCKET mới cho việc trao đổi dữ liệu, mỗi SOCKET mới thực hiện ở 1 thread. Thực hiện công việc ở hàm clientHandler.

B6. Nếu không có lỗi hay yêu cầu dừng, quay lại B5.

```
// Dừng Server
void stop();

// xử lý các yêu cầu của client, cần viết lại hàm khi kế thừa
virtual void clientHandler(SOCKET clientSocket) = 0;

// hàm hiển thị lỗi
int Error(const char* msg);
```

3. MyHTTPServer

Vì cài đặt Server HTTP đơn giản nên các phương thức gửi/nhận áp dụng với các file nhỏ. File lớn chia ra truyền nhiều lần chưa được được cài đặt.

- Phương thức:

```
// xử lý request và gửi đi respone
void clientHandler(SOCKET clientSocket)
```

B1. Thực hiện nhận dữ liệu từ clientSocket.
B2. Tạo đối tượng RequestParser phân tích request, nếu method = "GET", gọi doGetRequest. Ngược lại, method = "POST", gọi doPostRequest.
B3. Từ B2 ta được chuỗi respone, gửi chuỗi đến client.
B4. Lặp lại B1 đến khi không còn nhận được dữ liệu.

Vì cài đặt Server HTTP đơn giản nên ta bỏ qua một số trường header.

```
// xử lý request
string doGetRequest(RequestParser requestParser)
```

B1. Gán các trường mặc định vào string
HTTP/1.1 200 OK\r\n
Cache-control: no-cache, private\r\n
Connection: keep-alive\r\n
B2. Đọc URL. Nếu URL tồn tại đọc file theo địa chỉ. Ngược lại đọc 404.html (thay status code = 404 Not Found).
B3. Gán Content-Type, Content-Length, Content vào string ở B1
Content-Type: …\r\n
Content-Length: size(content)\r\n
\r\n
Data data data …
B4. Trả về kết quả chuỗi respone.

```
// xử lý respone
string doPostRequest(RequestParser requestParser)
```

B1. Lấy tên user và password từ request (trong content). Có dạng: username=*example*&password=*example*.

B2. So sánh user và password với *"admin"*. Nếu đúng đến **B3.**

B3. Gửi yêu cầu redirect đến trang info cho client

```
HTTP/1.1 302 Found\r\n
Connection: keep-alive\r\n
Location: /info.html\r\n\r\n
```

B4. Gửi yêu cầu redirect đến trang info cho client

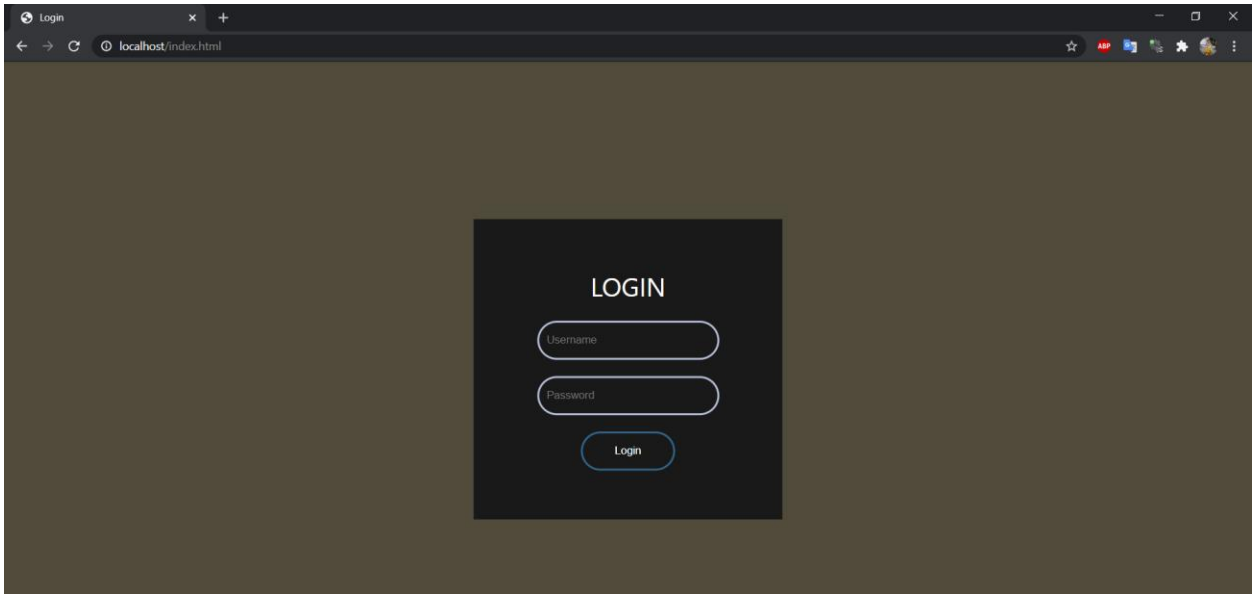
```
HTTP/1.1 302 Found\r\n
Connection: keep-alive\r\n
Location: /404.html\r\n\r\n
```

B5. Trả về chuỗi request.

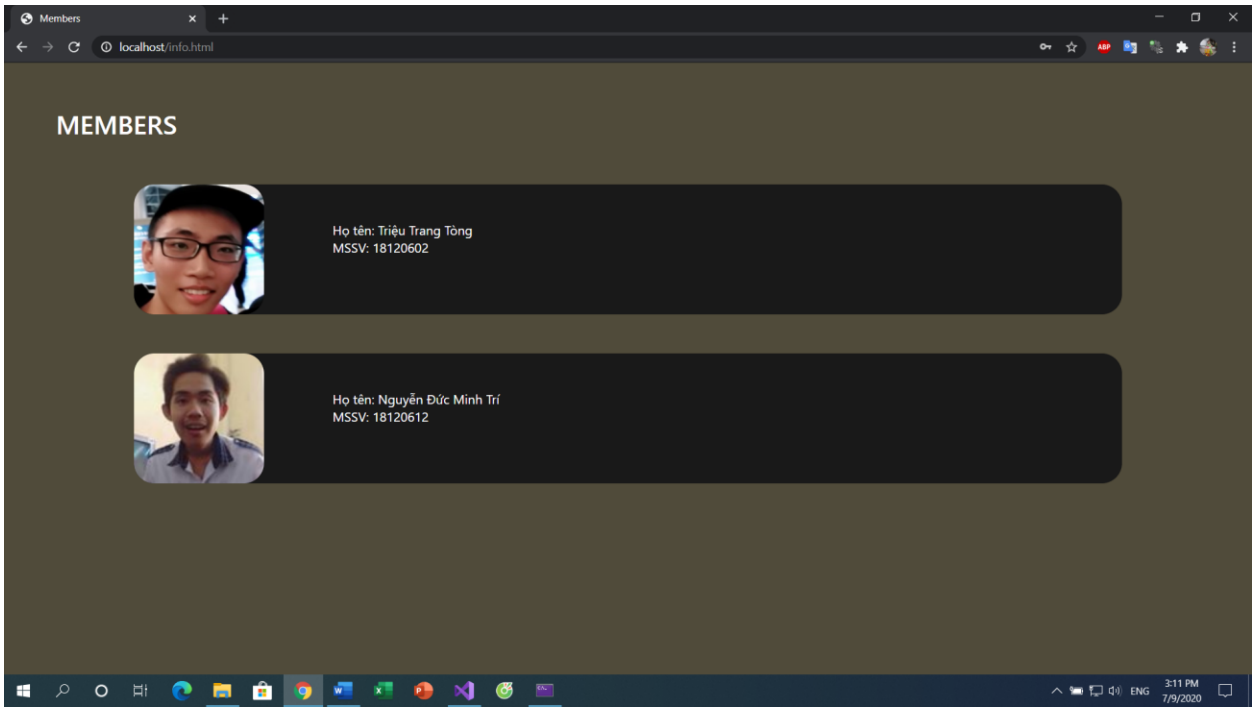
IV. Chạy thử chương trình

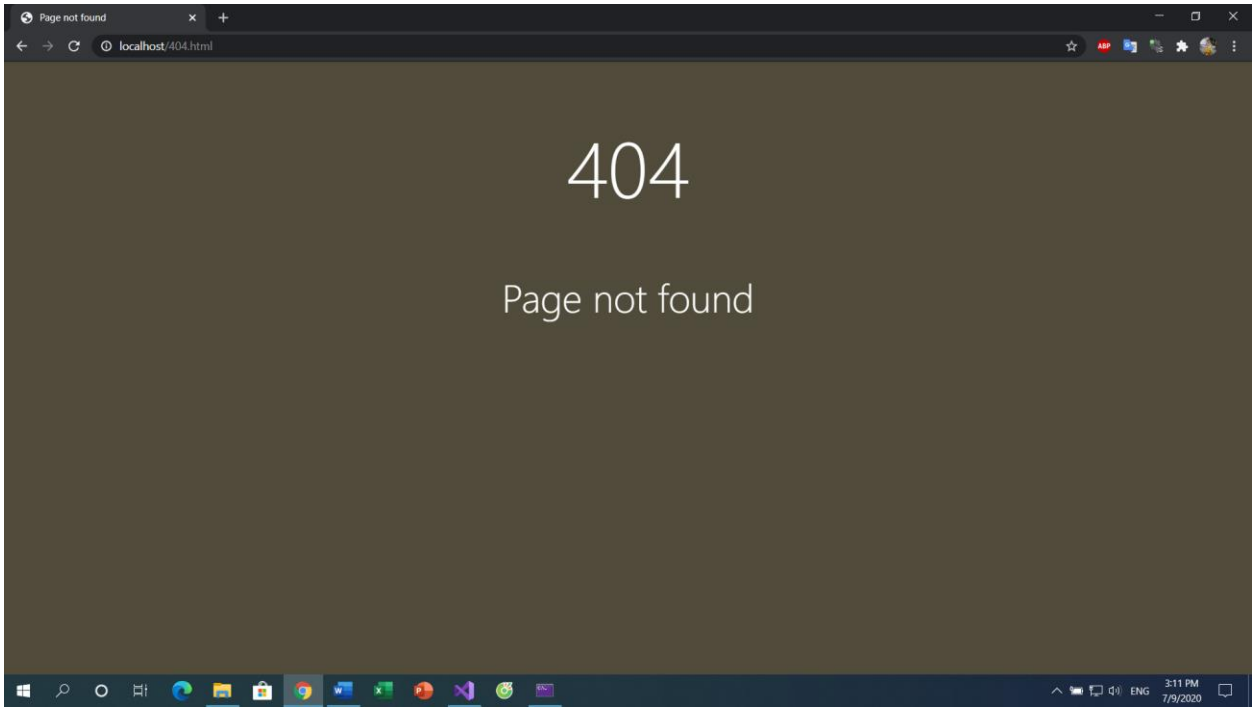
B1. Chạy chương trình phía Server: server.exe trong thư mục.

B2. Nhập trên trình duyệt cú pháp "localhost:80/index.html" để truy cập *(Nếu chạy trình duyệt nằm trên Server thì có thể nhập cú pháp "ipServer:80/index.html", với ipServer là ip máy Server của bạn)*



B3. Nhập user: admin và pass: admin, trình duyệt chuyển hướng đến trang /info.html. Nếu nhập sai thì trang 404.html sẽ được hiển thị.


Members
localhost/info.html
MEMBERS
Họ tên: Triệu Trang Tòng
MSSV: 18120602
Họ tên: Nguyễn Đức Minh Trí
MSSV: 18120612
ENG
3:11 PM
7/9/2020


Page not found
localhost/404.html
404
Page not found
ENG
3:11 PM
7/9/2020