

Cách thức thực hiện ứng dụng

--- Hàm Init : Khởi tạo Socket server, bind và listen tại port

-Khởi tạo Winsock để có thể khởi tạo và sử dụng socket bằng hàm WSStartup()

```
if (WSAStartup(MAKEWORD(2, 2), &wsadata) != 0)
{
    cerr << "Khoi tao winsock that bai \n";
    return false;
}

if (LOBYTE(wsadata.wVersion) != 2 || HIBYTE(wsadata.wVersion) != 2)
{
    cerr << "Sai phien ban winsock \n";
    return false;
}

return true;
```

-Khởi tạo 1 socket bằng hàm socket()

```
// Khoi tao listening socket
m_socket = socket(AF_INET, SOCK_STREAM, 0);
```

-Thực hiện chuyển địa chỉ IP từ dạng presentation sang network, để có thể bind socket vào địa chỉ đó

```
inet_pton(AF_INET, m_ipAddress, &hint.sin_addr);

if (bind(m_socket, (sockaddr*)&hint, sizeof(hint)) == SOCKET_ERROR)
```

-Thực hiện listen tại socket đó

```
// Tell Winsock the socket is for listening
if (listen(m_socket, SOMAXCONN) == SOCKET_ERROR)
```

-Tạo 1 fd_set để có thể mở nhiều kết nối tới server, và server có thể trả response lại cho tất cả các kết nối với mình

```
fd_set m_master;
```

```
typedef struct fd_set {
    u_int  fd_count;
    SOCKET fd_array[FD_SETSIZE];
} fd_set, FD_SET, *PFD_SET, *LPFD_SET;
```

-Khởi tạo m_master bằng cách gán tất cả các bit của các file descriptor = 0. Sau đó thêm listening socket vào mảng socket trong m_master

```
FD_ZERO(&m_master);

FD_SET(m_socket, &m_master);
```

--- Hàm Run() : nếu có client muốn kết nối, mở socket mới, accept client và gửi nhận dữ liệu với client từ socket đó

-Sử dụng hàm select, chương trình sẽ bị dừng ở đây cho tới khi có client muốn kết nối hay gửi dữ liệu cho server, socketCount sẽ cho ta biết có bao nhiêu socket muốn liên lạc hay gửi dữ liệu

```
int socketCount = select(0, &copy, nullptr, nullptr, nullptr);
```

-Nếu socketCount > 0, chạy vòng lặp kiểm tra tất cả các socket muốn kết nối hay gửi dữ liệu cho server

-Ta kiểm tra đây là kết nối mới tới server hay là truyền nhận dữ liệu

+) Nếu là kết nối mới, tạo 1 SOCKET client để accept kết nối từ client, sau đó thêm SOCKET client này vào mảng socket trong m_master

```
for (int i = 0; i < socketCount; i++)
{
    SOCKET sock = copy.fd_array[i];

    if (sock == m_socket)
    {
        SOCKET client = accept(m_socket, nullptr, nullptr);

        FD_SET(client, &m_master);

        onClientConnected(client);
    }
}
```

+) Nếu là truyền nhận dữ liệu, tạo mảng buf để nhận dữ liệu từ phía client. Nếu không nhận được dữ liệu thì đóng socket, đồng thời xóa socket đó khỏi mảng socket trong m_master. Còn nếu nhận được dữ liệu từ client, sẽ thực hiện gửi trả dữ liệu cho Client qua hàm onMessageReceived

```
char buf[4096];
ZeroMemory(buf, 4096);

int bytesIn = recv(sock, buf, 4096, 0);
if (bytesIn <= 0)
{
    closesocket(sock);
    FD_CLR(sock, &m_master);
}
else
{
    onMessageReceived(sock, buf, bytesIn);
}
```

----- Hàm onMessageReceived() : xử lý dữ liệu client gửi, gửi trả dữ liệu tương ứng cho client

-Tách msg từ client thành các từ cách nhau bởi khoảng trắng, lưu vào vector parsed

```
void WebServer::onMessageReceived(int clientSocket, const char* msg, int length)
{
    cout << msg << endl;
    std::istringstream iss(msg);
    std::vector<std::string> parsed((std::istream_iterator<std::string>(iss), std::istream_iterator<std::string>()));
}
```

-Ví dụ khi ta truy cập từ Chrome localhost:8080, server lắng nghe tại port 8080, địa chỉ 0.0.0.0, parsed sẽ như sau

A screenshot of a web browser's address bar. It shows navigation icons (back, forward, refresh) on the left, followed by an information icon and the text 'localhost:8080'.

```
GET
/
HTTP/1.1
Host:
localhost:8080
Connection:
keep-alive
Upgrade-Insecure-Requests:
1
User-Agent:
Mozilla/5.0
(Windows
NT
10.0;
Win64;
x64)
AppleWebKit/537.36
(KHTML,
like
Gecko)
Chrome/83.0.4103.106
Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site:
none
Sec-Fetch-Mode:
navigate
Sec-Fetch-User:
?1
Sec-Fetch-Dest:
document
Accept-Encoding:
gzip,
deflate,
br
Accept-Language:
en-US,en;q=0.9
HTTP/1.1 200 OK
Cache-Control: no-cache, private
Content-Type: text/html
Content-Length: 4571
```

-Ta quan tâm tới phần tử đầu tiên trong parsed

```
else if (parsed[0] == "GET")
{
    // Lay filename qua parsed[1], co dang /name.html
    string filename = parsed[1].substr(1);
    if (filename == "")
    {
        filename = "index.html";
    }
    response = getResponseToClient(filename, 200);
}
```

-Ta sẽ lấy filename mà client request, sau đó đọc file rồi trả html cho client. Nếu parsed[1] có dạng '/', ta sẽ hiểu là client muốn nhận file index.html

```

string pathFile = ".\\wwwroot/" + filename;
std::ifstream f(pathFile);
string fileContent;
if (f.good())
{
    std::string str((std::istreambuf_iterator<char>(f)), std::istreambuf_iterator<char>());
    fileContent = str;
}
else
{
    statusCode = 404;
}

```

-Ta mở file html từ wwwroot/filename. Nếu mở đọc file thành công, ta lưu nội dung file html vào biến content, nếu ko mở được sẽ chuyển statusCode thành 404

-Ta nhận response dựa trên status code và file

```

if (statusCode == 200)
{
    return get200Response(fileContent);
}
else if (statusCode == 301)
{
    return get301Response(fileContent);
}
else
{
    return get404Response(fileContent);
}

```

Chi tiết:

Get200Response: tham số là nội dung file html, gửi status line HTTP/1.1 200 OK cho biết server đã tìm được file yêu cầu và sẽ gửi cho client

```

string get200Response(string fileContent)
{
    ostream oss;
    oss << "HTTP/1.1 200 OK\r\n";
    oss << "Cache-Control: no-cache, private\r\n";
    oss << "Content-Type: text/html\r\n";
    oss << "Content-Length: " << fileContent.size() << "\r\n";
    oss << "\r\n";
    oss << fileContent;
    string result = oss.str();
    return result;
}

```

Get301Response : chương trình sẽ redirect qua location info.html, nên sẽ trả về status line 301 Moved Permanently kèm Location để redirect

```

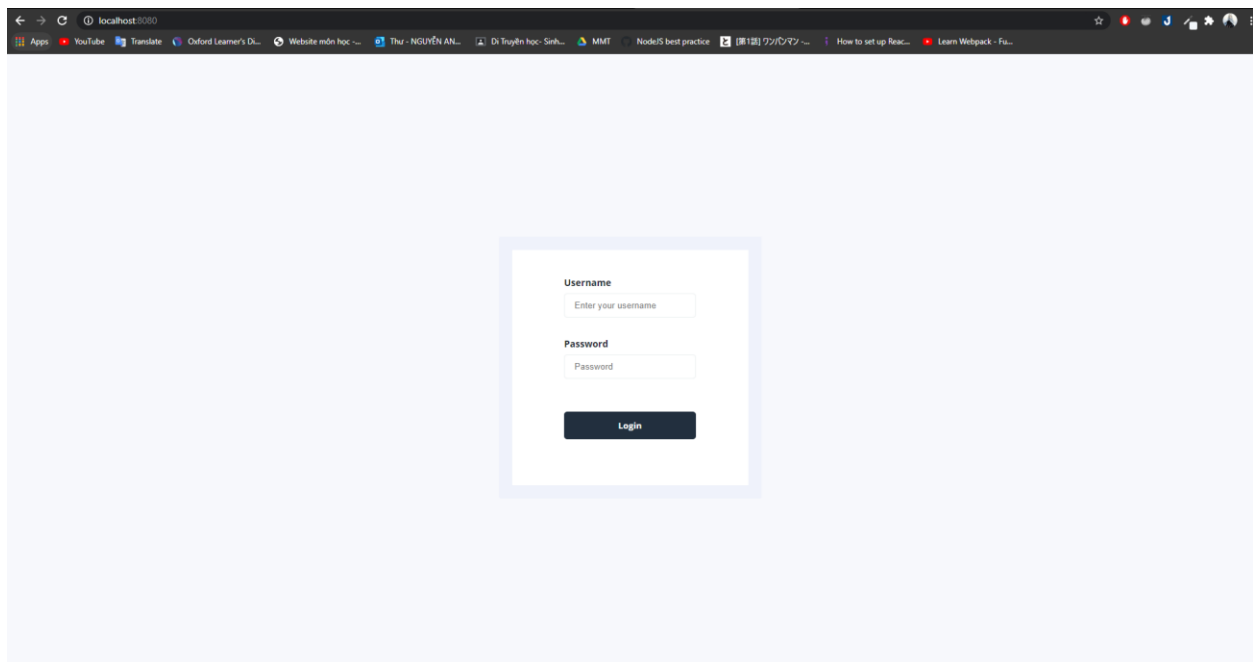
string get301Response(string fileContent)
{
    ostream oss;
    oss << "HTTP/1.1 301 Moved Permanently\r\n";
    oss << "Location: /info.html\r\n";
    oss << "Cache-Control: no-cache, private\r\n";
    oss << "Content-Type: text/html\r\n";
    oss << "Content-Length: " << fileContent.size() << "\r\n";
    oss << "\r\n";
    oss << fileContent;
    string result = oss.str();
    return result;
}

```

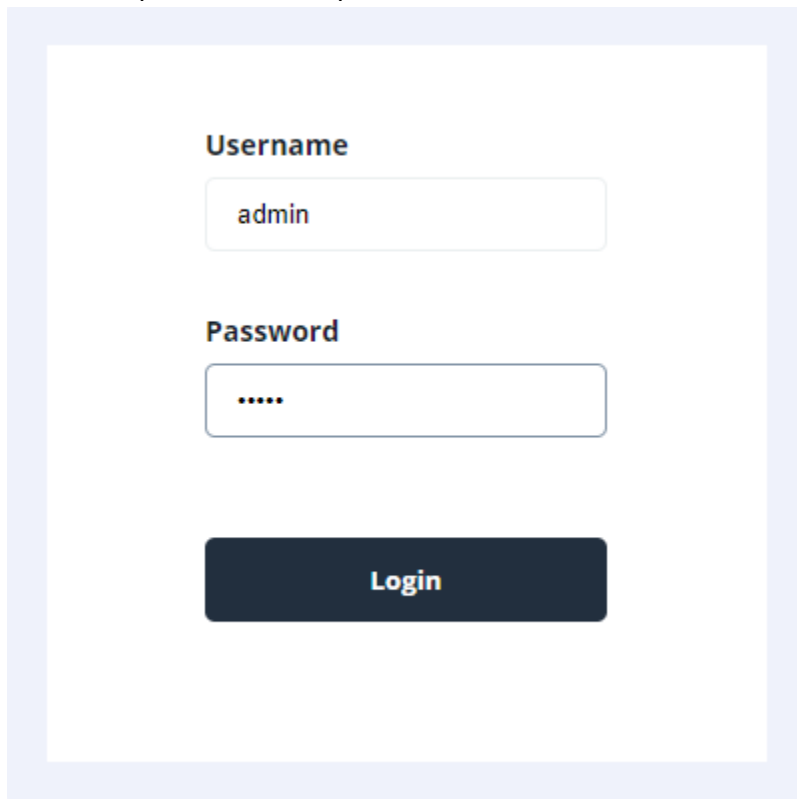
Get404Response: chương trình trả về status code 404 Not found, load file 404.html

```
string get404Response(string fileContent)
{
    stringstream oss;
    oss << "HTTP/1.1 404 Not Found\r\n";
    oss << "Cache-Control: no-cache, private\r\n";
    oss << "Content-Type: text/html\r\n";
    oss << "Content-Length: " << fileContent.size() << "\r\n";
    oss << "\r\n";
    oss << fileContent;
    string result = oss.str();
    return result;
}
```

Khi client gửi request lên port 8080, server trả về nội dung file index.html, hiển thị lên Chrome như sau



Client nhập username và password vào form.



Setup form như hình để khi nhấn Login, sẽ gửi request POST lên server, kèm body chứa username và password

```
<form action="http://localhost:8080" method="POST" class="form-login">
  <div class="input-field"> <label for="username">Username</label> <input name="username" type="username"
    placeholder="Enter your username" required autocomplete="off" /> <label
    for="password">Password</label> <input name="password" type="password" placeholder="Password"
    required /> </div>
  <div class="form-footer"> <button class="btn" type="submit">Login</button> </div>
</form>
```

-Xem nội dung vector parsed, ta thấy username và password được lưu ở phía cuối vector

```
POST / HTTP/1.1
Host: localhost:8080
Connection: keep-alive
Content-Length: 31
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
Origin: http://localhost:8080
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/83.0.4103.106 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: http://localhost:8080/
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9

username=khiem&password=deptrai
```

-Ta lấy được chuỗi username và password


```

if (parsed[0] == "POST")
{
    // Lay username va password
    string ttDangNhap = parsed[parsed.size() - 1];
    int foundAmp = ttDangNhap.find('&');
    int startUsername = ttDangNhap.find("=") + 1;
    string username = ttDangNhap.substr(startUsername, foundAmp - startUsername);
    string password = ttDangNhap.substr(foundAmp + 10, ttDangNhap.size() - foundAmp - 10);
}

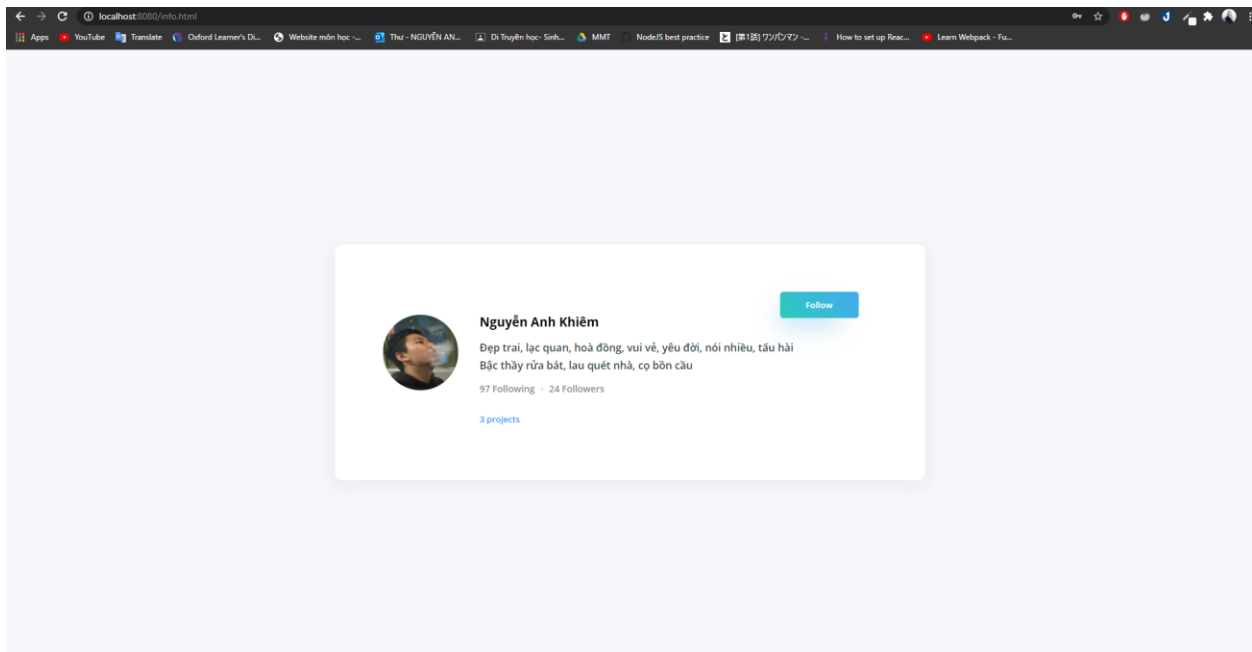
```

-So sánh username và password với “admin”, nếu đúng thì redirect qua location /info.html

```

// Nếu đăng nhập đúng
if (username == "admin" && password == "admin")
{
    // Redirect qua /info.html
    response = responseToClient("info.html", 301);
}

```



-Nếu không đúng thì trả về nội dung file 404.html

```

response = responseToClient("404.html", 404);

```



OOPS! PAGE NOT FOUND

404

WE ARE SORRY, BUT THE PAGE YOU REQUESTED WAS
NOT FOUND