

BÁO CÁO ĐỀ ÁN 1 – SOCKET

Lớp: 19CTT2

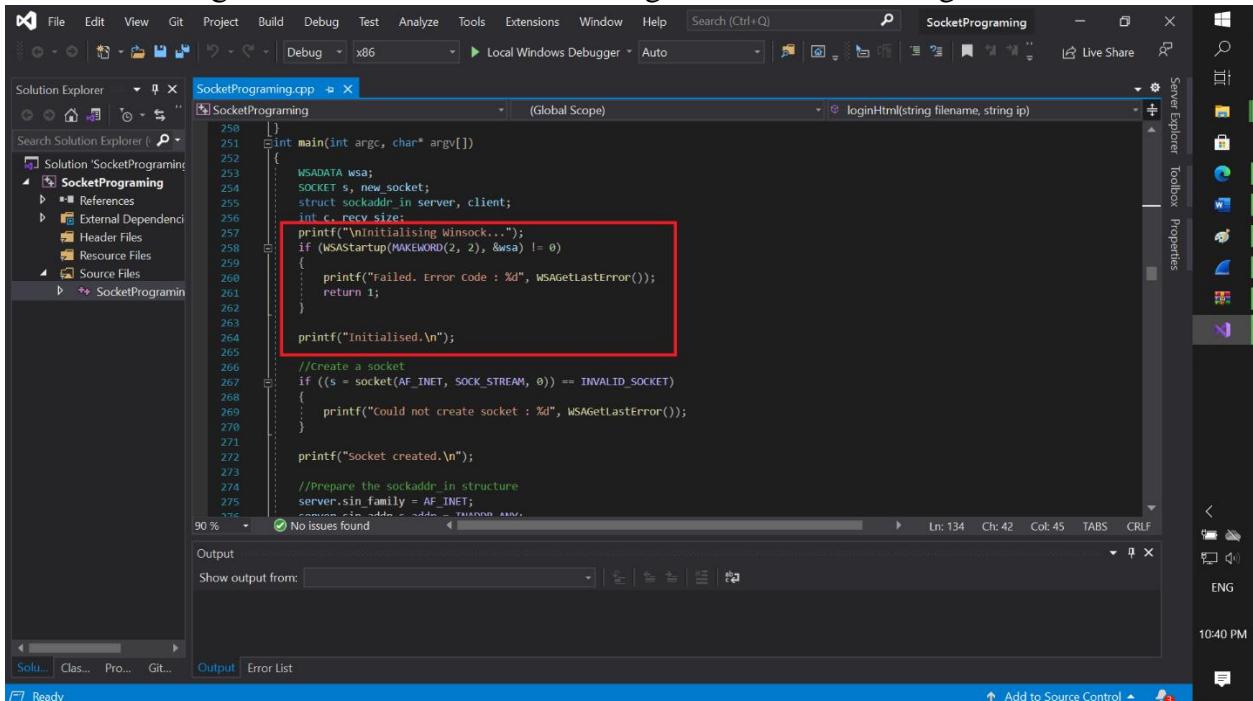
1/ Thành viên:

Họ và tên	MSSV
Nguyễn Huy Tùng	19120422
Lục Minh Bửu	19120462

2/ Nội dung báo cáo:

2.1/ Ý tưởng triển khai ứng dụng:

- Khi khởi chạy chương trình, chương trình sẽ in ra cửa sổ console thông báo đang khởi tạo Winsock (Initialising Winsock...). Sau đó dùng hàm WSAStartup() để tạo Winsock. Nếu khởi tạo thất bại sẽ in ra cửa sổ thông báo thất bại cùng với mã lỗi và kết thúc chương trình. Nếu khởi tạo thành công sẽ in ra cửa sổ thông báo Initialised.



The screenshot shows the Microsoft Visual Studio IDE interface. The main window displays a C++ source file named "SocketProgramming.cpp". The code is as follows:

```
250 } 251 int main(int argc, char* argv[])
252 {
253     WSADATA wsa;
254     SOCKET s, new_socket;
255     struct sockaddr_in server, client;
256     int c, recv_size;
257     printf("\nInitialising Winsock...");
258     if (WSAStartup(MAKEWORD(2, 2), &wsa) != 0)
259     {
260         printf("Failed. Error code : %d", WSAGetLastError());
261         return 1;
262     }
263     printf("Initialised.\n");
264
265     //create a socket
266     if ((s = socket(AF_INET, SOCK_STREAM, 0)) == INVALID_SOCKET)
267     {
268         printf("could not create socket : %d", WSAGetLastError());
269     }
270
271     printf("socket created.\n");
272
273     //Prepare the sockaddr_in structure
274     server.sin_family = AF_INET;
275     server.sin_port = htons(4444);
276     server.sin_addr.s_addr = htonl(INADDR_ANY);
```

A red box highlights the initial initialization code from line 257 to line 263. The status bar at the bottom right shows the time as 10:40 PM.

- Chương trình tạo ra 1 socket bằng hàm socket. Nếu khởi tạo thất bại sẽ in ra cửa sổ thông báo tạo thất bại cùng với mã lỗi và kết thúc chương trình. Nếu khởi tạo thành công sẽ in ra cửa sổ thông báo Socket created.

```
259     {
260         printf("Failed. Error Code : %d", WSAGetLastError());
261         return 1;
262     }
263
264     printf("Initialised.\n");
265
266     //Create a socket
267     if ((s = socket(AF_INET, SOCK_STREAM, 0)) == INVALID_SOCKET)
268     {
269         printf("could not create socket : %d", WSAGetLastError());
270     }
271
272     printf("Socket created.\n");
273
274     //Prepare the sockaddr_in structure
275     server.sin_family = AF_INET;
276     server.sin_addr.s_addr = INADDR_ANY;
277     server.sin_port = htons(8888);
278
279     //Bind
280     if (bind(s, (struct sockaddr*)&server, sizeof(server)) == SOCKET_ERROR)
281     {
282         printf("Bind failed with error code : %d", WSAGetLastError());
283         exit(EXIT_FAILURE);
284     }
285
286     puts("Bind done");
287
288     //Listen to incoming connections
289     listen(s, 3);
290
291     //Accept and incoming connection
292     puts("Waiting for incoming connections... \n");
293 }
```

3. Chương trình sẽ khởi tạo một vài thông số và chạy hàm bind để đăng ký tên cục bộ cho socket. Nếu khởi tạo thất bại sẽ in ra cửa sổ thông báo tạo thất bại cùng với mã lỗi và kết thúc chương trình. Nếu khởi tạo thành công sẽ in ra cửa sổ thông báo Bind done.

```
268     {
269         printf("Could not create socket : %d", WSAGetLastError());
270     }
271
272     printf("Socket created.\n");
273
274     //Prepare the sockaddr_in structure
275     server.sin_family = AF_INET;
276     server.sin_addr.s_addr = INADDR_ANY;
277     server.sin_port = htons(8888);
278
279     //bind
280     if (bind(s, (struct sockaddr*)&server, sizeof(server)) == SOCKET_ERROR)
281     {
282         printf("Bind failed with error code : %d", WSAGetLastError());
283         exit(EXIT_FAILURE);
284     }
285
286     puts("Bind done");
287
288     //Listen to incoming connections
289     listen(s, 3);
290
291     //Accept and incoming connection
292     puts("Waiting for incoming connections... \n");
293 }
```

4. Chương trình sẽ dùng hàm listen để socket lắng nghe kết nối (ở đây tối đa là 3 clients) và hiện lên thông báo Waiting for incoming connections... ở cửa sổ console.

```

280     if (bind(s, (struct sockaddr*)&server, sizeof(server)) == SOCKET_ERROR)
281     {
282         printf("Bind failed with error code : %d", WSAGetLastError());
283         exit(EXIT_FAILURE);
284     }
285
286     puts("Bind done");
287
288     //Listen to incoming connections
289     listen(s, 3);
290
291     //Accept incoming connection
292     puts("Waiting for incoming connections...\n");
293
294     c = sizeof(struct sockaddr_in);
295
296     while ((new_socket = accept(s, (struct sockaddr*)&client, &c)) != INVALID_SOCKET)
297     {
298         char* ipAddr = new char[15];
299         if (strstr(inet_ntoa(client.sin_addr), "127.0.0.1") != NULL) strcpy(ipAddr, "127.0.0.1");
300         else strcpy(ipAddr, getIpv4().c_str());
301         char* request = new char[2000];
302         if ((recv_size = recv(new_socket, request, 2000, 0)) == SOCKET_ERROR)
303         {
304             puts("Error");
305         }
306     }

```

5. Chương trình khởi tạo biến new_socket và gán giá trị cho nó bằng giá trị trả về của hàm accept(). Hàm này sẽ chấp nhận kết nối của 1 client, nếu có lỗi sẽ trả về INVALID_SOCKET. Ngược lại, vòng lặp sẽ được thực hiện. Trong vòng lặp này:
 - 5.1 Chương trình sẽ khởi tạo mảng char* ipAddr gồm 15 phần tử. Sau đó chương trình sẽ chuyển địa chỉ internet của client về dạng string bằng hàm inet_ntoa() rồi kiểm tra xem trong địa chỉ internet của client có chuỗi “127.0.0.1” hay không bằng hàm strstr(). Nếu có thì gán copy chuỗi “127.0.0.1” vào ipAddr bằng hàm strcpy(). Nếu không thì sẽ dùng hàm getIpv4() để lấy địa chỉ IP của client rồi chuyển sang dạng char* bằng hàm c_str() rồi copy vào ipAddr bằng hàm strcpy().

```

289     listen(s, 3);
290
291     //Accept and incoming connection
292     puts("Waiting for incoming connections...\n");
293
294     c = sizeof(struct sockaddr_in);
295
296     while ((new_socket = accept(s, (struct sockaddr*)&client, &c)) != INVALID_SOCKET)
297     {
298         char* ipAddr = new char[15];
299         if (strncpy(_inet_ntoa(client.sin_addr), "127.0.0.1") != NULL) strcpy(ipAddr, "127.0.0.1");
300         else strcpy(ipAddr, getIpv4().c.str());
301
302         char* request = new char[2000];
303         if ((recv_size = recv(new_socket, request, 2000, 0)) == SOCKET_ERROR)
304         {
305             puts("Error");
306         }
307         else
308         {
309             request[recv_size] = '\0';
310             cout << request << "\n";
311             if (strstr(request, "GET / HTTP/1.1") != NULL)
312             {
313                 char headers[83];
314                 sprintf(headers, "HTTP/1.1 301 Moved Permanently\r\n"
315                         "Location: http://%s:8888/index.html\r\n"
316                         "Content-Type: text/html\r\n");
317             }
318         }
319     }
320
321     request[recv_size] = '\0';
322     cout << request << "\n";
323     if (strstr(request, "GET / HTTP/1.1") != NULL)
324     {
325         char headers[83];
326         sprintf(headers, "HTTP/1.1 301 Moved Permanently\r\n"
327                 "Location: http://%s:8888/index.html\r\n"
328                 "Content-Type: text/html\r\n");
329     }
330 }

```

5.1.1 Hàm getIpv4() sẽ gọi hàm ipconfig và lưu kết quả trả về vào file ip.txt trong ổ đĩa D, sau đó mở file này lên và đọc từng dòng vào chuỗi data. Sau đó tìm trong data dòng có dữ liệu: “Default Gateway : 192.168.1.1” rồi dùng biến I đánh dấu tại đó. Sau đó trừ i đi 68 để đến được vị trí của địa chỉ Ipv4 của máy client rồi dùng vòng lặp để đọc địa chỉ này vào chuỗi ip được khai báo ở trên rồi ra về giá trị ip.

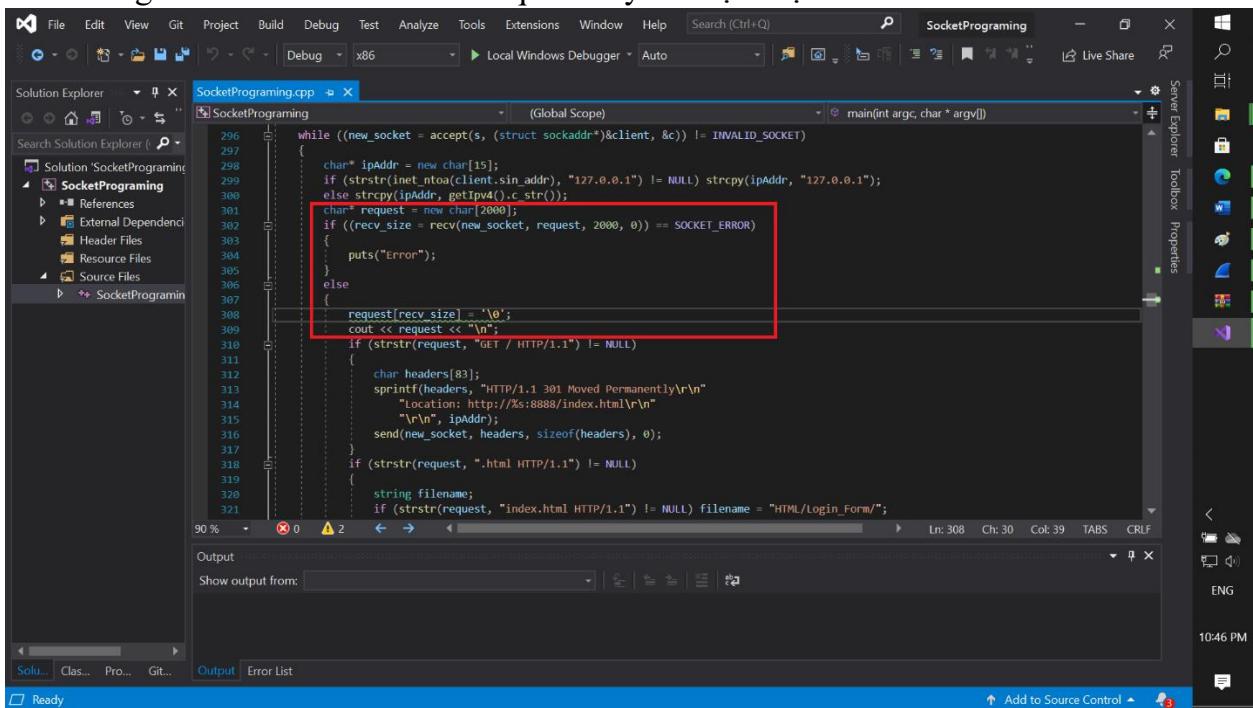
```

16     string getIpv4()
17     {
18         system("ipconfig > D:/ip.txt");
19         string line, data, ip;
20         ifstream f;
21         f.open("D:/ip.txt", ifstream::in);
22         if (!f)
23         {
24             cout << "Cannot open file\n";
25             return "";
26         }
27         while (!f.eof())
28         {
29             getline(f, line);
30             data += line + "\n";
31         }
32         size_t i = data.find("Default Gateway . . . . . : 192.168.1.1");
33         i -= 68;
34         do
35         {
36             ip += data[i];
37             i++;
38         } while (data[i] != '\n');
39         f.close();
40         return ip;
41     }

```

5.2 Khởi tạo mảng động request có kiểu char* gồm 2000 phần tử.

5.3 Khởi chạy hàm recv. Hàm này sẽ nhận dữ liệu từ client và truyền vào mảng request đã khai báo và trả về kích thước của dữ liệu nhận được khi không có lỗi và trả về giá trị SOCKET_ERROR khi xảy ra lỗi, ta gán giá trị trả về này vào biến recv_size. Nếu biến này nhận giá trị SOCKET_ERROR thì in ra cửa sổ console thông báo lỗi. Nếu không có lỗi, chương trình sẽ gán ký hiệu kết thúc chuỗi vào sau phần dữ liệu của mảng request. Lúc này ta đã có mảng request lưu toàn bộ yêu cầu từ phía client. Sau đó chương trình sẽ in ra màn hình request này và thực hiện các bước sau:



```

296     while ((new_socket = accept(s, (struct sockaddr*)&client, &c)) != INVALID_SOCKET)
297     {
298         char* ipAddr = new char[15];
299         if (strstr(inet_ntoa(client.sin_addr), "127.0.0.1") != NULL) strcpy(ipAddr, "127.0.0.1");
300         else strcpy(ipAddr, getIpv4().c_str());
301         char* request = new char[2000];
302         if ((recv_size = recv(new_socket, request, 2000, 0)) == SOCKET_ERROR)
303         {
304             puts("Error");
305         }
306         else
307         {
308             request[recv_size] = '\0';
309             cout << request << "\n";
310             if (strstr(request, "GET / HTTP/1.1") != NULL)
311             {
312                 char headers[83];
313                 sprintf(headers, "HTTP/1.1 301 Moved Permanently\r\n"
314                         "Location: http://%s:8888/index.html\r\n"
315                         "\r\n", ipAddr);
316                 send(new_socket, headers, sizeof(headers), 0);
317             }
318             if (strstr(request, ".html HTTP/1.1") != NULL)
319             {
320                 string filename;
321                 if (strstr(request, "index.html HTTP/1.1") != NULL) filename = "HTML/Login_Form/";

```

5.3.1 Chương trình sẽ dùng hàm strstr để kiểm tra xem trong chuỗi có dòng “GET / HTTP/1.1” hay không. Nếu có thì chương trình sẽ khai báo mảng char headers lưu thông tin của phần header của gói tin mà server phải gửi về cho client. Header này yêu cầu client chuyển sang trang index.html có địa chỉ ipAddr:8888/index.html. Sau đó server gửi gói tin này về cho client bằng hàm send().

File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help Search (Ctrl+Q) SocketProgramming Live Share

Solution Explorer

Search Solution Explorer (P)

Solution 'SocketProgramming' (1 item)

- SocketProgramming (1 item)

- External Dependencies
- Header Files
- Resource Files
- Source Files
- SocketProgramm

SocketProgramming.cpp (Global Scope)

```
302     if ((recv_size = recv(new_socket, request, 2000, 0)) == SOCKET_ERROR)
303     {
304         puts("Error");
305     }
306     else
307     {
308         request[recv_size] = '\0';
309         cout << request << endl;
310         if (strstr(request, "GET / HTTP/1.1") != NULL)
311         {
312             char headers[83];
313             sprintf(headers, "HTTP/1.1 301 Moved Permanently\r\n"
314                     "Location: http://%s:8888/index.html\r\n"
315                     "\r\n", ipAddr);
316             send(new_socket, headers, sizeof(headers), 0);
317         }
318         if (strstr(request, ".html HTTP/1.1") != NULL)
319         {
320             string filename;
321             if (strstr(request, "index.html HTTP/1.1") != NULL) filename = "HTML/Login_Form/";
322             else
323             {
324                 if (strstr(request, "files.html HTTP/1.1"))
325                 {
326                     filename = "HTML/Files_Page/";
327                 }
328             }
329         }
330     }
331 }
```

Output

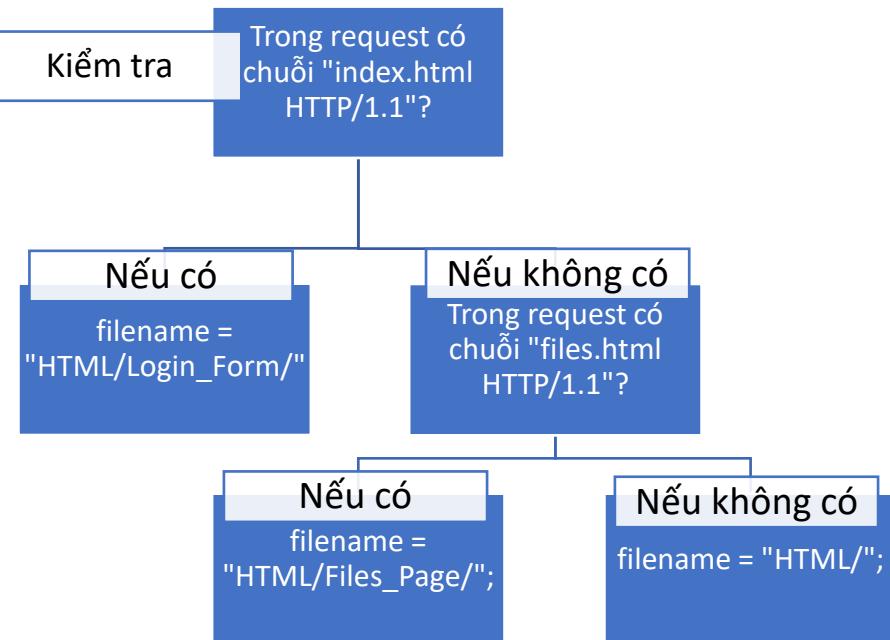
Show output from:

Output Error List

Ready

Add to Source Control

5.3.2 Chương trình sẽ dùng hàm strstr() để kiểm tra xem trong chuỗi có dòng “.html HTTP/1.1” hay không. Nếu có thì chương trình sẽ khai báo chuỗi filename. Sau đó gán filename theo sơ đồ sau (tất cả các bước kiểm tra đều dùng hàm strstr()):



Chương trình sẽ gán thêm giá trị của hàm readPathFile(request) vào cuối chuỗi filename. Hàm này sẽ trả về tên file mà server cần gửi. Đến đây thì filename có giá trị là đường dẫn đến file mà server cần gửi.

```

315     "\r\n", ipAddr);
316     send(new_socket, headers, sizeof(headers), 0);
317 }
318 if (strstr(request, ".html HTTP/1.1") != NULL)
319 {
320     string filename;
321     if (strstr(request, "index.html HTTP/1.1") != NULL) filename = "HTML/Login_Form/";
322     else
323     {
324         if (strstr(request, "files.html HTTP/1.1"))
325         {
326             filename = "HTML/Files_Page/";
327         }
328         else filename = "HTML/";
329     }
330     filename += readPathFile(request);
331     if (strstr(request, "index.html HTTP/1.1") != NULL) loginHTML(filename,ipAddr);
332     if (strstr(request, "files.html HTTP/1.1")) filesHTML(filename, "HTML/DOWNLOAD/*");
333     string html_s = readTextFile(filename);
334     int n = html_s.length();
335     if (countDigit(n) == 3)
336     {
337         char headers[88];
338         sprintf(headers, "HTTP/1.1 200 OK\r\n"
339                 "Content-Type: text/html\r\n"
340                 "Accept-Ranges: bytes\r\n");

```



```

91     pbuf->pubseekpos(0, fin.in);
92     char* buffer = new char[size];
93     fin.read(buffer, size);
94
95     fin.close();
96     return buffer;
97 }
98 Estring readPathfile(char* request)
99 {
100     string path;
101     int i = 5;
102     do
103     {
104         if (request[i] == '?') break;
105         path += request[i];
106         i++;
107     } while (request[i] != ' ');
108     replace(path, "%20", " ");
109     return path;
110 }
111 EAccount getAccount(char* request)
112 {
113     Account acc;
114     char* find = strstr(request, "Username=");
115     int i = (int)(find - request + 9);
116     ...

```

- 5.3.3 Chương trình kiểm tra xem trong request có dòng “index.html HTTP/1.1” không, nếu có thì chạy hàm loginHTML(filename,ipAddr). Hàm này sẽ tạo ra file index.html.

```

315         "\r\n", ipAddr);
316         send(new_socket, headers, sizeof(headers), 0);
317     }
318     if (strstr(request, ".html HTTP/1.1") != NULL)
319     {
320         string filename;
321         if (strstr(request, "index.html HTTP/1.1") != NULL) filename = "HTML/Login_Form/";
322         else
323         {
324             if (strstr(request, "files.html HTTP/1.1"))
325             {
326                 filename = "HTML/Files_Page/";
327             }
328             else filename = "HTML/";
329         }
330         filename += readPathFile(request);
331         if (strstr(request, "index.html HTTP/1.1") != NULL) loginHTML(filename, ipAddr);
332         if (strstr(request, "files.html HTTP/1.1")) filesHTML(filename, "HTML/DOWNLOAD/*");
333         string html = readTextFile(filename);
334         int n = html.length();
335         if (countDigit(n) == 3)
336         {
337             char headers[88];
338             sprintf(headers, "HTTP/1.1 200 OK\r\n"
339             "Content-Type: text/html\r\n"
340             "Accept-Ranges: bytes\r\n"

```

- 5.3.4 Chương trình sẽ kiểm tra tiếp xem trong request có dòng "files.html HTTP/1.1" hay không. Nếu có thì chương trình sẽ chạy hàm filesHTML(filename,"HTML/DOWNLOAD/*").

```

315         "\r\n", ipAddr);
316         send(new_socket, headers, sizeof(headers), 0);
317     }
318     if (strstr(request, ".html HTTP/1.1") != NULL)
319     {
320         string filename;
321         if (strstr(request, "index.html HTTP/1.1") != NULL) filename = "HTML/Login_Form/";
322         else
323         {
324             if (strstr(request, "files.html HTTP/1.1"))
325             {
326                 filename = "HTML/Files_Page/";
327             }
328             else filename = "HTML/";
329         }
330         filename += readPathFile(request);
331         if (strstr(request, "index.html HTTP/1.1") != NULL) loginHTML(filename, ipAddr);
332         if (strstr(request, "files.html HTTP/1.1")) filesHTML(filename, "HTML/DOWNLOAD/*");
333         string html = readTextFile(filename);
334         int n = html.length();
335         if (countDigit(n) == 3)
336         {
337             char headers[88];
338             sprintf(headers, "HTTP/1.1 200 OK\r\n"
339             "Content-Type: text/html\r\n"
340             "Accept-Ranges: bytes\r\n"

```

- 5.3.4.1 Hàm này có công dụng viết ra một file là files.html hiển thị một bảng gồm tên của các file có trong thư mục DOWNLOAD và kích thước của các file đó, đồng thời cho phép client download các file được hiển thị về thiết bị.

5.3.5 Kế đến, chương trình khai báo 1 chuỗi html_s và truyền giá trị trả về của hàm readTextFile(filename) vào chuỗi. Hàm readTextFile(filename) sẽ trả về nội dung của file có đường dẫn là filename.

```

325     {
326         filename = "HTML/Files_Page/";
327     }
328     else filename = "HTML/";
329 }
330 filename += readPathFile(request);
331 if (strstr(request, "index.html HTTP/1.1") != NULL) loginHTML(filename, ipAddr);
332 if (strstr(request, "files.html HTTP/1.1")) filesHTML(filename, "HTML/ DOWNLOAD/*");
333 string html_s = readTextFile(filename);
334 int n = html_s.length();
335 if (countDigit(n) == 3)
336 {
337     char headers[88];
338     sprintf(headers, "HTTP/1.1 200 OK\r\n"
339             "Content-Type: text/html\r\n"
340             "Accept-Ranges: bytes\r\n"
341             "Content-Length: %d\r\n"
342             "\r\n", n);
343     send(new_socket, headers, sizeof(headers), 0);
344 }
345 if (countDigit(n) == 4)
346 {
347     char headers[89];
348     sprintf(headers, "HTTP/1.1 200 OK\r\n"
349             "Content-type: text/html\r\n"
350             "Accept-Ranges: bytes\r\n"
351             "Content-Length: %d\r\n"
352             "\r\n", n);
353     send(new_socket, headers, sizeof(headers), 0);
354 }
355 }
356 }
```



```

58     : return true;
59 }
60 
```

```

61     =string readTextfile(string filename)
62     {
63         fstream f;
64         string line;
65         string data;
66         f.open(filename, ios::in);
67         if (!f)
68         {
69             cout << "cannot open file\n";
70             return "";
71         }
72         while (!f.eof())
73         {
74             getline(f, line);
75             data += line + "\r\n";
76         }
77         data += "\r\n";
78         f.close();
79         return data;
80     }
81 
```

```

82     =char* readBinaryFile(string filename, size_t& size)
83     {
84         ifstream fin;
85         fin.open(filename, ifstream::binary);
86     }
87 }
```

5.3.6 Sau đó, chương trình khai báo 1 biến n=html_s.length() để lưu độ dài của chuỗi html_s.

5.3.7 Tiếp đến, chương trình sẽ chạy hàm countDigit(n) để tìm độ dài thích hợp cho mảng headers. Nếu countDigit(n)=3 thì headers sẽ có độ dài là 88, nếu

`countDigit(n)=4` thì headers sẽ có độ dài là 89. Sau đó điền nội dung vào header bằng hàm `sprint()` và gửi header này qua client bằng hàm `send()`.

```

330     filename += readPathFile(request);
331     if (strstr(request, "index.html HTTP/1.1") != NULL) loginHTML(filename, ipAddr);
332     if (strstr(request, "files.html HTTP/1.1") != NULL) filesHTML(filename, "HTML/DOWNLOAD/*");
333     strcpy(html_s, readTextfile(filename));
334     int n = html_s.length();
335     if (countDigit(n) == 3)
336     {
337         char headers[88];
338         sprintf(headers, "HTTP/1.1 200 OK\r\n"
339             "Content-type: text/html\r\n"
340             "Accept-Ranges: bytes\r\n"
341             "Content-Length: %d\r\n"
342             "\r\n", n);
343         send(new_socket, headers, sizeof(headers), 0);
344     }
345     if (countDigit(n) == 4)
346     {
347         char headers[89];
348         sprintf(headers, "HTTP/1.1 200 OK\r\n"
349             "Content-type: text/html\r\n"
350             "Accept-Ranges: bytes\r\n"
351             "Content-Length: %d\r\n"
352             "\r\n", n);
353         send(new_socket, headers, sizeof(headers), 0);
354     }
355     char html[10000];
356     strcpy(html_s, html);
357 
```

5.3.7.1 Hàm `countDigit(n)` hiểu đơn giản là hàm tính xem n có bao nhiêu chữ số.

```

232     html += "<script src='vendor/jquery/jquery-3.2.1.min.js'></script>\n";
233     html += "<script src='vendor/bootstrap/js/popper.js'></script>\n";
234     html += "<script src='vendor/bootstrap/js/bootstrap.min.js'></script>\n";
235     html += "<script src='vendor/select2/select2.min.js'></script>\n";
236     html += "<script src='js/main.js'></script>\n";
237     html += "</body></html>";
238     f << html;
239     f.close();
240
241     int countDigit(int n)
242     {
243         int count = 1;
244         while (n / 10 > 0)
245         {
246             count++;
247             n /= 10;
248         }
249         return count;
250     }
251     main(argc, argv);
252 }
253 WSADATA wsa;
254 SOCKET s, new_socket;
255 struct sockaddr_in server, client;
256 int c, recv_size;
257 printf("\nInitialising Winsock...\n");
258 
```

5.3.8 Chương trình tạo 1 mảng tên là `html` gồm 10000 phần tử rồi copy dữ liệu của chuỗi `html_s` sang nó bằng hàm `strcpy(html, html_s.c_str())` rồi gửi chuỗi này đi bằng hàm `send()`.

```

341         "Content-Length: %d\r\n"
342         "\r\n", n);
343         send(new_socket, headers, sizeof(headers), 0);
344     }
345     if (countDigit(n) == 4)
346     {
347         char headers[89];
348         sprintf(headers, "HTTP/1.1 200 OK\r\n"
349             "Content-Type: text/html\r\n"
350             "Accept-Ranges: bytes\r\n"
351             "Content-Length: %d\r\n"
352             "\r\n", n);
353         send(new_socket, headers, sizeof(headers), 0);
354     }
355     char html[10000];
356     strcpy(html, html_s_c.str());
357     send(new_socket, html, sizeof(html), 0);
358 }
359 if (strstr(request, ".css") != NULL)
360 {
361     string filename;
362     if (strstr(request, "files.html") != NULL)
363     {
364         filename = "HTML/Files_Page/";
365     }
366     else filename = "HTML/Login_Form/";
367     filename += readPathFile(request);

```

- 5.3.9 Kết thúc dòng if này, chương trình tiếp tục xét đến các dòng if tương tự với những điều kiện thích hợp để truyền tìm đường dẫn đến file cần gửi qua chuỗi filename, tìm kích thước file, truy xuất nội dung file bằng các hàm đọc thích hợp đã được viết như readTextFile() hay readBinaryFile(), dùng hàm countDigit(n) để chọn độ lớn của headers, truyền nội dung vào headers và gửi headers này về cho phía clients, sau đó gửi nội dung của cái file cần gửi bằng hàm send(). Mục đích là để có thể hiển thị các lớp css, js, các font chữ đã được cài đặt sẵn bên phía server cho client.

The image displays two side-by-side screenshots of the Microsoft Visual Studio IDE interface, both showing the same C++ code editor window.

Top Window:

- Solution Explorer:** Shows the project 'SocketProgramming' with files like 'SocketProgramming.h', 'SocketProgramming.cpp', and 'main.cpp'.
- Code Editor:** Displays the 'SocketProgramming.cpp' file. The code handles HTTP requests for files like 'index.css', 'index.html', and 'Login_Form/'. It reads the request, determines the file type, and sends an appropriate HTTP response header followed by the file content.
- Status Bar:** Shows 'Ln: 354 Ch: 6 Col: 18 TABS CRLF'.
- Output Window:** Shows 'Item(s) Saved'.
- Bottom Status Bar:** Shows 'Ready'.

Bottom Window:

- Solution Explorer:** Shows the project 'SocketProgramming' with files like 'SocketProgramming.h', 'SocketProgramming.cpp', and 'main.cpp'.
- Code Editor:** Displays the 'SocketProgramming.cpp' file. The code handles HTTP requests for files like 'index.css', 'index.html', and 'Login_Form/'. It reads the request, determines the file type, and sends an appropriate HTTP response header followed by the file content.
- Status Bar:** Shows 'Ln: 408 Ch: 6 Col: 18 TABS CRLF'.
- Output Window:** Shows 'Item(s) Saved'.
- Bottom Status Bar:** Shows 'Ready'.

The screenshot shows the Microsoft Visual Studio IDE interface. The top menu bar includes File, Edit, View, Git, Project, Build, Debug, Test, Analyze, Tools, Extensions, Window, Help, and a Search bar. The title bar displays "SocketProgramming". The left sidebar contains the Solution Explorer with the "SocketProgramming" solution selected, showing its structure with Source Files and Header Files. The main workspace shows the "SocketProgramming.cpp" file in the Global Scope, containing C++ code for handling HTTP requests for font files. The bottom pane is the Output window, which is currently empty. The status bar at the bottom shows "90 %", "0 x 0 2", and "LN: 408 Ch: 6 Col: 18 TABS CRLF". The bottom right corner shows the date and time as "11:04 PM".

```
408
409     else
410     {
411         size_t sizeFile = 0;
412         if (strstr(request, "OpenSans-Regular.ttf") != NULL)
413         {
414             filename = "HTML/Files_Page/";
415             filename += readpathfile(request);
416         }
417         char* font = readBinaryFile(filename, sizeFile);
418         if (strstr(request, ".woff") != NULL)
419         {
420             char headers[66];
421             sprintf(headers, "HTTP/1.1 200 OK\r\n"
422                     "Content-Type: font/woff\r\n"
423                     "Content-Length: %d\r\n\r\n", sizeFile);
424             send(new_socket, headers, sizeof(headers), 0);
425         }
426         else
427         {
428             char headers[67];
429             sprintf(headers, "HTTP/1.1 200 OK\r\n"
430                     "Content-Type: font/ttf\r\n"
431                     "Content-Length: %d\r\n\r\n", sizeFile);
432             send(new_socket, headers, sizeof(headers), 0);
433         }
434     }
435     send(new_socket, font, sizeFile, 0);
```

The screenshot shows the Microsoft Visual Studio IDE interface. The main window displays the code for `SocketProgramming.cpp` in the `SocketProgramming` project. The code is a C program that handles HTTP requests for files like `.js` and `.html`. The interface includes the Solution Explorer, Properties, and Output windows.

```
432     : send(new_socket, headers, sizeof(headers), 0);
433     :
434     : send(new_socket, font, sizeFile, 0);
435     : send(new_socket, "\r\n", 3, 0);
436     :
437     if (strstr(request, ".js") != NULL)
438     {
439         :
440         string filename;
441         if (strstr(request, "Files.html") != NULL)
442         {
443             filename = "HTML/Files_Page/";
444         }
445         else filename = "HTML/Login_Form/";
446         filename += readPathfile(request);
447         string js_s = readTextfile(filename);
448         int n = js_s.length();
449         if (countDigit(n) == 4)
450         {
451             char headers[126];
452             sprintf(headers, "HTTP/1.1 200 OK\r\n"
453                     "Content-type: text/javascript\r\n"
454                     "Content-transfer-Encoding: utf-8"
455                     "Accept-Ranges: bytes\r\n"
456                     "Content-Length: %d\r\n"
457                     "\r\n", n);
```

The screenshot shows the Microsoft Visual Studio IDE interface. The top menu bar includes File, Edit, View, Git, Project, Build, Debug, Test, Analyze, Tools, Extensions, Window, Help, and a Search bar. The title bar displays "SocketProgramming". The left sidebar contains the Solution Explorer, which lists the solution 'SocketProgramming' with its projects, references, header files, resource files, and source files. The main workspace shows the code editor with the file "SocketProgramming.cpp" open. The code is written in C and handles socket communication, specifically sending HTTP headers and JSON payloads. Below the code editor is the Output window, which is currently empty. The status bar at the bottom provides information about the current line (Ln: 408), column (Col: 18), and tabs. A status bar at the very bottom indicates "Ready".

```
460
461     if (countDigit(n) == 5)
462     {
463         char headers[127];
464         sprintf(headers, "HTTP/1.1 200 OK\r\n"
465             "Content-Type: text/javascript\r\n"
466             "Content-Transfer-Encoding: utf-8"
467             "Accept-Ranges: bytes\r\n"
468             "Content-Length: %d\r\n"
469             "\r\n", n);
470         send(new_socket, headers, sizeof(headers), 0);
471     }
472     if (countDigit(n) == 6)
473     {
474         char headers[128];
475         sprintf(headers, "HTTP/1.1 200 OK\r\n"
476             "Content-Type: text/javascript\r\n"
477             "Content-Transfer-Encoding: utf-8"
478             "Accept-Ranges: bytes\r\n"
479             "Content-Length: %d\r\n"
480             "\r\n", n);
481         send(new_socket, headers, sizeof(headers), 0);
482     }
483     char js[200000];
484     strcpy(js, js.s.c_str());
485     send(new_socket, js, sizeof(js), 0);
486 }
```

5.3.10 Ở các font chữ hoặc ảnh chỉ xuất hiện trong 1 trường hợp, ta không cần dùng biến filename để tìm đường dẫn mà dùng đường dẫn cứng đến file font chữ hoặc ảnh đó, và do kích thước dữ liệu cố định nên ta cũng không cần dùng hàm countDigit() để xác định độ dài header.

The screenshot shows the Microsoft Visual Studio IDE interface. The top menu bar includes File, Edit, View, Git, Project, Build, Debug, Test, Analyze, Tools, Extensions, Window, Help, and a Search bar. The title bar displays "SocketProgramming". The left sidebar contains the Solution Explorer with the "SocketProgramming" solution selected, showing its structure with Source Files and Header Files. The main area is the Code Editor displaying a C++ file named "SocketProgramming.cpp". The code implements a simple HTTP server that serves static files like fonts and images. Below the code editor is the Output window, which is currently empty. The status bar at the bottom shows "90 %", "Ln: 408 Ch: 6 Col: 18 TABS CRLF", and the current time "11:06 PM".

```
if (strstr(request, "GET /fonts/COURIER.ttf HTTP/1.1") != NULL)
{
    size_t sizefile = 0;
    char* font = readBinaryFile("HTML/fonts/COURIER.ttf", sizefile);
    char headers[66];
    sprintf(headers, "HTTP/1.1 200 OK\r\n");
    "Content-type: font/ttf\r\n";
    "Content-length: %d\r\n", sizefile);
    send(new_socket, headers, sizeof(headers), 0);
    send(new_socket, font, sizefile, 0);
    send(new_socket, "\r\n", 3, 0);
}
if (strstr(request, "GET /images/image1.jpg HTTP/1.1") != NULL)
{
    size_t sizefile = 0;
    char* image = readBinaryFile("HTML/images/image1.jpg", sizefile);
    char headers[69];
    sprintf(headers, "HTTP/1.1 200 OK\r\n");
    "Content-type: image/jpeg\r\n";
    "Content-length: %d\r\n", sizefile);
    send(new_socket, headers, sizeof(headers), 0);
    send(new_socket, image, sizefile, 0);
    send(new_socket, "\r\n", 3, 0);
}
if (strstr(request, "GET /images/image2.jpg HTTP/1.1") != NULL)
```

```
if (strstr(request, "GET /images/image2.jpg HTTP/1.1") != NULL)
{
    size_t sizefile = 0;
    char* image = readBinaryFile("HTML/images/image2.jpg", sizefile);
    char headers[68];
    sprintf(headers, "HTTP/1.1 200 OK\r\n"
                  "Content-Type: image/jpeg\r\n\r\n"
                  "Content-Length: %d\r\n", sizefile);
    send(new_socket, headers, sizeof(headers), 0);
    send(new_socket, image, sizefile, 0);
    send(new_socket, "\r\n", 3, 0);
}

if (strstr(request, "GET /images/bg-01.jpg HTTP/1.1") != NULL)
{
    size_t sizefile = 0;
    char* icon = readBinaryFile("HTML/Login_Form/images/bg-01.jpg", sizefile);
    char headers[69];
    sprintf(headers, "HTTP/1.1 200 OK\r\n"
                  "Content-Type: image/jpeg\r\n\r\n"
                  "Content-Length: %d\r\n", sizefile);
    send(new_socket, headers, sizeof(headers), 0);
    send(new_socket, icon, sizefile, 0);
    send(new_socket, "\r\n", 3, 0);
}

if (strstr(request, "GET /images/1.jpg HTTP/1.1") != NULL)
```

```
if (strstr(request, "GET /images/1.jpg HTTP/1.1") != NULL)
{
    size_t sizefile = 0;
    char* icon = readBinaryFile("HTML/images/1.jpg", sizefile);
    char headers[69];
    sprintf(headers, "HTTP/1.1 200 OK\r\n"
                  "Content-Type: image/jpeg\r\n\r\n"
                  "Content-Length: %d\r\n", sizefile);
    send(new_socket, headers, sizeof(headers), 0);
    send(new_socket, icon, sizefile, 0);
    send(new_socket, "\r\n", 3, 0);
}

if (strstr(request, "/DOWNLOAD") != NULL)
{
    string filename = "HTML/" + readPathfile(request);
    size_t sizefile = 0;
    char* DOWNLOADfile = readBinaryFile(filename, sizefile);
    if (countDigit(sizefile) == 1)
    {
        char headers[55];
```

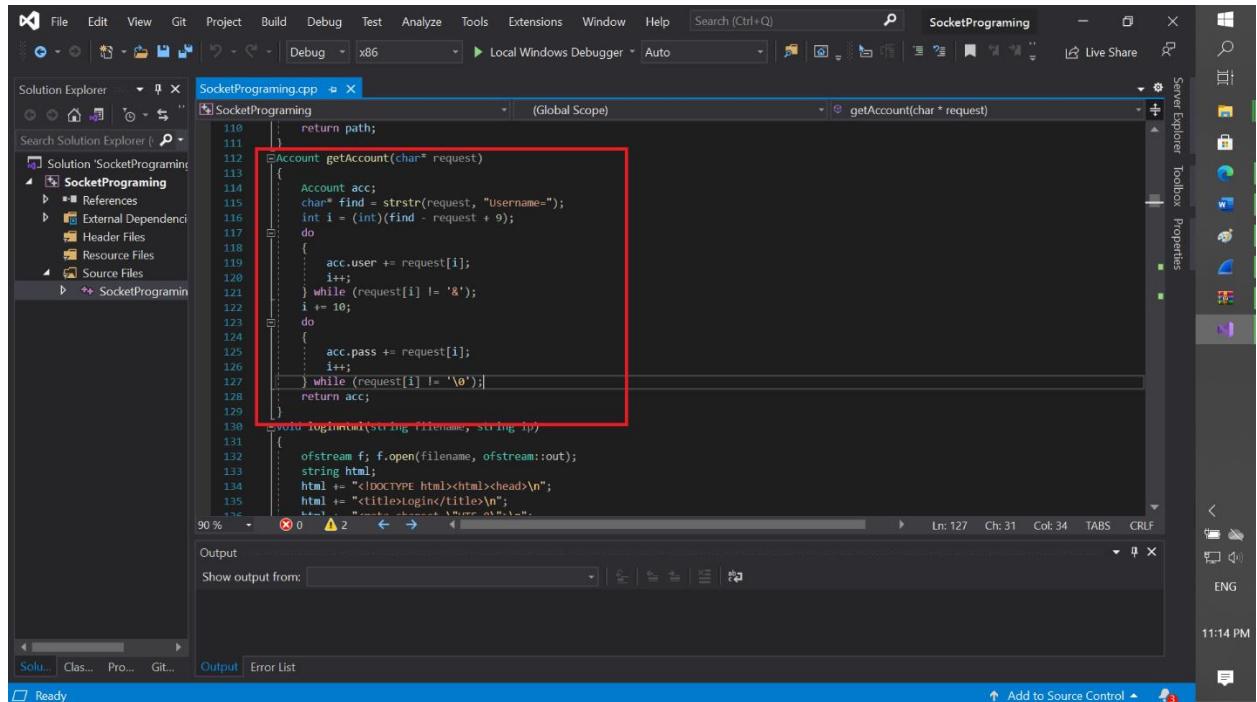
```

622         send(new_socket, headers, sizeof(headers), 0);
623     }
624     send(new_socket, DOWNLOADfile, sizeFile, 0);
625     send(new_socket, "\r\n", 3, 0);
626
627     if (strstr(request, "GET /favicon.ico HTTP/1.1") != NULL)
628     {
629         size_t sizefile = 0;
630         char* icon = readbinary_file("HTML/Login_Form/images/icons/favicon.ico", sizefile);
631         char headers[68];
632         sprintf(headers, "HTTP/1.1 200 OK\r\n");
633         sprintf(headers, "Content-Type: image/png\r\n");
634         sprintf(headers, "Content-Length: %d\r\n", sizefile);
635         send(new_socket, headers, sizeof(headers), 0);
636         send(new_socket, icon, sizefile, 0);
637         send(new_socket, "\r\n", 3, 0);
638     }
639     if (strstr(request, "POST / HTTP/1.1") != NULL)
640     {
641         Account acc = getAccount(request);
642         if (acc.user == "admin" && acc.pass == "admin")
643         {
644             char headers[83];
645             sprintf(headers, "HTTP/1.1 301 Moved Permanently\r\n");
646             sprintf(headers, "Location: http://%s:8888/info.html\r\n");
647         }
648     }

```

5.3.11 Tiếp theo, chương trình xét xem trong request có chuỗi "POST / HTTP/1.1" hay không. Nếu có thì khởi tạo biến acc có kiểu Account đã được xây dựng ở đầu chương trình và gán giá trị bằng giá trị trả về của hàm getAccount(request).

5.3.11.1 Hàm này sử dụng hàm strstr(request,"USERNAME=") để tìm vị trí bắt đầu của chuỗi USERNAME. Sau đó khởi tạo biến I đánh dấu vị trí của phần username trong struct Account. Sau đó tiến hành chạy vòng lặp để sao chép từng ký tự username trong request về thành phần user của một biến tạm đặt là acc có kiểu dữ liệu là Account. Vòng lặp sẽ dừng lại khi gặp khi request[i] = '&'. Kết thúc vòng lặp, ta cộng I cho 10 để I đánh dấu được vị trí của phần password (vì ký tự cuối username và ký tự đầu của password cách nhau 10 ký tự). sau đó tiến hành vòng lặp tương tự để gán giá trị cho acc.pass cho đến hết chuỗi request. Hàm sẽ trả về giá trị của biến acc.

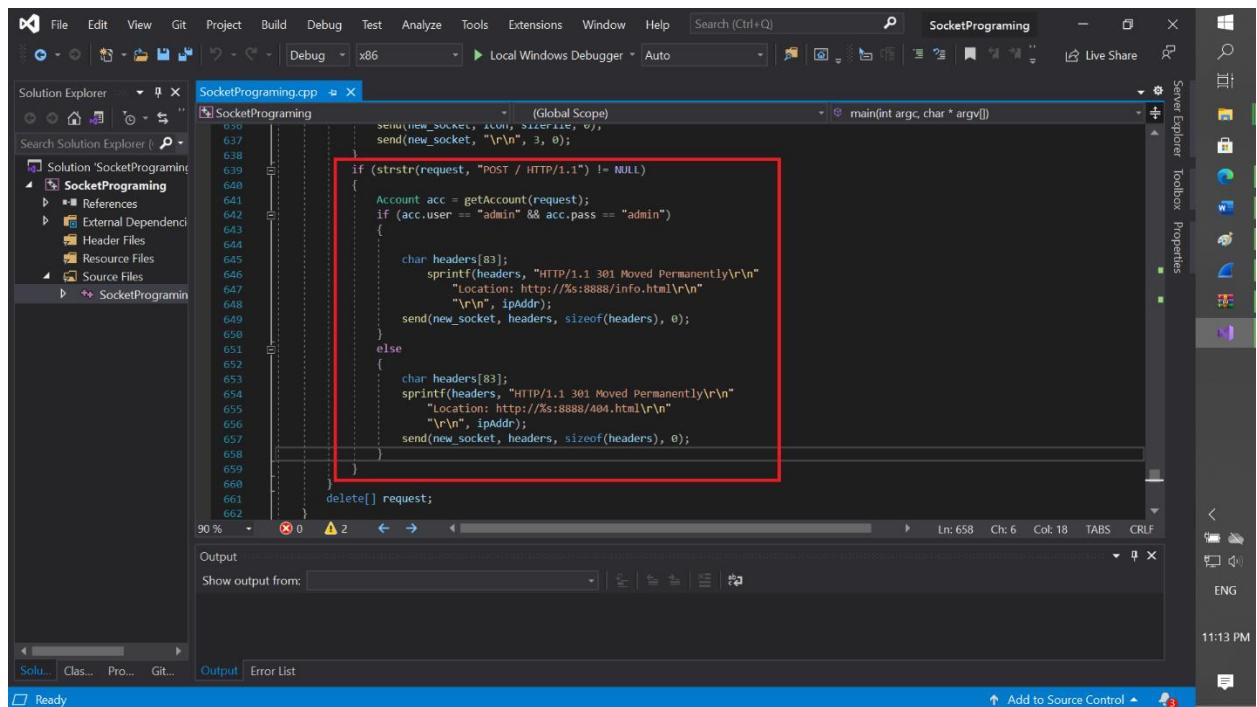


```

110     return path;
111 }
112 Account getAccount(char* request)
113 {
114     Account acc;
115     char* find = strstr(request, "username=");
116     int i = (int)(find - request + 9);
117     do
118     {
119         acc.user += request[i];
120         i++;
121     } while (request[i] != '&');
122     i += 10;
123     do
124     {
125         acc.pass += request[i];
126         i++;
127     } while (request[i] != '\0');
128     return acc;
129 }
130 void loginOn(string filename, string ip)
131 {
132     ofstream f; f.open(filename, ofstream::out);
133     string html;
134     html += "<!DOCTYPE html><html><head>\n";
135     html += "<title>Login</title>\n";
136     html += "<meta name='viewport' content='width=device-width, initial-scale=1.0'>\n";

```

5.3.12 Đến đây thì chương trình đã có tài khoản và mật khẩu mà client nhập vào. Chương trình chỉ cần kiểm tra xem đã đúng với tài khoản và mật khẩu mặc định hay chưa. Nếu đúng thì tạo một header yêu cầu client chuyển hướng sang <http://ipAddr:8888/info.html> và gửi về client qua hàm send. Nếu sai thì ta sẽ tạo một header yêu cầu client chuyển hướng sang <http://ipAddr:8888/404.html> rồi gửi về cho client. Với ipAddr là địa chỉ đã tìm được ở trên.



```

569     send(new_socket, login, strlen(login), 0);
570     send(new_socket, "\r\n", 2, 0);
571     if (strstr(request, "POST / HTTP/1.1") != NULL)
572     {
573         Account acc = getAccount(request);
574         if (acc.user == "admin" && acc.pass == "admin")
575         {
576             char headers[83];
577             sprintf(headers, "HTTP/1.1 301 Moved Permanently\r\n"
578                     "Location: http://%s:8888/info.html\r\n"
579                     "\r\n", ipAddr);
580             send(new_socket, headers, sizeof(headers), 0);
581         }
582         else
583         {
584             char headers[83];
585             sprintf(headers, "HTTP/1.1 301 Moved Permanently\r\n"
586                     "Location: http://%s:8888/404.html\r\n"
587                     "\r\n", ipAddr);
588             send(new_socket, headers, sizeof(headers), 0);
589         }
590     }
591     delete[] request;
592 }

```

5.3.12.1 Trong <http://ipAddr:8888/info.html> sẽ có một button để chuyển hướng client sang trang <http://ipAddr:8888/files.html> đã được viết ở trên.

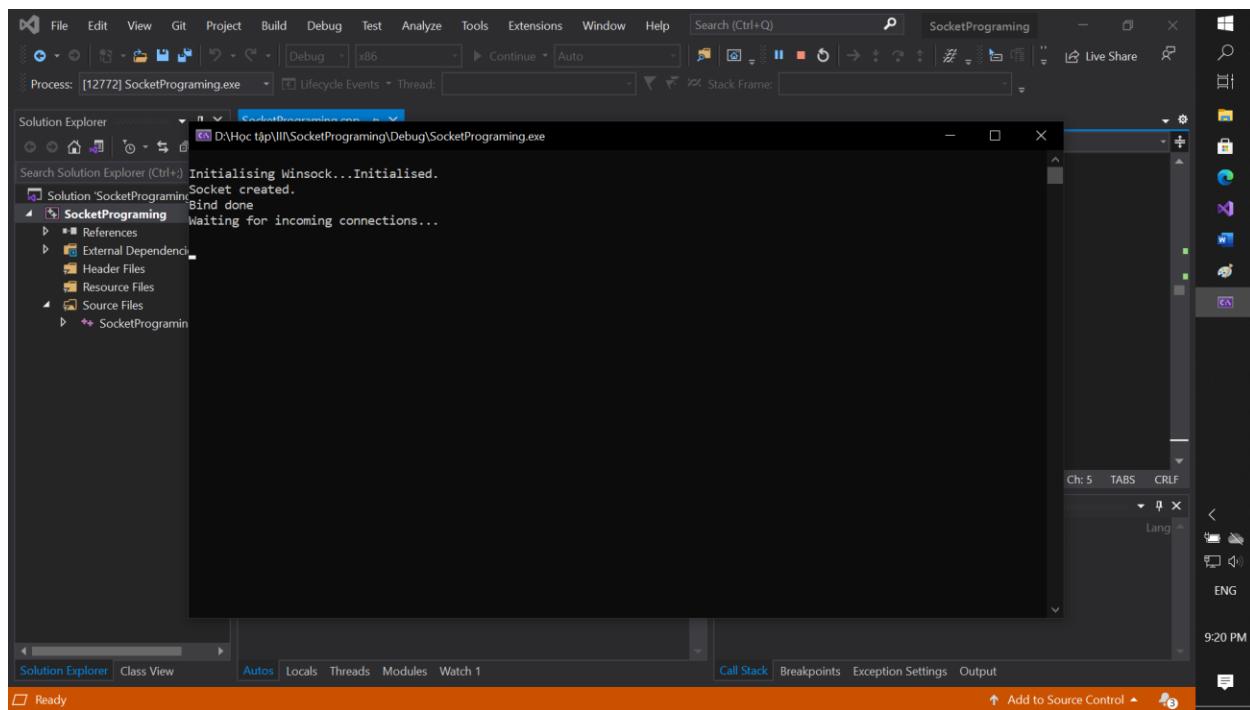
5.4 Khi kết thúc vòng lặp, ta sẽ xóa mảng request.

- 6 Cuối cùng, xét nếu new_socket=INVALID_SOCKET thì ta in ra màn hình báo thất bại và mã lỗi rồi kết thúc chương trình.
- 7 Nếu không thì ta đóng socket đã tạo lại rồi kết thúc chương trình.

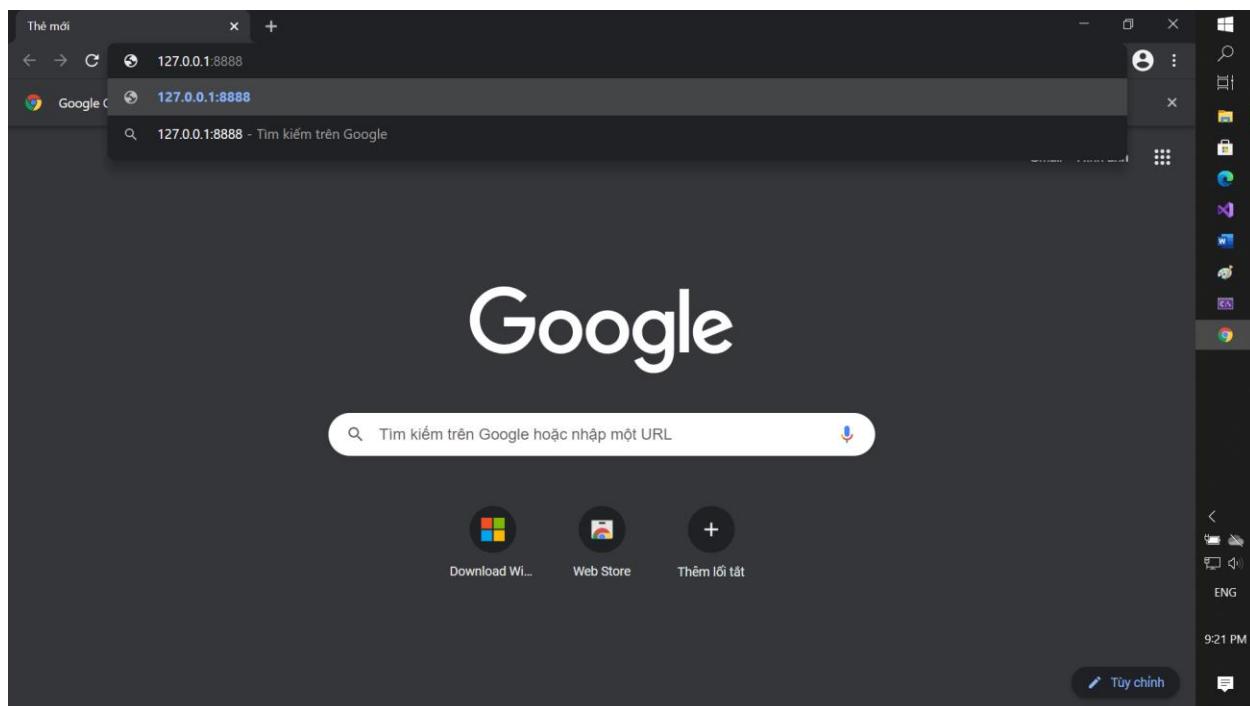
```
657     send(new_socket, headers, sizeof(headers), 0);
658 }
659 }
660 delete[] request;
661 }
662 if (new_socket == INVALID_SOCKET)
663 {
664     printf("accept failed with error code : %d", WSAGetLastError());
665     return 1;
666 }
667 closesocket(s);
668 WSACleanup();
669 return 0;
670 }
671 }
672 }
673 }
674 }
```

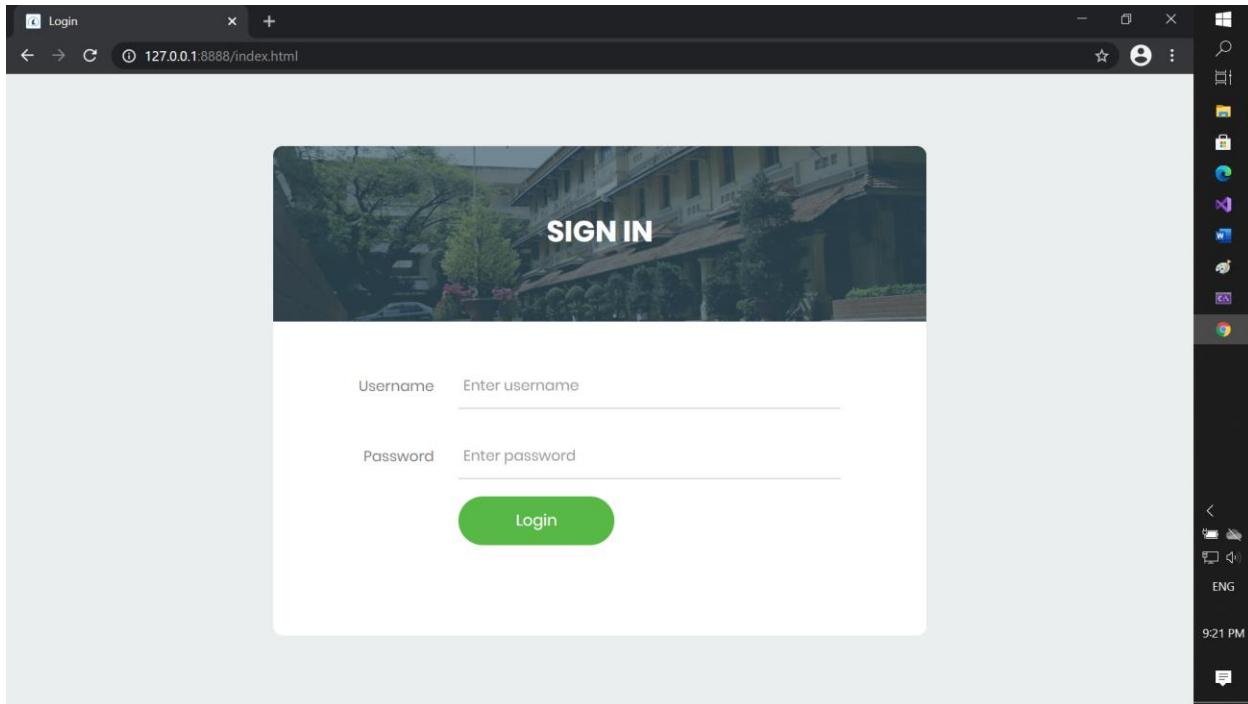
2.3/ Cách kiểm tra ứng dụng:

Bước 1: Chạy phần code cpp của server.

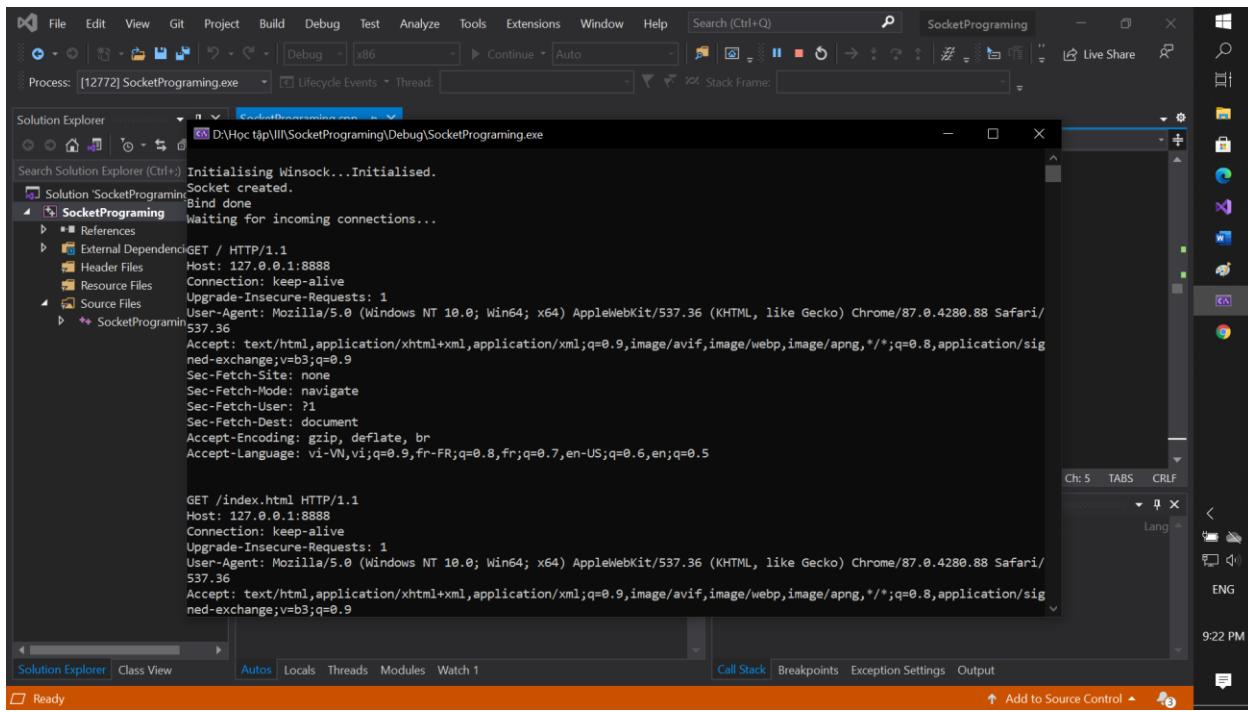


Bước 2: Truy cập vào đường dẫn 127.0.0.1:8888. Nếu server dẫn ta qua đường dẫn 127.0.0.1:8888/index.html thì đoạn code này đã chạy đúng.

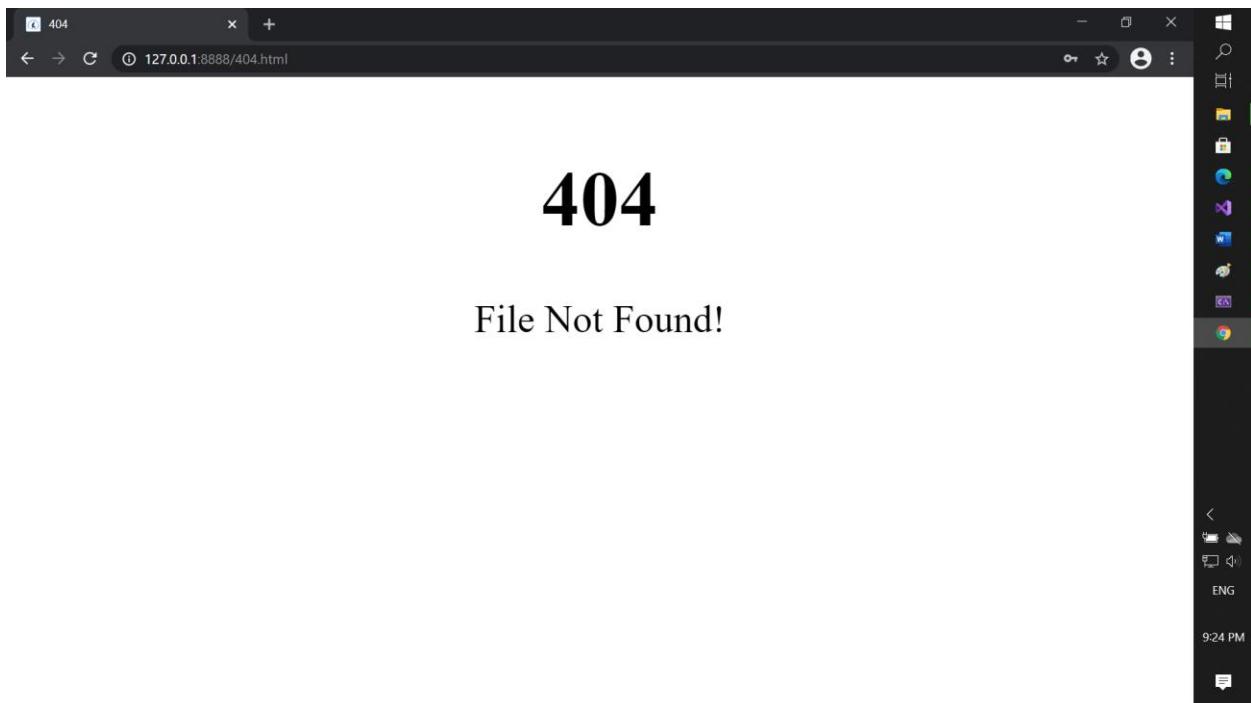
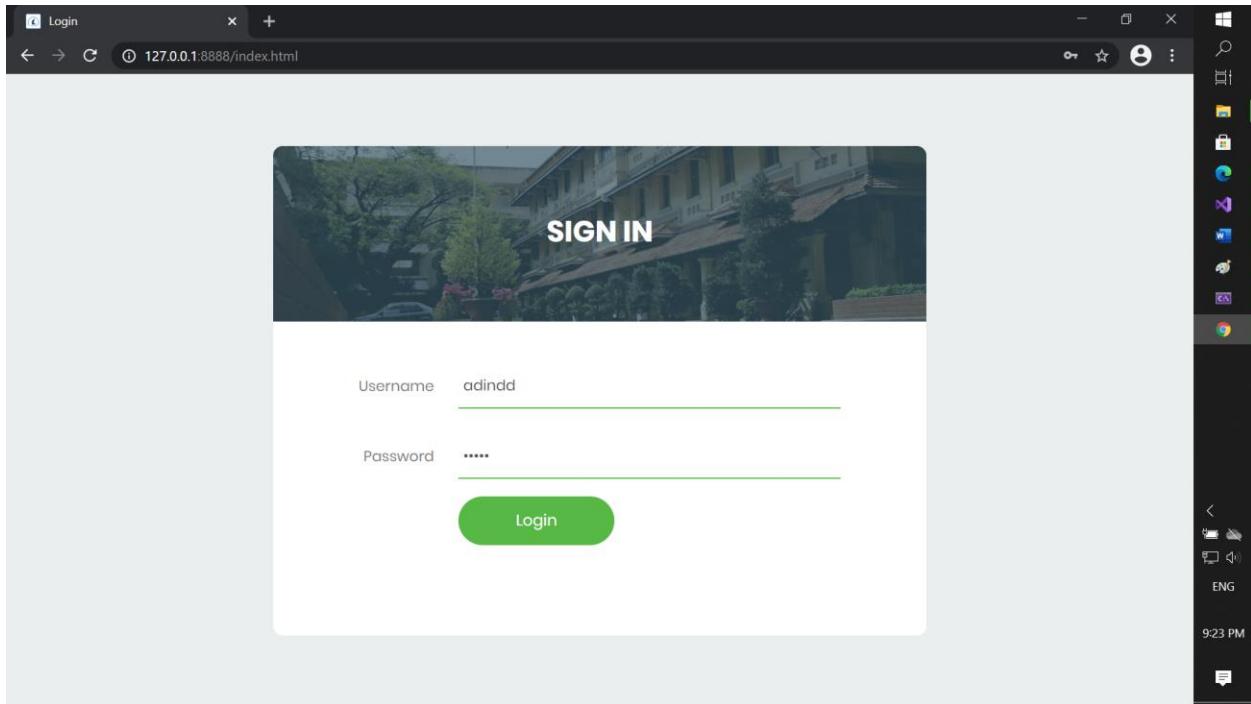




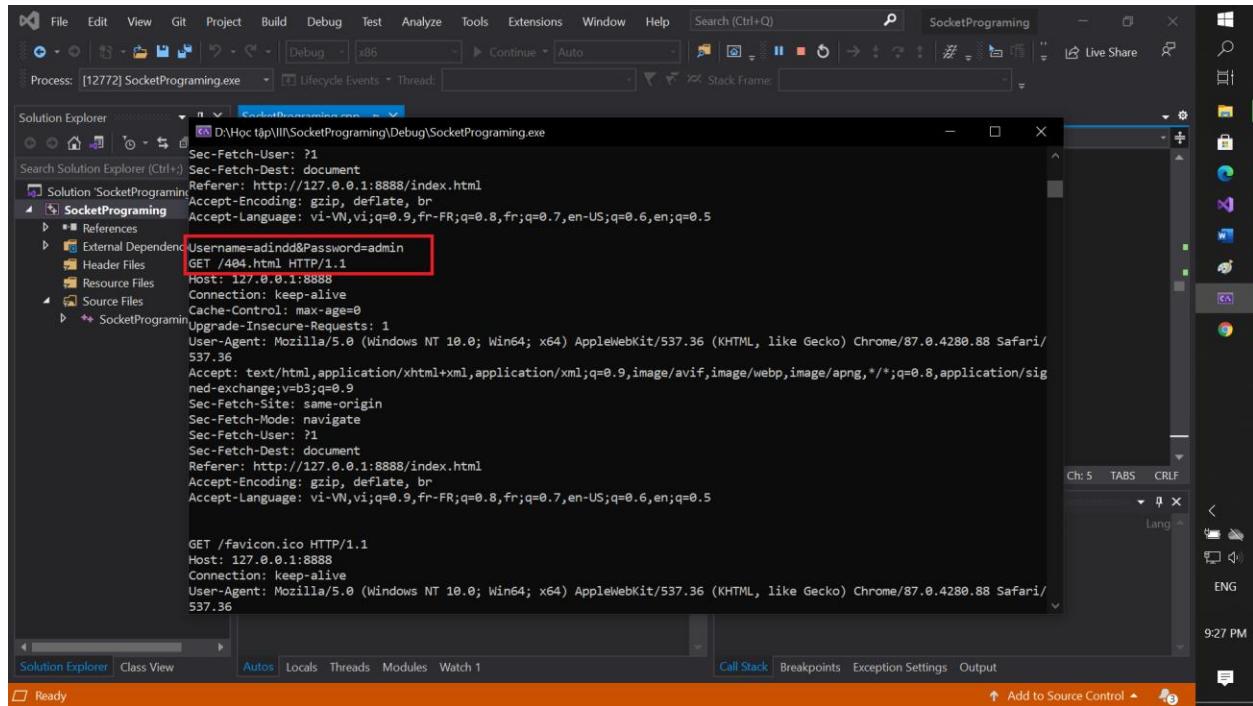
Bước 3: Xem thông tin ở cửa sổ console



Bước 4: Nhập thử tài khoản sai hoặc mật khẩu sai trong trang index.html hoặc cả hai. Nếu chuyển đến trang info.html ở 1 trong 3 trường hợp thì đoạn code này đã sai. Ví dụ ở trường hợp 1:



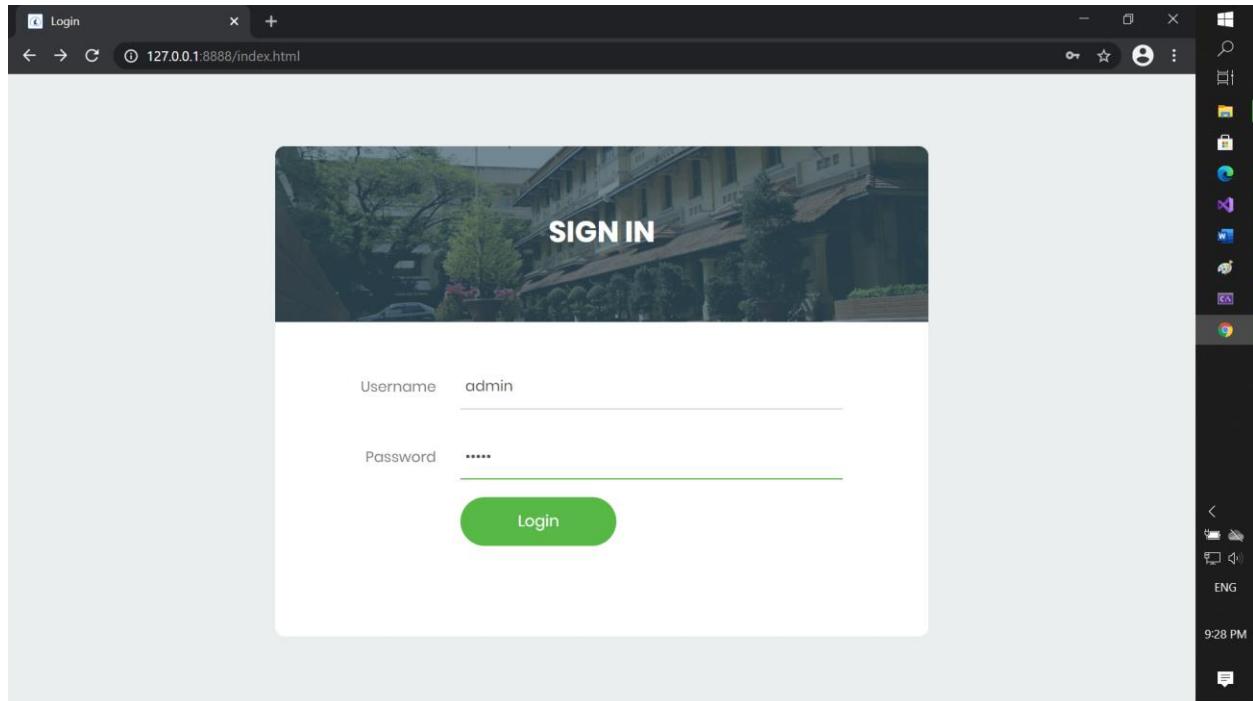
Bước 5: Kiểm tra cửa sổ console



```
Process: [12772] SocketProgramming.exe
Solution Explorer: D:\Học tập\II\SocketProgramming\Debug\SocketProgramming.exe
Search Solution Explorer (Ctrl+F)
Solution 'SocketProgramming' -> 
  Sec-Fetch-User: ?1
  Sec-Fetch-Dest: document
  Referer: http://127.0.0.1:8888/index.html
  Accept-Encoding: gzip, deflate, br
  Accept-Language: vi-VN,vi;q=0.9,fr-FR;q=0.8,fr;q=0.7,en-US;q=0.6,en;q=0.5
  <-- External Dependencies
  > References
  Header Files
  Resource Files
  Source Files
  <-- SocketProgramming
    Username=admin&Password=admin
    GET /404.html HTTP/1.1
    Host: 127.0.0.1:8888
    Connection: keep-alive
    Cache-Control: max-age=0
    Upgrade-Insecure-Requests: 1
    User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4280.88 Safari/537.36
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,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://127.0.0.1:8888/index.html
    Accept-Encoding: gzip, deflate, br
    Accept-Language: vi-VN,vi;q=0.9,fr-FR;q=0.8,fr;q=0.7,en-US;q=0.6,en;q=0.5

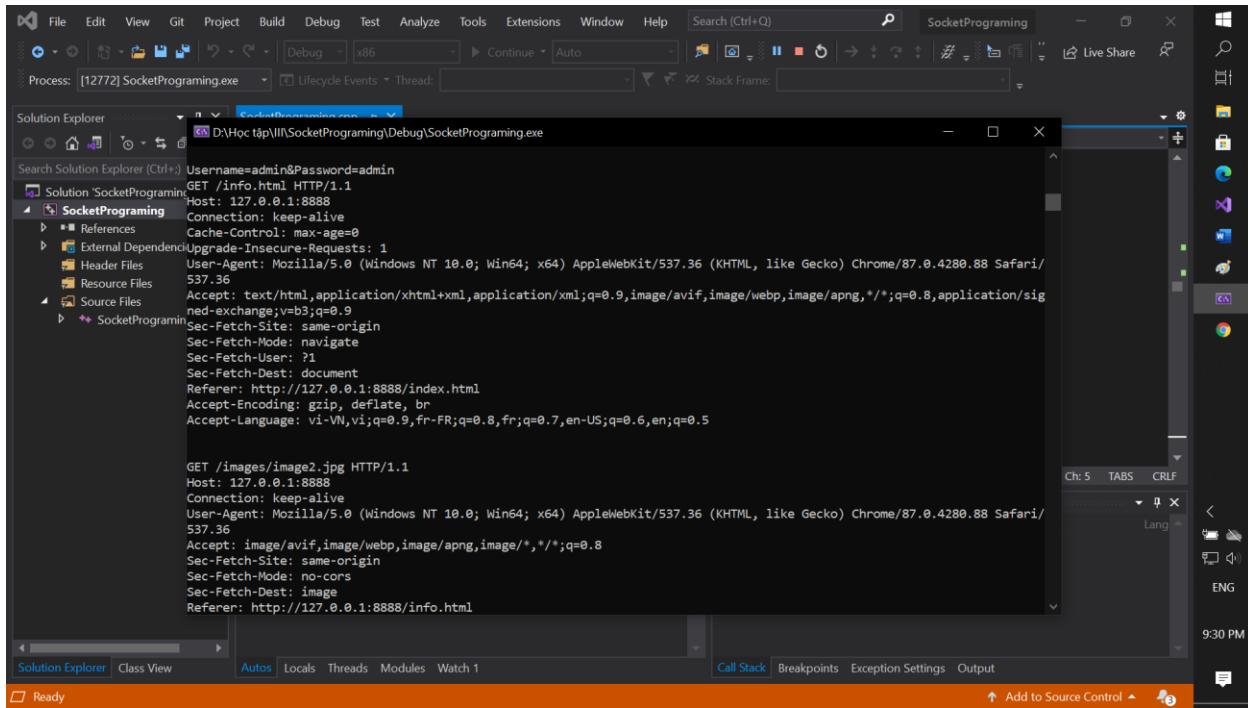
    GET /favicon.ico HTTP/1.1
    Host: 127.0.0.1:8888
    Connection: keep-alive
    User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4280.88 Safari/537.36
```

Bước 6: Nhập tài khoản và mật khẩu đúng. Nếu chuyển sang trang info.html thì đoạn code này đã đúng.

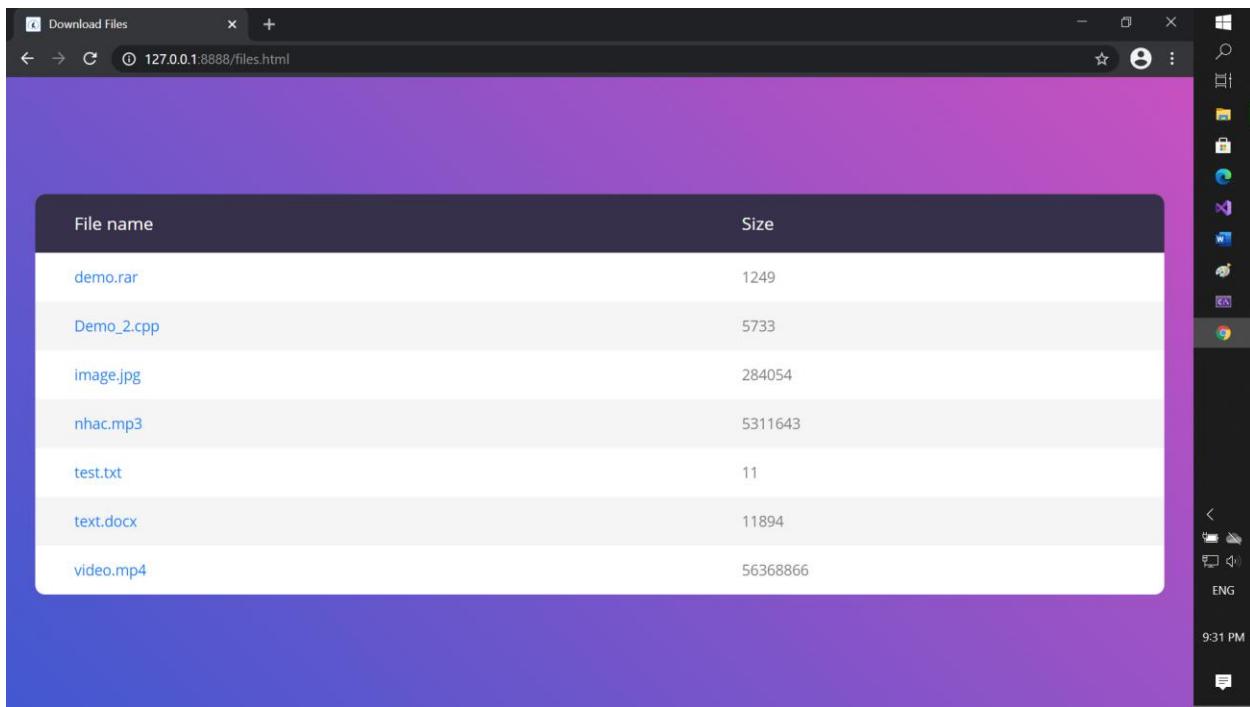
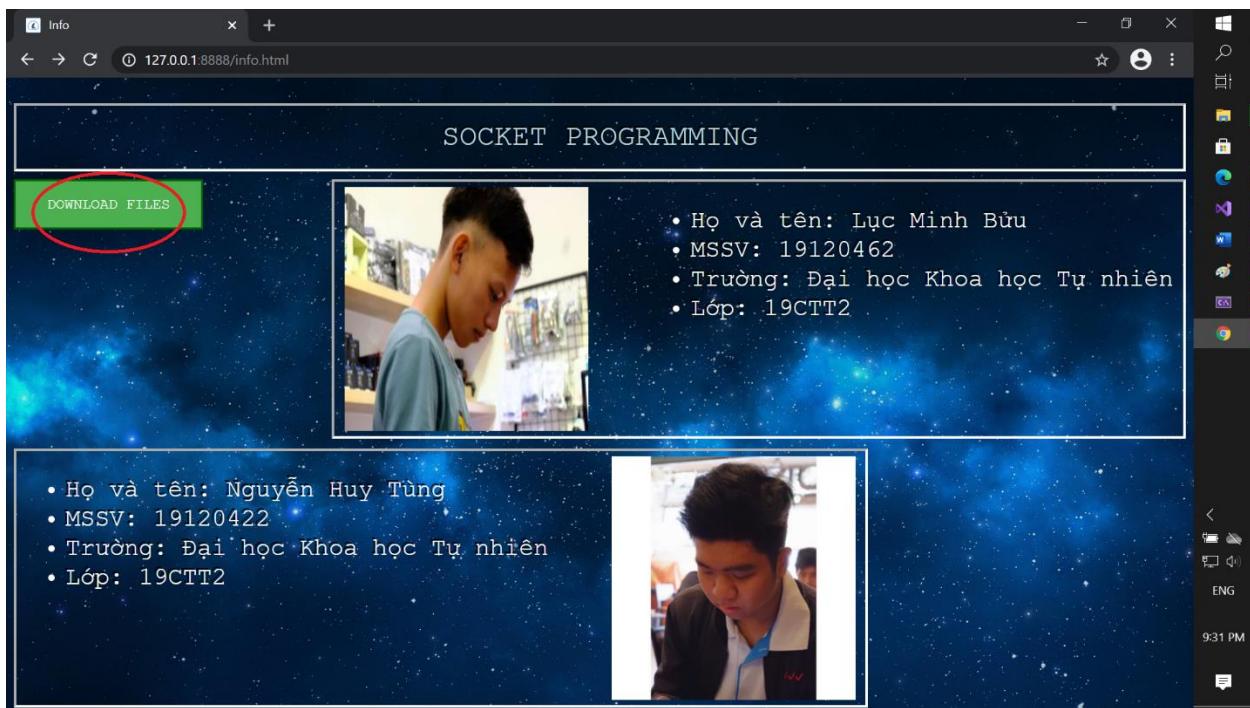




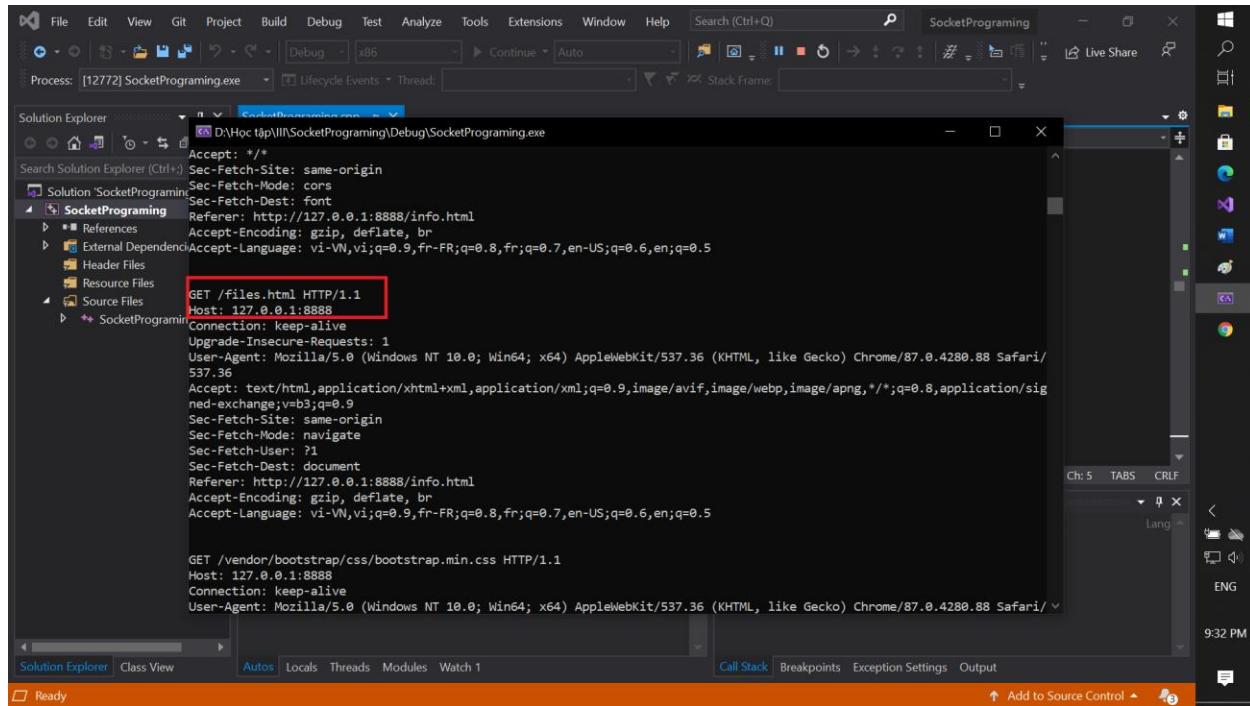
Bước 7: Kiểm tra cửa sổ console



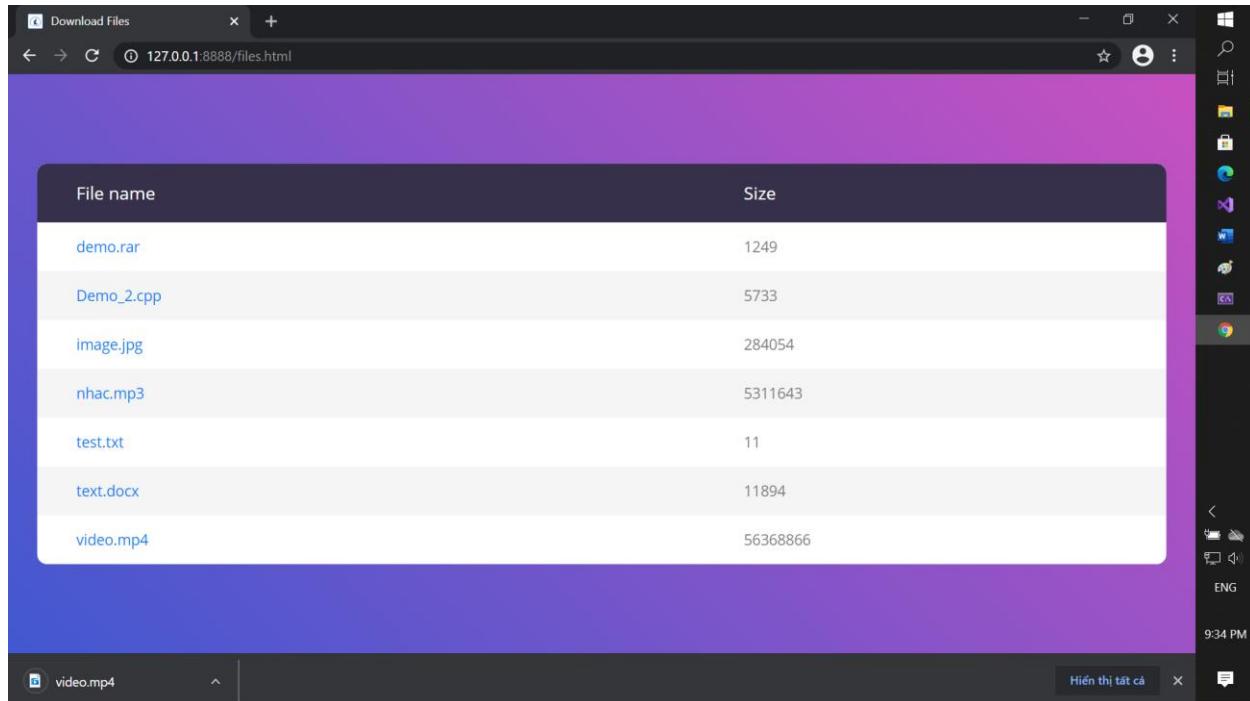
Bước 8: Bấm vào button “DOWNLOAD FILES” bên góc trái. Nếu chuyển sang trang files.html thì đoạn code này đã đúng.



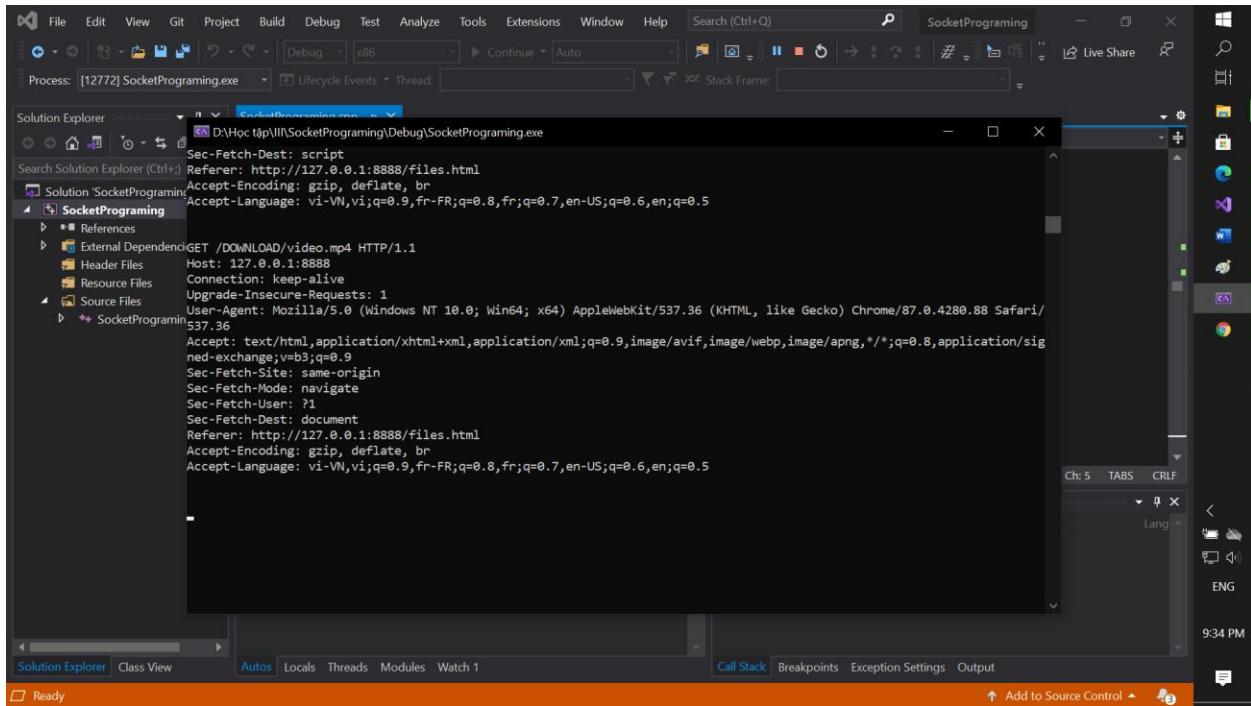
Bước 9: Kiểm tra cửa sổ console



Bước 10: Tải bất kỳ 1 file nào đó trong bảng về máy. Nếu có file nào bị lỗi thì đoạn code này không đúng



Bước 11: Kiểm tra cửa sổ console



2.4/ Các nguồn tham khảo:

- Tài liệu thực hành socket
- Các video hướng dẫn thực hành socket.
- Các trang web, video về code html, css.
- Các trang web về các hàm liên quan đến socket và địa chỉ IP.

[Winsock tutorial – Socket programming in C on windows – BinaryTides](#)

[MIME types \(IANA media types\) - HTTP | MDN \(mozilla.org\)](#)