CMPT 371 Mini-Project 1 Report

Team 32: Clement Lau, Yuanhang Wang

Step One: HTTP Status Code Documentation

Status Code 200: OK	
Requirements	The HTTP request format must be correct, and the server must be able to perform the request successfully
HTTP Methods	GET
HTTP Request Section	All request and header lines must contain all necessary information for the request to succeed
Simplified Example HTTP Request Message	

`GET /test.html HTTP/1.1`
`Host: localhost:8088`	

Status Code 304: Not Modified	
Requirements	The HTTP request format must be correct, and the modification date of the server file must not be newer than the modification date of the if-modified-since header
HTTP Methods	GET
HTTP Request Section	The timestamp of the HTTP if-modified-since header if it is present
Simplified Example HTTP Request Message	

`GET /test.html HTTP/1.1`
`Host: localhost:8088`
`If-Modified-Since: Tue, 25 Jun 2024 12:53:25 GMT`	

Status Code 400: Bad Request	
Requirements	The HTTP request is incorrect, and the server cannot parse or process the request. On our web server, this can be caused by unsupported query strings
HTTP Methods	GET
HTTP Request Section	The URL in the HTTP request line
Simplified Example HTTP Request Message	

`GET /test.html?key=1 HTTP/1.1`
`Host: localhost:8088`	

Status Code 403: Forbidden	
Requirements	The HTTP request format is correct, but the credentials or permissions to perform the request are absent or incorrect. On our web server, we will require the `Authorization` field for the `auth.html` file
HTTP Methods	GET
HTTP Request Section	Missing or incorrect authorization field in the HTTP header if the path is `auth.html`
Simplified Example HTTP Request Message	

`GET /auth.html HTTP/1.1`
`Host: localhost:8088`
`Authorization: Basic dGVzdDppbmNvcnJlY3QgcGFzc3dvcmQ=`	

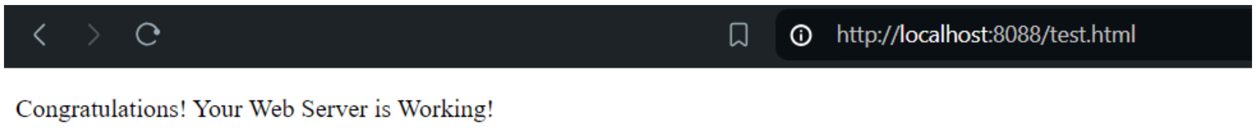
Status Code 404: Not Found	
Requirements	The HTTP request format is correct, but the specified resource cannot be found by the server
HTTP Methods	GET
HTTP Request Section	The request line specifying the target resource
Simplified Example HTTP Request Message	

`GET /nonexistentfile.html HTTP/1.1`
`Host: localhost:8088`	

We did not implement HTTP methods other than GET because they are not required by the project description.

Step Two: Web server and testing

The zip folder contains the Python server files and three test files. Place the test.html and auth.html files in the same folder as the server. To run the server, navigate to the server file directory and run web_server.py using `python`. Enter http://localhost:8088/test.html in your browser, and you should be greeted with the following screen:

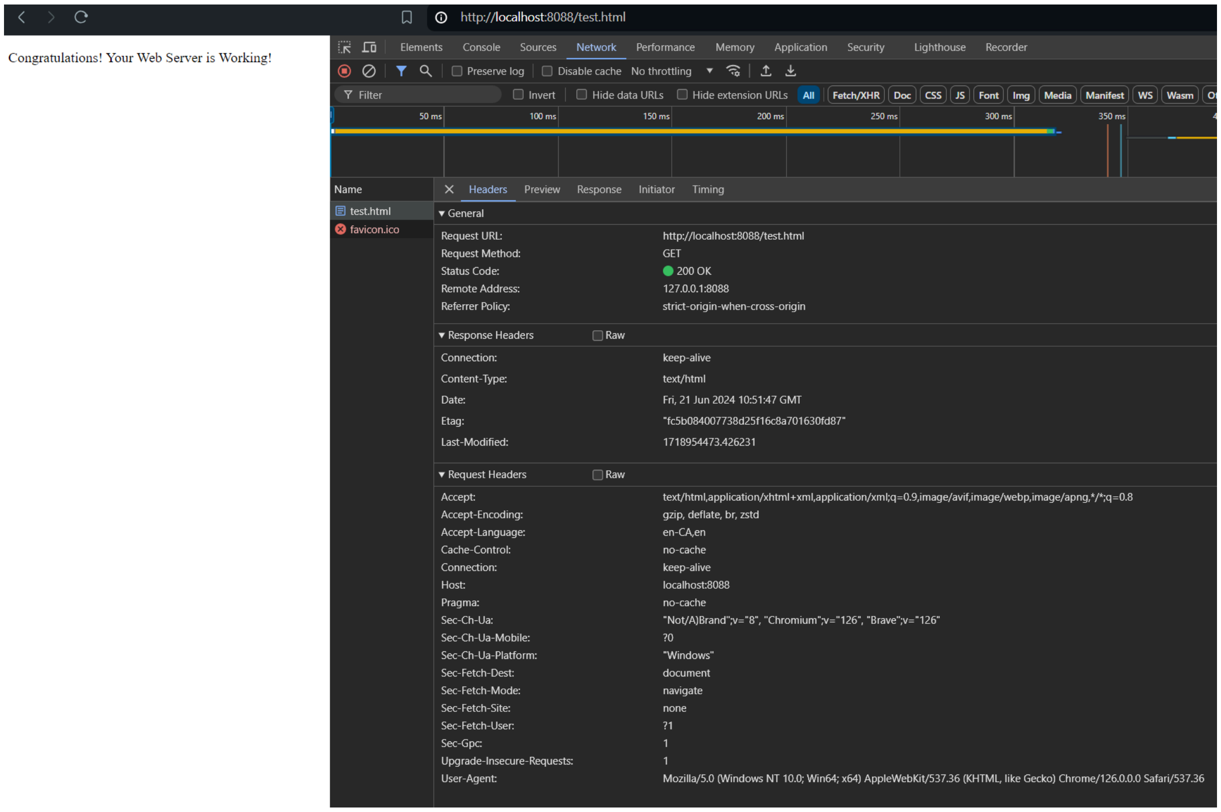


Congratulations! Your Web Server is Working!

Web Server Specifications
Host name: localhost or 127.0.0.1
Port: 8088
Socket: TCP
Threading: Multi-threaded
Resources: test.html & auth.html (requires authorization)
Username: test
Password: test
Authorization base64 encoding: Basic dGVzdDp0ZXN0

Testing the web server
Note: Some additional headers are included when compared to the example HTTP requests specified earlier. These are added by the browser, but functionally, they should have no effect.

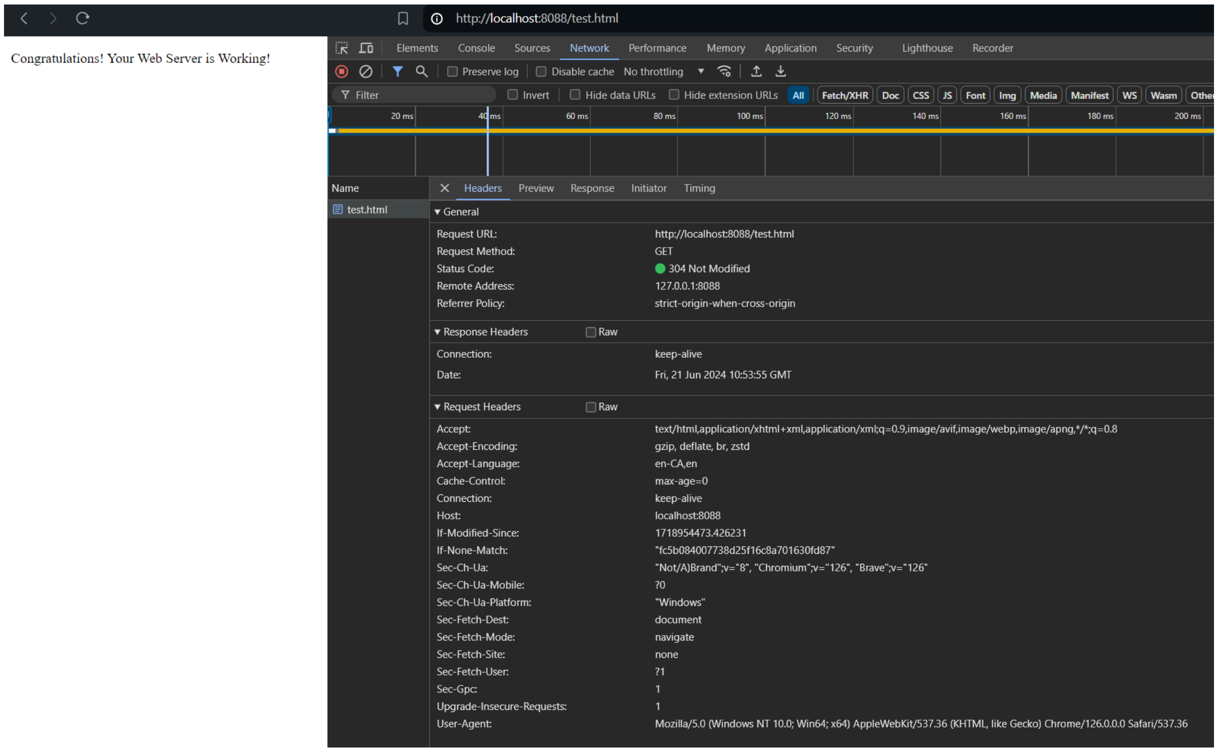
Let’s test the web server. When we fetch the resource for the first time, it should return a 200 OK status code.



Congratulations! Your Web Server is Working!

```
▼ General
Request URL:               http://localhost:8088/test.html
Request Method:            GET
Status Code:               ● 200 OK
Remote Address:            127.0.0.1:8088
Referrer Policy:           strict-origin-when-cross-origin

▼ Response Headers    ☐ Raw
Connection:                keep-alive
Content-Type:              text/html
Date:                      Fri, 21 Jun 2024 10:51:47 GMT
Etag:                      "fc5b084007738d25f16c8a701630fd87"
Last-Modified:             1718954473.426231

▼ Request Headers    ☐ Raw
Accept:                    text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8
Accept-Encoding:           gzip, deflate, br, zstd
Accept-Language:           en-CA,en
Cache-Control:             no-cache
Connection:                keep-alive
Host:                      localhost:8088
Pragma:                    no-cache
Sec-Ch-Ua:                 "Not/A)Brand";v="8", "Chromium";v="126", "Brave";v="126"
Sec-Ch-Ua-Mobile:          ?0
Sec-Ch-Ua-Platform:        "Windows"
Sec-Fetch-Dest:            document
Sec-Fetch-Mode:            navigate
Sec-Fetch-Site:            none
Sec-Fetch-User:            ?1
Sec-Gpc:                   1
Upgrade-Insecure-Requests: 1
User-Agent:                Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/126.0.0.0 Safari/537.36
```

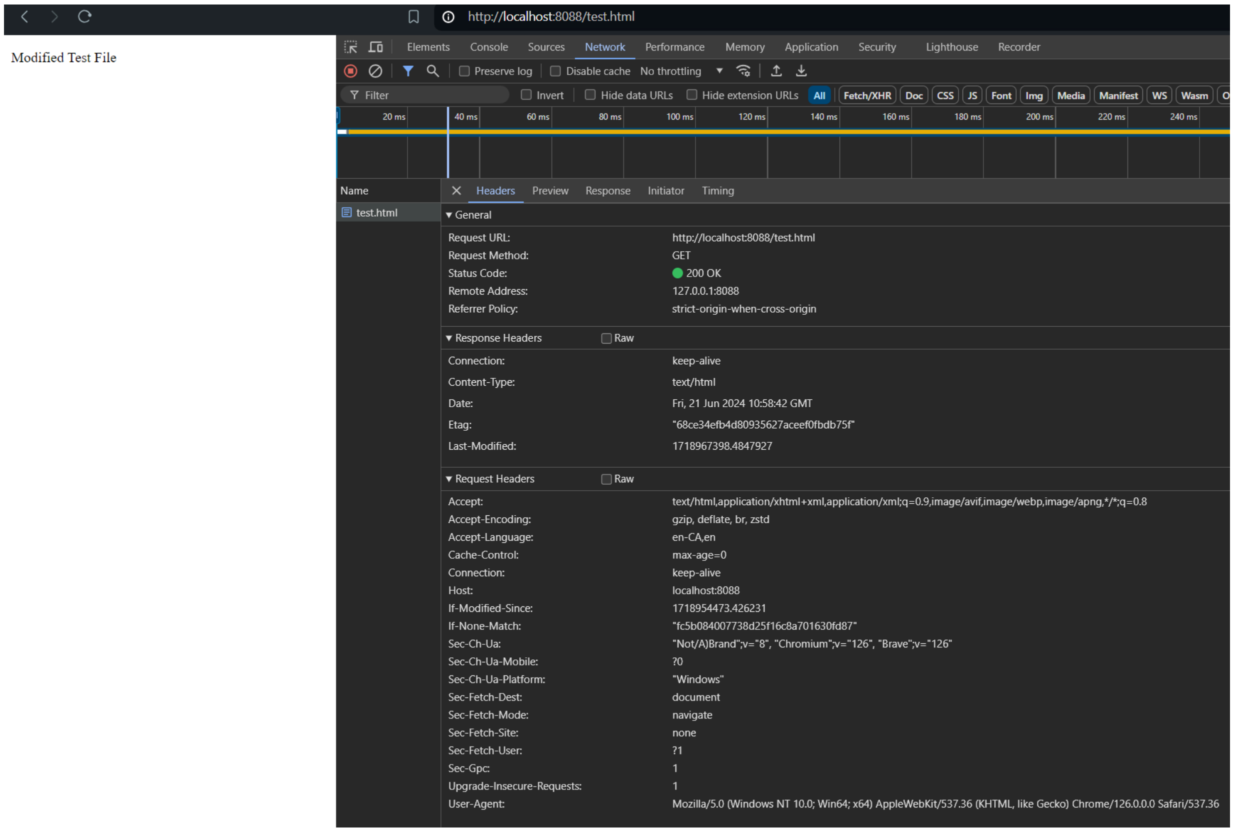
This worked as intended. If we request the same resource again, it should give a 304 Not Modified response because we have not modified it.



Congratulations! Your Web Server is Working!

```
▼ General
Request URL:               http://localhost:8088/test.html
Request Method:            GET
Status Code:               ● 304 Not Modified
Remote Address:            127.0.0.1:8088
Referrer Policy:           strict-origin-when-cross-origin

▼ Response Headers    ☐ Raw
Connection:                keep-alive
Date:                      Fri, 21 Jun 2024 10:53:55 GMT

▼ Request Headers    ☐ Raw
Accept:                    text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8
Accept-Encoding:           gzip, deflate, br, zstd
Accept-Language:           en-CA,en
Cache-Control:             max-age=0
Connection:                keep-alive
Host:                      localhost:8088
If-Modified-Since:         1718954473.426231
If-None-Match:             "fc5b084007738d25f16c8a701630fd87"
Sec-Ch-Ua:                 "Not/A)Brand";v="8", "Chromium";v="126", "Brave";v="126"
Sec-Ch-Ua-Mobile:          ?0
Sec-Ch-Ua-Platform:        "Windows"
Sec-Fetch-Dest:            document
Sec-Fetch-Mode:            navigate
Sec-Fetch-Site:            none
Sec-Fetch-User:            ?1
Sec-Gpc:                   1
Upgrade-Insecure-Requests: 1
User-Agent:                Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/126.0.0.0 Safari/537.36
```

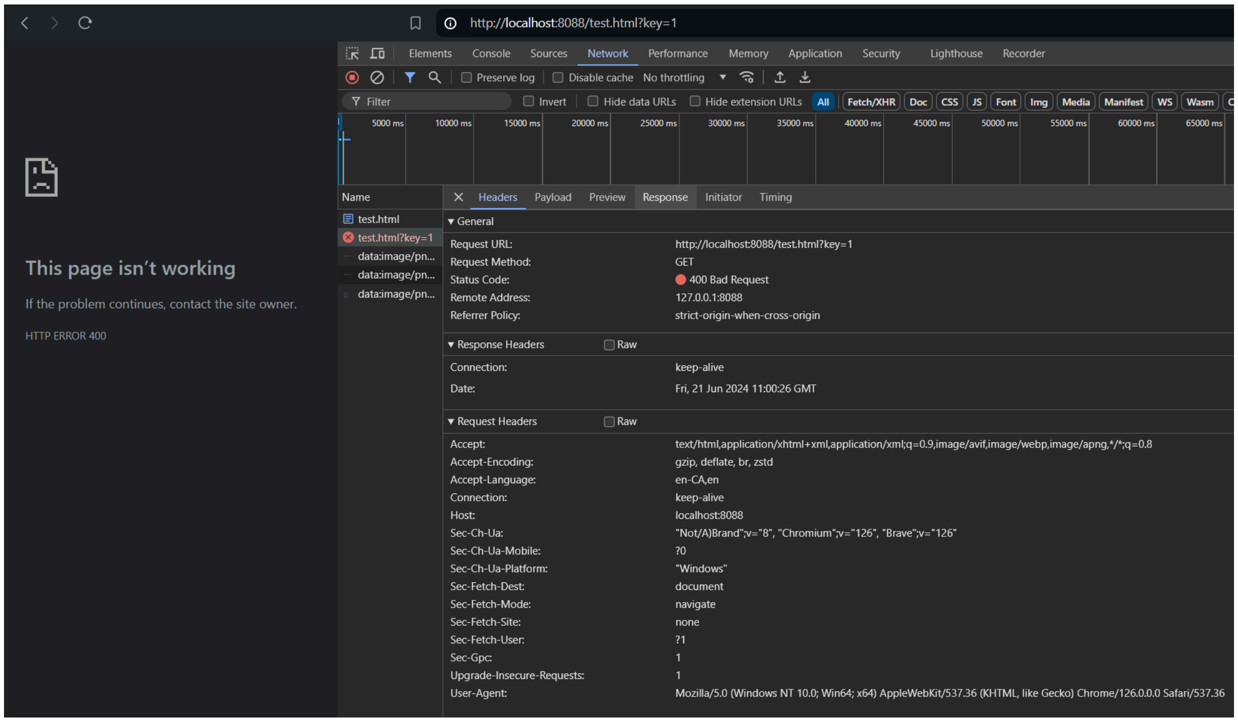
If we modify and save test.html on the server, then when we request test.html again, it should return the newly modified file along with the status code 200.


http://localhost:8088/test.html

Modified Test File

```
General
Request URL:              http://localhost:8088/test.html
Request Method:           GET
Status Code:              200 OK
Remote Address:           127.0.0.1:8088
Referrer Policy:          strict-origin-when-cross-origin

Response Headers
Connection:               keep-alive
Content-Type:             text/html
Date:                     Fri, 21 Jun 2024 10:58:42 GMT
Etag:                     "68ce34efb4d80935627aceef0fbdb75f"
Last-Modified:            1718967398.4847927

Request Headers
Accept:                   text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8
Accept-Encoding:          gzip, deflate, br, zstd
Accept-Language:          en-CA,en
Cache-Control:            max-age=0
Connection:               keep-alive
Host:                     localhost:8088
If-Modified-Since:        1718954473.426231
If-None-Match:            "fc5b084007738d25f16c8a701630fd87"
Sec-Ch-Ua:                "Not/A)Brand";v="8", "Chromium";v="126", "Brave";v="126"
Sec-Ch-Ua-Mobile:         ?0
Sec-Ch-Ua-Platform:       "Windows"
Sec-Fetch-Dest:           document
Sec-Fetch-Mode:           navigate
Sec-Fetch-Site:           none
Sec-Fetch-User:           ?1
Sec-Gpc:                  1
Upgrade-Insecure-Requests: 1
User-Agent:               Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/126.0.0.0 Safari/537.36
```

If we add some unsupported query strings to our request, then the server will return a 400 Bad Request response.


http://localhost:8088/test.html?key=1

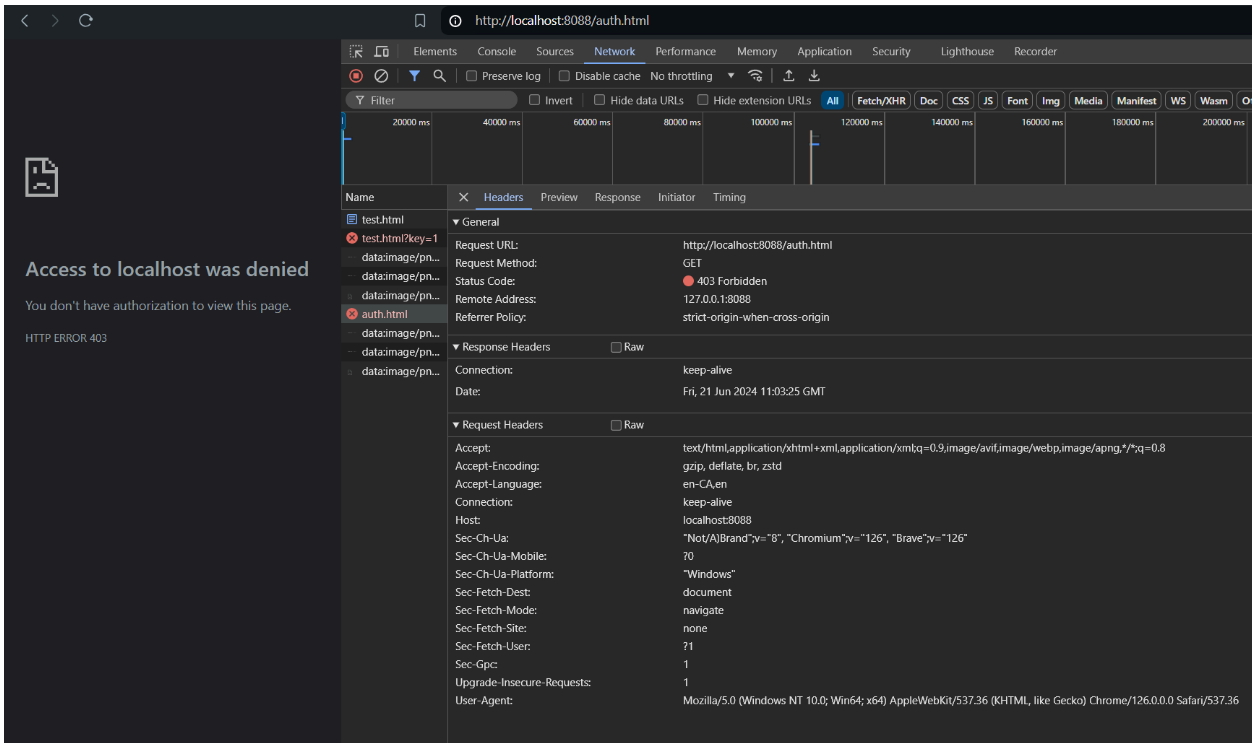
This page isn't working

If the problem continues, contact the site owner.

HTTP ERROR 400

```
General
Request URL:              http://localhost:8088/test.html?key=1
Request Method:           GET
Status Code:              400 Bad Request
Remote Address:           127.0.0.1:8088
Referrer Policy:          strict-origin-when-cross-origin

Response Headers
Connection:               keep-alive
Date:                     Fri, 21 Jun 2024 11:00:26 GMT

Request Headers
Accept:                   text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8
Accept-Encoding:          gzip, deflate, br, zstd
Accept-Language:          en-CA,en
Connection:               keep-alive
Host:                     localhost:8088
Sec-Ch-Ua:                "Not/A)Brand";v="8", "Chromium";v="126", "Brave";v="126"
Sec-Ch-Ua-Mobile:         ?0
Sec-Ch-Ua-Platform:       "Windows"
Sec-Fetch-Dest:           document
Sec-Fetch-Mode:           navigate
Sec-Fetch-Site:           none
Sec-Fetch-User:           ?1
Sec-Gpc:                  1
Upgrade-Insecure-Requests: 1
User-Agent:               Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/126.0.0.0 Safari/537.36
```

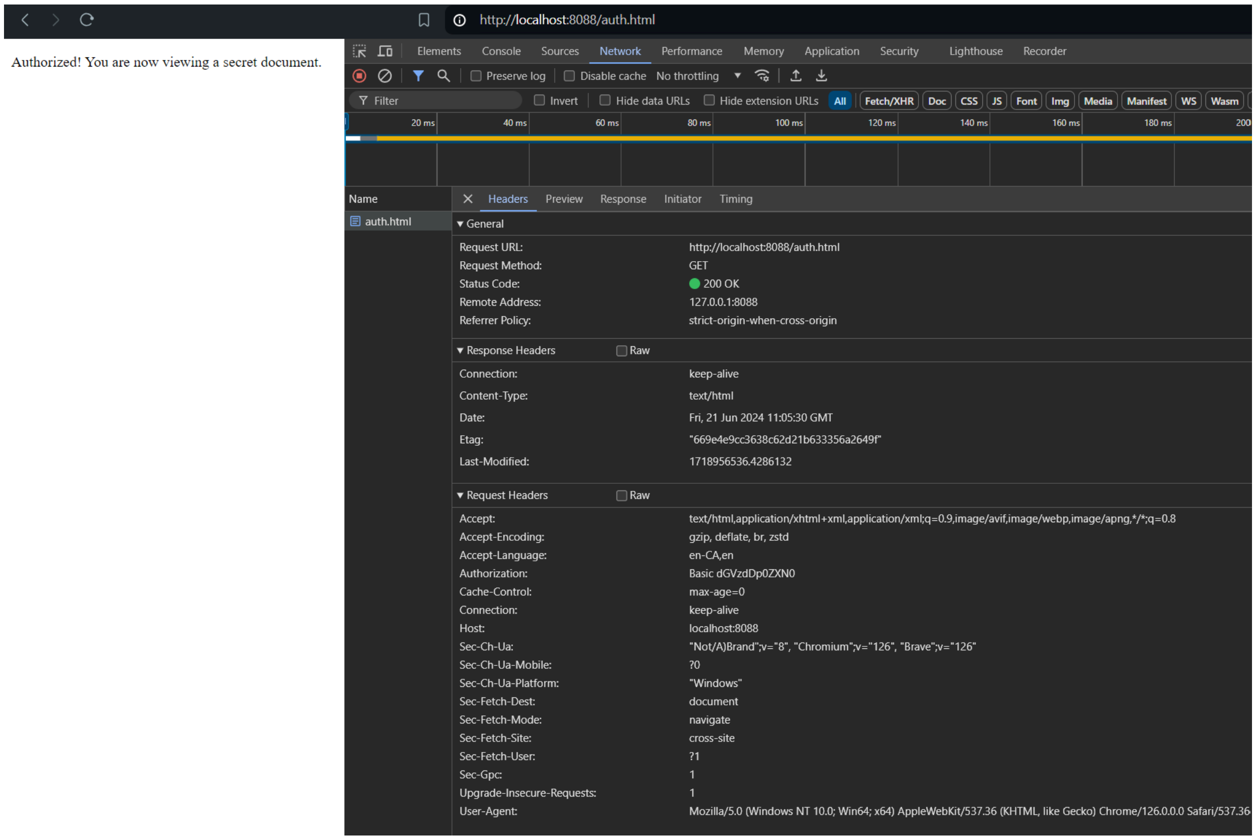
Let's try to access auth.html now.



```
http://localhost:8088/auth.html

Access to localhost was denied
You don't have authorization to view this page.
HTTP ERROR 403
```

```
Name
test.html
test.html?key=1
data:image/pn...
data:image/pn...
data:image/pn...
auth.html
data:image/pn...
data:image/pn...
data:image/pn...

▼ General
Request URL:                http://localhost:8088/auth.html
Request Method:             GET
Status Code:                ● 403 Forbidden
Remote Address:             127.0.0.1:8088
Referrer Policy:            strict-origin-when-cross-origin

▼ Response Headers
Connection:                 keep-alive
Date:                       Fri, 21 Jun 2024 11:03:25 GMT

▼ Request Headers
Accept:                     text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8
Accept-Encoding:            gzip, deflate, br, zstd
Accept-Language:            en-CA,en
Connection:                 keep-alive
Host:                       localhost:8088
Sec-Ch-Ua:                  "Not/A)Brand";v="8", "Chromium";v="126", "Brave";v="126"
Sec-Ch-Ua-Mobile:           ?0
Sec-Ch-Ua-Platform:         "Windows"
Sec-Fetch-Dest:             document
Sec-Fetch-Mode:             navigate
Sec-Fetch-Site:             none
Sec-Fetch-User:             ?1
Sec-Gpc:                    1
Upgrade-Insecure-Requests:  1
User-Agent:                 Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/126.0.0.0 Safari/537.36
```

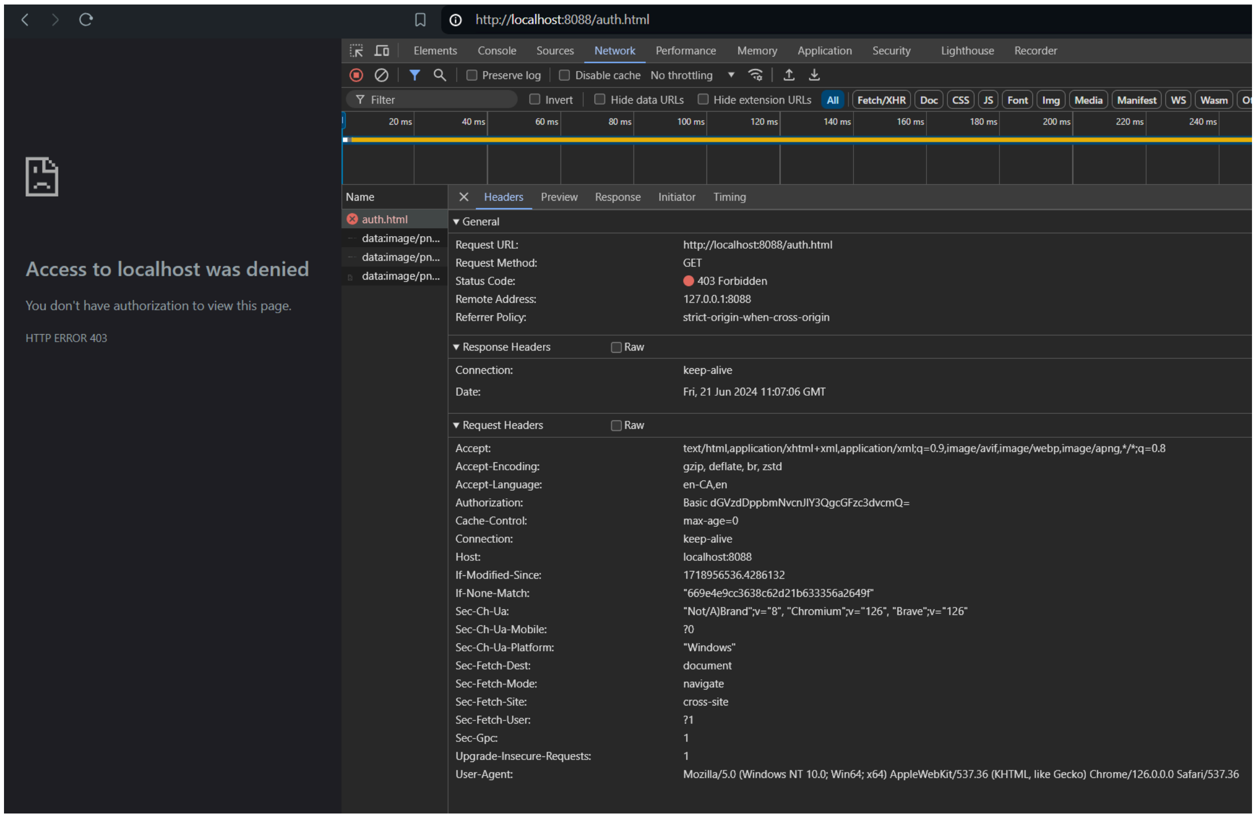
It throws a 403 Forbidden error because an authorization is not provided. Now, let's add the required authorization headers and values.



```
http://localhost:8088/auth.html

Authorized! You are now viewing a secret document.
```

```
Name
auth.html

▼ General
Request URL:                http://localhost:8088/auth.html
Request Method:             GET
Status Code:                ● 200 OK
Remote Address:             127.0.0.1:8088
Referrer Policy:            strict-origin-when-cross-origin

▼ Response Headers
Connection:                 keep-alive
Content-Type:               text/html
Date:                       Fri, 21 Jun 2024 11:05:30 GMT
Etag:                       "669e4e9cc3638c62d21b633356a2649f"
Last-Modified:              1718956536.4286132

▼ Request Headers
Accept:                     text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8
Accept-Encoding:            gzip, deflate, br, zstd
Accept-Language:            en-CA,en
Authorization:              Basic dGVzdDp0ZXN0
Cache-Control:              max-age=0
Connection:                 keep-alive
Host:                       localhost:8088
Sec-Ch-Ua:                  "Not/A)Brand";v="8", "Chromium";v="126", "Brave";v="126"
Sec-Ch-Ua-Mobile:           ?0
Sec-Ch-Ua-Platform:         "Windows"
Sec-Fetch-Dest:             document
Sec-Fetch-Mode:             navigate
Sec-Fetch-Site:             cross-site
Sec-Fetch-User:             ?1
Sec-Gpc:                    1
Upgrade-Insecure-Requests:  1
User-Agent:                 Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/126.0.0.0 Safari/537.36
```

If we enter an incorrect username or password, then 403 Forbidden is thrown again.



http://localhost:8088/auth.html

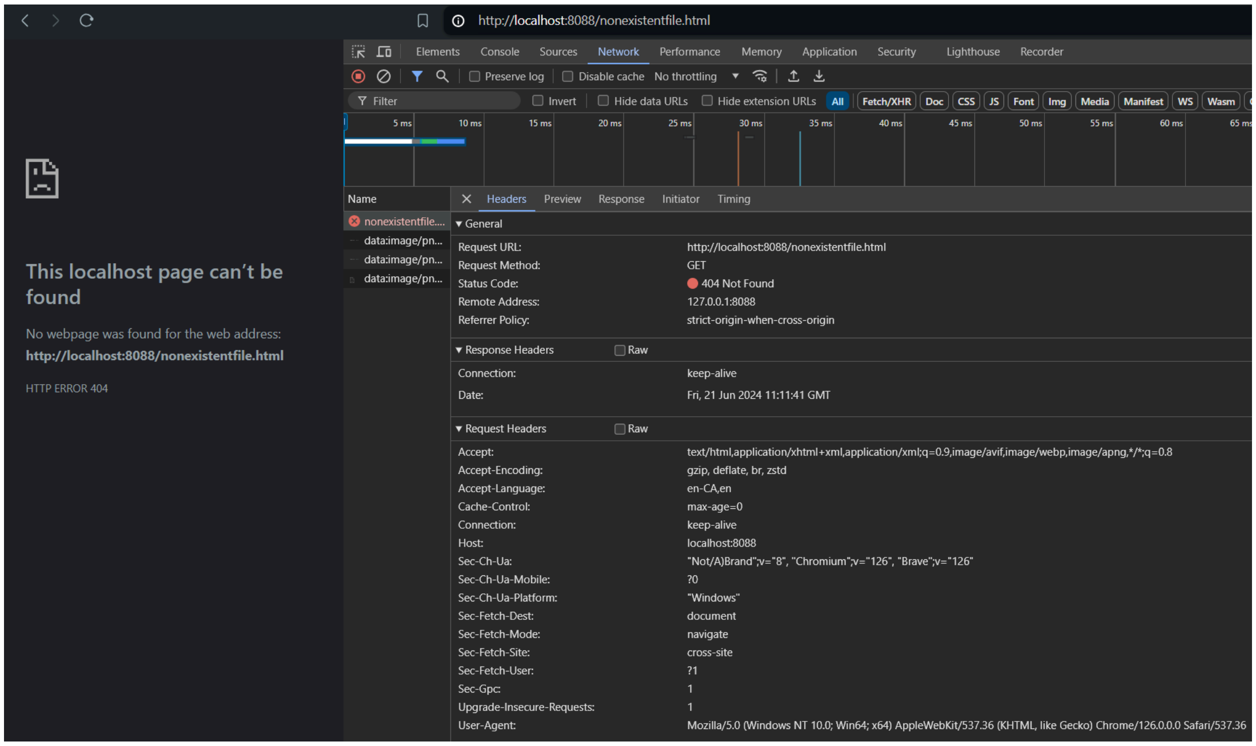
Access to localhost was denied

You don't have authorization to view this page.

HTTP ERROR 403

```
General
Request URL:                 http://localhost:8088/auth.html
Request Method:              GET
Status Code:                 ● 403 Forbidden
Remote Address:              127.0.0.1:8088
Referrer Policy:             strict-origin-when-cross-origin

Response Headers
Connection:                  keep-alive
Date:                        Fri, 21 Jun 2024 11:07:06 GMT

Request Headers
Accept:                      text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8
Accept-Encoding:             gzip, deflate, br, zstd
Accept-Language:             en-CA,en
Authorization:               Basic dGVzdDppbmNvcnJlY3QgcGFzc3dvcmQ=
Cache-Control:               max-age=0
Connection:                  keep-alive
Host:                        localhost:8088
If-Modified-Since:           1718956536.4286132
If-None-Match:               "669e4e9cc3638c62d21b633356a2649f"
Sec-Ch-Ua:                   "Not/A)Brand";v="8", "Chromium";v="126", "Brave";v="126"
Sec-Ch-Ua-Mobile:            ?0
Sec-Ch-Ua-Platform:          "Windows"
Sec-Fetch-Dest:              document
Sec-Fetch-Mode:              navigate
Sec-Fetch-Site:              cross-site
Sec-Fetch-User:              ?1
Sec-Gpc:                     1
Upgrade-Insecure-Requests:   1
User-Agent:                  Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/126.0.0.0 Safari/537.36
```

Finally, let's try to access a non-existent file.



http://localhost:8088/nonexistentfile.html

This localhost page can't be found

No webpage was found for the web address: **http://localhost:8088/nonexistentfile.html**

HTTP ERROR 404

```
General
Request URL:                 http://localhost:8088/nonexistentfile.html
Request Method:              GET
Status Code:                 ● 404 Not Found
Remote Address:              127.0.0.1:8088
Referrer Policy:             strict-origin-when-cross-origin

Response Headers
Connection:                  keep-alive
Date:                        Fri, 21 Jun 2024 11:11:41 GMT

Request Headers
Accept:                      text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8
Accept-Encoding:             gzip, deflate, br, zstd
Accept-Language:             en-CA,en
Cache-Control:               max-age=0
Connection:                  keep-alive
Host:                        localhost:8088
Sec-Ch-Ua:                   "Not/A)Brand";v="8", "Chromium";v="126", "Brave";v="126"
Sec-Ch-Ua-Mobile:            ?0
Sec-Ch-Ua-Platform:          "Windows"
Sec-Fetch-Dest:              document
Sec-Fetch-Mode:              navigate
Sec-Fetch-Site:              cross-site
Sec-Fetch-User:              ?1
Sec-Gpc:                     1
Upgrade-Insecure-Requests:   1
User-Agent:                  Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/126.0.0.0 Safari/537.36
```

This returned a 404 Not Found response.

Step Three: Proxy Server and Performance

In our minimal proxy server, we assume that the proxy always caches the newest version of a web server file. This simplifies the implementation and reduces the need for the proxy to contact the web server to check if its cache is up-to-date. The proxy will process non-authorization requests and return either 200 OK or 304 Not Modified if a copy exists in its cache. Otherwise, 400 Bad Request will be returned by the proxy if the request is invalid, and the remaining requests will be forwarded to the web server, which then returns a response to the proxy, and the proxy relays it to the client.

Our proxy server does not include additional features such as saving cache from the origin server and cache freshness. This is because they are not outlined in the project requirements or lecture slides.

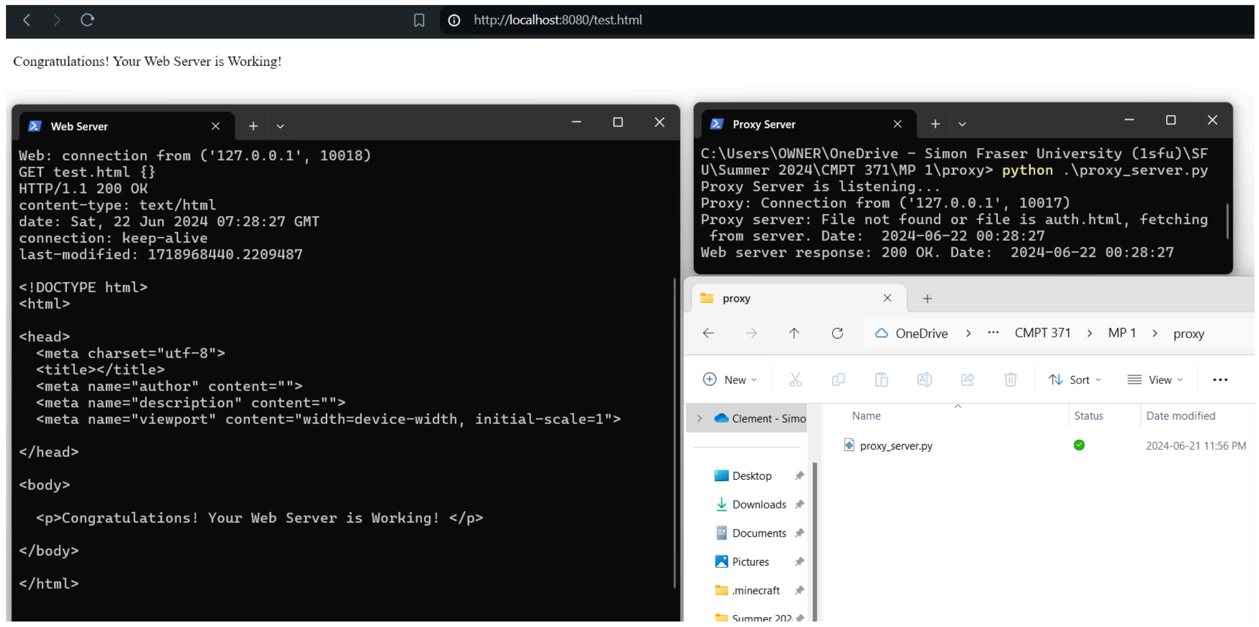
To run the proxy server, navigate to the proxy folder inside the zip folder and run the proxy_server.py file. The proxy folder will be the proxy server's cache directory, and any files placed inside the folder will be part of the proxy's cache. When running the proxy server, ensure that the main web server is also online.

Proxy Server Specifications

Host name: localhost or 127.0.0.1
Port: 8080
Socket: TCP
Threading: Multi-threaded
Supported response codes: 200, 304, 400
Forwarded requests: If cache miss or authorization-required files

Testing the proxy server

We can first try to request test.html on port 8080 when the proxy server cache is empty.



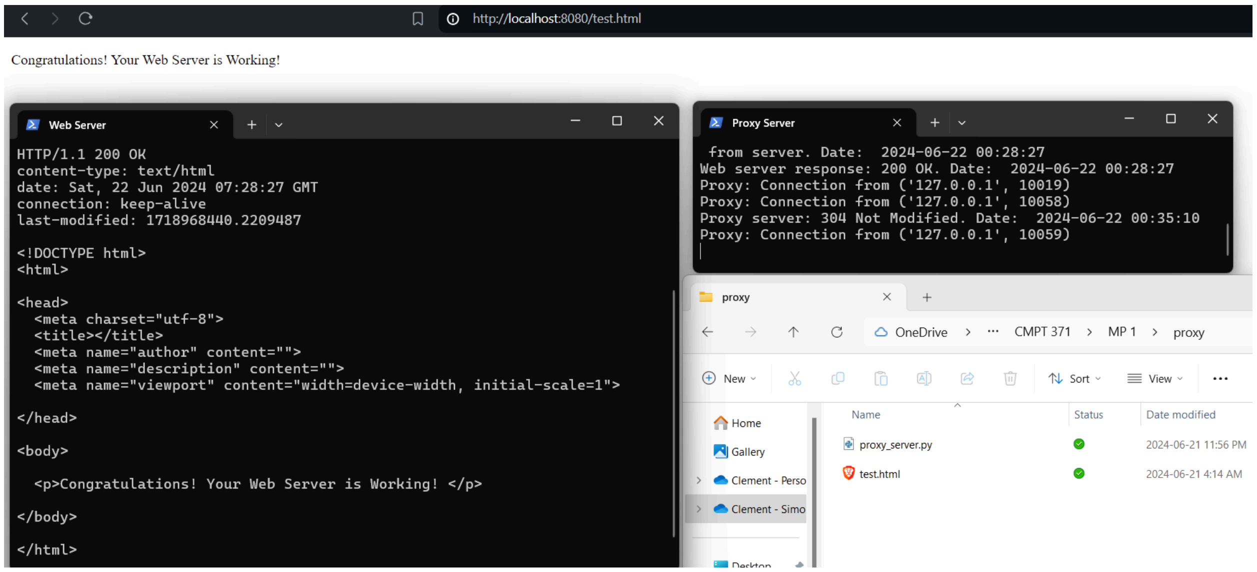
Congratulations! Your Web Server is Working!

```
Web: connection from ('127.0.0.1', 10018)
GET test.html {}
HTTP/1.1 200 OK
content-type: text/html
date: Sat, 22 Jun 2024 07:28:27 GMT
connection: keep-alive
last-modified: 1718968440.2209487

<!DOCTYPE html>
<html>

<head>
  <meta charset="utf-8">
  <title></title>
  <meta name="author" content="">
  <meta name="description" content="">
  <meta name="viewport" content="width=device-width, initial-scale=1">

</head>

<body>

  <p>Congratulations! Your Web Server is Working! </p>

</body>

</html>
```

```
C:\Users\OWNER\OneDrive - Simon Fraser University (1sfu)\SF
U\Summer 2024\CMPT 371\MP 1\proxy> python .\proxy_server.py
Proxy Server is listening...
Proxy: Connection from ('127.0.0.1', 10017)
Proxy server: File not found or file is auth.html, fetching
 from server. Date:  2024-06-22 00:28:27
Web server response: 200 OK. Date:  2024-06-22 00:28:27
```

proxy_server.py 2024-06-21 11:56 PM

From the server logs, we can see that the proxy server fails to locate the requested resource, and the request is fulfilled by the web server. The proxy receives the 200 OK response from the web server and relays it to the client. Now, let's try to add the same test.html to the proxy server cache.



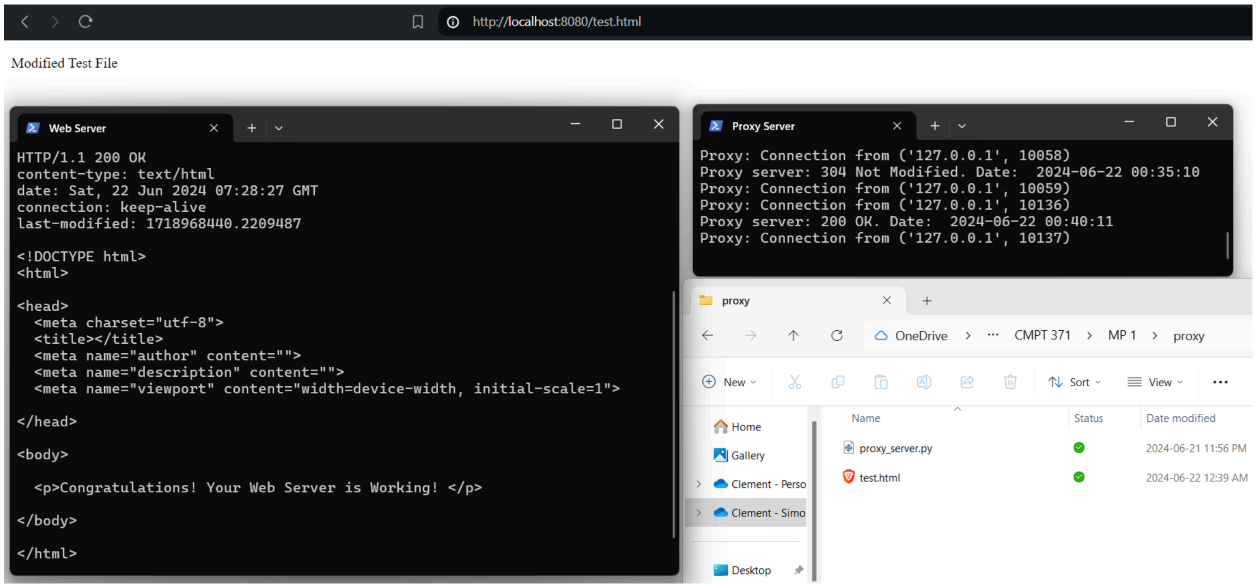
Congratulations! Your Web Server is Working!

```
HTTP/1.1 200 OK
content-type: text/html
date: Sat, 22 Jun 2024 07:28:27 GMT
connection: keep-alive
last-modified: 1718968440.2209487

<!DOCTYPE html>
<html>

<head>
  <meta charset="utf-8">
  <title></title>
  <meta name="author" content="">
  <meta name="description" content="">
  <meta name="viewport" content="width=device-width, initial-scale=1">

</head>

<body>

  <p>Congratulations! Your Web Server is Working! </p>

</body>

</html>
```

```
 from server. Date:  2024-06-22 00:28:27
Web server response: 200 OK. Date:  2024-06-22 00:28:27
Proxy: Connection from ('127.0.0.1', 10019)
Proxy: Connection from ('127.0.0.1', 10058)
Proxy server: 304 Not Modified. Date:  2024-06-22 00:35:10
Proxy: Connection from ('127.0.0.1', 10059)
```

Name	Status	Date modified
proxy_server.py	✓	2024-06-21 11:56 PM
test.html	✓	2024-06-21 4:14 AM

The proxy server processes the request by responding with a 304 Not Modified, while no new requests are sent to the web server. This works as expected because the test.html file is the same and was not modified. We can try to modify the test.html file in the cache, and it should return a 200 OK only on the proxy server.



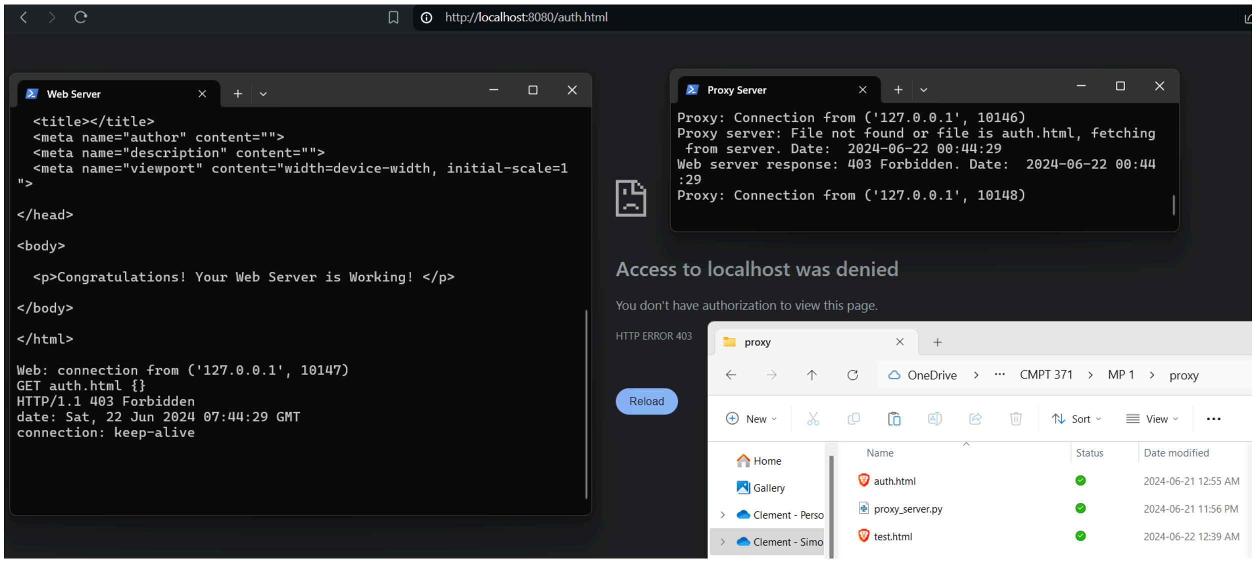
Modified Test File

```
HTTP/1.1 200 OK
content-type: text/html
date: Sat, 22 Jun 2024 07:28:27 GMT
connection: keep-alive
last-modified: 1718968440.2209487

<!DOCTYPE html>
<html>

<head>
  <meta charset="utf-8">
  <title></title>
  <meta name="author" content="">
  <meta name="description" content="">
  <meta name="viewport" content="width=device-width, initial-scale=1">

</head>

<body>

  <p>Congratulations! Your Web Server is Working! </p>

</body>

</html>
```

```
Proxy: Connection from ('127.0.0.1', 10058)
Proxy server: 304 Not Modified. Date:  2024-06-22 00:35:10
Proxy: Connection from ('127.0.0.1', 10059)
Proxy: Connection from ('127.0.0.1', 10136)
Proxy server: 200 OK. Date:  2024-06-22 00:40:11
Proxy: Connection from ('127.0.0.1', 10137)
```

Name	Status	Date modified
proxy_server.py	✓	2024-06-21 11:56 PM
test.html	✓	2024-06-22 12:39 AM

The proxy server responds as expected without invoking the web server. If we try to access the auth.html document (even if it is in the proxy cache somehow) with an invalid authorization, the proxy server should redirect the authentication process to the web server.



```
  <title></title>
  <meta name="author" content="">
  <meta name="description" content="">
  <meta name="viewport" content="width=device-width, initial-scale=1
">

</head>

<body>

  <p>Congratulations! Your Web Server is Working! </p>

</body>

</html>

Web: connection from ('127.0.0.1', 10147)
GET auth.html {}
HTTP/1.1 403 Forbidden
date: Sat, 22 Jun 2024 07:44:29 GMT
connection: keep-alive
```

```
Proxy: Connection from ('127.0.0.1', 10146)
Proxy server: File not found or file is auth.html, fetching
 from server. Date:  2024-06-22 00:44:29
Web server response: 403 Forbidden. Date:  2024-06-22 00:44
:29
Proxy: Connection from ('127.0.0.1', 10148)
```

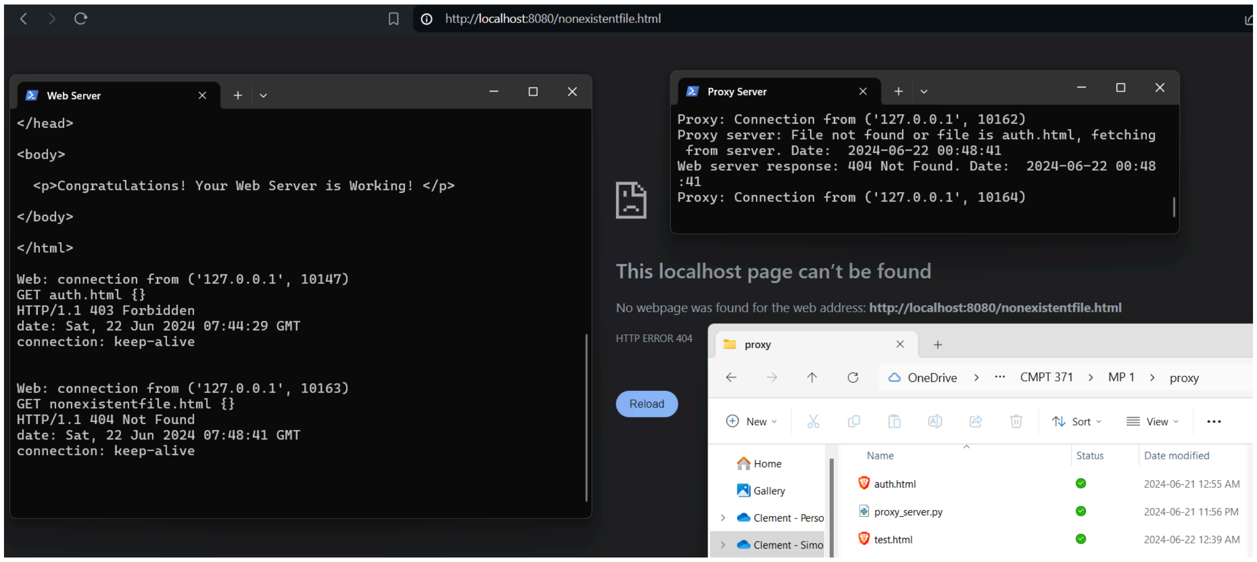
Access to localhost was denied

You don't have authorization to view this page.

HTTP ERROR 403

Name	Status	Date modified
auth.html	✓	2024-06-21 12:55 AM
proxy_server.py	✓	2024-06-21 11:56 PM
test.html	✓	2024-06-22 12:39 AM

As you can see, the proxy server redirects the request to the web server, which responds with a 403 Forbidden. Let's test with a non-existent file and see the 404 Not Found generation process.



```
</head>

<body>

  <p>Congratulations! Your Web Server is Working! </p>

</body>

</html>

Web: connection from ('127.0.0.1', 10147)
GET auth.html {}
HTTP/1.1 403 Forbidden
date: Sat, 22 Jun 2024 07:44:29 GMT
connection: keep-alive


Web: connection from ('127.0.0.1', 10163)
GET nonexistentfile.html {}
HTTP/1.1 404 Not Found
date: Sat, 22 Jun 2024 07:48:41 GMT
connection: keep-alive
```

```
Proxy: Connection from ('127.0.0.1', 10162)
Proxy server: File not found or file is auth.html, fetching
 from server. Date:  2024-06-22 00:48:41
Web server response: 404 Not Found. Date:  2024-06-22 00:48
:41
Proxy: Connection from ('127.0.0.1', 10164)
```

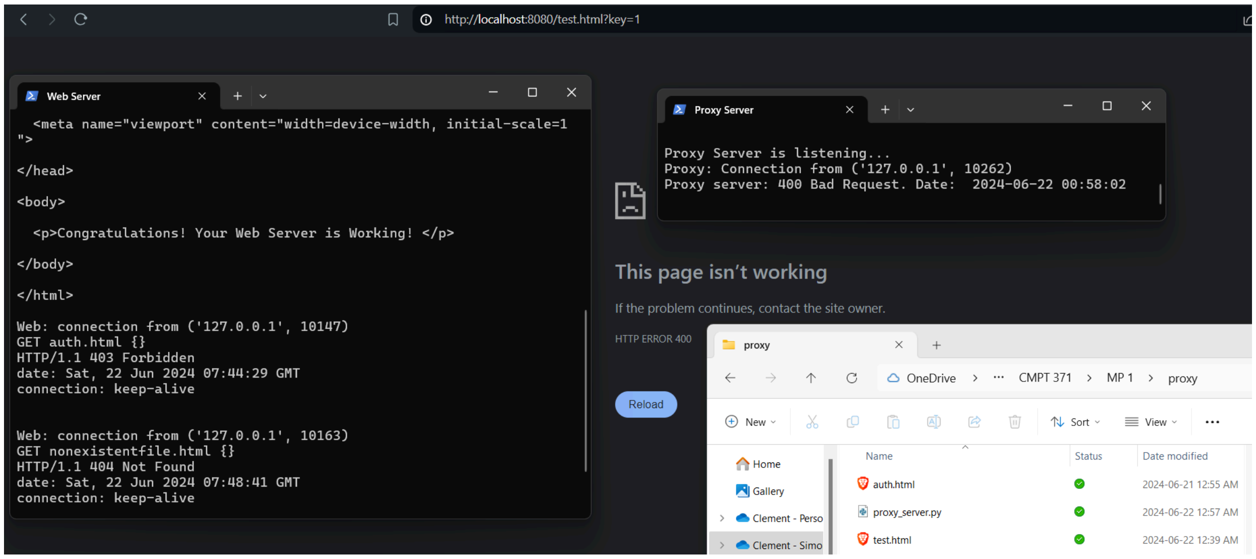
This localhost page can't be found

No webpage was found for the web address: **http://localhost:8080/nonexistentfile.html**

HTTP ERROR 404

Name	Status	Date modified
auth.html	✓	2024-06-21 12:55 AM
proxy_server.py	✓	2024-06-21 11:56 PM
test.html	✓	2024-06-22 12:39 AM

Once again, the proxy server forwards the request to the web server because the requested file was not found in its cache (i.e. a cache miss). The web server checks its directory as well and returns a 404 Not Found when the resource cannot be found. Finally, let's try with an invalid request. The proxy server should intercept the bad request and respond directly to the client.



```
  <meta name="viewport" content="width=device-width, initial-scale=1
">
</head>
<body>
  <p>Congratulations! Your Web Server is Working! </p>
</body>
</html>
Web: connection from ('127.0.0.1', 10147)
GET auth.html {}
HTTP/1.1 403 Forbidden
date: Sat, 22 Jun 2024 07:44:29 GMT
connection: keep-alive


Web: connection from ('127.0.0.1', 10163)
GET nonexistentfile.html {}
HTTP/1.1 404 Not Found
date: Sat, 22 Jun 2024 07:48:41 GMT
connection: keep-alive
```

```
Proxy Server is listening...
Proxy: Connection from ('127.0.0.1', 10262)
Proxy server: 400 Bad Request. Date:  2024-06-22 00:58:02
```

This page isn't working

If the problem continues, contact the site owner.

HTTP ERROR 400

Name	Status	Date modified
auth.html	✓	2024-06-21 12:55 AM
proxy_server.py	✓	2024-06-22 12:57 AM
test.html	✓	2024-06-22 12:39 AM

The proxy server handles the bad request directly without calling the web server.

Multi-threading

Our server uses multi-threading for each TCP connection. New threads are created when a new incoming connection is intercepted. This is done in the main function of our server code using Python's threading.Thread function:

```
    while True:
        # Establish a connection
        client_socket, addr = server_socket.accept()
        # main process
        threading.Thread(target=handle_client, args=(client_socket,
addr)).start()
```

After a connection is closed, the thread is closed as well.

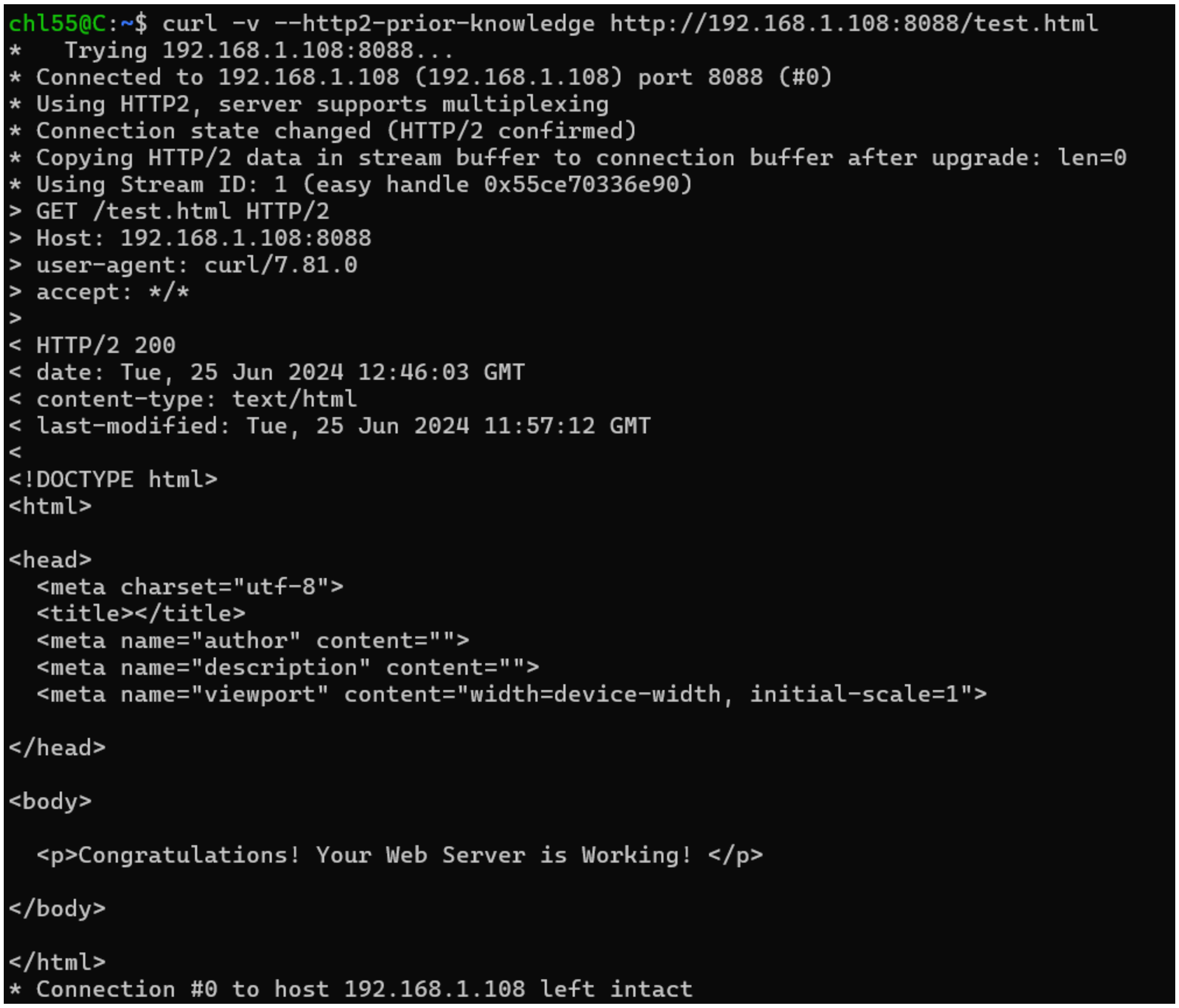
In terms of performance, multi-threading significantly speeds up the server processing time if there are multiple connections, as each connection can be handled concurrently. A single-threaded server will need to handle and finish a connection before processing another, which can create long queues if a server has a high traffic volume.

Step Four: HOL Problem

According to module 2, slide 37, using HTTP/2 can be an effective approach to mitigate and avoid HOL blocking. Therefore, we rewrote our web server using the hyper-h2 library to serve HTTP/2 instead of HTTP/1.1. The new server can be found in the zip file under the name web_server_http2.py. Note that it will not be compatible with the proxy as it serves HTTP/1.1 instead. The HTTP/2 server has the same specifications as the HTTP/1.1 web server, except for the HTTP version.

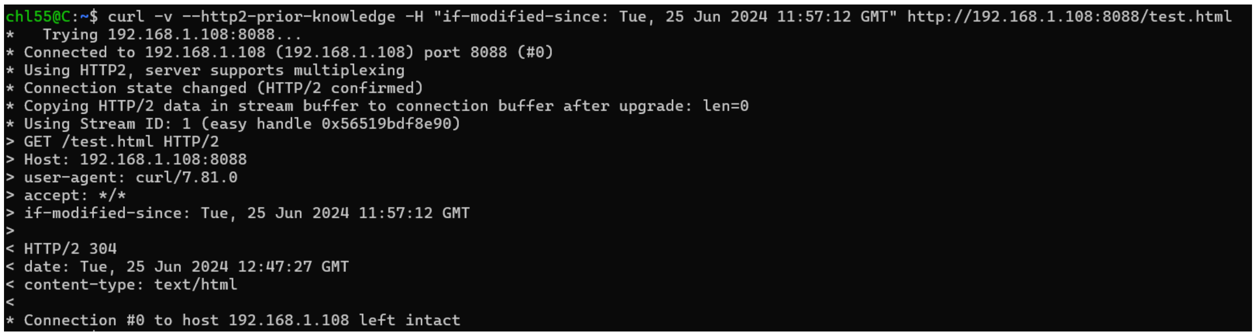
To run the server, you need to install the hyper-h2 library using the command `pip install h2`. In addition, the server will most likely not work in browsers due to them sending HTTP/1.1

requests initially. As a result, you will need to use the curl command line tool along with the --http2-prior-knowledge flag. The following screenshots show some functionalities of our HTTP/2 server. (Note that the IP is 192.168.1.108 instead of localhost because of WSL restrictions).



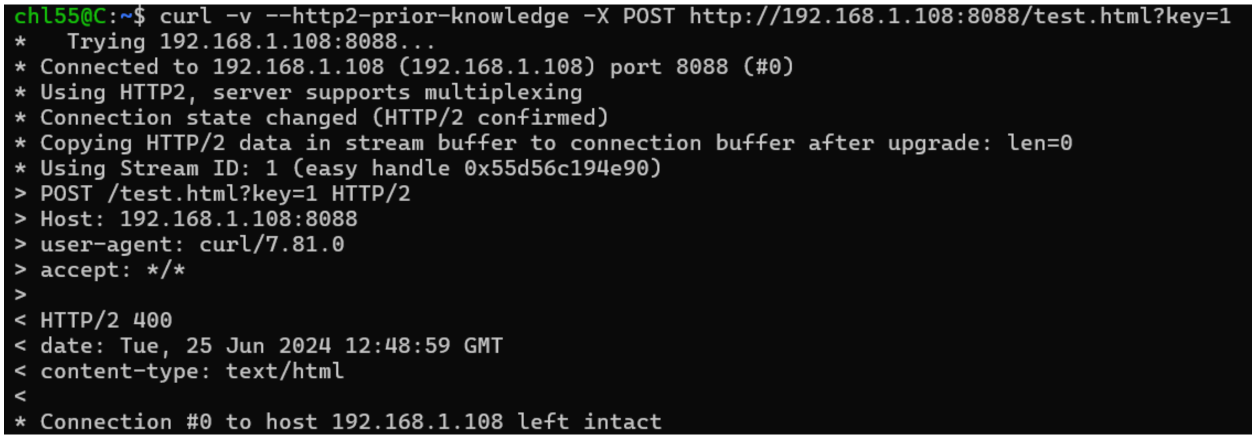
```
chl55@C:~$ curl -v --http2-prior-knowledge http://192.168.1.108:8088/test.html
*   Trying 192.168.1.108:8088...
* Connected to 192.168.1.108 (192.168.1.108) port 8088 (#0)
* Using HTTP2, server supports multiplexing
* Connection state changed (HTTP/2 confirmed)
* Copying HTTP/2 data in stream buffer to connection buffer after upgrade: len=0
* Using Stream ID: 1 (easy handle 0x55ce70336e90)
> GET /test.html HTTP/2
> Host: 192.168.1.108:8088
> user-agent: curl/7.81.0
> accept: */*
>
< HTTP/2 200
< date: Tue, 25 Jun 2024 12:46:03 GMT
< content-type: text/html
< last-modified: Tue, 25 Jun 2024 11:57:12 GMT
<
<!DOCTYPE html>
<html>

<head>
  <meta charset="utf-8">
  <title></title>
  <meta name="author" content="">
  <meta name="description" content="">
  <meta name="viewport" content="width=device-width, initial-scale=1">

</head>

<body>

  <p>Congratulations! Your Web Server is Working! </p>

</body>

</html>
* Connection #0 to host 192.168.1.108 left intact
```

Simple GET for 200 OK



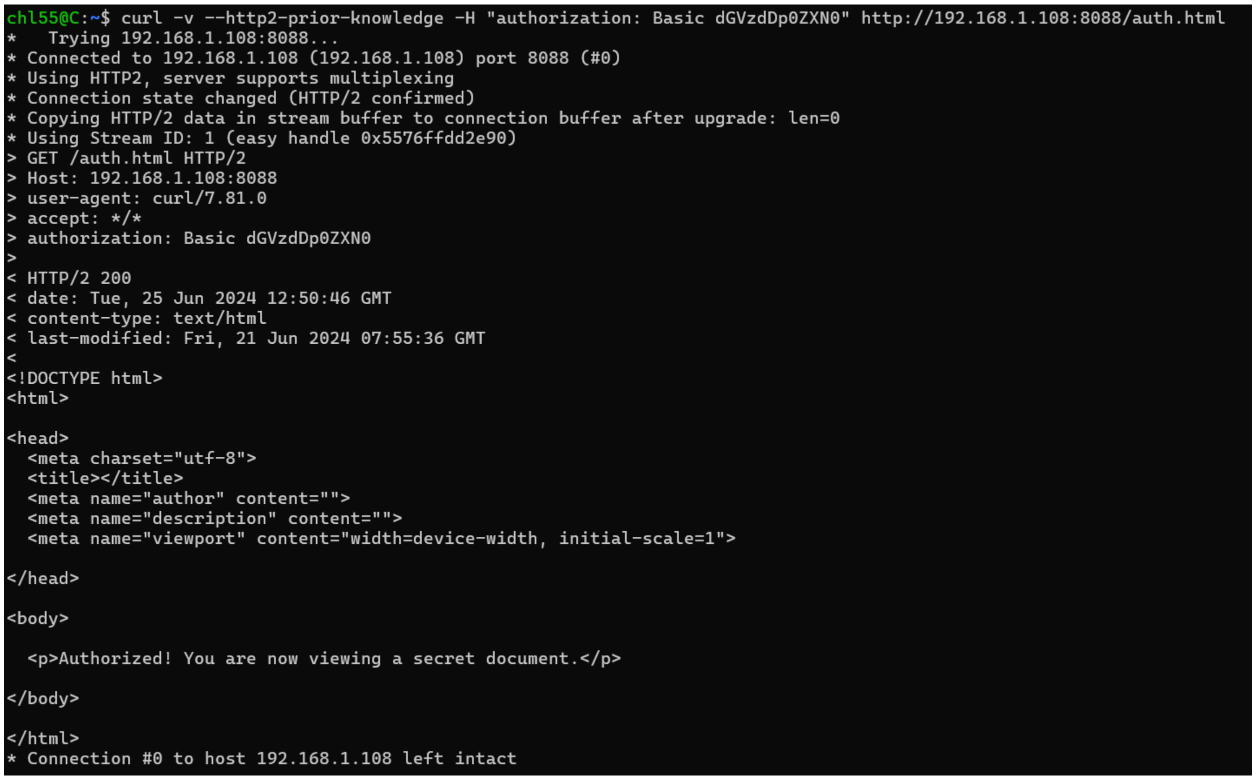
```
chl55@C:~$ curl -v --http2-prior-knowledge -H "if-modified-since: Tue, 25 Jun 2024 11:57:12 GMT" http://192.168.1.108:8088/test.html
*   Trying 192.168.1.108:8088...
* Connected to 192.168.1.108 (192.168.1.108) port 8088 (#0)
* Using HTTP2, server supports multiplexing
* Connection state changed (HTTP/2 confirmed)
* Copying HTTP/2 data in stream buffer to connection buffer after upgrade: len=0
* Using Stream ID: 1 (easy handle 0x56519bdf8e90)
> GET /test.html HTTP/2
> Host: 192.168.1.108:8088
> user-agent: curl/7.81.0
> accept: */*
> if-modified-since: Tue, 25 Jun 2024 11:57:12 GMT
>
< HTTP/2 304
< date: Tue, 25 Jun 2024 12:47:27 GMT
< content-type: text/html
<
* Connection #0 to host 192.168.1.108 left intact
```

Another GET for 304 Not Modified



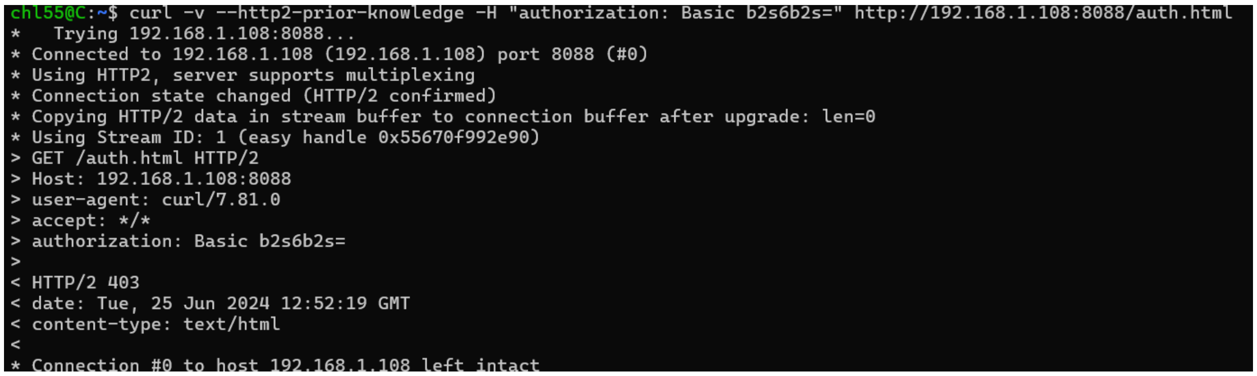
```
chl55@C:~$ curl -v --http2-prior-knowledge -X POST http://192.168.1.108:8088/test.html?key=1
*   Trying 192.168.1.108:8088...
* Connected to 192.168.1.108 (192.168.1.108) port 8088 (#0)
* Using HTTP2, server supports multiplexing
* Connection state changed (HTTP/2 confirmed)
* Copying HTTP/2 data in stream buffer to connection buffer after upgrade: len=0
* Using Stream ID: 1 (easy handle 0x55d56c194e90)
> POST /test.html?key=1 HTTP/2
> Host: 192.168.1.108:8088
> user-agent: curl/7.81.0
> accept: */*
>
< HTTP/2 400
< date: Tue, 25 Jun 2024 12:48:59 GMT
< content-type: text/html
<
* Connection #0 to host 192.168.1.108 left intact
```

A 400 Bad Request using the POST method and unsupported query strings



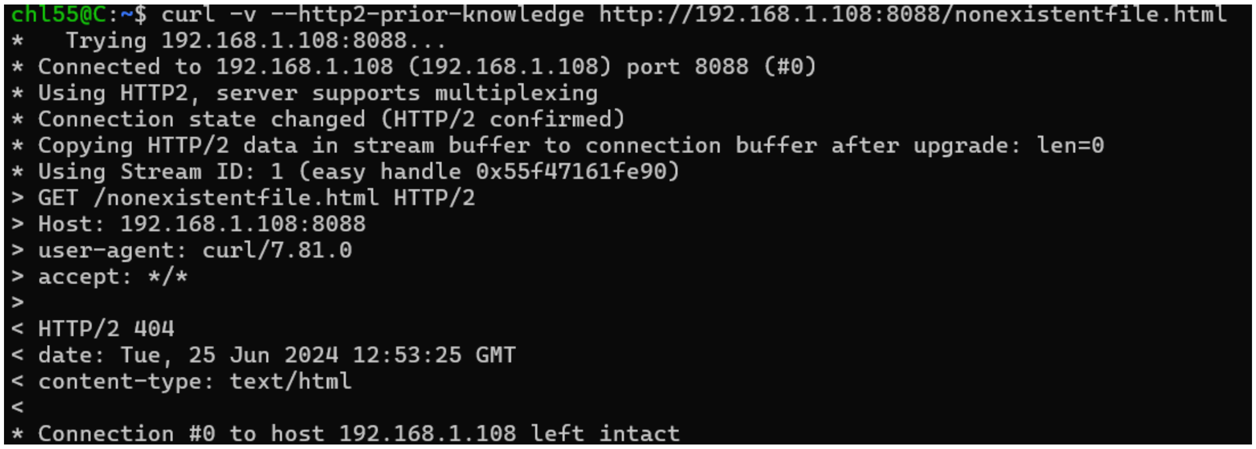
```
chl55@C:~$ curl -v --http2-prior-knowledge -H "authorization: Basic dGVzdDp0ZXN0" http://192.168.1.108:8088/auth.html
*   Trying 192.168.1.108:8088...
* Connected to 192.168.1.108 (192.168.1.108) port 8088 (#0)
* Using HTTP2, server supports multiplexing
* Connection state changed (HTTP/2 confirmed)
* Copying HTTP/2 data in stream buffer to connection buffer after upgrade: len=0
* Using Stream ID: 1 (easy handle 0x5576ffdd2e90)
> GET /auth.html HTTP/2
> Host: 192.168.1.108:8088
> user-agent: curl/7.81.0
> accept: */*
> authorization: Basic dGVzdDp0ZXN0
>
< HTTP/2 200
< date: Tue, 25 Jun 2024 12:50:46 GMT
< content-type: text/html
< last-modified: Fri, 21 Jun 2024 07:55:36 GMT
<
<!DOCTYPE html>
<html>

<head>
  <meta charset="utf-8">
  <title></title>
  <meta name="author" content="">
  <meta name="description" content="">
  <meta name="viewport" content="width=device-width, initial-scale=1">

</head>

<body>

  <p>Authorized! You are now viewing a secret document.</p>

</body>

</html>
* Connection #0 to host 192.168.1.108 left intact
```

Successful access to auth.html using the correct authorization



```
chl55@C:~$ curl -v --http2-prior-knowledge -H "authorization: Basic b2s6b2s=" http://192.168.1.108:8088/auth.html
*   Trying 192.168.1.108:8088...
* Connected to 192.168.1.108 (192.168.1.108) port 8088 (#0)
* Using HTTP2, server supports multiplexing
* Connection state changed (HTTP/2 confirmed)
* Copying HTTP/2 data in stream buffer to connection buffer after upgrade: len=0
* Using Stream ID: 1 (easy handle 0x55670f992e90)
> GET /auth.html HTTP/2
> Host: 192.168.1.108:8088
> user-agent: curl/7.81.0
> accept: */*
> authorization: Basic b2s6b2s=
>
< HTTP/2 403
< date: Tue, 25 Jun 2024 12:52:19 GMT
< content-type: text/html
<
* Connection #0 to host 192.168.1.108 left intact
```

403 Forbidden from incorrect authorization



```
chl55@C:~$ curl -v --http2-prior-knowledge http://192.168.1.108:8088/nonexistentfile.html
*   Trying 192.168.1.108:8088...
* Connected to 192.168.1.108 (192.168.1.108) port 8088 (#0)
* Using HTTP2, server supports multiplexing
* Connection state changed (HTTP/2 confirmed)
* Copying HTTP/2 data in stream buffer to connection buffer after upgrade: len=0
* Using Stream ID: 1 (easy handle 0x55f47161fe90)
> GET /nonexistentfile.html HTTP/2
> Host: 192.168.1.108:8088
> user-agent: curl/7.81.0
> accept: */*
>
< HTTP/2 404
< date: Tue, 25 Jun 2024 12:53:25 GMT
< content-type: text/html
<
* Connection #0 to host 192.168.1.108 left intact
```

404 Not Found for an non-existent file

Reference

We would like to acknowledge that we referenced the hyper-h2 documentation (https://python-hyper.org/projects/h2/en/stable/basic-usage.html) for parts of the implementation phase for our HTTP/2 server. Any additional features compared to the tutorial are our own work.