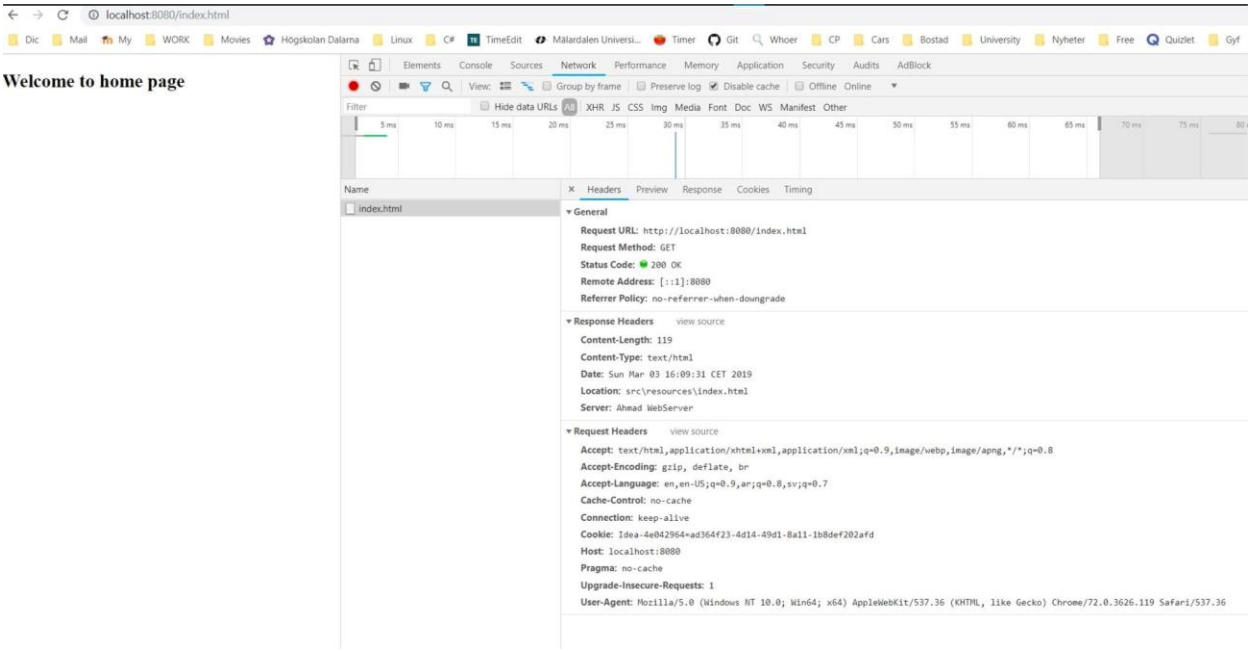
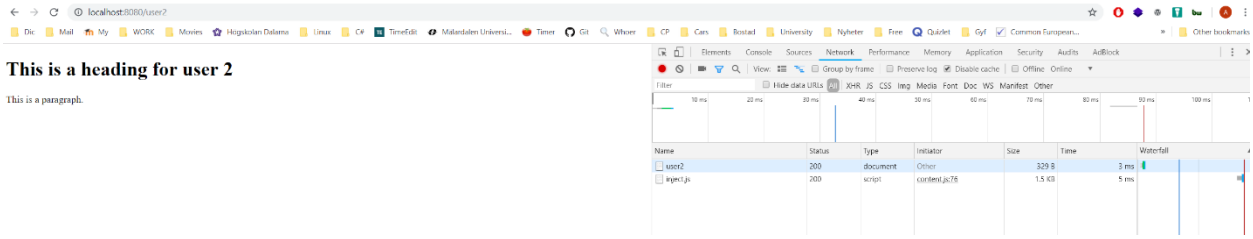
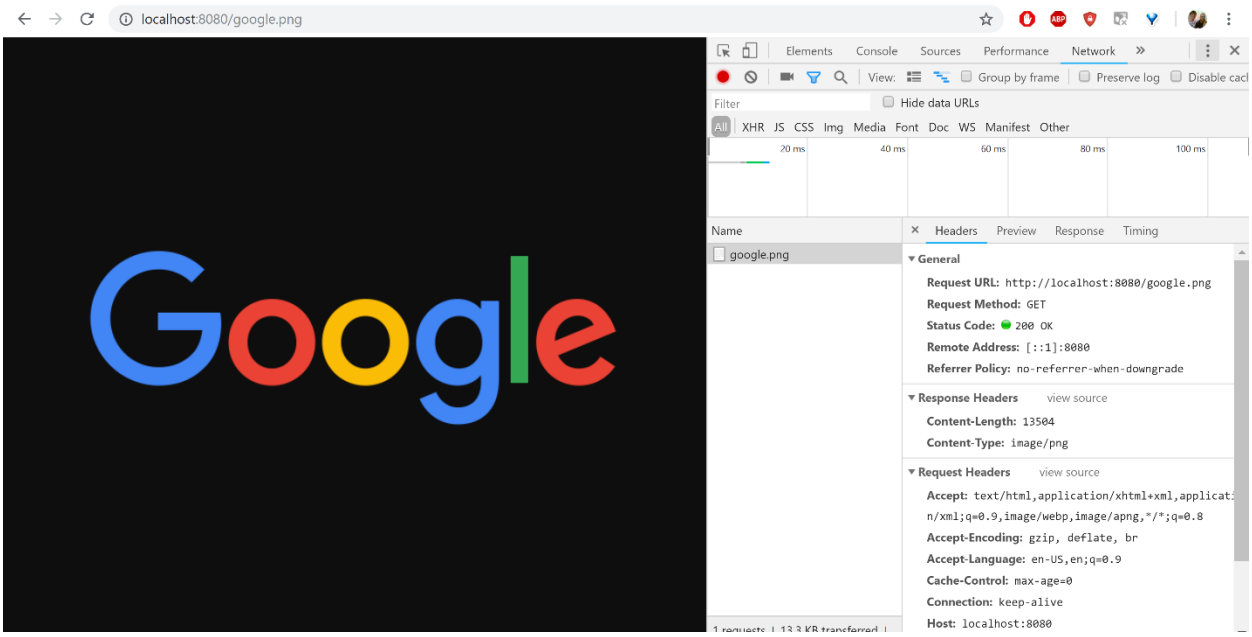
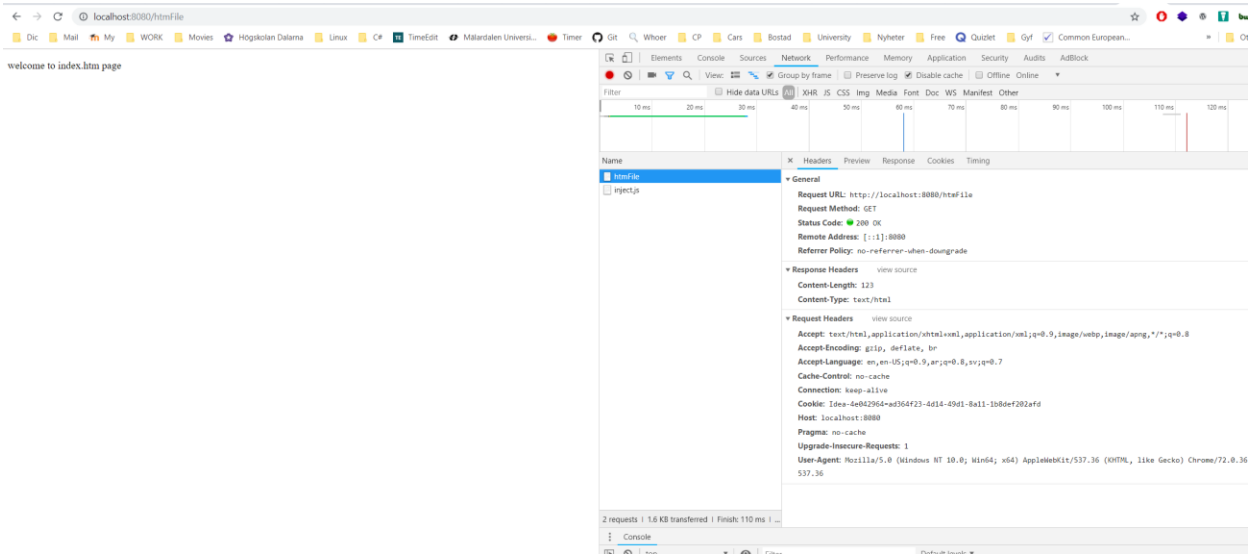
Problem 1

the screenshots listed below show the output of a GET request:





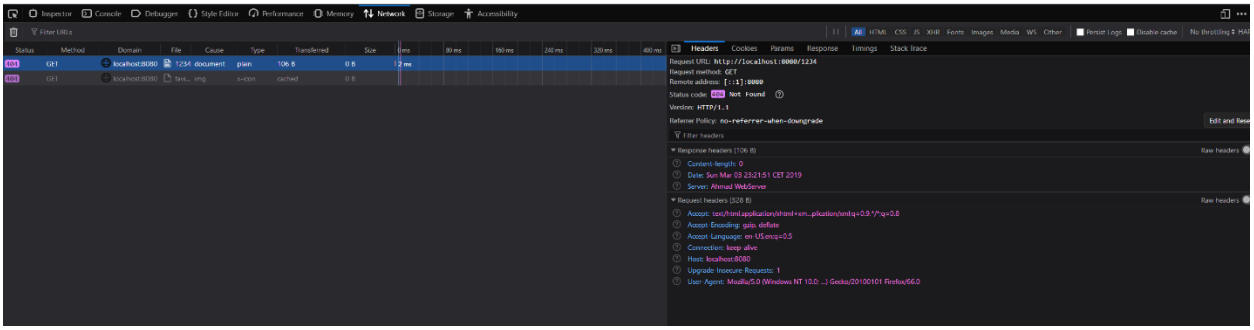

localhost:8080/google.png
Google
Elements
Console
Sources
Performance
Network
Filter
Hide data URLs
All XHR JS CSS Img Media Font Doc WS Manifest Other
20 ms
40 ms
60 ms
80 ms
100 ms
Name
google.png
Headers
Preview
Response
Timing
General
Request URL: http://localhost:8080/google.png
Request Method: GET
Status Code: 200 OK
Remote Address: [::1]:8080
Referrer Policy: no-referrer-when-downgrade
Response Headers
view source
Content-Length: 13504
Content-Type: image/png
Request Headers
view source
Accept: text/html,application/xhtml+xml,applicat
n/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Cache-Control: max-age=0
Connection: keep-alive
Host: localhost:8080


localhost:8080/htmFile
welcome to index.htm page
htmFile
inject.js
Request URL: http://localhost:8080/htmFile
Request Method: GET
Status Code: 200 OK
Remote Address: [::1]:8080
Referrer Policy: no-referrer-when-downgrade
Content-Length: 123
Content-Type: text/html

Problem 2

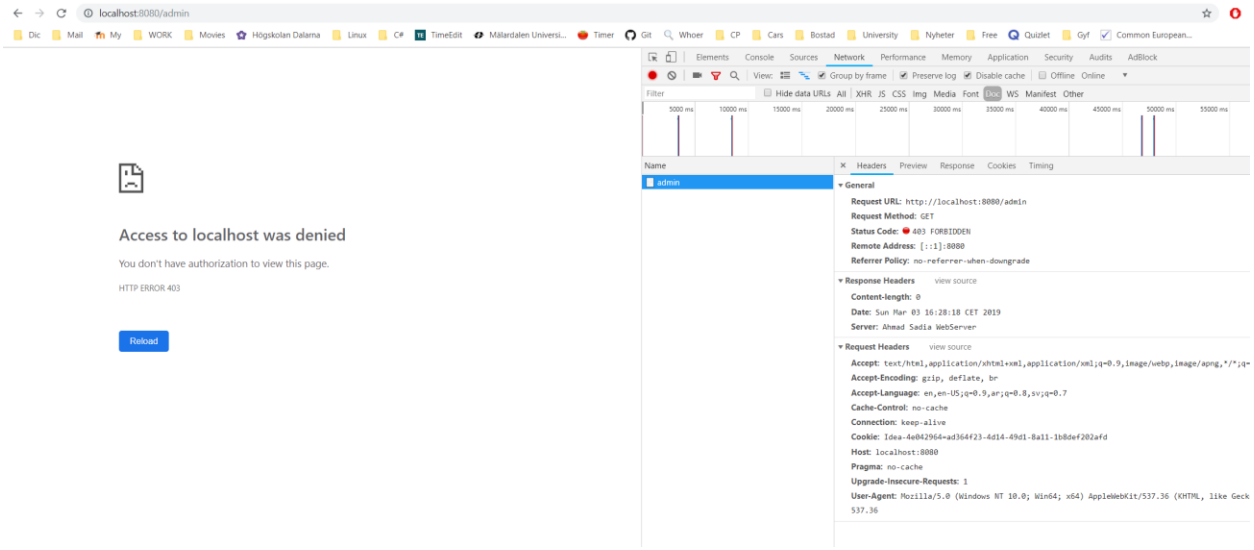
404 Not Found

The picture below shows the result when a client tries to access a file that is not available in the directory.



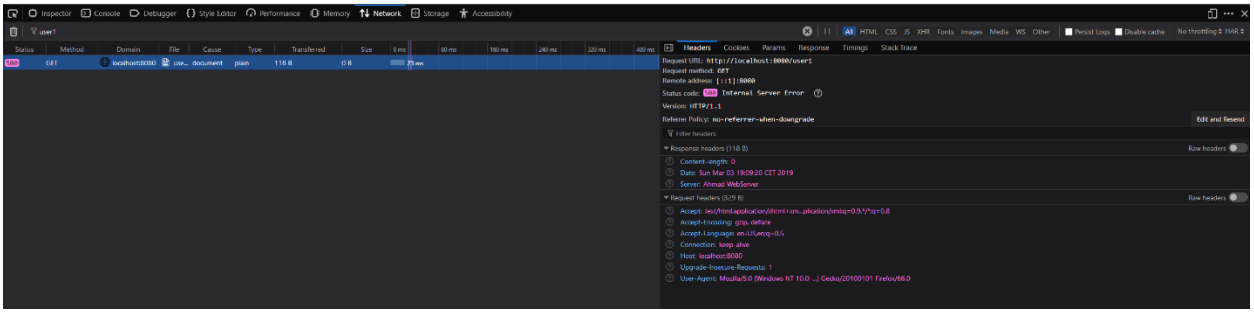
403 Forbidden

the screenshot below shows the results of trying to access the admin file with a response of 403. The admin file directory was restricted if the client enters localhost:8080/admin or localhost:8080/admin/ in the path.



500 Internal Server Error

The screenshot below shows the results of an Internal server error. To get a screenshot of an Internal Server Error from the browser we had to generate it by ourselves, because it is tough to catch an internal server error



VG-task 2

Put a text file 1.txt using curl

```
C:\Users\Ahmad\Desktop\curl-7.64.0-win64-mingw\bin>curl -T C:\1.txt localhost:8080
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100    14    0     0  100    14      0      1  0:00:14  0:00:09  0:00:05     0
```

PUT a png image 4.png using curl

```
curl: try 'curl --help' or 'curl --manual' for more information

C:\Users\Ahmad\Desktop\curl-7.64.0-win64-mingw\bin>curl -T C:\4.png localhost:8080
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  209k    0     0  100  209k      0   196k  0:00:01  0:00:01 --:--:--  196k

C:\Users\Ahmad\Desktop\curl-7.64.0-win64-mingw\bin>
```

Post vs Put

Put:

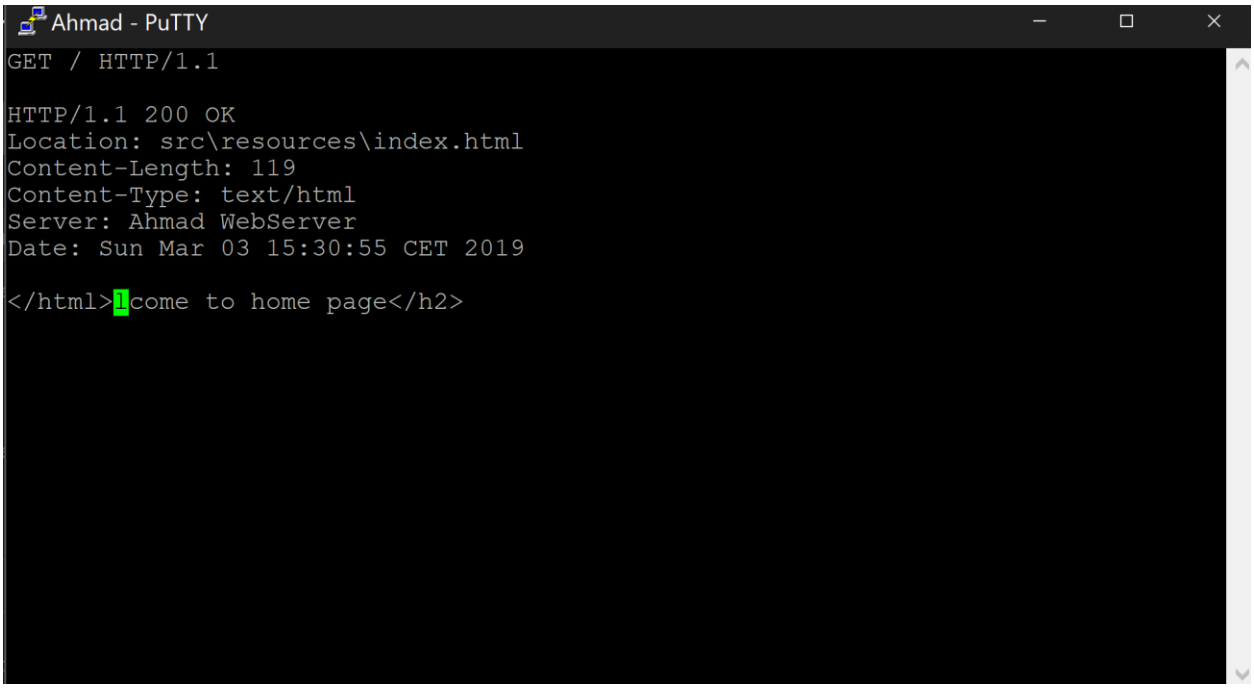
Used for replacing the existing resources or creating new incase if the resources does not exist. It is idempotent which means calling the request once or multiply times has the same effect the request is carried on without modifying the results beyond the initial application. It is repeatable and same, that means that for example if we have x= 1 if we repeat it many times the result will be the same x= 1.

Post:

Used for creating new resources. It is non-idempotent which means that calling the request once or multiply times may have additional effects the request is carried on and may modify the results beyond the initial application. It is non-repeatable and if requested multiply times an alert message is presented by the browser. It is not safe and non-repeatable due to the method producing several copies of the same resources.

Problem 3

200 OK



Using the GET / *HTTP*/1.1 we get the index file in the root.

the headers response:

Version of the HTTP 1.1.

Response status: 200 OK.

Location: the path of the file requested by the client.

Content-Length: the size of the file in bytes.

Content-Type: the type of the file requested which is text/html which indicate the browser that the file of a text type.

Server: name of the server

Date: the time and date when the request was created.

The last row is the content of index.html file (the response body).

Using the GET /user1 *HTTP*/1.1 we get the index file in the file user1.

the headers response:

Version of the HTTP 1.1.

Response status: 200 OK.

Location: the path of the file requested by the client.

Content-Length: the size of the file in bytes.

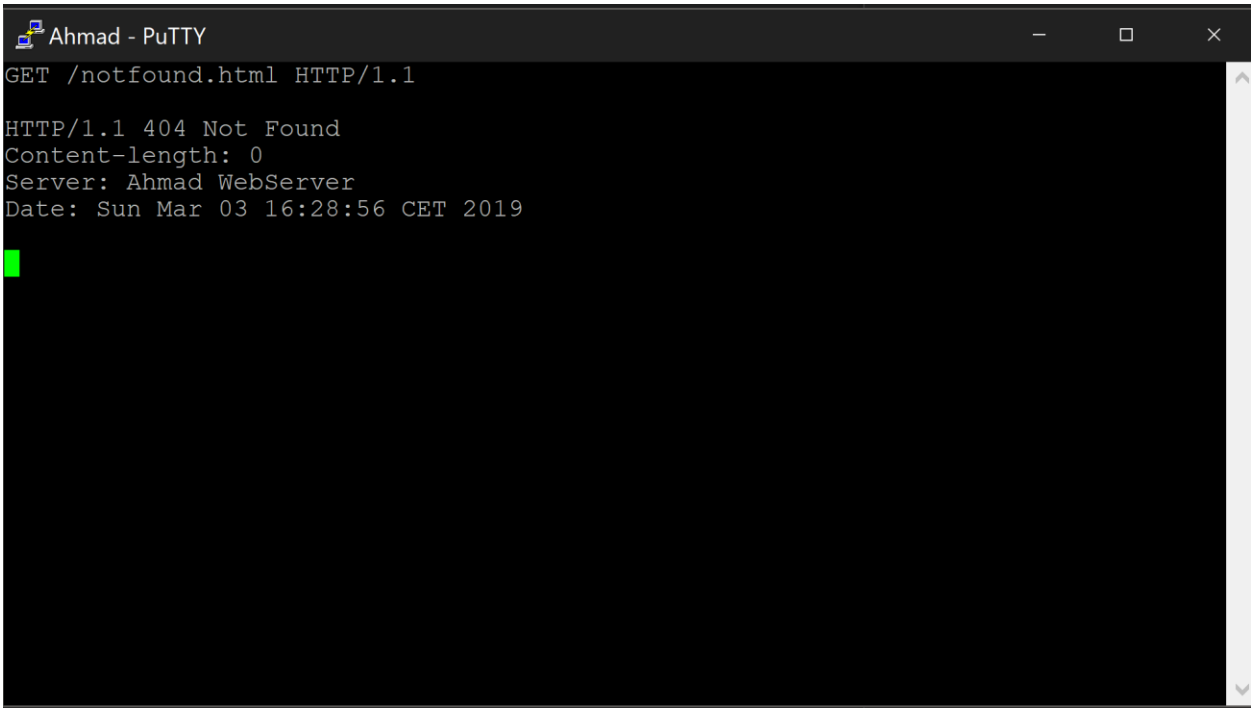
Content-Type: the type of the file requested which is text/html which indicate the browser that the file of a text type.

Server: name of the server.

Date: the time and date when the request was created.

the content of the body is a html tag.

404 Not Found



Using the GET /notfound.html *HTTP*/1.1 which is a file that is not in the directory.

the headers response:

Version of the HTTP 1.1.

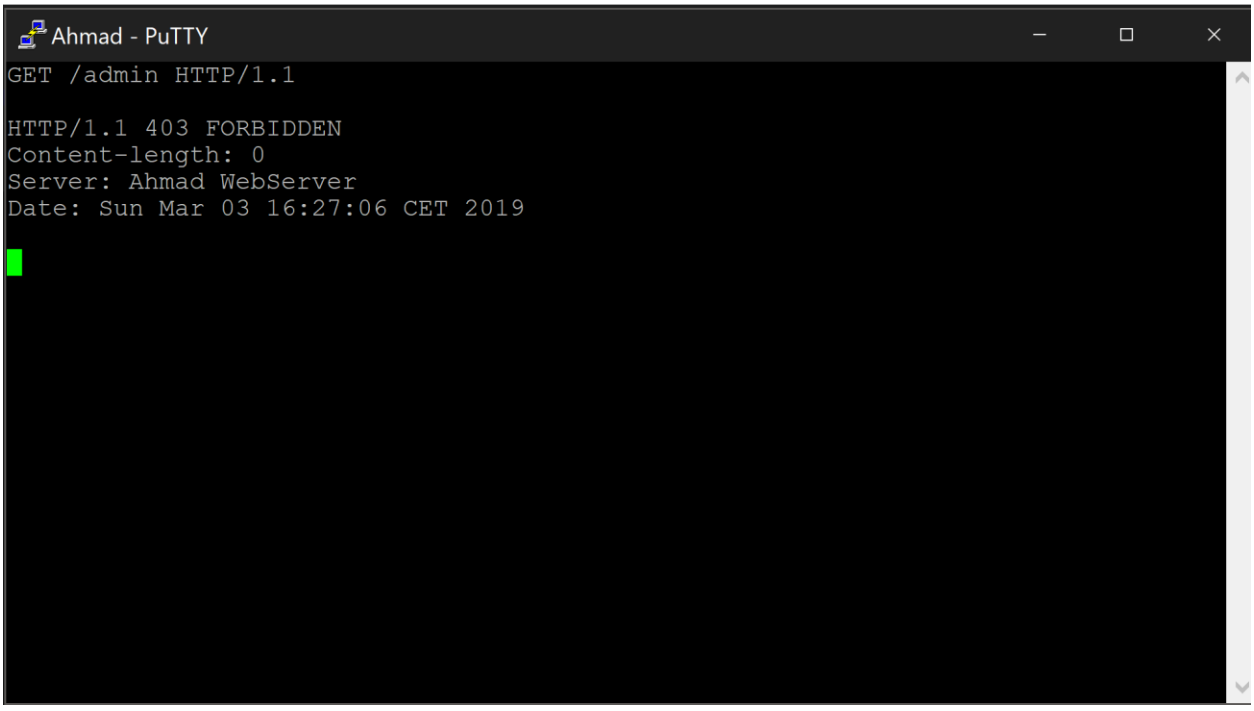
Response status: 404 Not Found.

Content-Length: the size of the file in bytes.

Server: name of the server.

Date: the time and date when the request was created.

403 Forbidden



Using the GET /admin *HTTP*/1.1 dined access when trying to access the admin file.

the headers response:

Version of the HTTP 1.1.

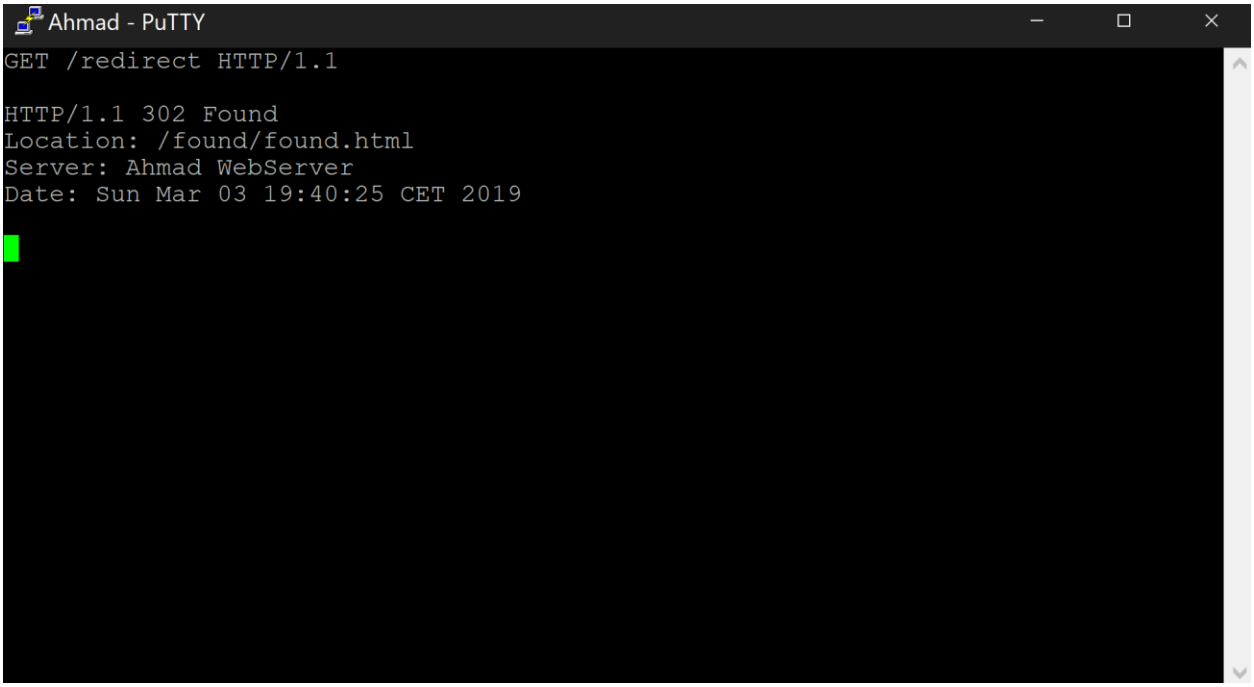
Response status: 403 FORBIDDEN.

Content-Length: the size of the file in bytes.

Server: name of the server.

Date: the time and date when the request was created.

302 Found



Using the GET /redirect *HTTP*/1.1 results in a redirect to the new path.

the headers response:

Version of the HTTP 1.1.

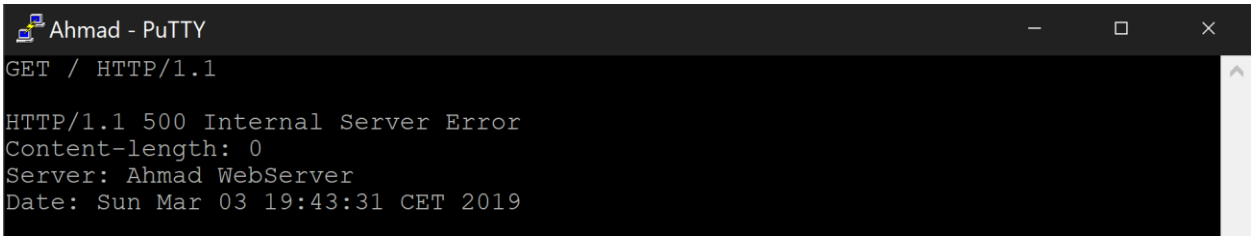
Response status: 302 Found.

Location: the path of the new file requested by the client using the old path.

Server: name of the server.

Date: the time and date when the request was created.

500 Internal Server Error



Using the GET / *HTTP*/1.1 results in Internal Server Error

(Note: in order to show 500 error we had to generate it by ourselves in the code)

the headers response:

Version of the HTTP 1.1.

Response status: 500 Internal Server Error.

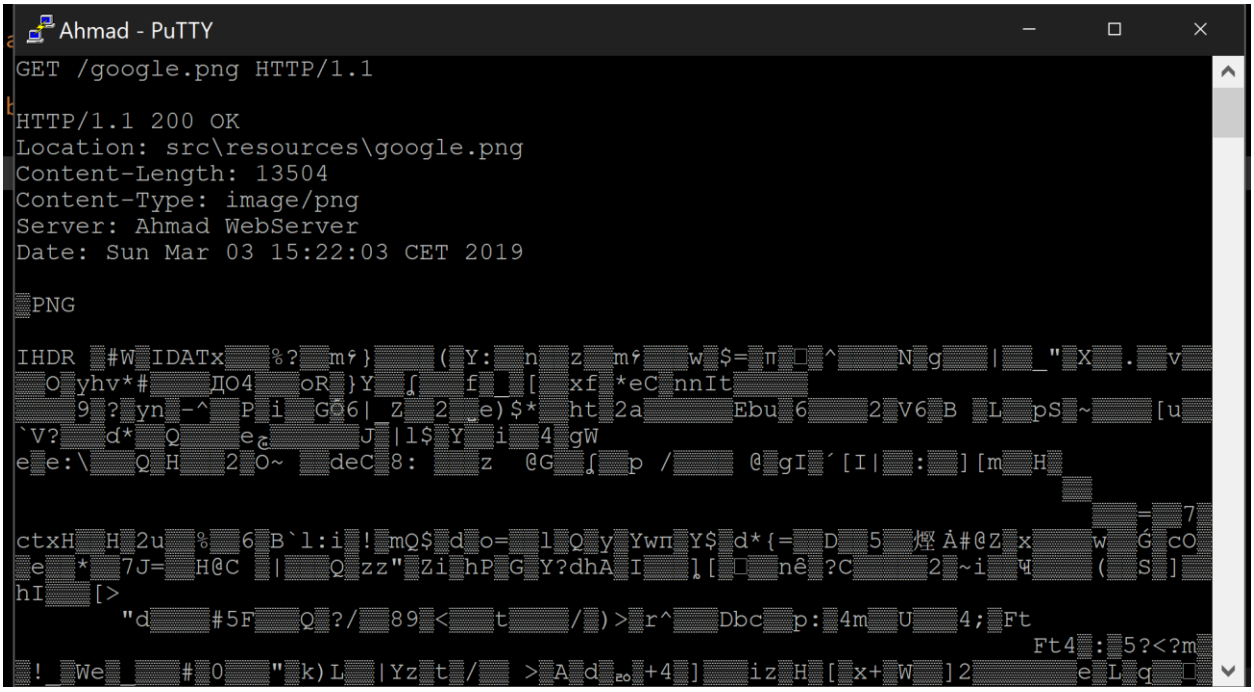
Content-Length: the size of the file in bytes.

Server: name of the server.

Date: the time and date when the request was created.

Get png image using Telnet

the picture below shows a request by telnet for getting a png image. It shows the headers of the response and a part of the response body.



Using the GET /google.png *HTTP*/1.1 we get the png image in the file user1.

the headers response:

Version of the HTTP 1.1.

Response status: 200 OK.

Location: the path of the file requested by the client.

Content-Length: the size of the file in bytes.

Content-Type: the type of the file requested which is image/png which indicate the browser that the file of a image type.

Server: name of the server.

Date: the time and date when the request was created.

the content of the body is a binary form of the png.