BIRZEIT UNIVERSITY

Department of Electrical and Computer Engineering

ENCS3320-Computer Networks

Project #1

Students:

Obada Tahayna	**1191319**
Kareem Halayqa	**1192087**
Khalid Mustafa	**1191523**

Instructor: Dr. Abdalkarim Awad

Section: 1
Date: 14/08/2021

Part 1:

1- Ping a device in the same network

The command “arp -a” was used to list all devices that are connected to the network with their IP address.

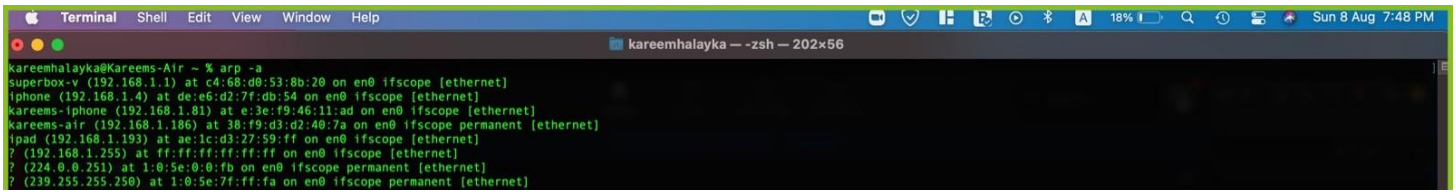


Figure 1: arp -a Command

Then, the command “ping” was used to ping one of the devices that were listed. It shows the amount the amount of packets sent. As a result, it shows the amount of packets received, the IP it was received from, the time-to-live (TTL) which is the amount of hops the packets did before exiting the network, and lastly the time it took to be received.

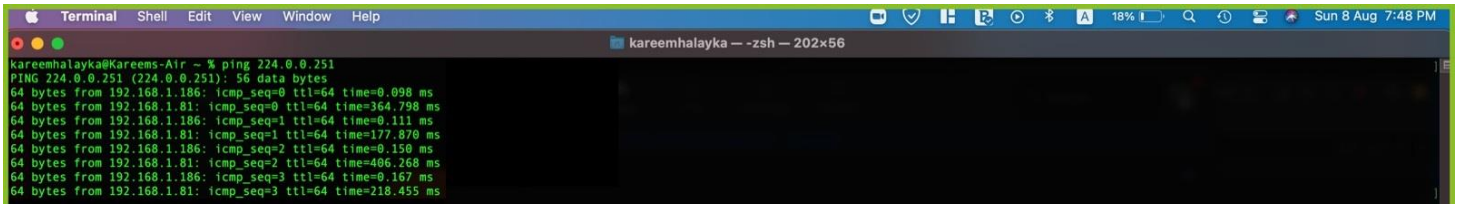


Figure 2: ping Command - Device on the same Network

2- ping www.stanford.edu

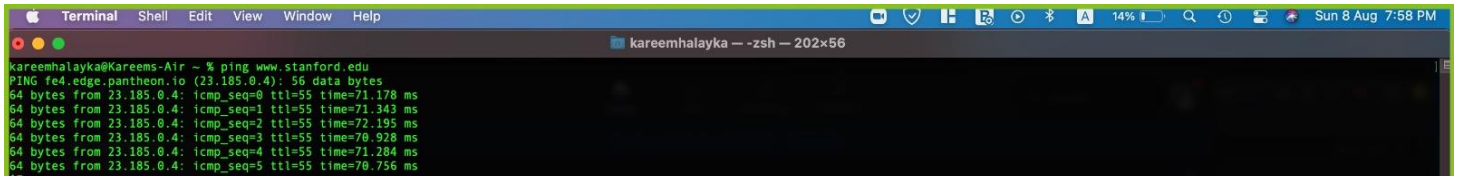
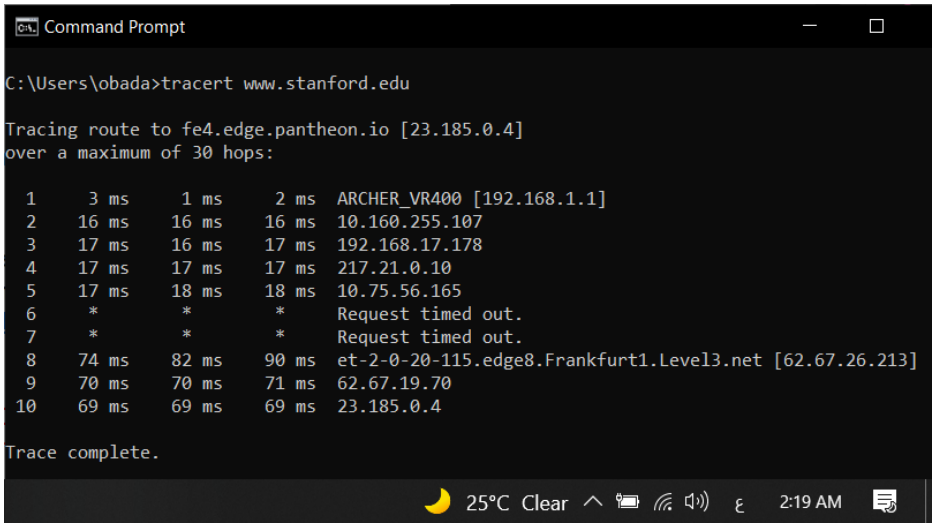


Figure 3: ping Command - Online Website

3- tracert www.stanford.edu

Tracert tests the different paths taken by sent packets to reach the destination from source. The result consists of 5 columns, the firs being the hop number (TTL). Tracert actually sends 3 packets, so the 3 columns after (TTL) are the time it takes for the packets to make each hop. The last column is the server at the specified hop. In our results, it took us 69ms to retrieve data from the destination server as shown in the last hop.

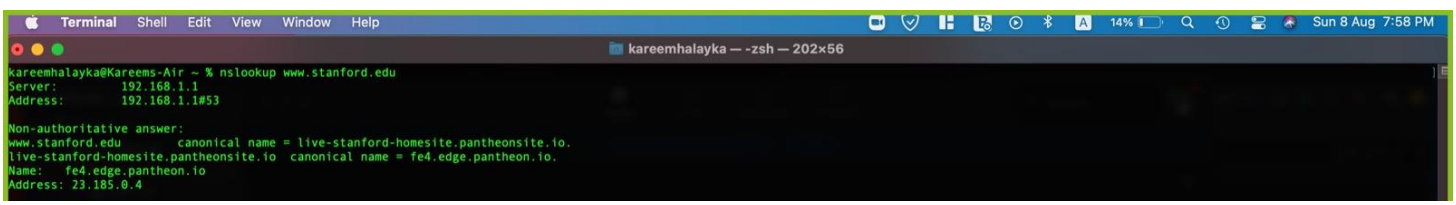


```
Command Prompt

C:\Users\obada>tracert www.stanford.edu

Tracing route to fe4.edge.pantheon.io [23.185.0.4]
over a maximum of 30 hops:

  1     3 ms     1 ms     2 ms  ARCHER_VR400 [192.168.1.1]
  2    16 ms    16 ms    16 ms  10.160.255.107
  3    17 ms    16 ms    17 ms  192.168.17.178
  4    17 ms    17 ms    17 ms  217.21.0.10
  5    17 ms    18 ms    18 ms  10.75.56.165
  6     *        *        *     Request timed out.
  7     *        *        *     Request timed out.
  8    74 ms    82 ms    90 ms  et-2-0-20-115.edge8.Frankfurt1.Level3.net [62.67.26.213]
  9    70 ms    70 ms    71 ms  62.67.19.70
 10    69 ms    69 ms    69 ms  23.185.0.4

Trace complete.
```

Figure 4: tracert Command

4- nslookup www.stanford.edu

nslookup is used to diagnose DNS problems. It's none interactive mode (putting an option like a domain URL in front of it) displays the corresponding server IP address. It also displays more information under ("None-authoritative information"), which is information retrieved from the DNS server cache.



```
kareemhalayka@Kareems-Air ~ % nslookup www.stanford.edu
Server:         192.168.1.1
Address:        192.168.1.1#53

Non-authoritative answer:
www.stanford.edu        canonical name = live-stanford-homesite.pantheonsite.io.
live-stanford-homesite.pantheonsite.io  canonical name = fe4.edge.pantheon.io.
Name:   fe4.edge.pantheon.io
Address: 23.185.0.4
```

Figure 5: nslookup Command

Part 2:

Screenshots:

Main Page (localhost:5000, localhost:5000/, localhost:5000/index.html, localhost:5000/main.html)

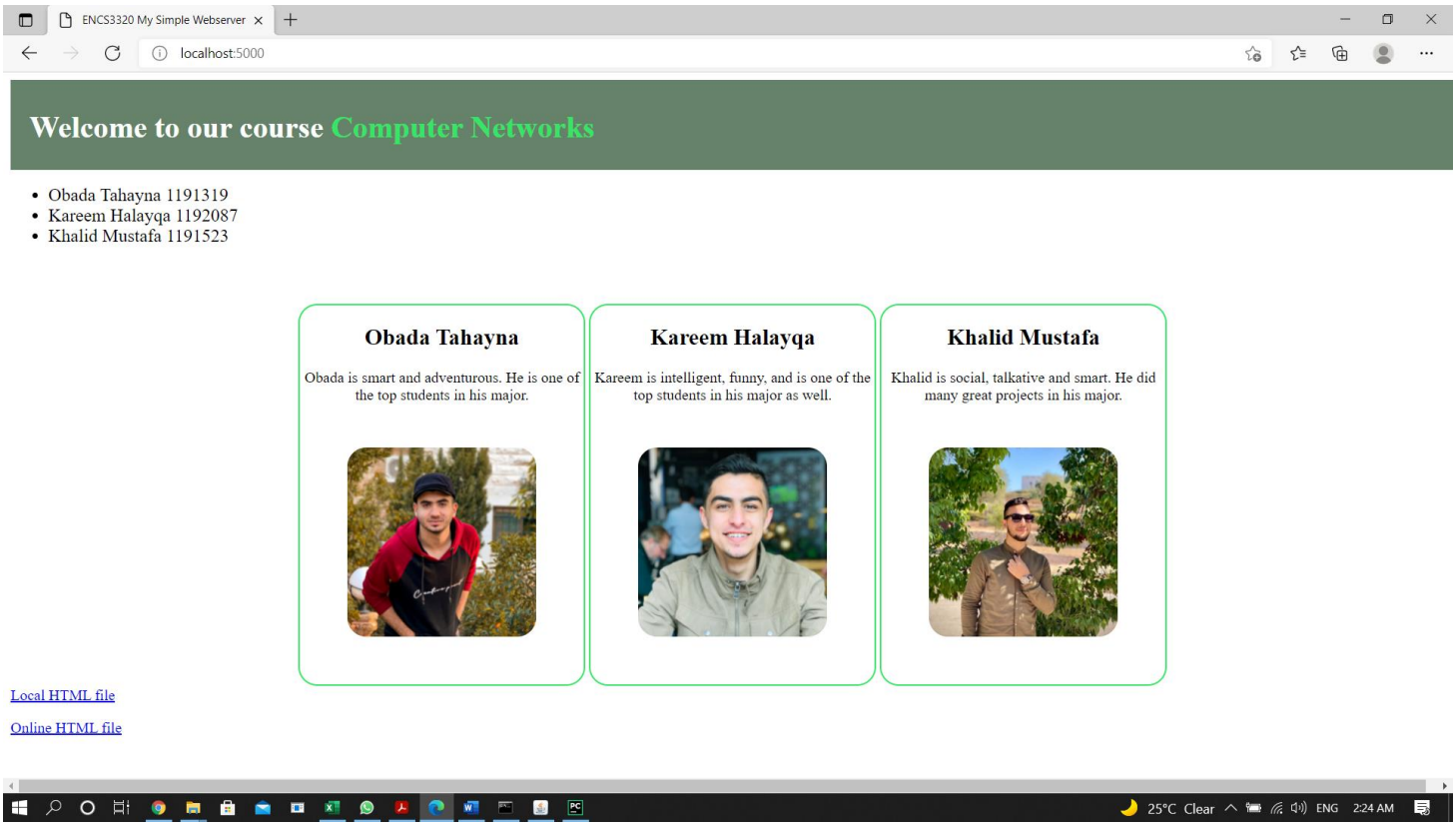


Figure 6: localhost:5000 Browser Window

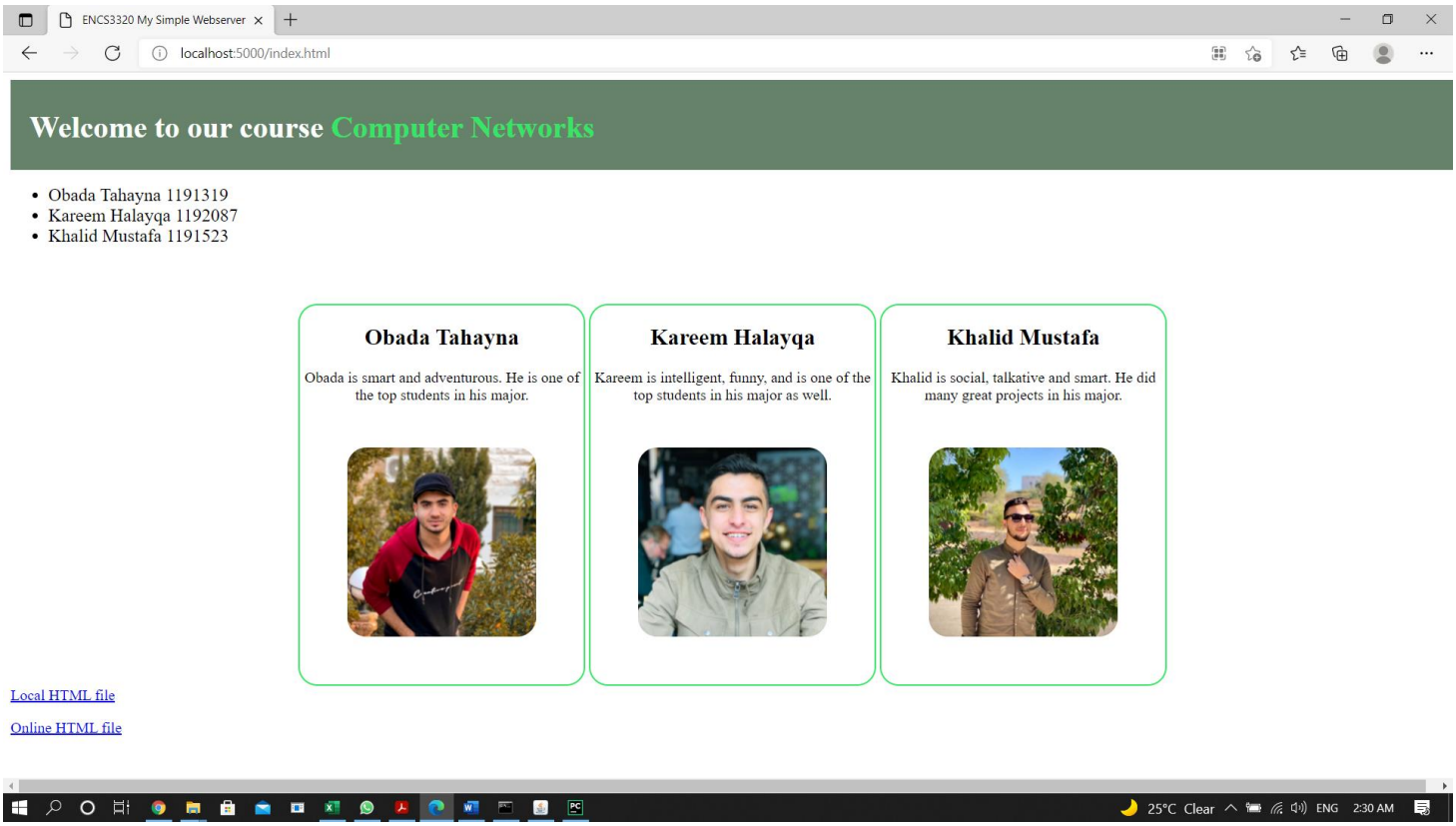


Figure 7: localhost:5000/index.html Browser Window

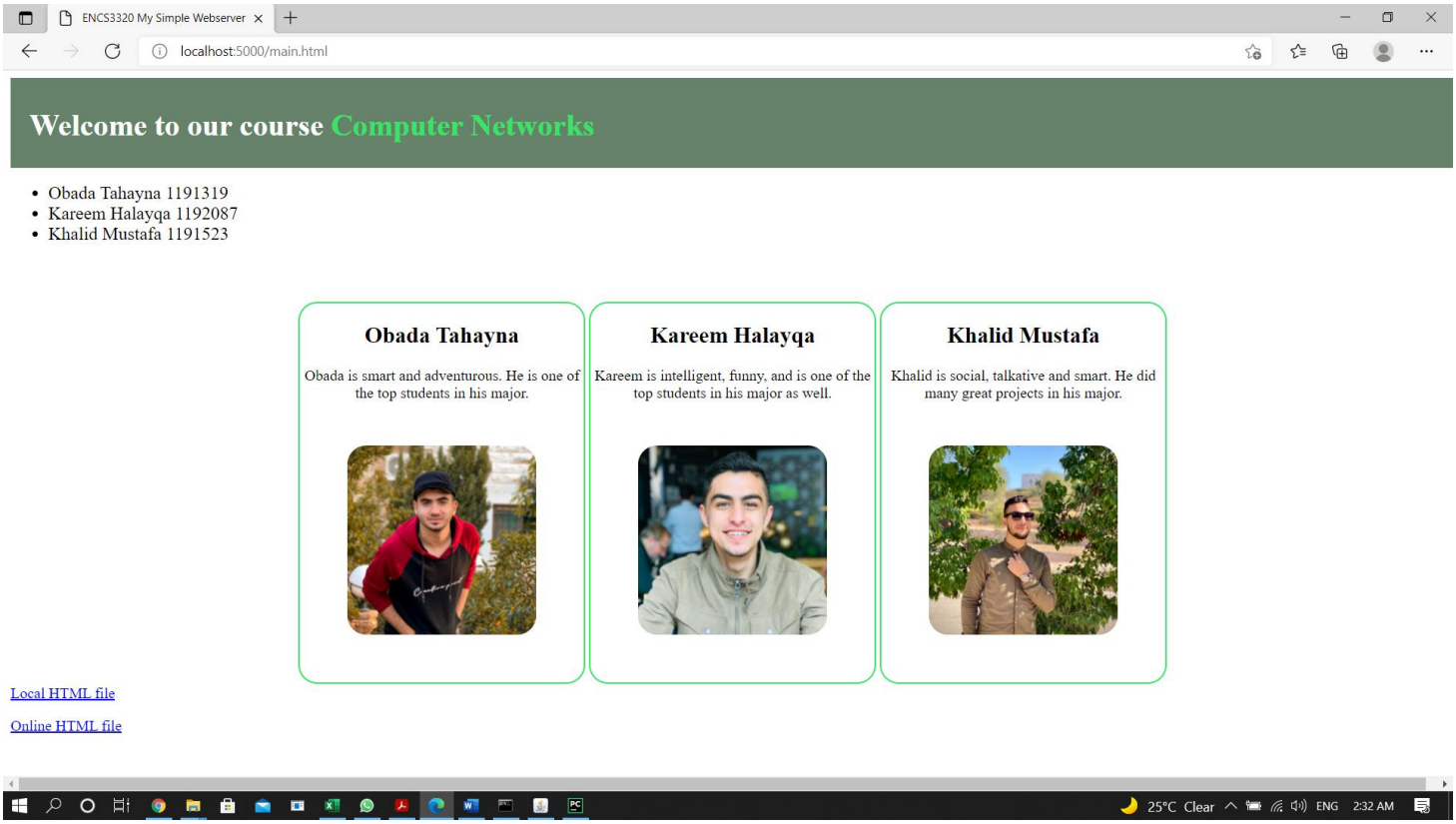


Figure 8: localhost:5000/main.html Browser Window

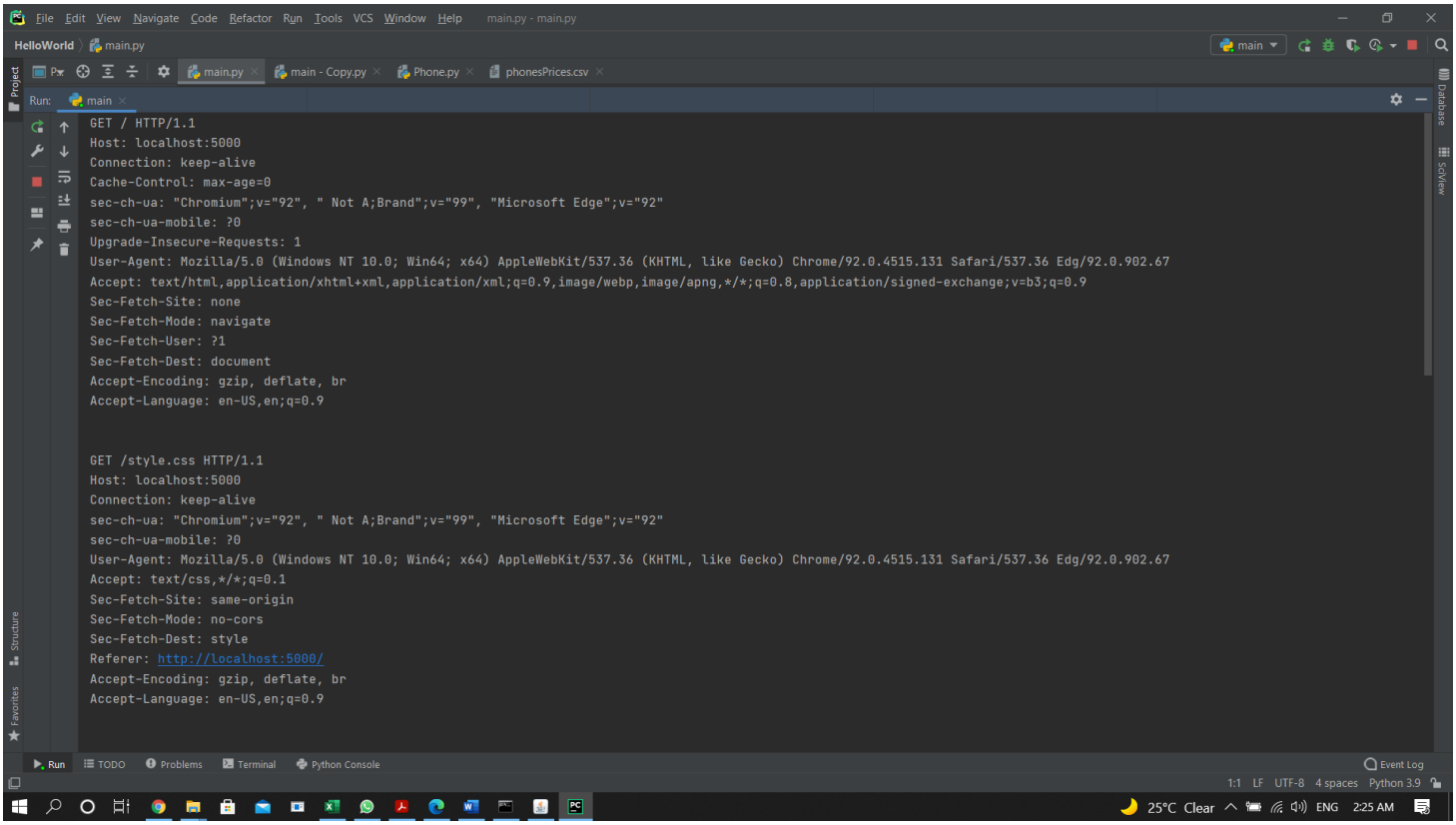


Figure 9: Main Page HTTP Requests Printed on Command Line (1)

```
GET /images/obada.jpg HTTP/1.1
Host: localhost:5000
Connection: keep-alive
sec-ch-ua: "Chromium";v="92", " Not A;Brand";v="99", "Microsoft Edge";v="92"
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/92.0.4515.131 Safari/537.36 Edg/92.0.902.67
Accept: image/webp,image/apng,image/svg+xml,image/*,*/*;q=0.8
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: no-cors
Sec-Fetch-Dest: image
Referer: http://localhost:5000/
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9


GET /images/Kareem.png HTTP/1.1
Host: localhost:5000
Connection: keep-alive
sec-ch-ua: "Chromium";v="92", " Not A;Brand";v="99", "Microsoft Edge";v="92"
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/92.0.4515.131 Safari/537.36 Edg/92.0.902.67
Accept: image/webp,image/apng,image/svg+xml,image/*,*/*;q=0.8
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: no-cors
Sec-Fetch-Dest: image
Referer: http://localhost:5000/
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
```

Figure 10: Main Page HTTP Requests Printed on Command Line (2)

```
GET /images/Khalid.jpeg HTTP/1.1
Host: localhost:5000
Connection: keep-alive
sec-ch-ua: "Chromium";v="92", " Not A;Brand";v="99", "Microsoft Edge";v="92"
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/92.0.4515.131 Safari/537.36 Edg/92.0.902.67
Accept: image/webp,image/apng,image/svg+xml,image/*,*/*;q=0.8
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: no-cors
Sec-Fetch-Dest: image
Referer: http://localhost:5000/
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
```

Figure 11: Main Page HTTP Requests Printed on Command Line (3)

CSS File

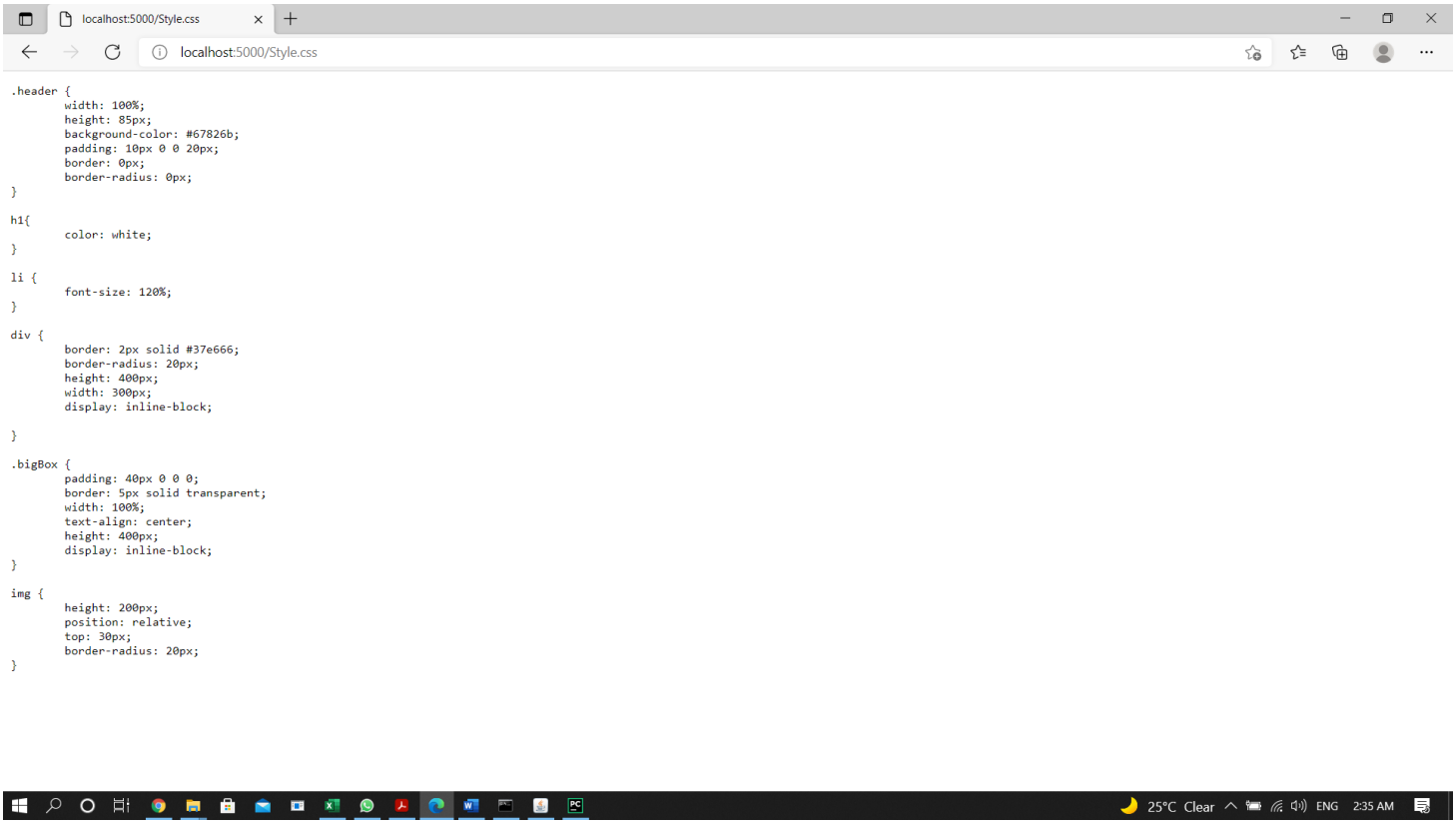


Figure 12: localhost:5000/Style.css Browser Window

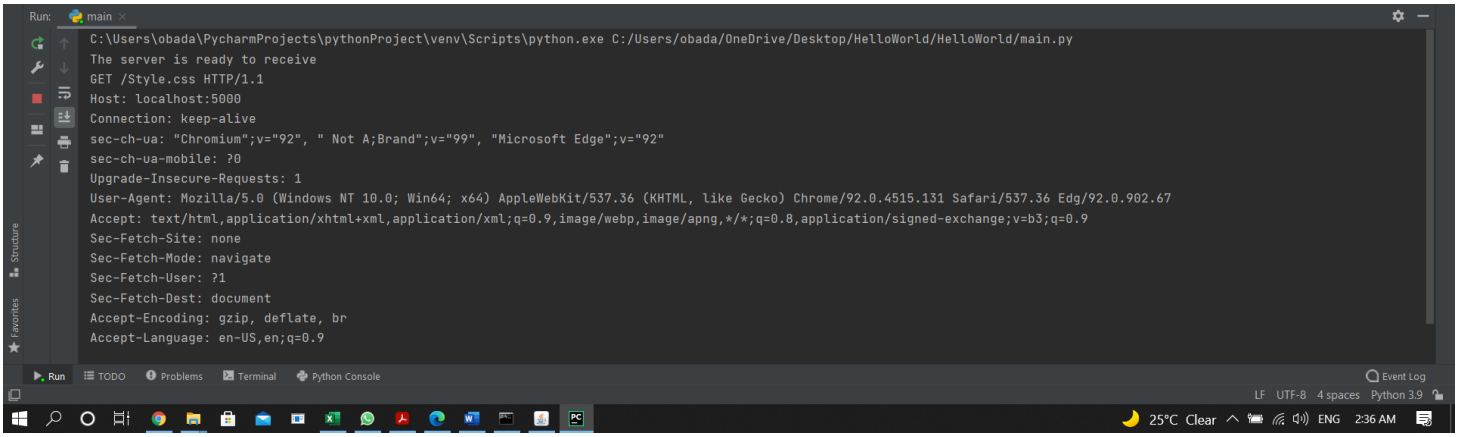


Figure 13: localhost:5000/Style.css HTTP Request Printed on Command Line

JPG Image

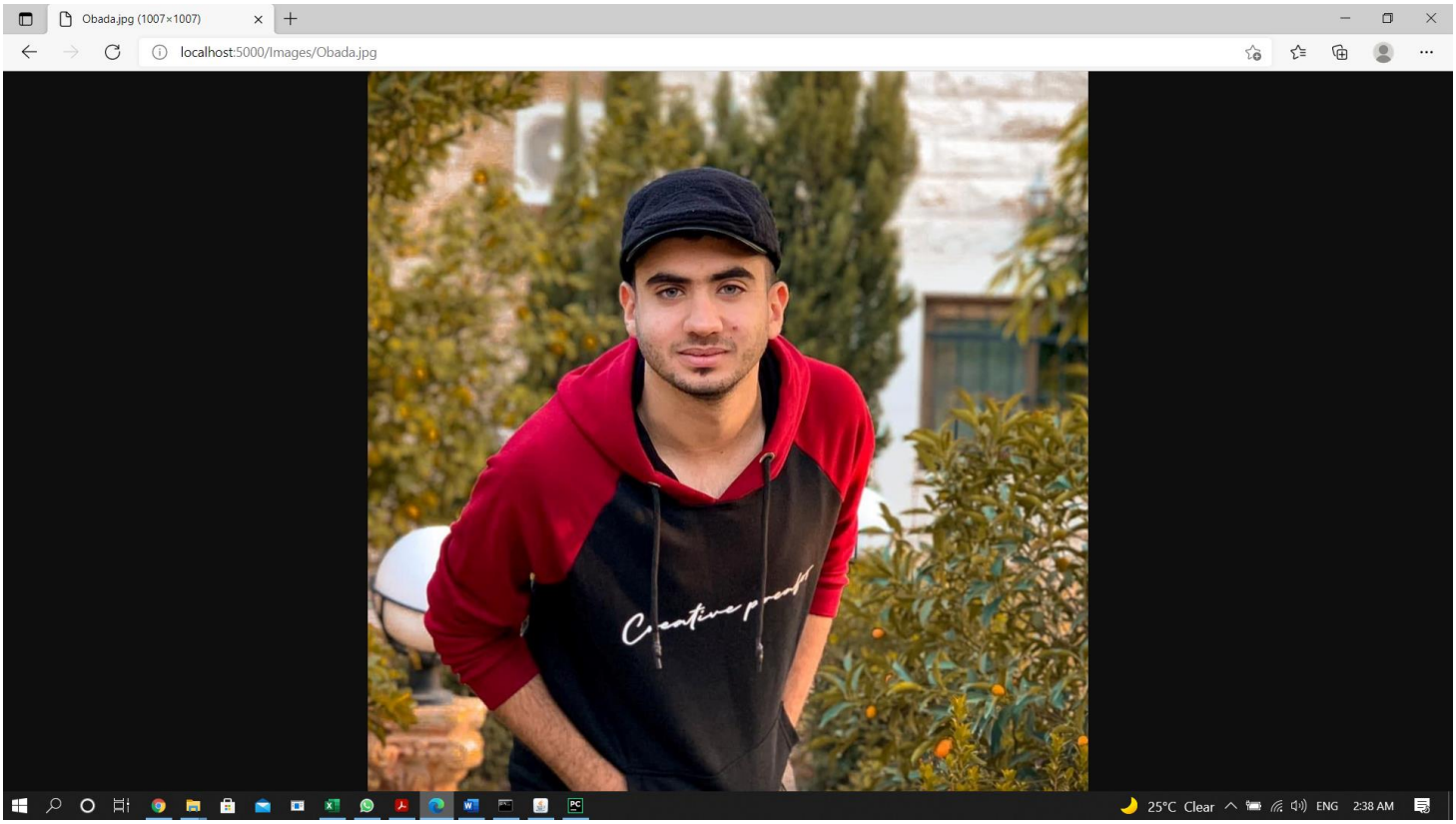


Figure 14: JPG image - Browser Window

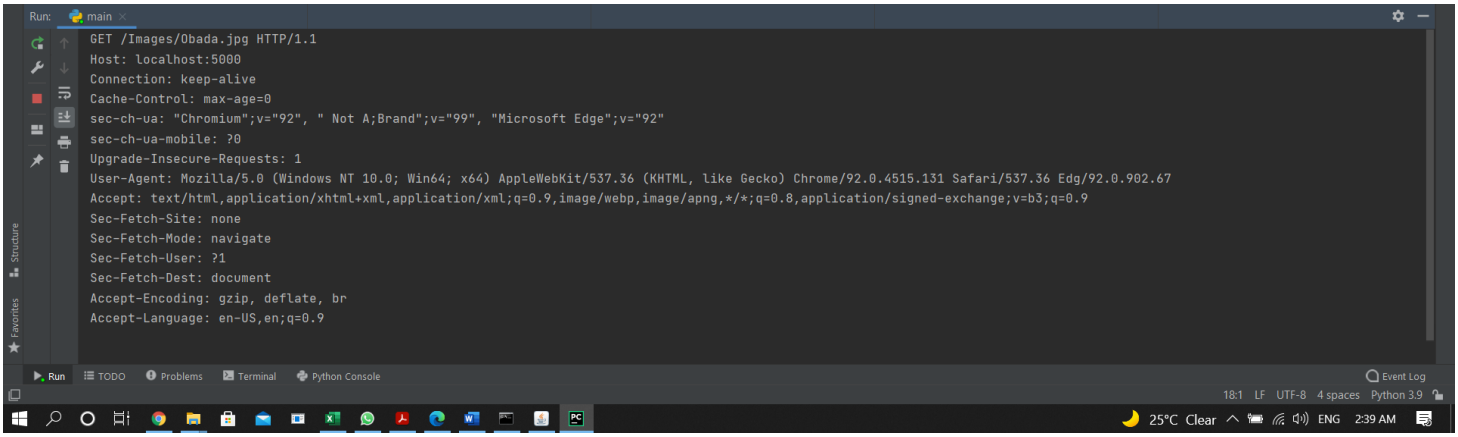


Figure 15: JPG image - HTTP Request Printed on Command Line

PNG Image

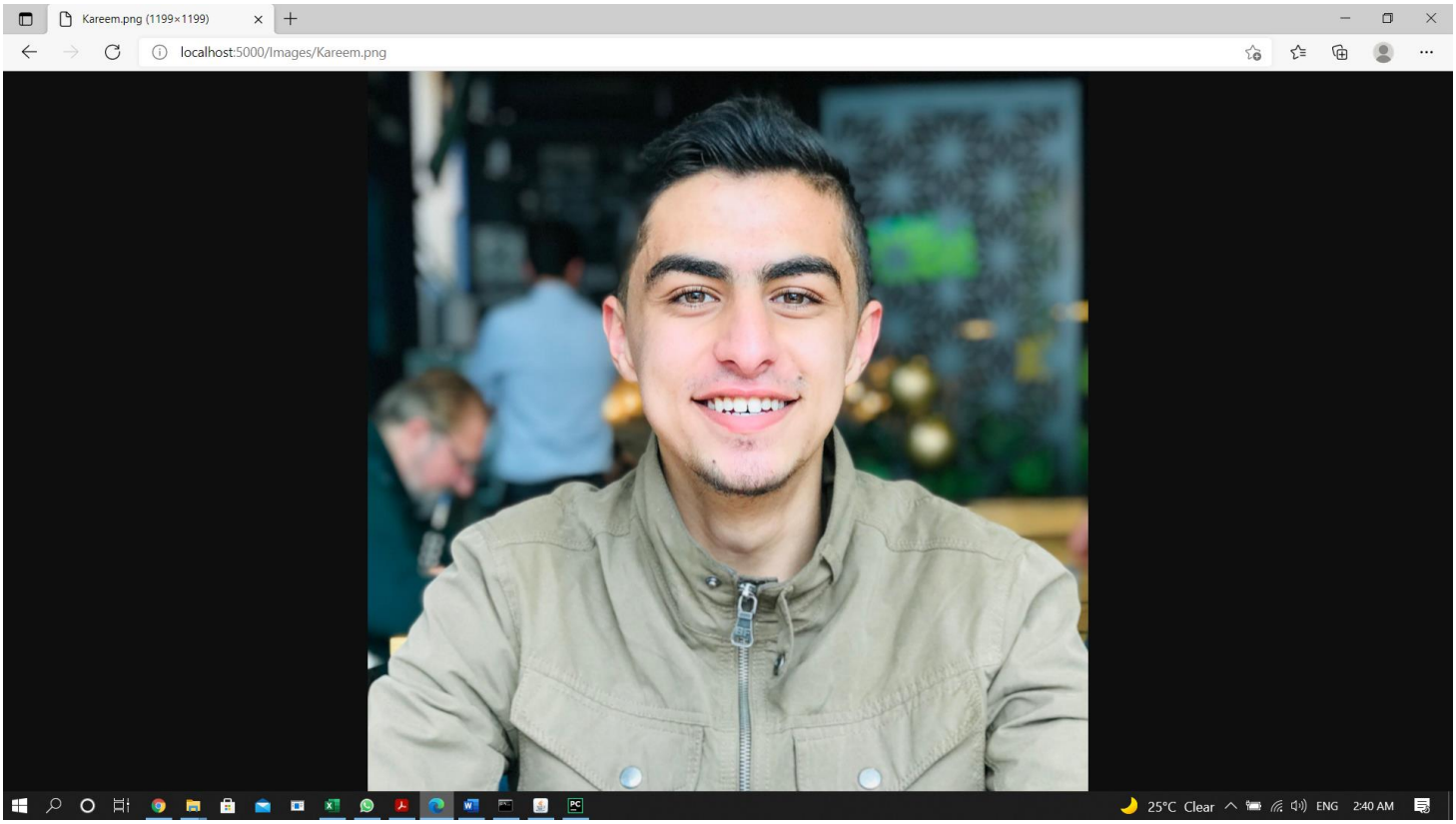


Figure 16: PNG image - Browser Window

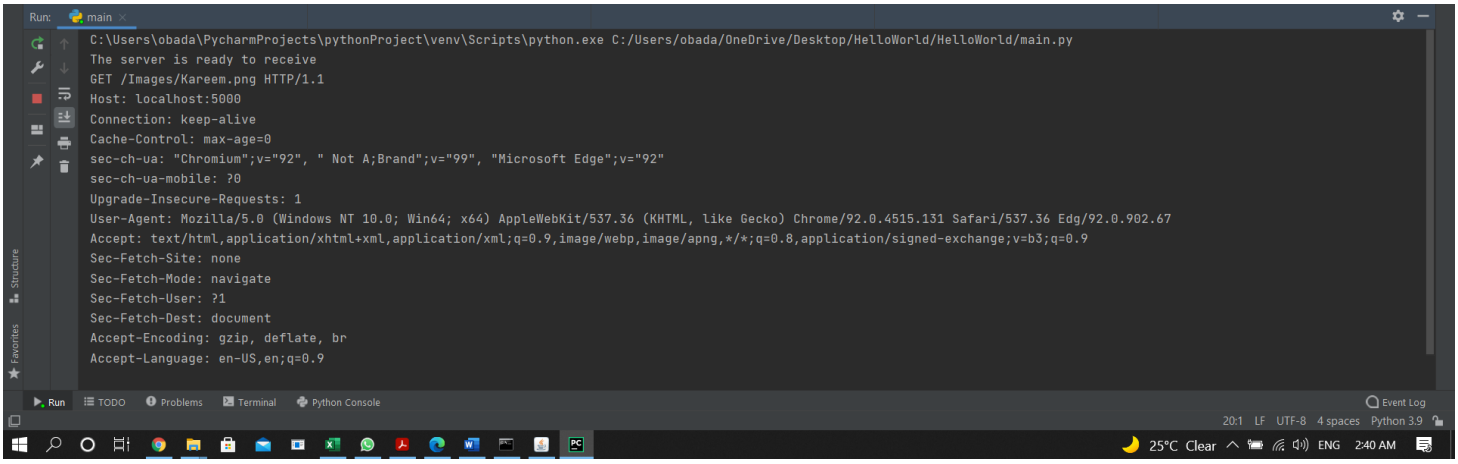


Figure 17: PNG image - HTTP Request Printed on Command Line

JPEG Image

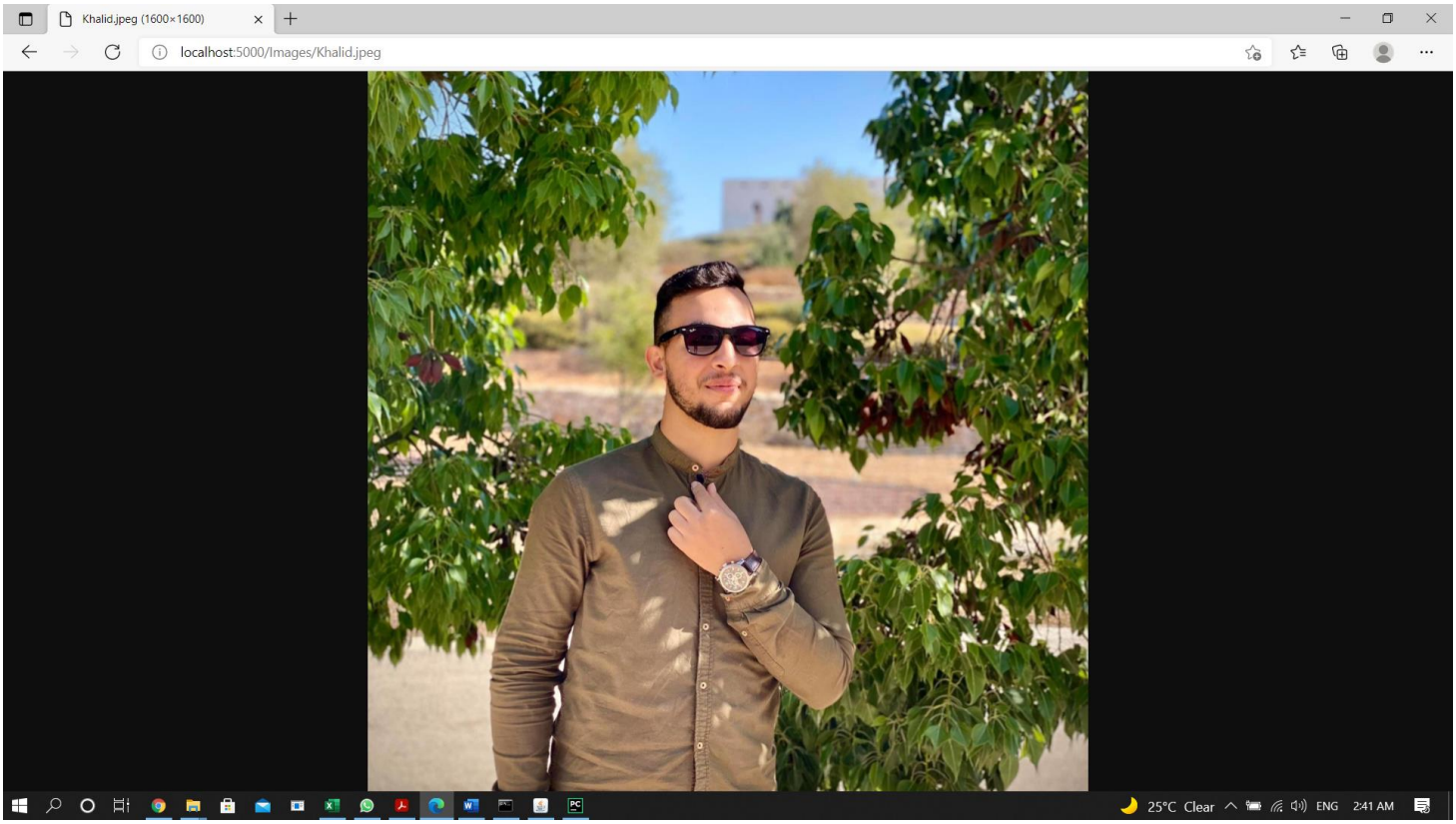


Figure 18: JPEG image - Browser Window

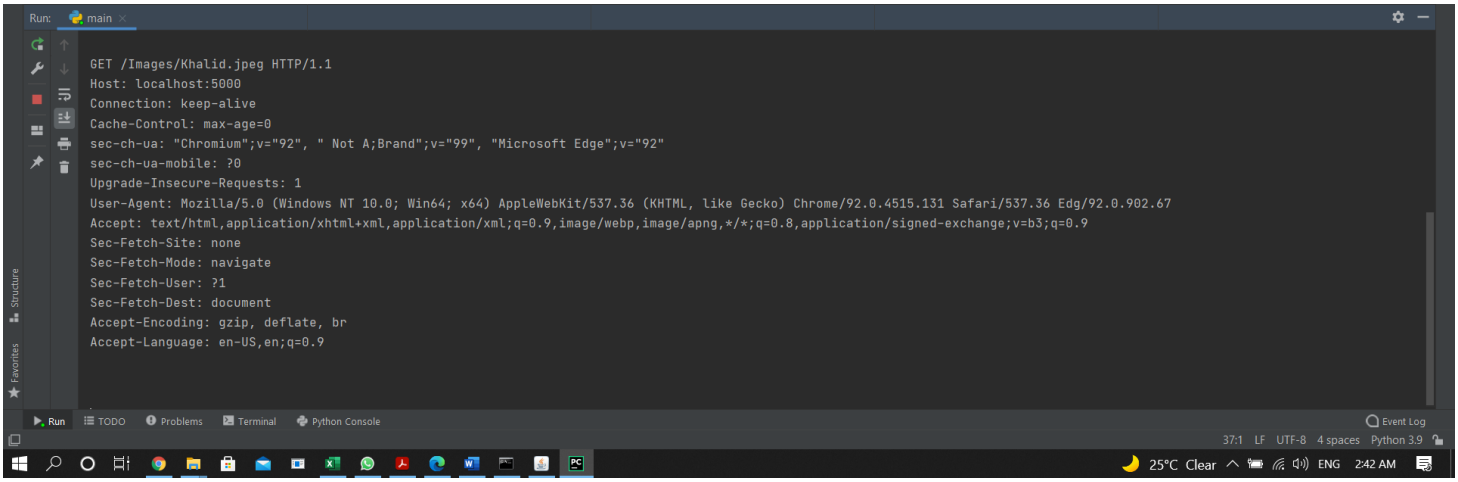
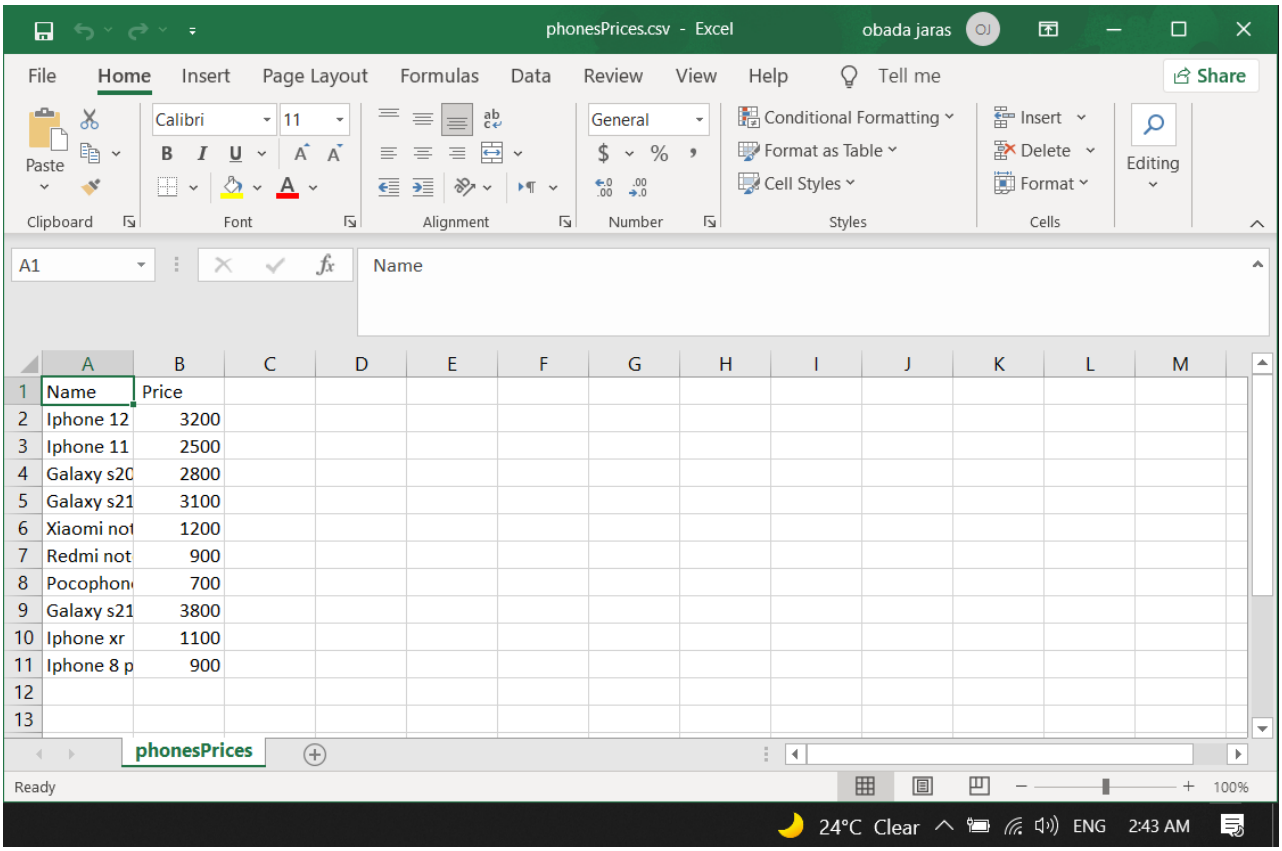


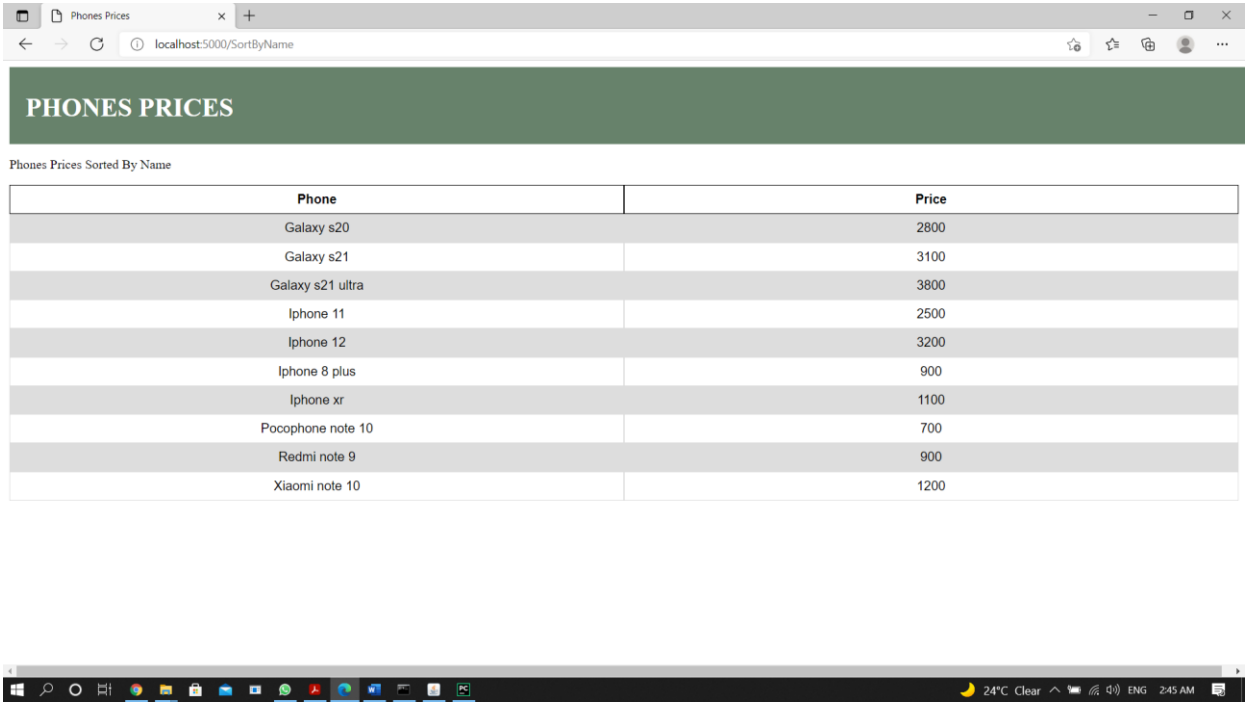
Figure 19: JPEG image - HTTP Request Printed on Command Line

Sort By Name



Name	Price
Iphone 12	3200
Iphone 11	2500
Galaxy s20	2800
Galaxy s21	3100
Xiaomi not	1200
Redmi not	900
Pocophon	700
Galaxy s21	3800
Iphone xr	1100
Iphone 8 p	900

Figure 20: CSV File to Get Data From



PHONES PRICES

Phones Prices Sorted By Name

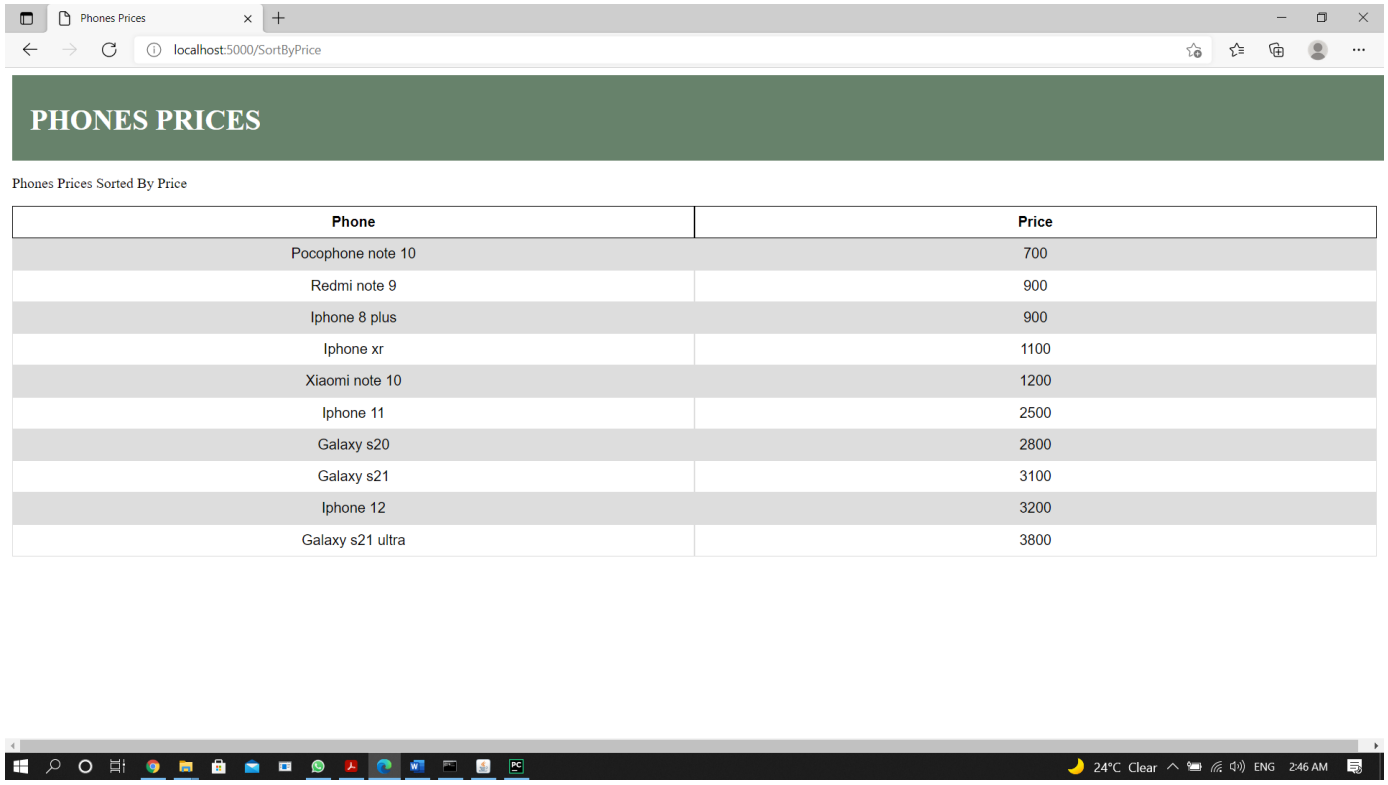
Phone	Price
Galaxy s20	2800
Galaxy s21	3100
Galaxy s21 ultra	3800
Iphone 11	2500
Iphone 12	3200
Iphone 8 plus	900
Iphone xr	1100
Pocophone note 10	700
Redmi note 9	900
Xiaomi note 10	1200

Figure 21: SortByName Browser Window

```
C:\Users\obada\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/obada/OneDrive/Desktop/HelloWorld/HelloWorld/main.py
The server is ready to receive
GET /SortByName HTTP/1.1
Host: localhost:5000
Connection: keep-alive
sec-ch-ua: "Chromium";v="92", " Not A;Brand";v="99", "Microsoft Edge";v="92"
sec-ch-ua-mobile: ?0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/92.0.4515.131 Safari/537.36 Edg/92.0.902.67
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
```

Figure 22: SortByName HTTP Request Printed on Command Line

Sort By Price



Phone	Price
Pocophone note 10	700
Redmi note 9	900
Iphone 8 plus	900
Iphone xr	1100
Xiaomi note 10	1200
Iphone 11	2500
Galaxy s20	2800
Galaxy s21	3100
Iphone 12	3200
Galaxy s21 ultra	3800

Figure 23: SortByPrice Browser Window

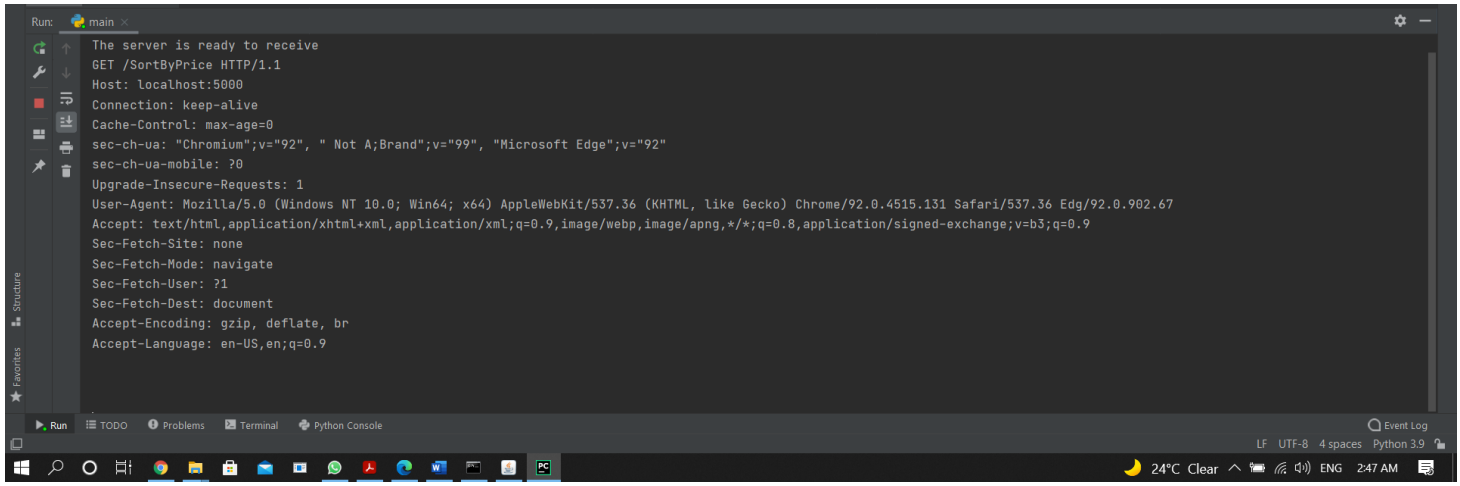


Figure 24: SortByPrice HTTP Request Printed on Command Line

Error 404

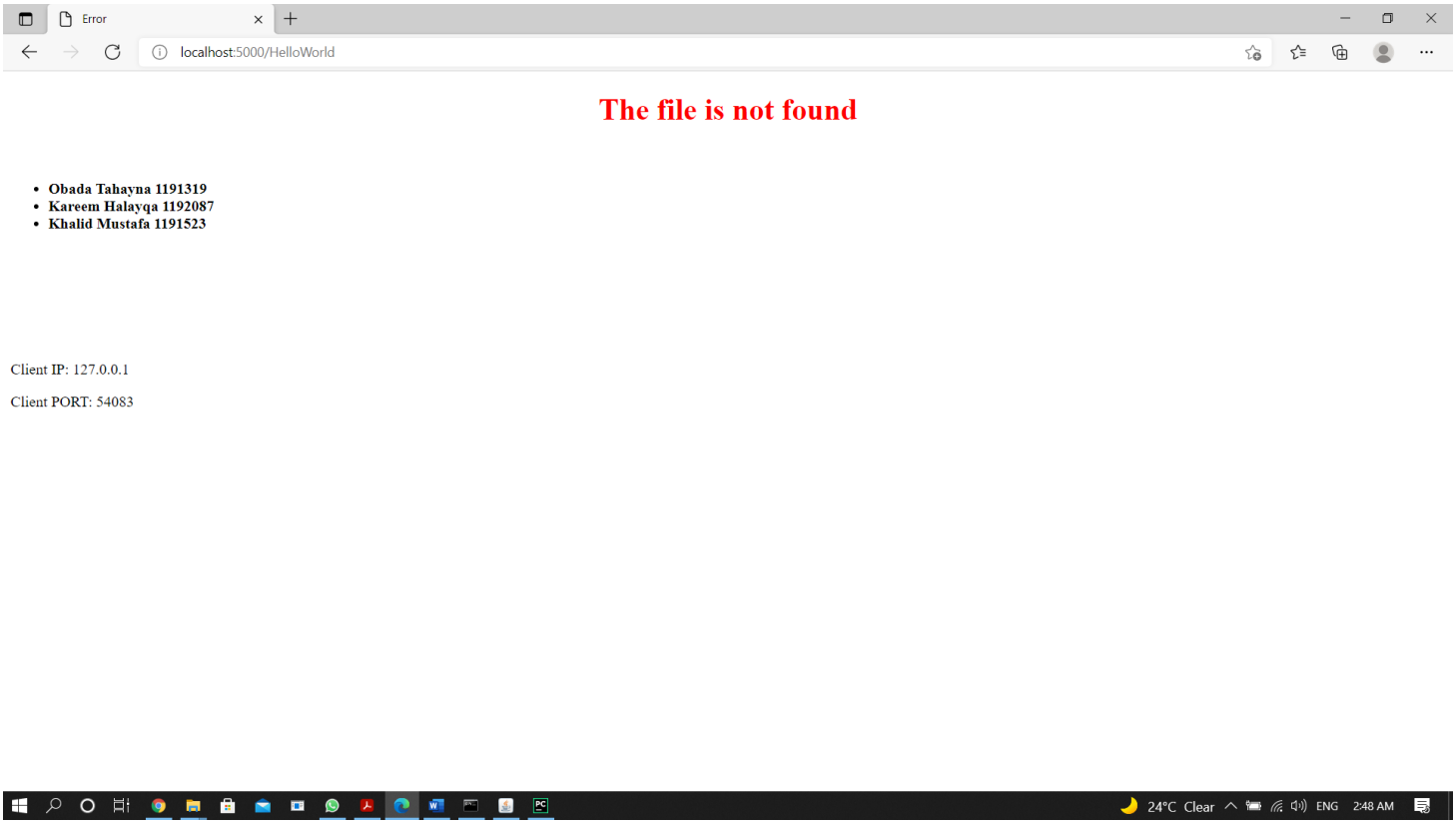


Figure 25: Error 404 Browser Window

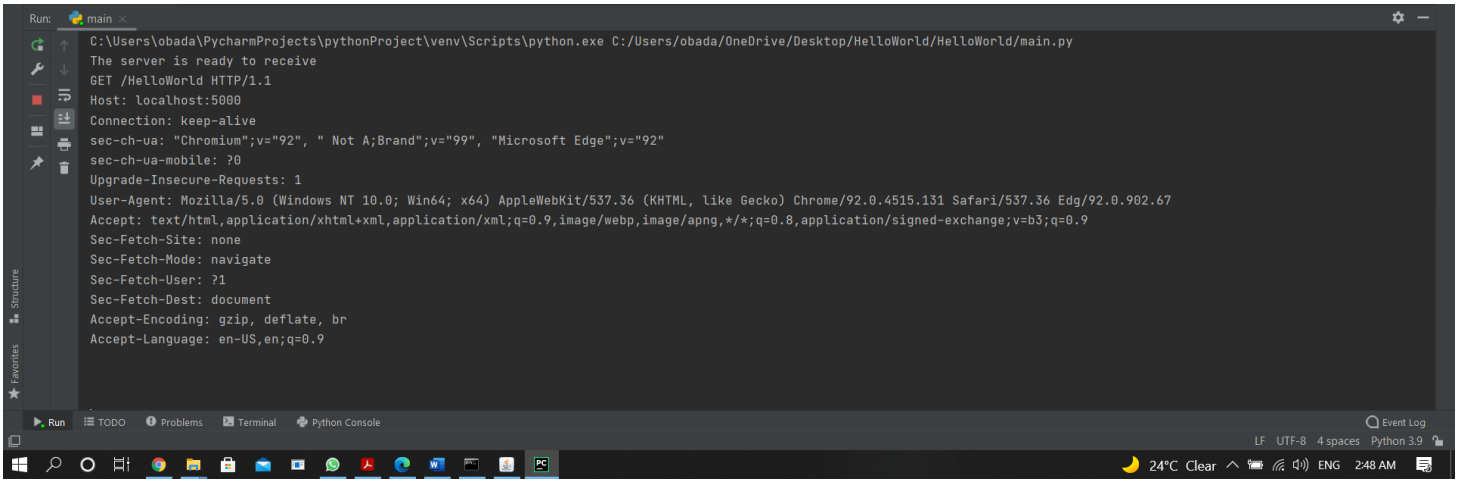


Figure 26: Not existed page HTTP request printed on command line

Codes:

Server Code – Python

```
from socket import *
from Phone import Phone

phonesList = []
PORT = 5000
# defining the socket, and binding it to the port
serverSocket = socket(AF_INET, SOCK_STREAM)
serverSocket.bind(('', PORT))
# socket listening for response
serverSocket.listen(1)

print('The server is ready to receive')


# a function to read the phones csv file
def readFile(fileName):
    phonesList.clear()
    with open(fileName) as f:
        line = " "
        next(f)  # To ignore reading the first line on the csv file (header line)
        while line:
            line = f.readline()

            if line != "":
                lineSplit = line.split(',')  # splitting the csv file lines by commas to
get the desired data

                phone = Phone(lineSplit[0], lineSplit[1])
                phonesList.append(phone)  # putting the file information in a list


# return name function to use as a key in the list sort function

def retName(phone):
    return phone.name


# return name function to use as a key in the list sort function

def retPrice(phone):
    return float(phone.price)


# sort the list by name after storing the file data in it

def sortByName():
    readFile('phonesPrices.csv')
    phonesList.sort(key=retName)


# sort the list by price after storing the file data in it
def sortByPrice():
    readFile('phonesPrices.csv')
    phonesList.sort(key=retPrice)


while True:
    connection, add = serverSocket.accept()
    sentence = connection.recv(1024).decode('utf-8')

    requesting_file = sentence.split(' ')[1]  # from the request sentence, getting the
```

```
requested file
    requestedFile = requesting_file.lstrip('/')  # removing the first / to get the
requested file name

    if requestedFile == '' or requestedFile == "index.html":  # default request
        requestedFile = 'main.html'  # Load main.html file as default

    try:
        sortedBy = ''
        # accepting different file formats
        if requestedFile.endswith(".jpg"):
            requestedType = 'image/jpg'

        elif requestedFile.endswith(".jpeg"):
            requestedType = 'image/jpeg'

        elif requestedFile.endswith(".png"):
            requestedType = 'image/png'

        elif requestedFile.endswith(".css"):
            requestedType = 'text/css'

        elif requestedFile.upper() == "SORTBYNAME":
            sortByName()
            sortedBy = 'Name'
            requestedType = 'text/html'

        elif requestedFile.upper() == "SORTBYPRICE":
            sortByPrice()
            sortedBy = 'Price'
            requestedType = 'text/html'

        else:
            requestedType = 'text/html'

        if requestedFile.upper() != "SORTBYNAME" and requestedFile.upper() !=
"SORTBYPRICE":
            file = open(requestedFile, 'rb')  # opening the requested file
            response = file.read()  # reading the file
            file.close()  # closing the file

        else:
            response = ('<!DOCTYPE html><html><head><title>Phones Prices</title><style
type="text/css">.header {'
                        'width:100%;height: 85px;background-color: #67826b;padding: 10px 0
0 20px;border: '
                        '0px;border-radius:0px;}table {font-family: arial, sans-
serif;border-collapse: '
                        'collapse;width: 100%;}td, th {border: 1px solid #dddddd;text-
align: center;padding: '
                        '8px; width: 50%; font-weight: normal;}tr:nth-child(even)
{background-color: '
                        '#dddddd;}</style></head><body><div class="header"><h1
style="color: white;">PHONES '
                        'PRICES</h1></div><p>Phones Prices Sorted By ' + sortedBy +
'</p><table><tr><th style="border: '
                        '1px solid black; font-weight: bold;">Phone</th><th style="border:
1px solid black; '
                        'font-weight: bold;">Price</th></tr>').encode()
            for item in phonesList:
                response += ('<tr><th>' + str(item.name) + '</th><th>' + str(item.price) +
'</th></tr>').encode()
            response += '</table></body></html>'.encode()

        header = 'HTTP/1.1 200 OK\r\n'  # the first part of the header to send.
        header += 'Content-Type: ' + str(requestedType) + '\r\n\r\n'
```

```
    except Exception as e:  # Exception if the request the user has entered doesn't exist
        header = 'HTTP/1.1 404 Not Found\r\n'
        header += 'Content-Type: text/html\r\n\r\n'
        response = '<!DOCTYPE html><html><head><title>Error</title><style
type="text/css">h1 {text-align: center;}li ' \
                   '{font-weight: bold;}</style></head><body><h1 style="color:red">The
file is not ' \
                   'found</h1><br><ul><li> Obada Tahayna 1191319</li><li> Kareem Halayqa
1192087</li><li> Khalid ' \
                   'Mustafa 1191523</li></ul><div style="position: relative; top:
120px;"><p>Client IP: ' + \
                   str(add[0]) + '</p><p>Client PORT: ' + str(add[1]) +
'</p></div></body></html>'
        response = response.encode()

    final_response = header.encode() + response  # encoding the header and adding the
response to the request
    connection.send(final_response)  # sending the final response with all parts of header
    connection.close()
    print(sentence)  # Print the HTTP request on the terminal window
```

```
class Phone:
    name = ''
    price = 0

    def __init__(self, name, price):
        self.name = name
        self.price = price

    def __repr__(self):
        return self.name + ',' + self.price
```

Main Page HTML Code

```
<!DOCTYPE html>
<html>
<head>
   <title>ENCS3320 My Simple Webserver</title>
   <link rel="stylesheet" href="style.css">

</head>


<body>
   <div class="header">
      <h1>Welcome to our course <span style="color:#37e666">Computer Networks</span></h1>
   </div>


   <ul>
      <li> Obada Tahayna 1191319</li>
      <li> Kareem Halayqa 1192087</li>
      <li> Khalid Mustafa 1191523</li>
   </ul>


   <div class="bigBox">
      <div>
         <h2>Obada Tahayna</h2>
         <p>Obada is smart and adventurous. He is one of the top students in his
major.</p>
         <img src="images/obada.jpg" alt="Obada Tahayna">
      </div>


      <div>
         <h2>Kareem Halayqa</h2>
         <p>Kareem is intelligent, funny, and is one of the top students in his major as
well.</p>
         <img src="images/Kareem.png" alt="Kareem Halayqa">
      </div>


      <div>
         <h2>Khalid Mustafa</h2>
         <p>Khalid is social, talkative and smart. He did many great projects in his
major.</p>
         <img src="images/Khalid.jpeg" alt="Khalid Mustafa">
      </div>
   </div>


   <a href="testFile.html" target="_blank">Local HTML file</a>
   <p></p>
   <a href="https://www.w3schools.com/tags/tag_div.ASP" target="_blank">Online HTML
file</a>
</body>
</html>
```

Cascading Style Sheet Code

```
.header {
   width: 100%;
   height: 85px;
   background-color: #67826b;
   padding: 10px 0 0 20px;
   border: 0px;
   border-radius: 0px;
}

h1{
   color: white;
}

li {
   font-size: 120%;
}

div {
   border: 2px solid #37e666;
   border-radius: 20px;
   height: 400px;
   width: 300px;
   display: inline-block;

}

.bigBox {
   padding: 40px 0 0 0;
   border: 5px solid transparent;
   width: 100%;
   text-align: center;
   height: 400px;
   display: inline-block;
}

img {
   height: 200px;
   position: relative;
   top: 30px;
   border-radius: 20px;
}
```