BIRZEIT UNIVERSITY

FACULTY OF ENGINEERING AND TECHNOLOGY
DEPARTMENT OF COMPUTER ENGINEERING

Computer Networks
ENCS3320

Project 1 Report

Prepared by:

Tariq Odeh	1190699
Qays Safa	1190880
Mahmoud Samara	1191602

Sec: 2

Instructor: Dr. Abdalkarim Awad

Date: 11th November 2021

Table of Contents

List Of Figures

1. Part I:

1.1. Ping a device in the same network

As we can see the result in figure 1, it displays the total number of packets sent. As a result, the number of packets received is displayed (here we sent 4 packets where all packets have the same TTL, we received a response from 192.168.1.100), All packets are received with different delays. Also, we sent out 32 bytes of data and we got back 32 bytes and this is stable connection.

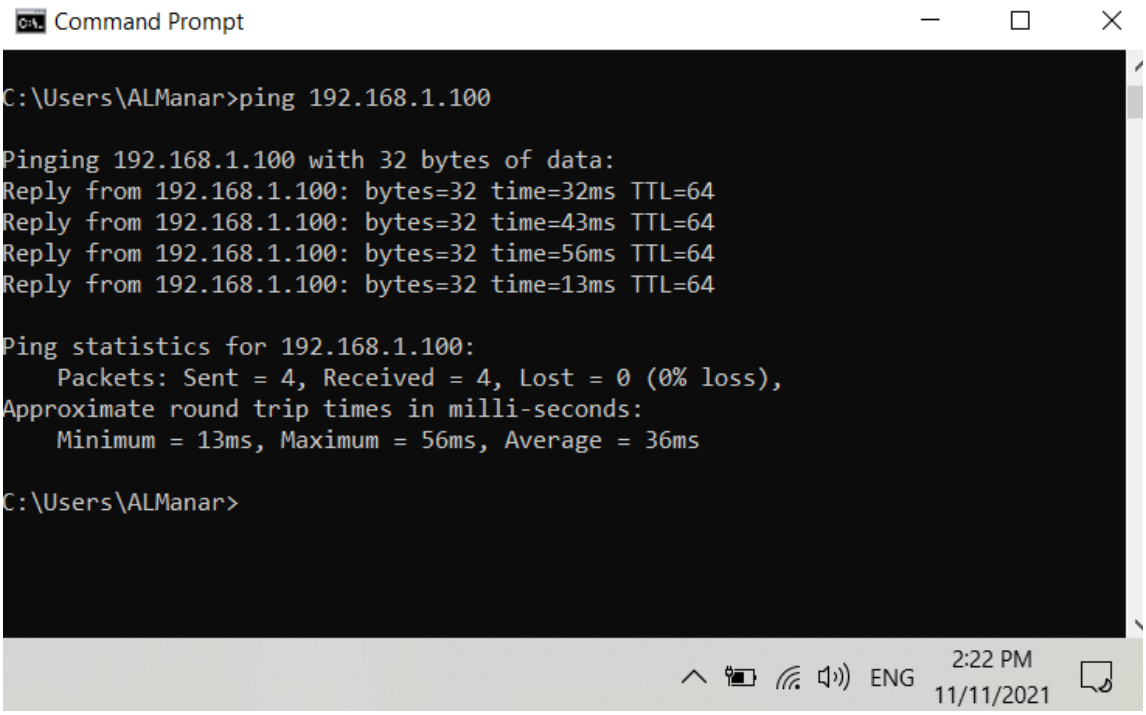


Figure 1: ping Command - Device on the same Network

1.2. ping b.root-servers.net

As we can see the result in figure 2, it displays the total number of packets sent. As a result, the number of packets received is displayed (here we sent 4 packets where all packets have the same TTL, we sent to b.root-server.net four packets to the destination with IP 199.9.14.201 and the destination response back with the same four packets, all packets are received with different delays. Also, we sent out 32 bytes of data and we got back 32 bytes and this is stable connection.

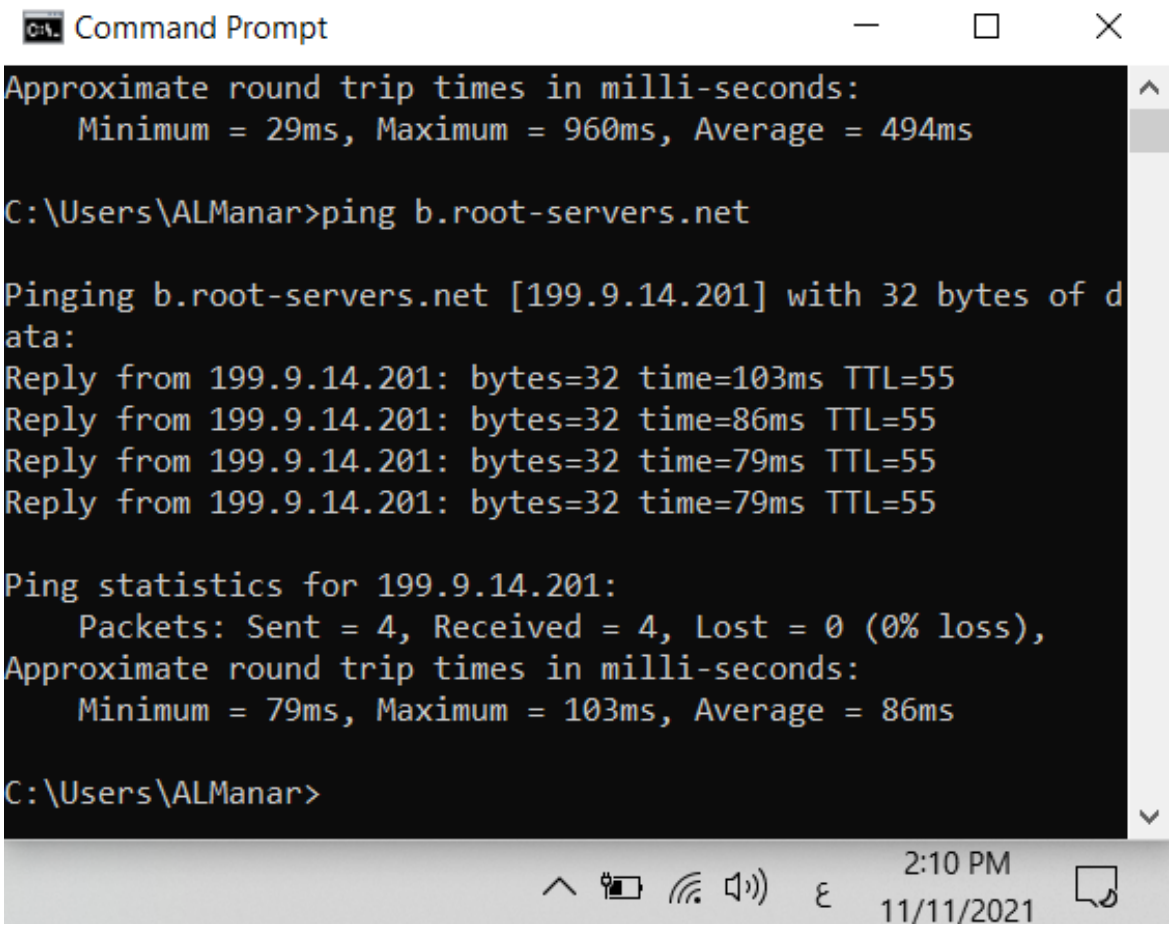


Figure 2: ping b.root-servers.net

1.3. tracert b.root-servers.net

The tracer command was used to display several details about a packet's path from the computer or device you're on to the destination you select. and this command sends 3 messages for every router and waits the response from the router, it continues in this process until it reaches the chosen IP. There are 5 columns in the end result, the number one is the hop number (TTL) and the time it takes for packets to make each hop is shown in the 3columns below (TTL), The last column is the server at the specified hop.

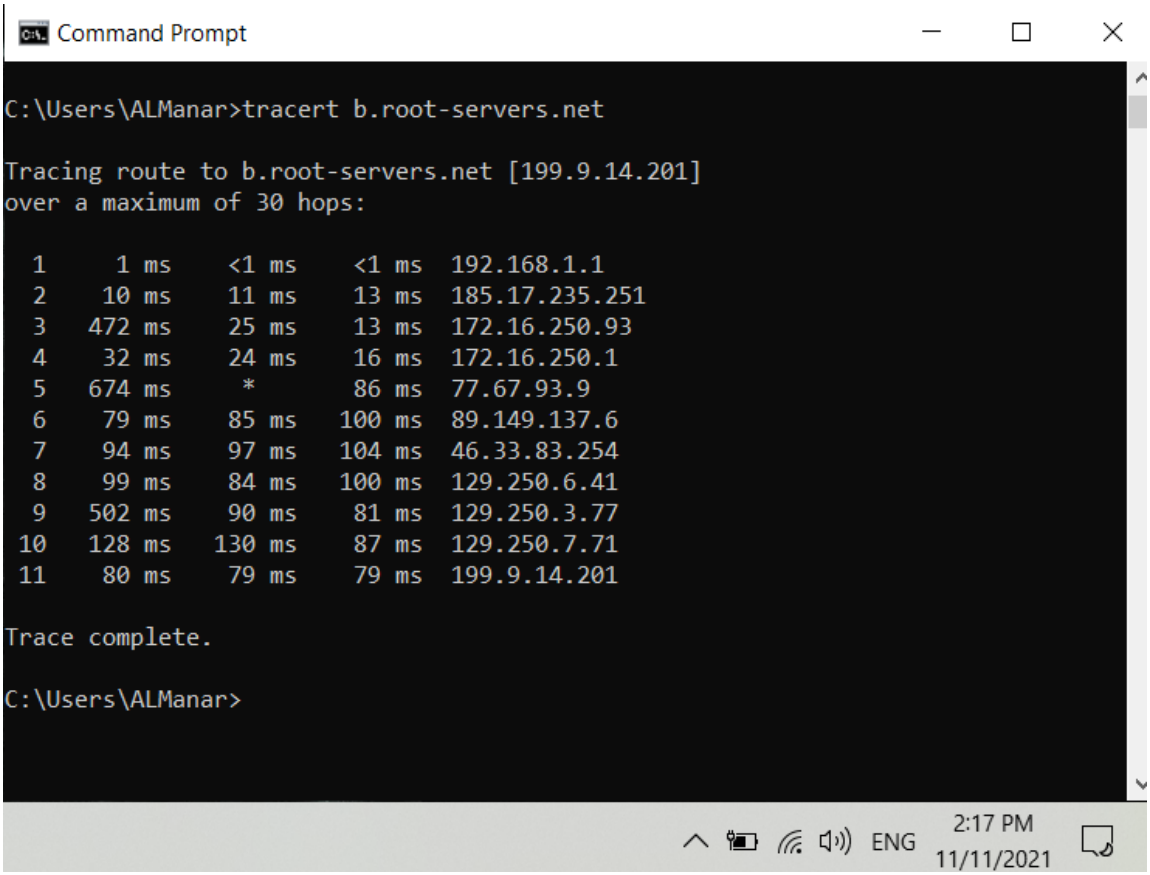


Figure 3: tracert Command

1.4. nslookup b.root-servers.net

As we can see in finger 4, when we used nslookup that is used to diagnose DNS problems first it prints the server and the address Unknown server is my router with the address 192.168.1.1 and prints the name and the 3 addresses of the server which is the host that we sent.

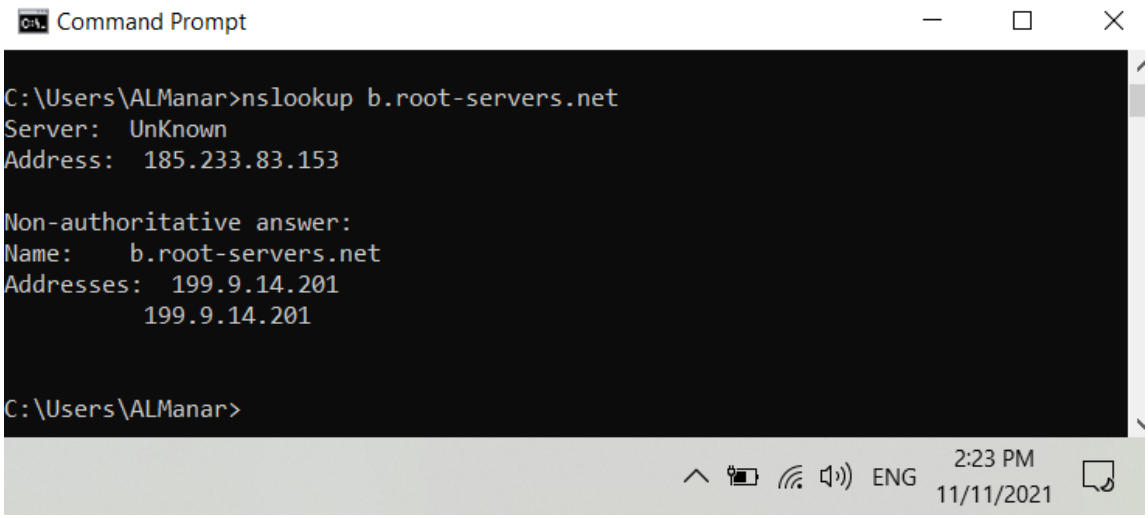


Figure 4: nslookup Command

2. Part II

2.1. Explanation

In this part we are asking the user to enter the website name to get HTTP response from a specific webserver and calculate the http response time for this webserver. First, we will create a socket object then we sent data and started the time and received data to finish the time and finally we found the response time. In addition, we will display the response using HEAD method.

2.2. Response

The response information shows that HTTP response status codes is 200 OK (request succeeded, requested object later in this message), with content type (text/html) and response time = 328.12 ms.

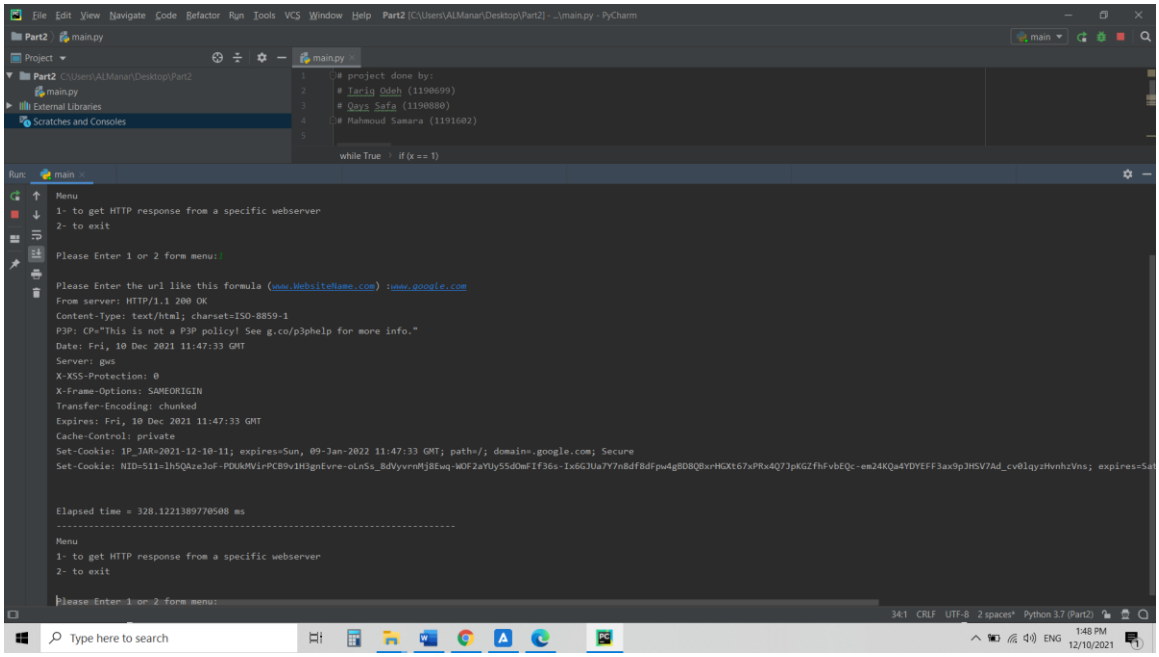


Figure 5: HTTP response from www.google.com webserve

2.3. Full Code with comments

```
# project done by:
# Tariq Odeh (1190699)
# Qays Safa (1190880)
# Mahmoud Samara (1191602)

import time
from socket import *

print("------------------------------------------------------------------------
---")
print("\t\t\t\t\t\t{Welcome to our program}")
print("\n\t\t{This program get HTTP response from a specific webserver}")

while True:
  print("------------------------------------------------------------------------
-----")
  print("Menu")
  print("1- to get HTTP response from a specific webserver")
  print("2- to exit")

  x = int(input("\nPlease Enter 1 or 2 form menu:"))

  if(x == 1):
    url = input("\nPlease Enter the url like this formula
(www.WebsiteName.com) :")
    NameOFServer = url
    PortOfServer = 80

    clinetSoket = socket(AF_INET, SOCK_STREAM)  # Create a socket object

    clinetSoket.connect((NameOFServer, PortOfServer))  # Connect the client
    StartTime = time.time()  # Time when send requst

    clinetSoket.send("HEAD / HTTP/1.1 \r\n".encode())  # Send some data
    clinetSoket.send(("Hostname:" + url + " \r\n\r\n").encode())

    modifiedSentence = clinetSoket.recv(1024)  # receive some data

    EndTime = time.time()  # Time when recive response

    print("From server:", modifiedSentence.decode())  # Display the response

    ElapsedTime = EndTime - StartTime  # Response time
    print(f"Elapsed time = {ElapsedTime * 1000} ms ")  # Display the response
time

    clinetSoket.close()

  elif(x == 2):
    break
```

3. Part III

In this part we will use socket programming, implement a web server in python that is listening on port 6500.

3.1. Main Page

http://localhost:6500/ **or** http://localhost:6500/index.html **or** http://localhost:6500/main.html

In the main page we used html language to design it and to put names, numbers and information about each student we used css language to arrange the boxes and the full design.

Main Page in the browser window:

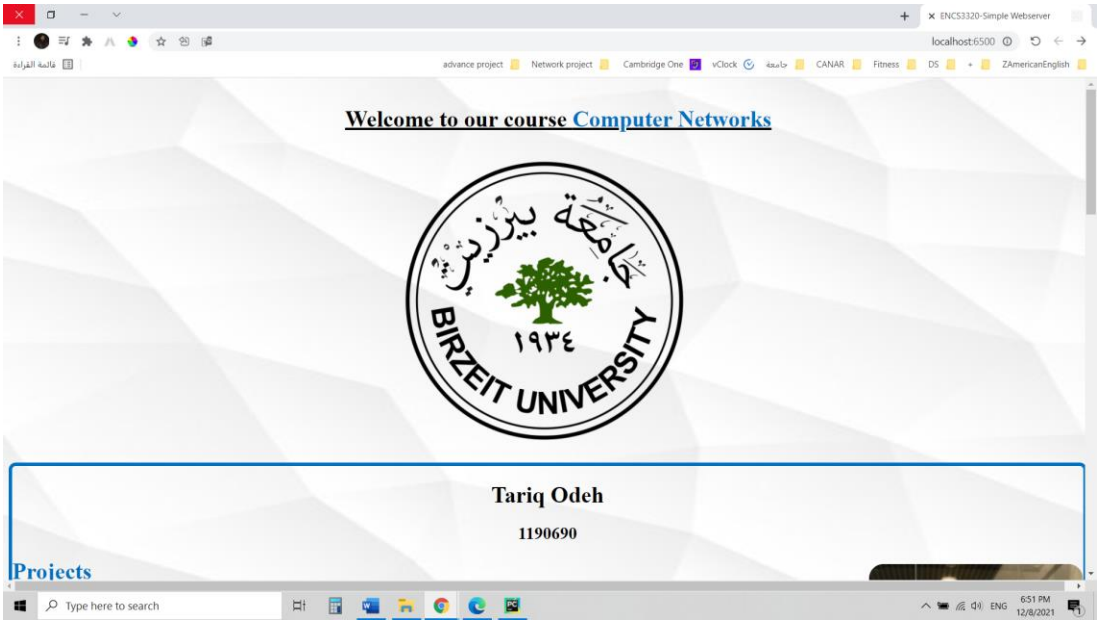


Figure 6: localhost:6500 browser window – 1

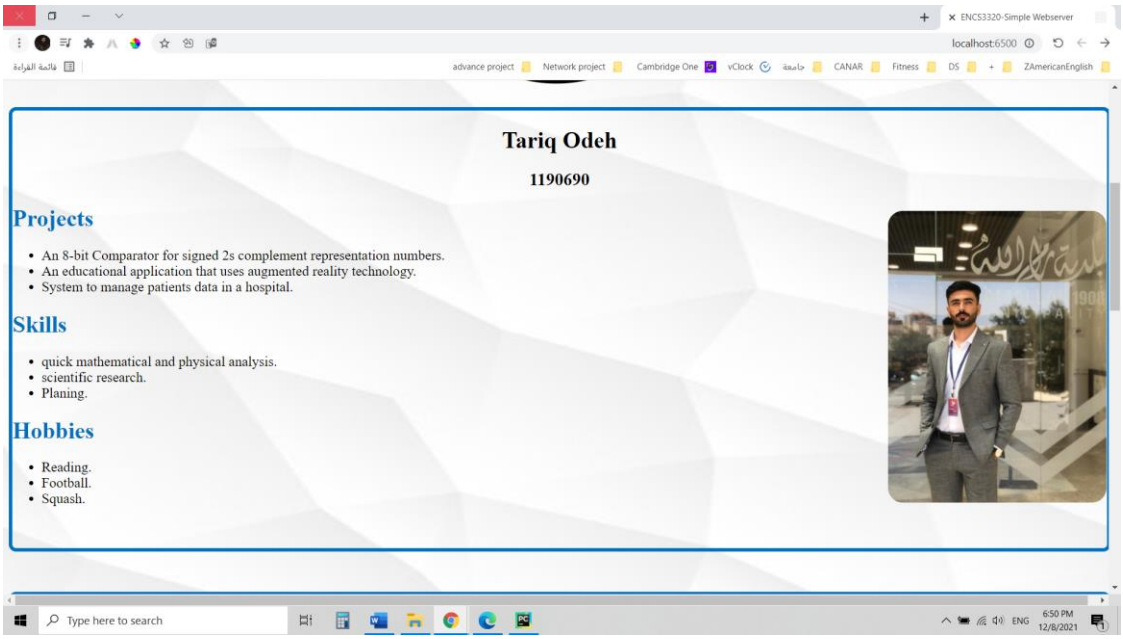


Figure 7: localhost:6500 browser window – 2

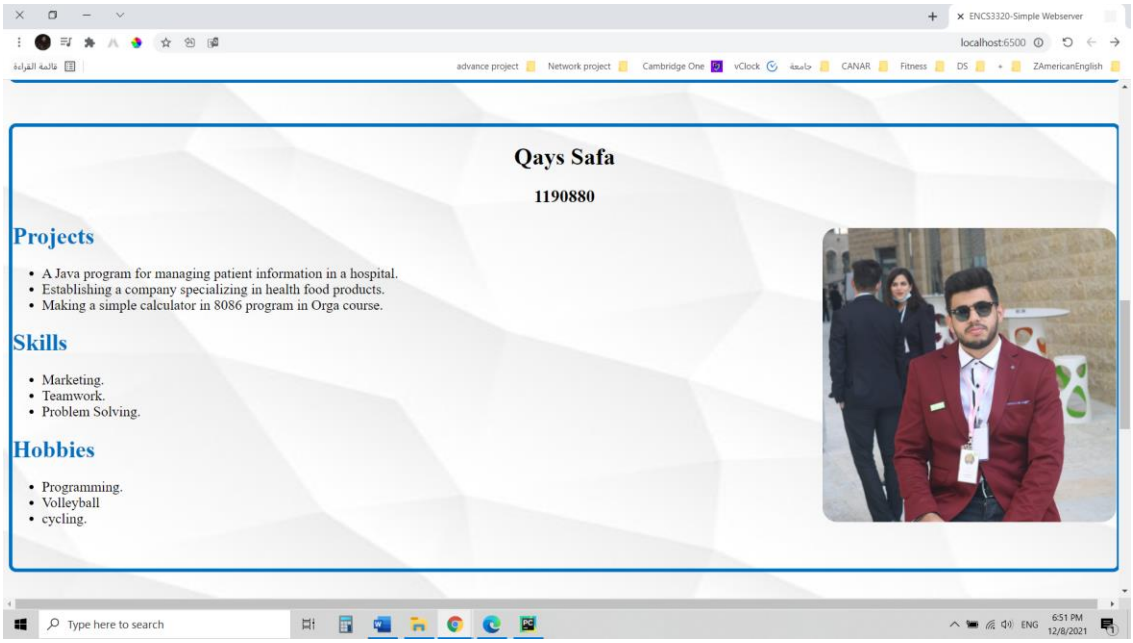


Figure 8: localhost:6500 browser window – 3

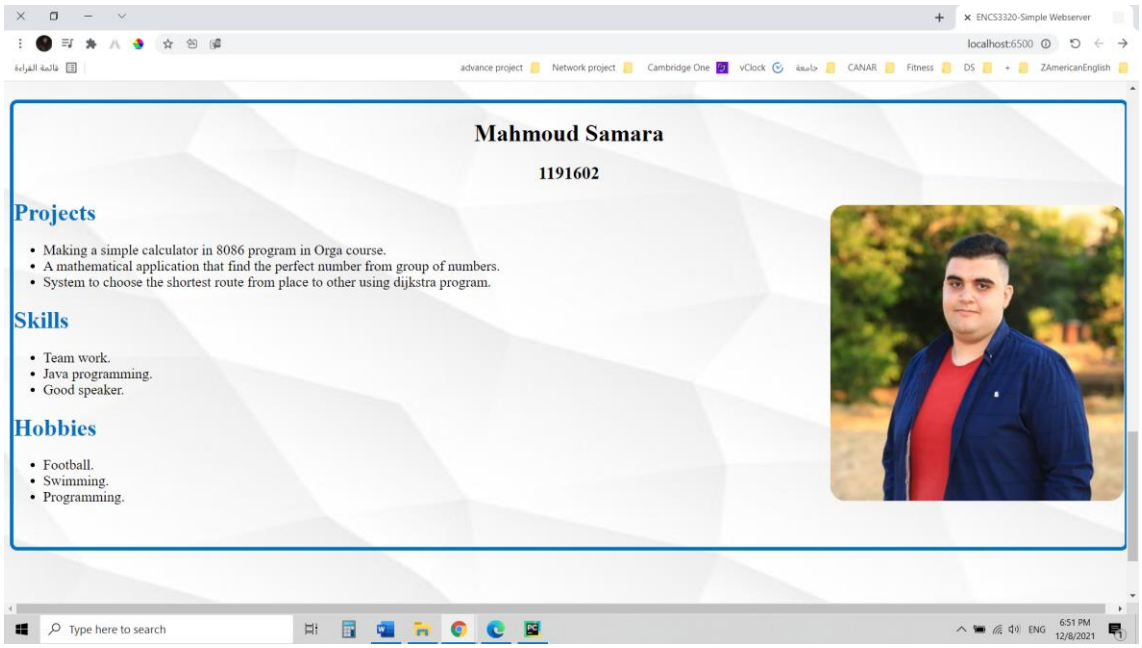


Figure 9: localhost:6500 browser window – 4

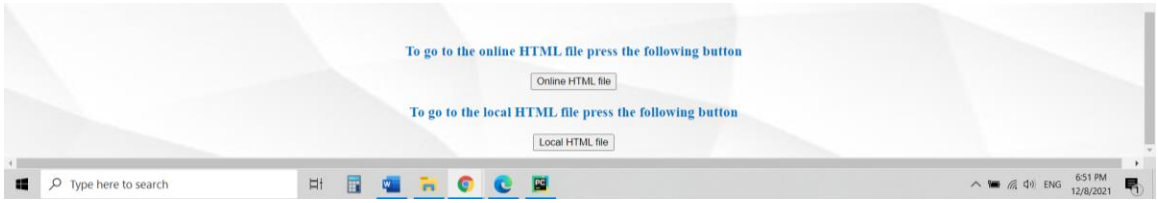


Figure 10: localhost:6500 browser window - 5

In the following figures, it will give the same results as the previous figures, but it will be using localhost:6500/index.html and localhost:6500/main.html in the browser.

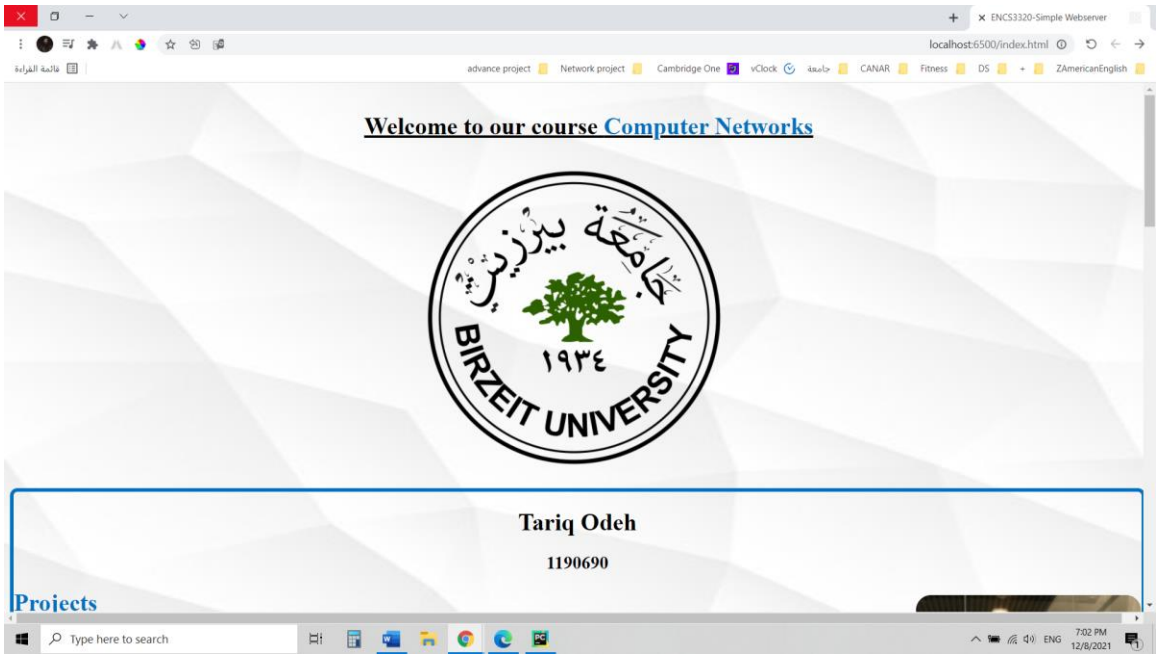


Figure 11: localhost:6500/index.html browser window

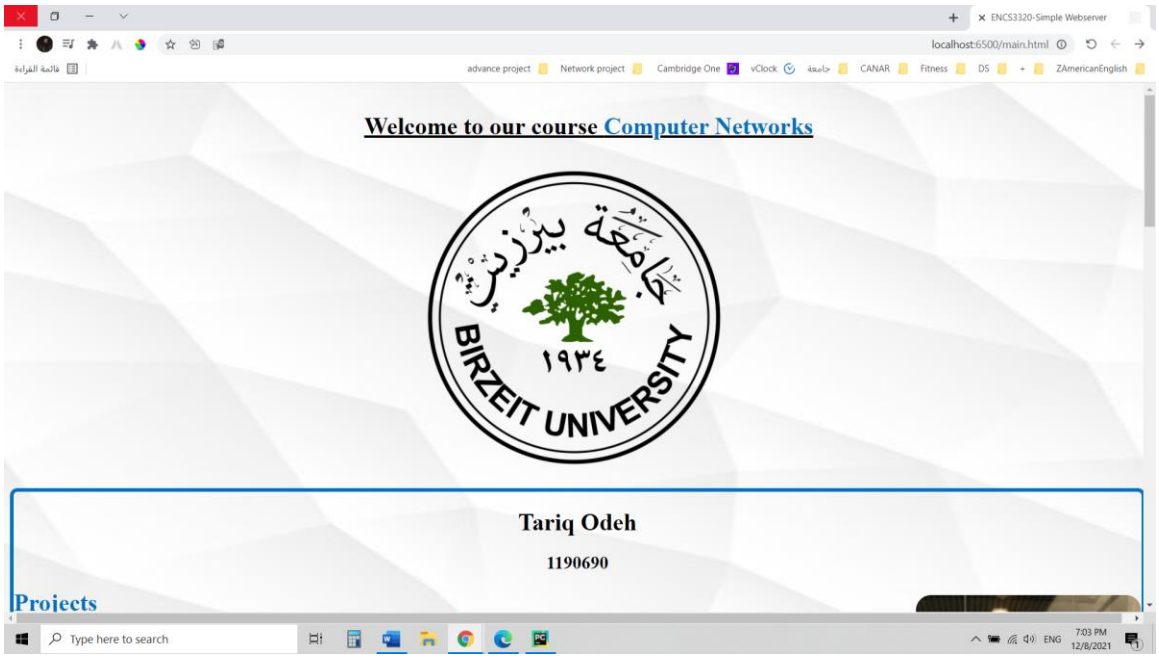


Figure 12: localhost:6500/main.html browser window

In the following figures we will see where we will go when we press online html file button and same thing for local html file button.

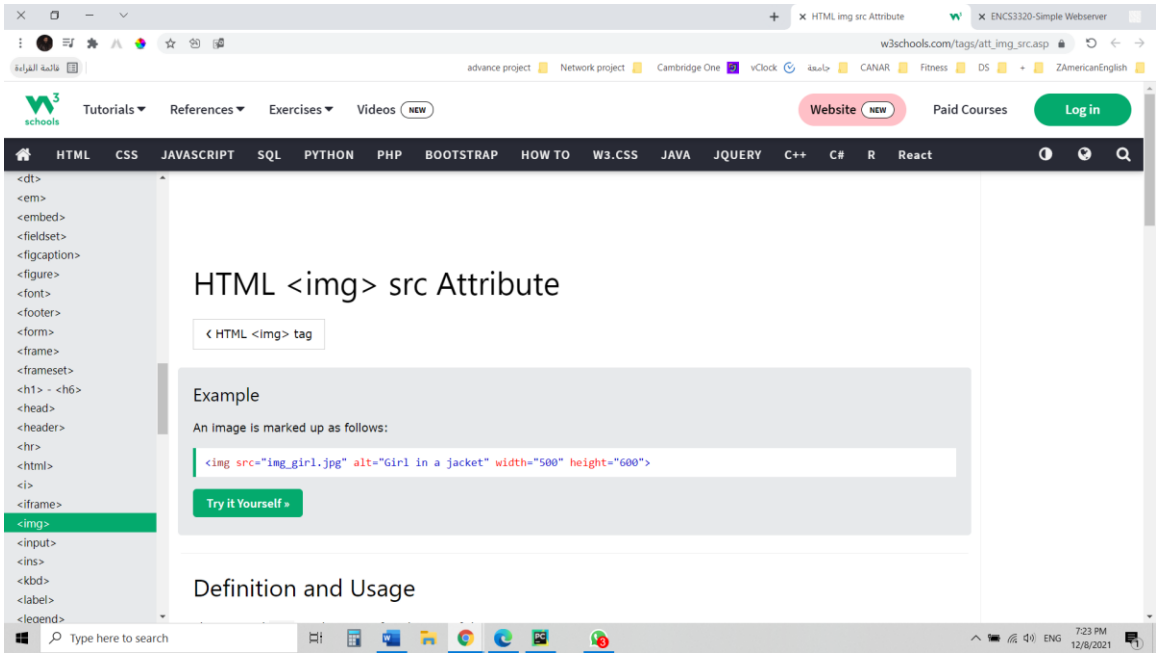


Figure 13: Online HTML file browser window (button)

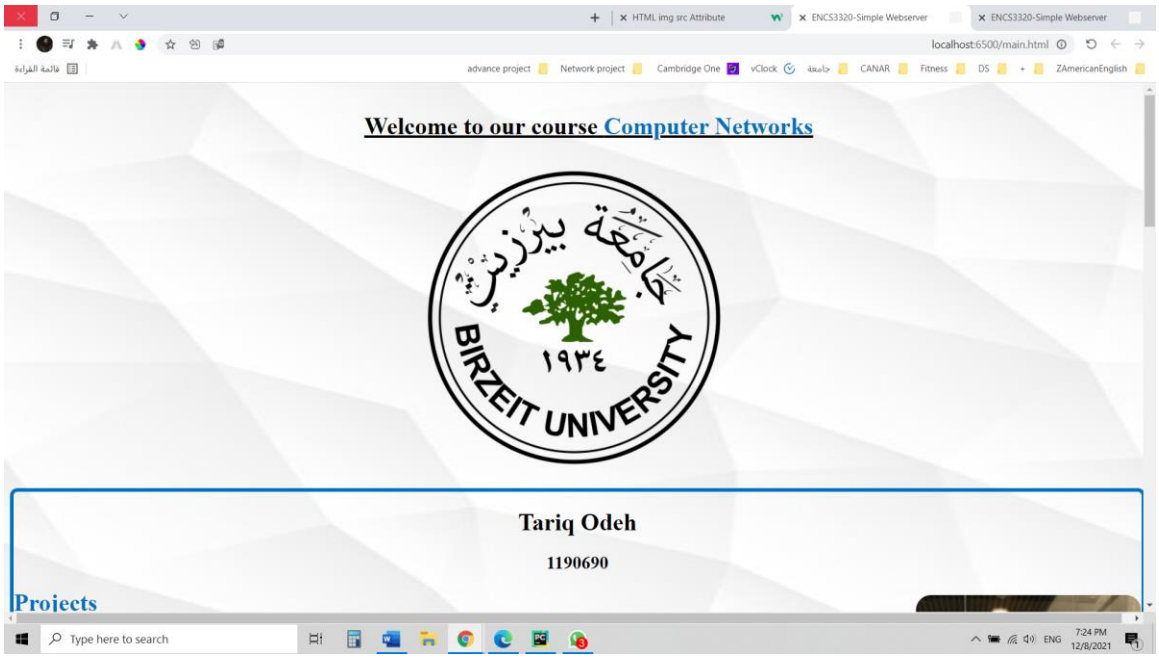


Figure 14: Local HTML file browser window (button)

Requests:

As we can see in the figures below that shown the http request for localhost:6500. We can see that http request is OK and everything is right, and keepalive means persistent, after that it specified all the contents in localhost with accepted content type. As we can see there are many responses like mahmoud or tariq image also the css file (style.css), and the method that we use is GETmethod.

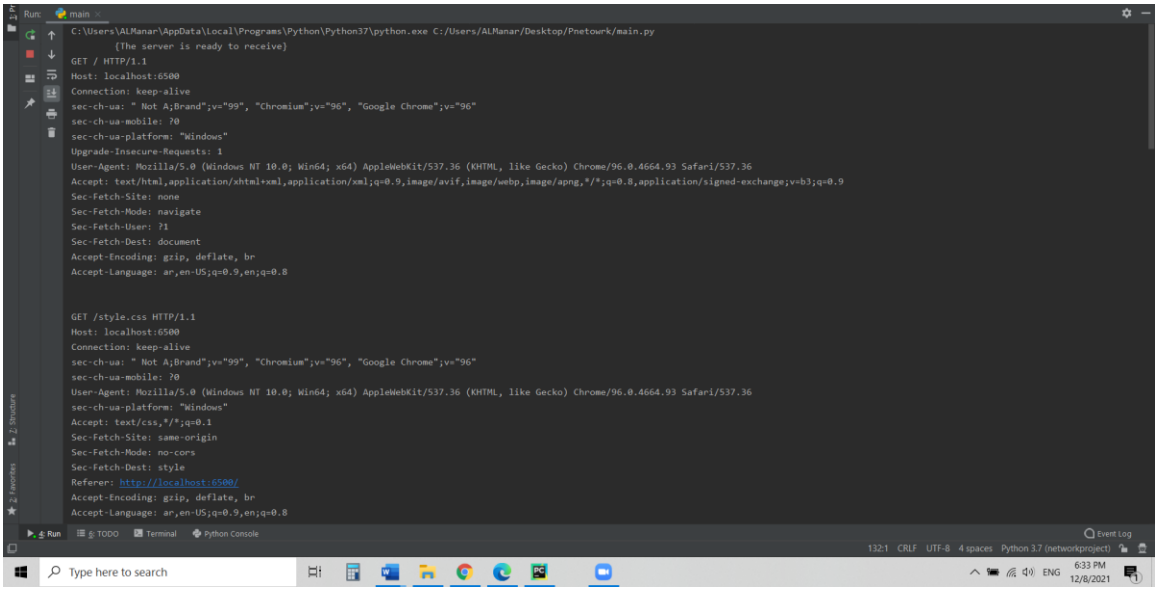


Figure 15: Main Page HTTP requests printed on command Line - 1

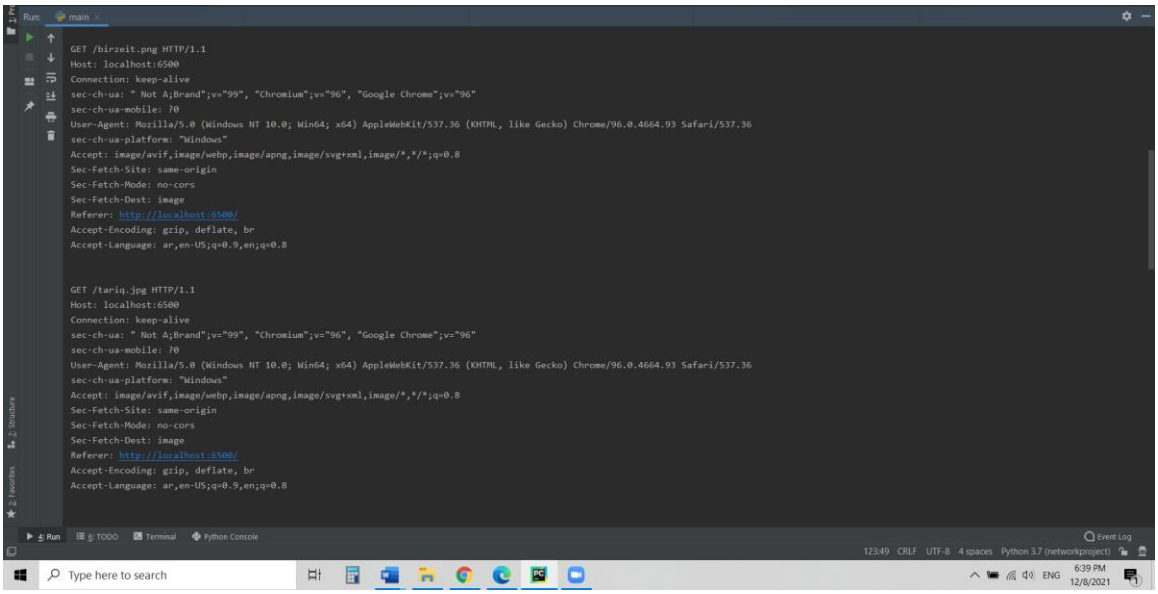


Figure 16: Main Page HTTP requests printed on command Line - 2

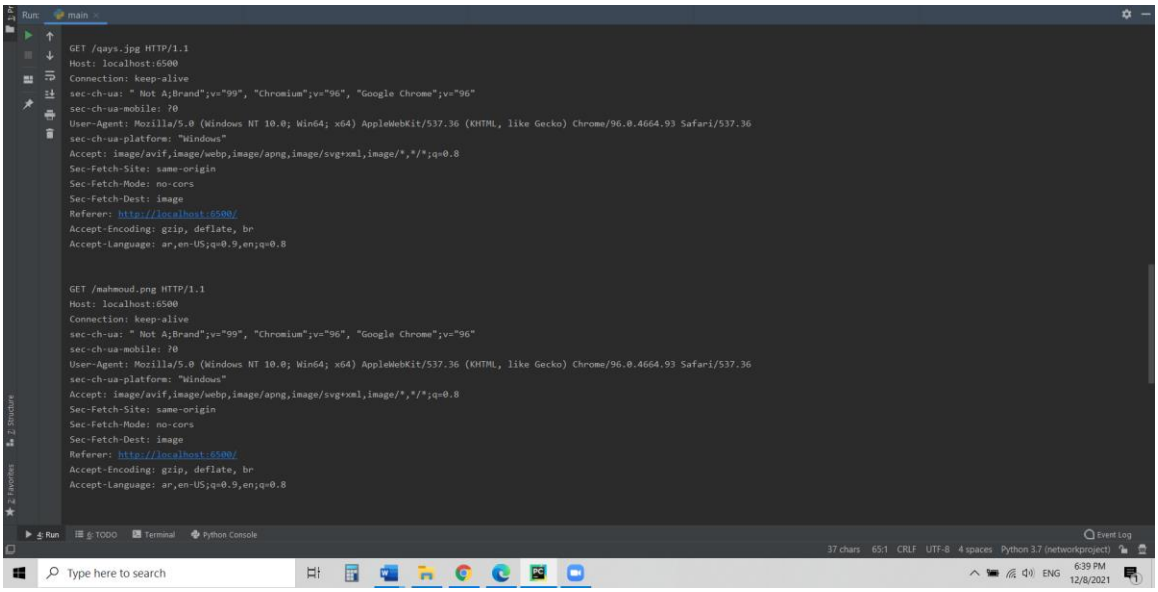


Figure 17: Main Page HTTP requests printed on command Line - 3

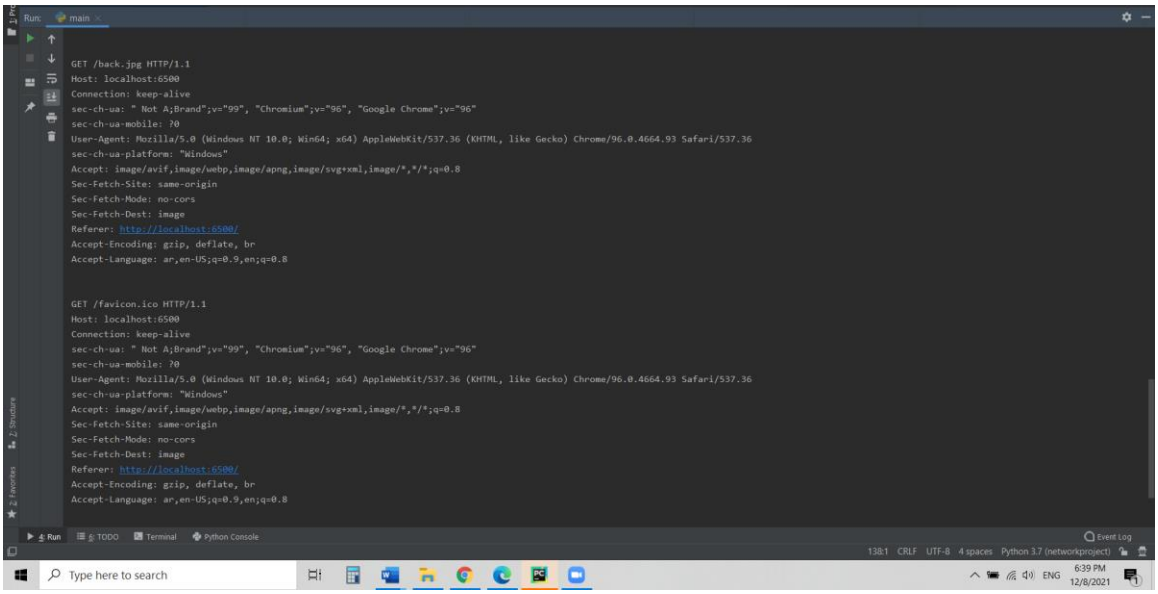


Figure 18: Main Page HTTP requests printed on command Line - 4

Screenshot from another device (phone):

To test our program from other divide: first we must know ipv4 for the device we work on it (origin device), then we make run for the code and note that both origin device and other device are on the same network. Finally, we used the following IP address to open the project: 192.168.1.1.158:6500.

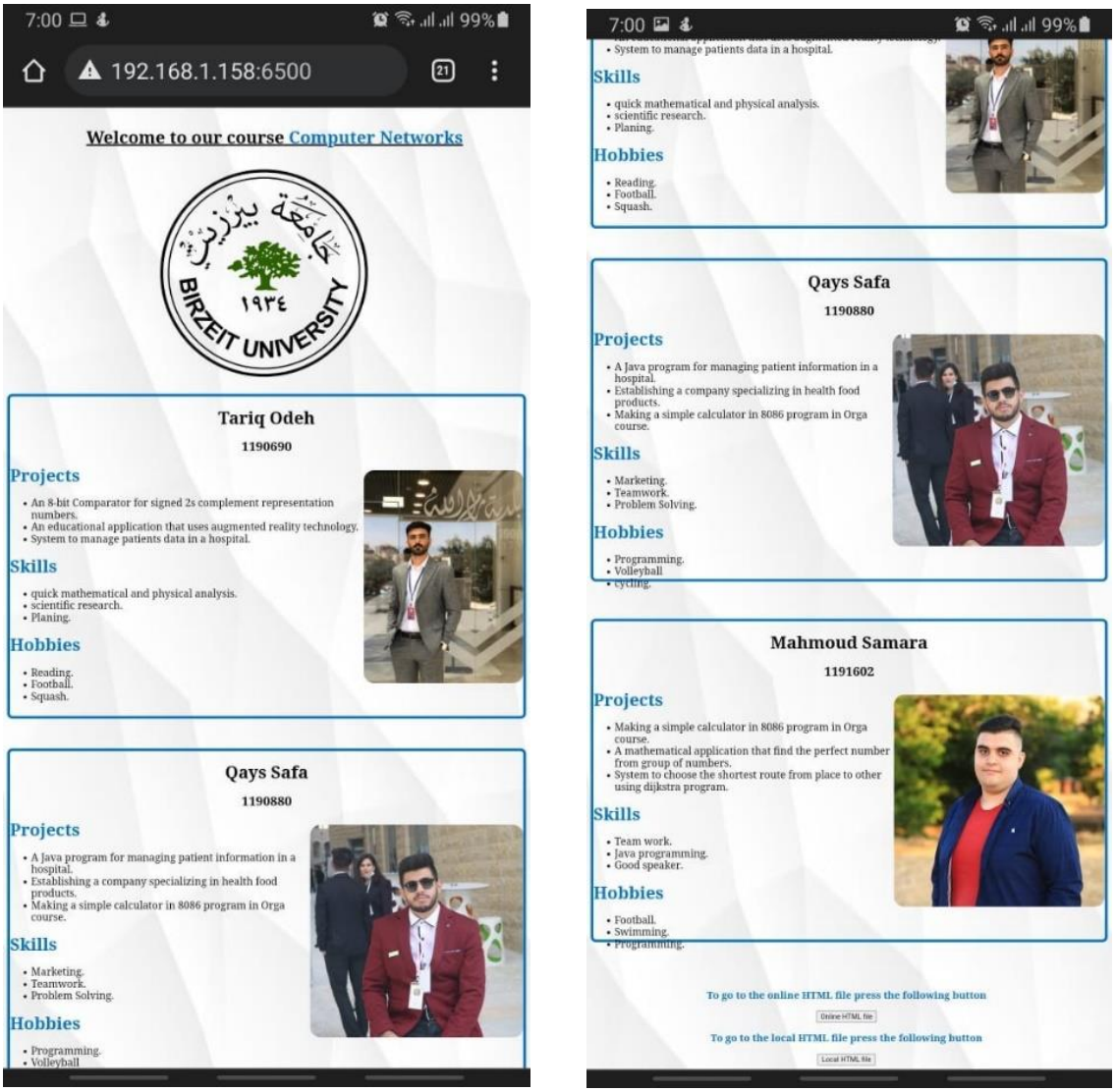


Figure 19: Localhost:6500 or Localhost:6500/index.html From phone

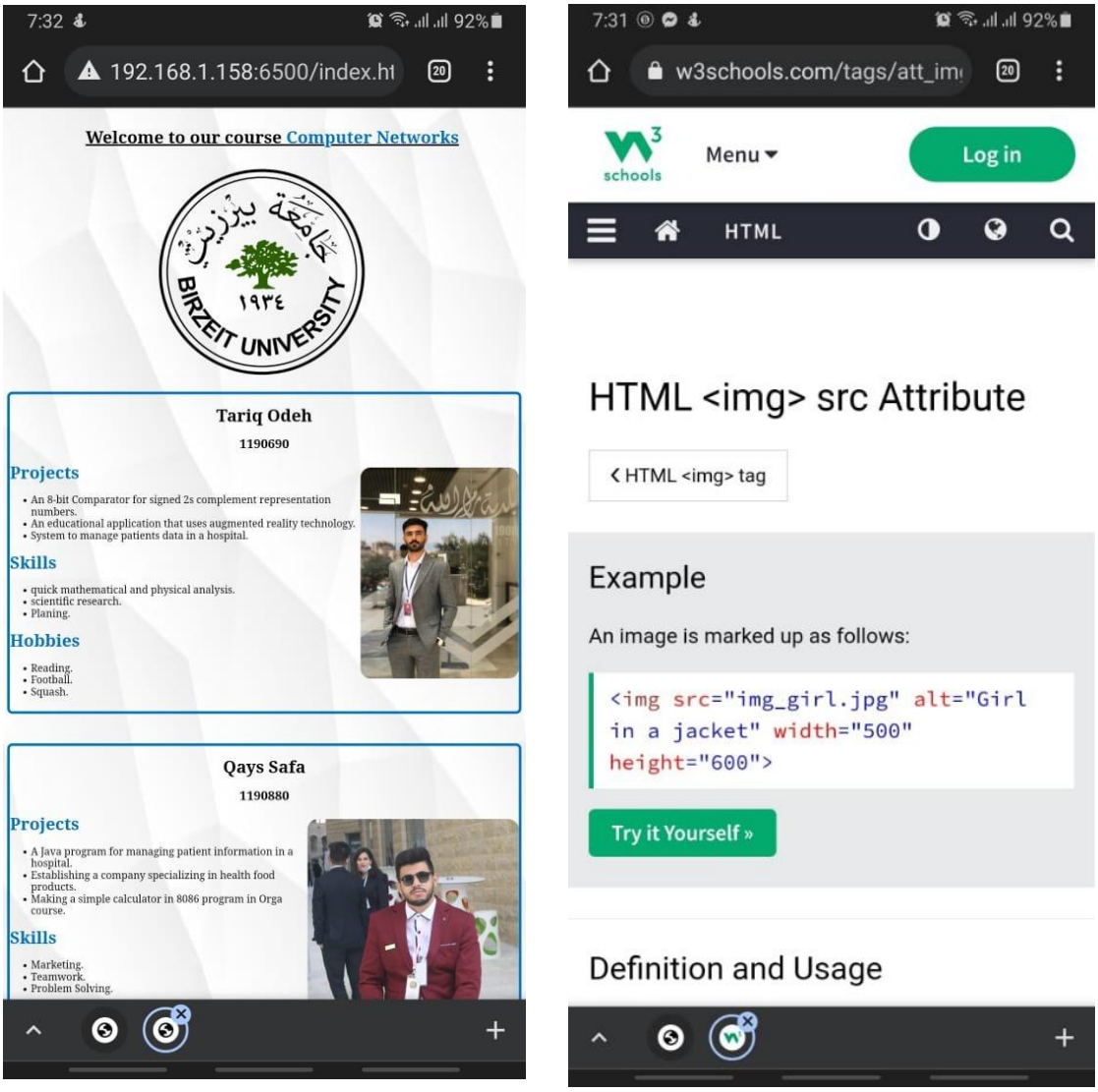


Figure 20: Local host:6500 and Online host (button) From phone

3.2. PNG Image

http://localhost:6500/mahmoud.png

Main Page in the browser window:

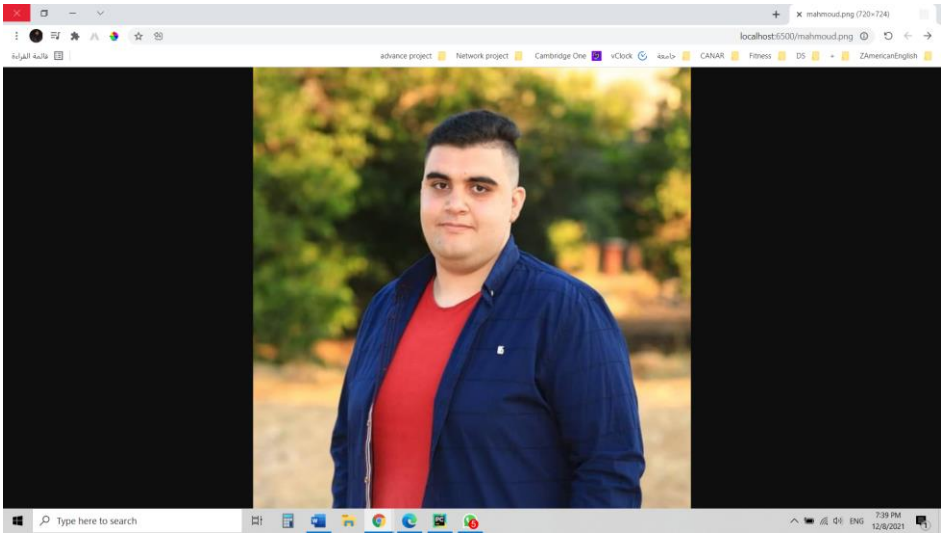


Figure 21: localhost:6500/mahmoud.png browser window

Requests:

As we can see in the figure below that shown the http request for an image with type (png). We can see that http request is OK and everything is right, and keepalive means persistent, after that it specified all the contents in localhost with accepted content type. As we can see there are a response for mahmoud image and in the website, it appears only the image as what we asked.

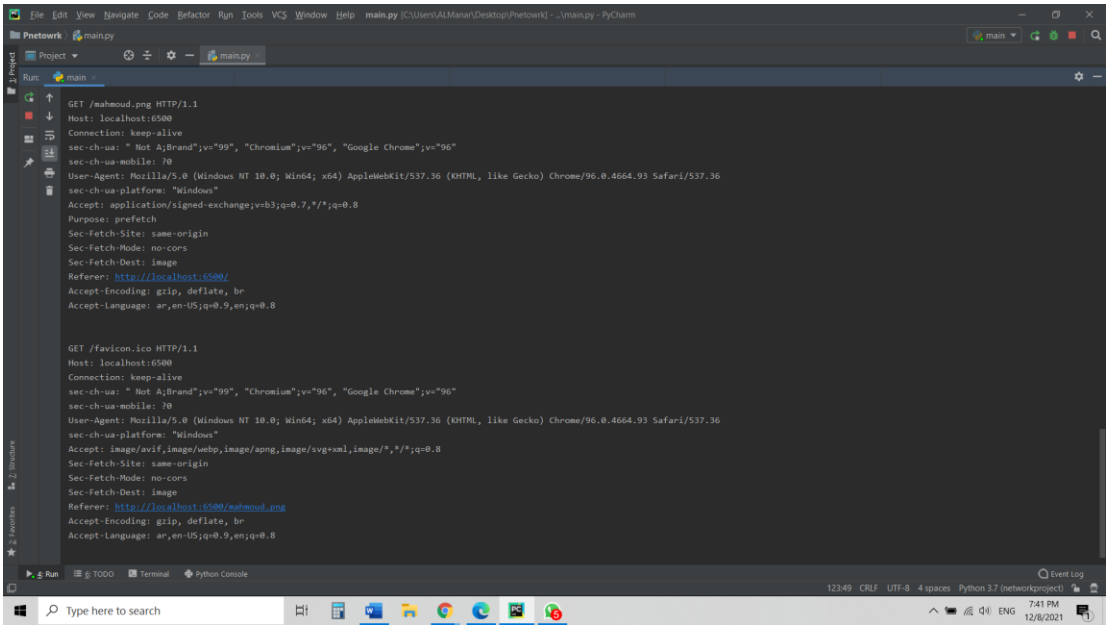


Figure 22: localhost:6500/mahmoud.png HTTP requests printed on command line

3.3. JPG Image

http://localhost:6500/qays.jpg

Main Page in the browser window:

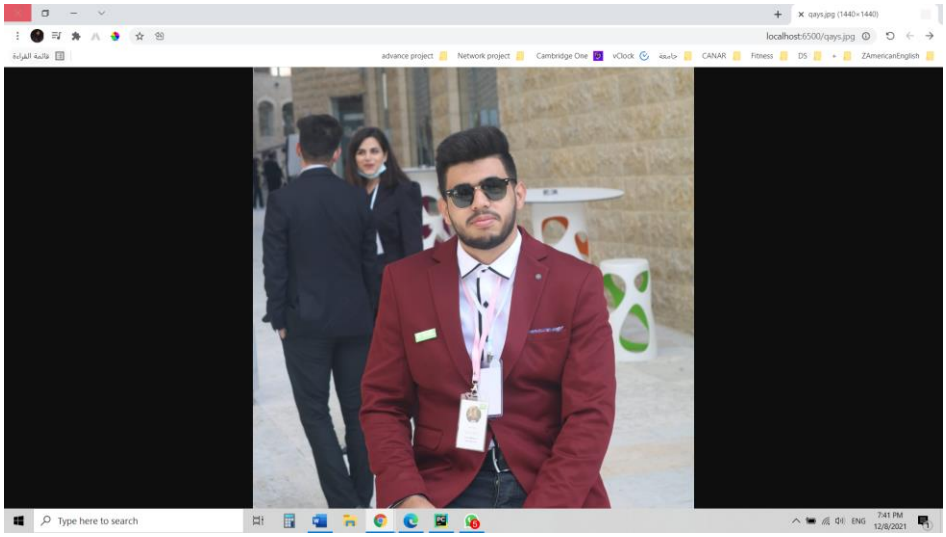


Figure 23: localhost:6500/qays.jpg browser window

Requests:

As we can see in the figure below that shown the http request for an image with type (jpg). We can see that http request is OK and everything is right, and keepalive means persistent, after that it specified all the contents in localhost with accepted content type. As we can see there are a response for qays image and in the website, it appears only the image as what we asked.

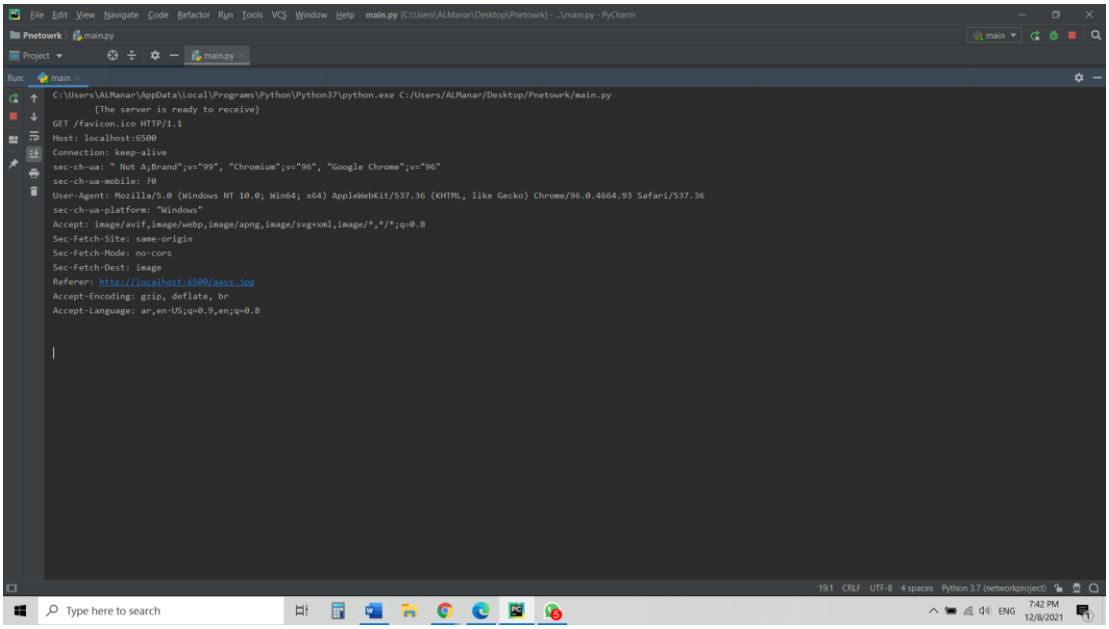


Figure 24: localhost:6500/qays.jpg HTTP requests printed on command line

Screenshot from another device (phone):

We used the following IP address to open the project:
192.168.1.1.158:6500/qays.jpg
192.168.1.1.158:6500/mahmoud.png

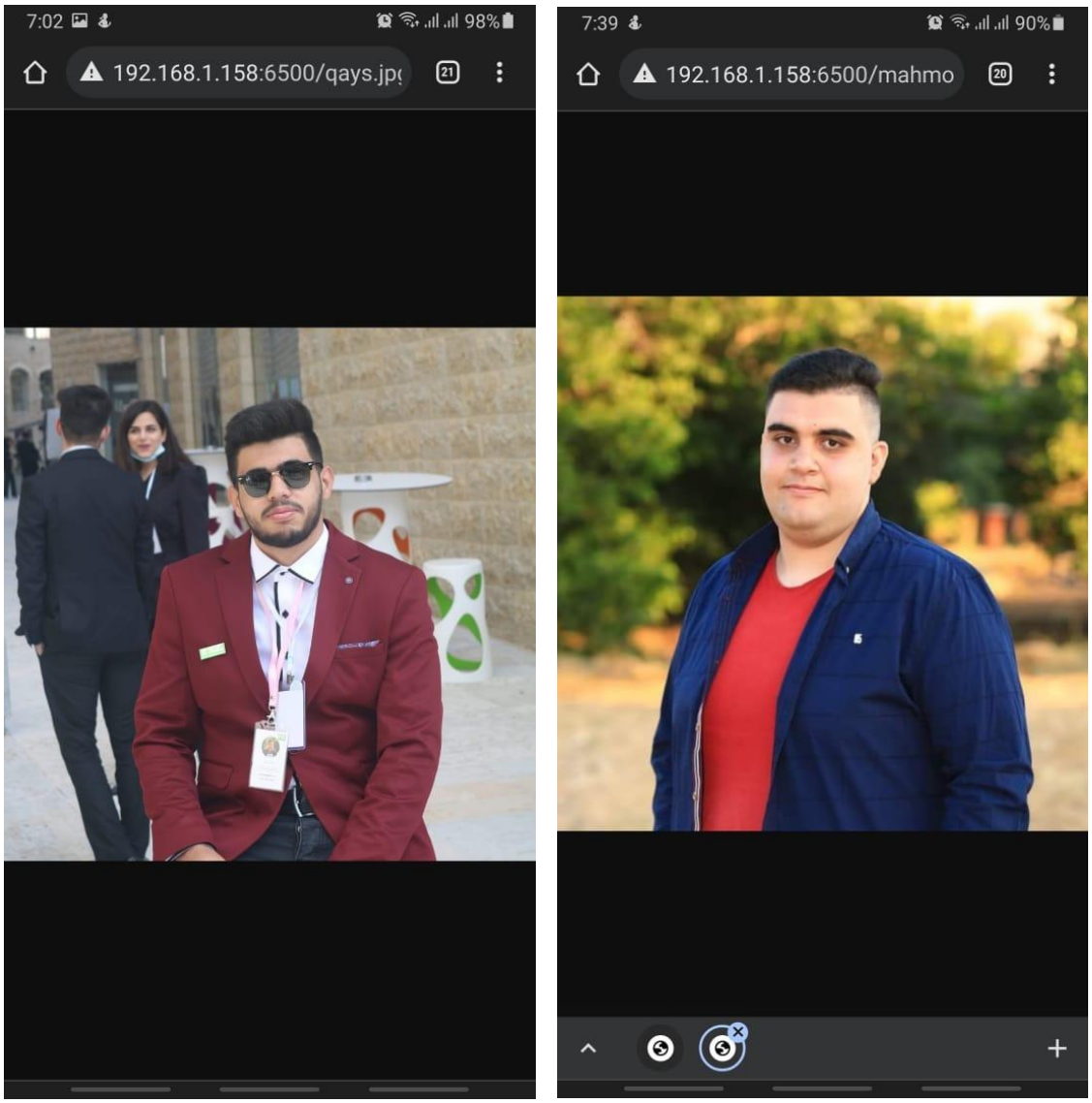


Figure 25: localhost:6500/qays.jpg and localhost:6500/ mahmoud.png browser from phone

3.4. Sort By Price

http://localhost:6500/SortByPrice

Text file:

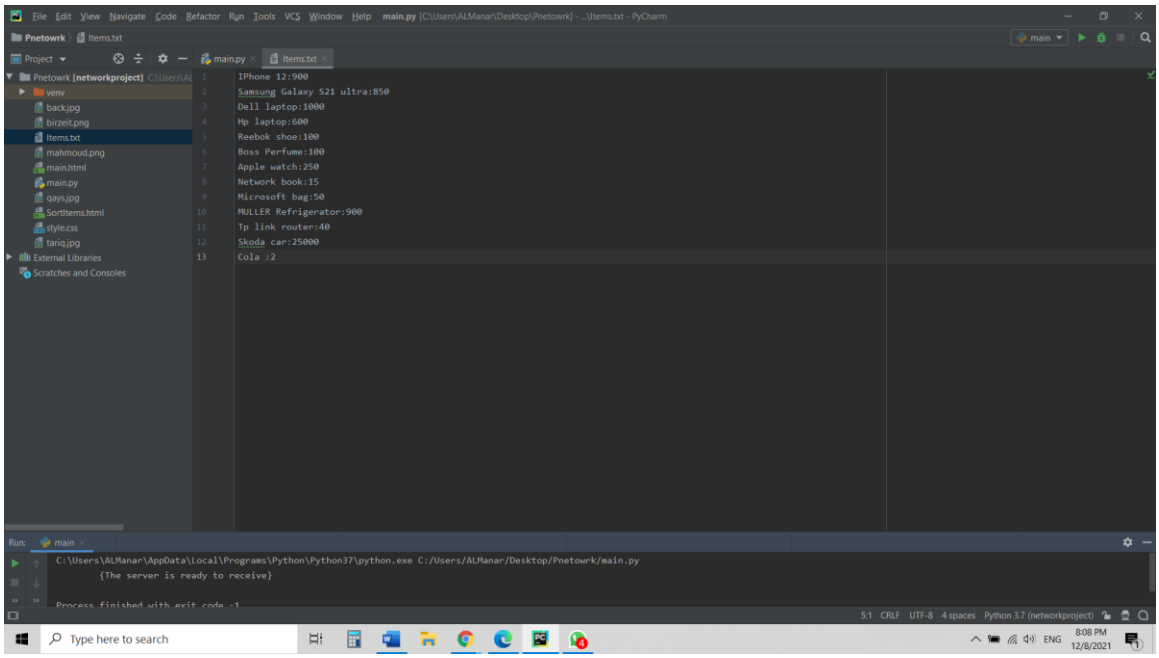


Figure 26: text file that contains the names of the items

Main Page in the browser window:

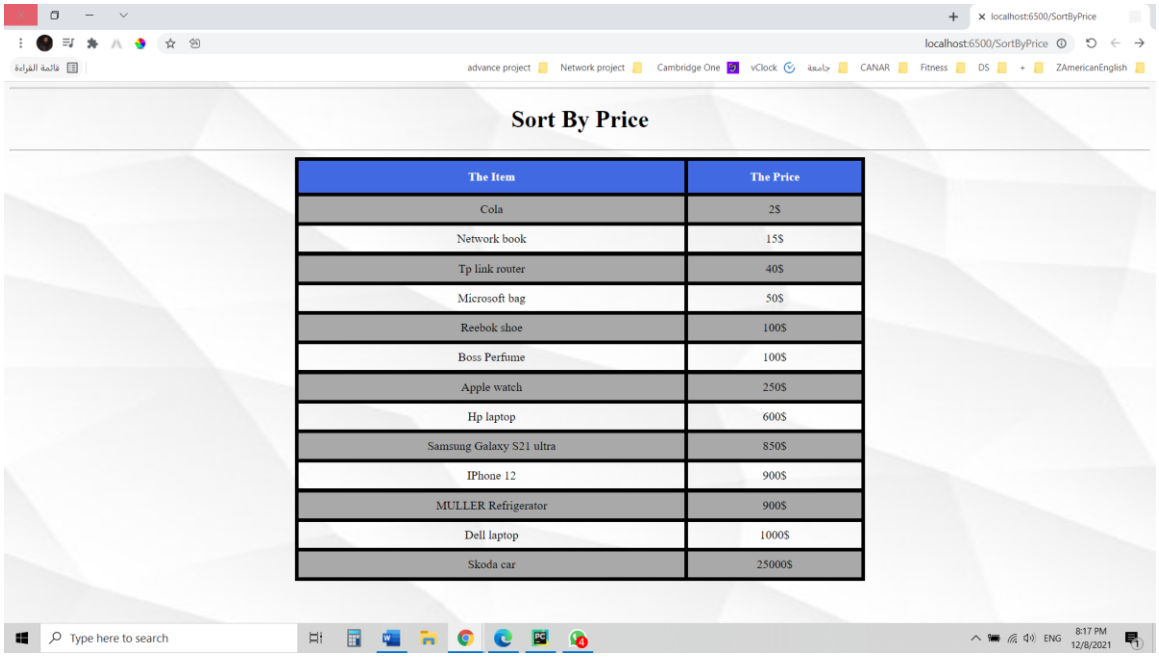


Figure 27: localhost:6500/SortByPrice browser window

Requests:

In the figure below we can see the http response for sort by price it accepted content type of text/plain, and the design of the page was arranged using html code and it was put in the main python. Note that the items were read from a text file in the python program.

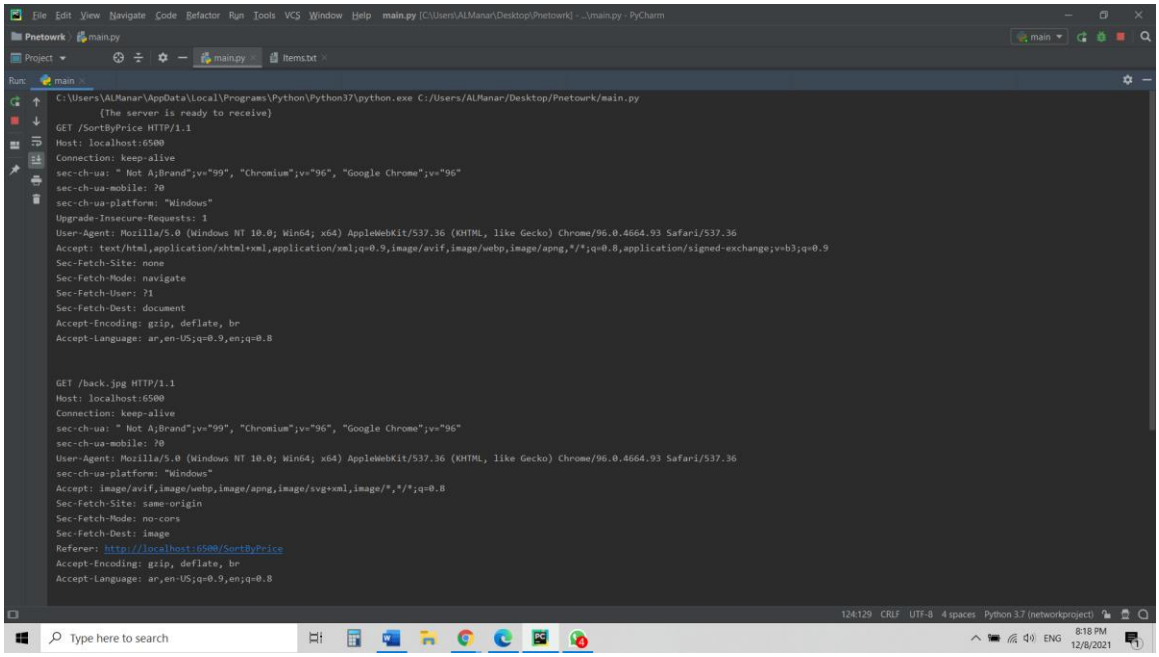


Figure 28: SortByPrice HTTP requests printed on command line - 1

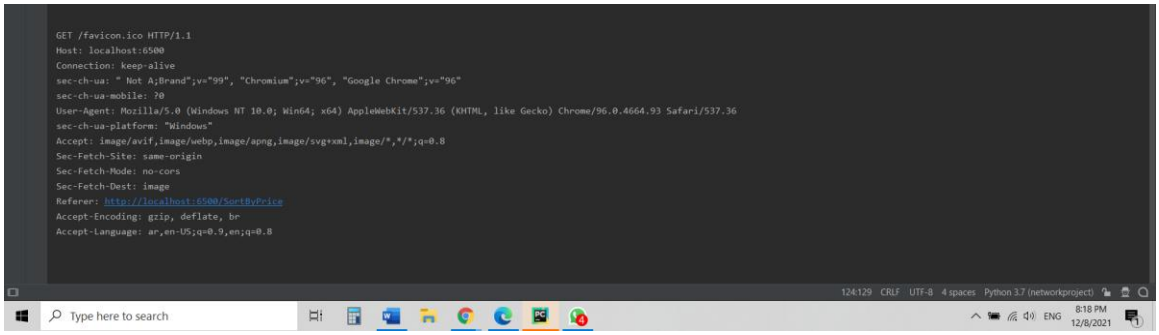


Figure 29: SortByPrice HTTP requests printed on command line – 2

3.5. Sort By Name

http://localhost:6500/SortByName

Text file:

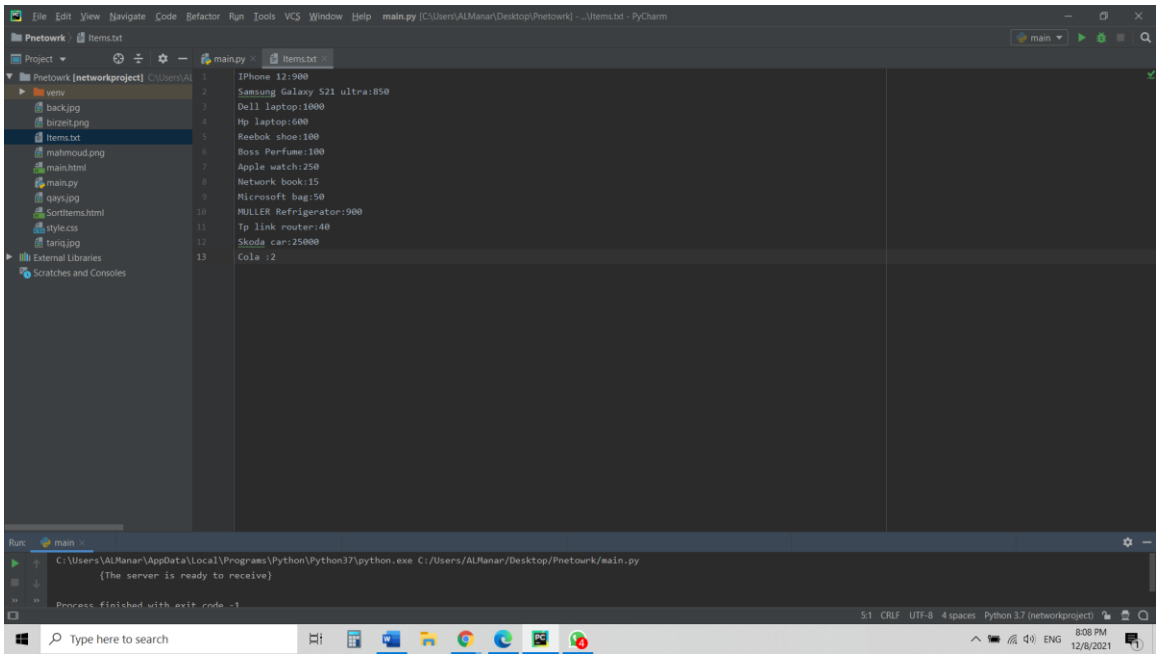
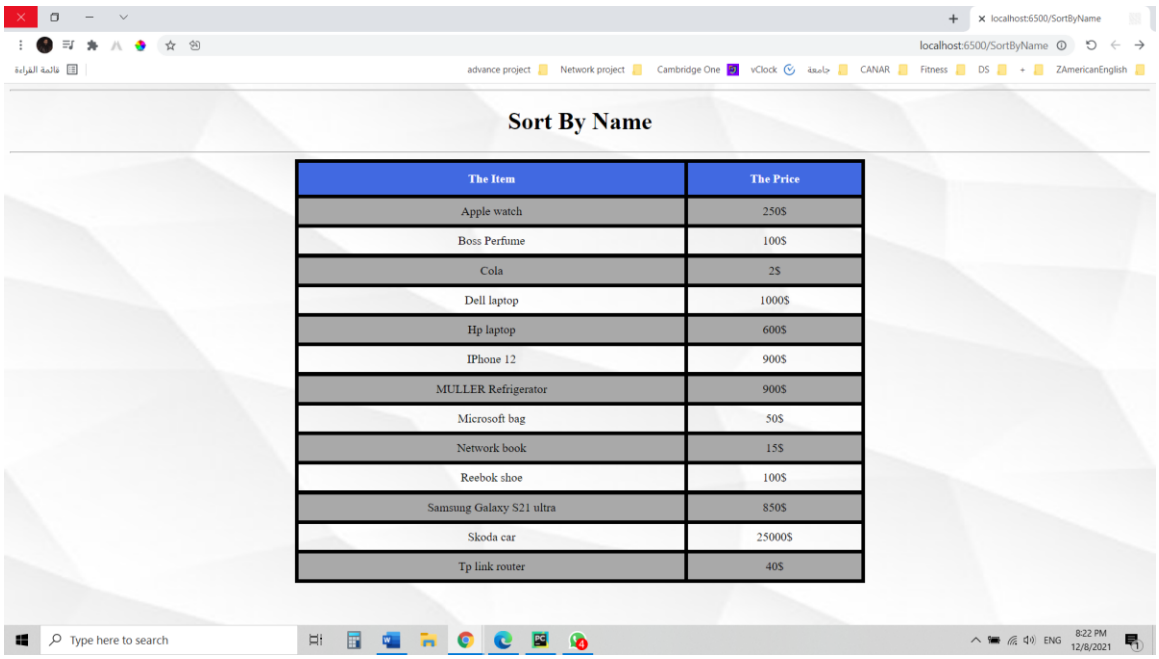


Figure 30: text file that contains the names of the items

Main Page in the browser window:



Sort By Name

The Item	The Price
Apple watch	250$
Boss Perfume	100$
Cola	2$
Dell laptop	1000$
Hp laptop	600$
IPhone 12	900$
MULLER Refrigerator	900$
Microsoft bag	50$
Network book	15$
Reebok shoe	100$
Samsung Galaxy S21 ultra	850$
Skoda car	25000$
Tp link router	40$

Figure 31: localhost:6500/SortByName browser window

Requests:

In the figure below we can see the http response for sort by name it accepted content type of text/plain, and the design of the page was arranged using html code and it was put in the main python. Note that the items were read from a text file in the python program.

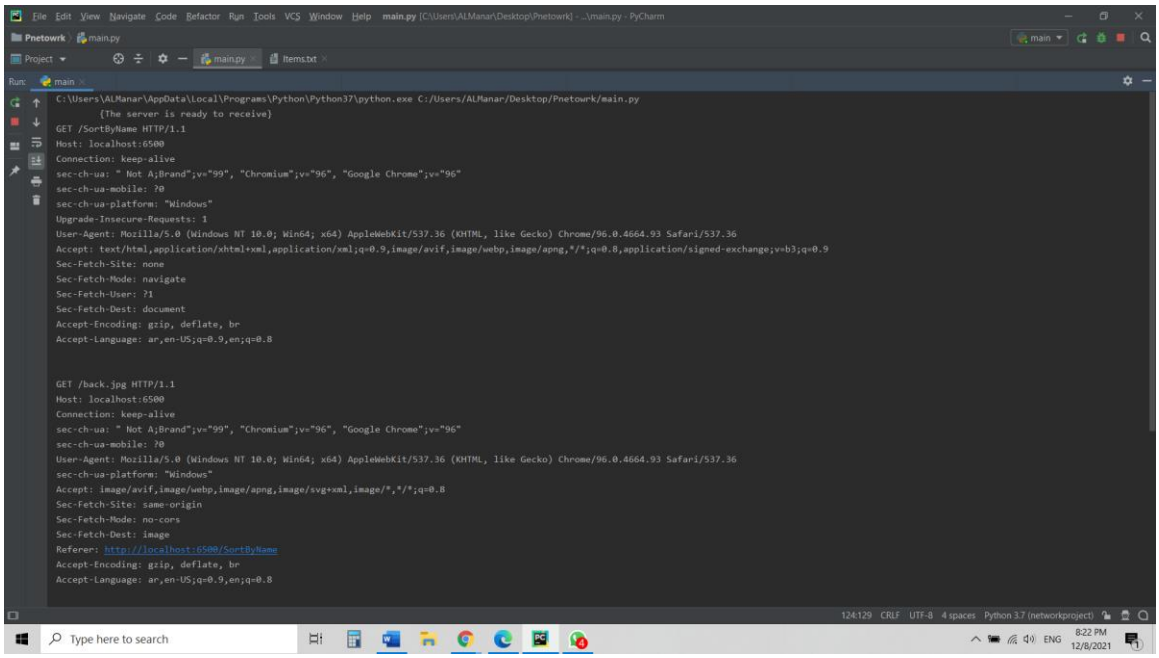


Figure 32: SortByName HTTP requests printed on command line - 1

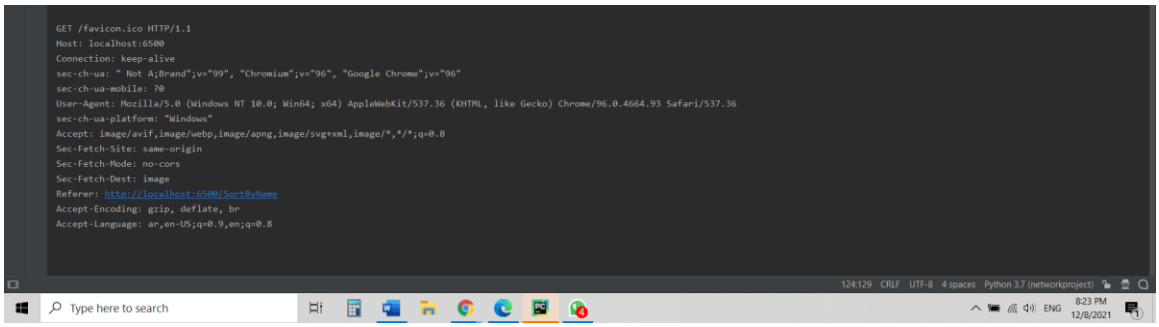


Figure 33: SortByName HTTP requests printed on command line - 2

Screenshot from another device (phone):

We used the following IP address to open the project:
192.168.1.1.158:6500/SortByPrice
192.168.1.1.158:6500/ SortByName

Figure 34: localhost:6500/SortByPrice and localhost:6500/SortByName browser from phone

3.6. Error 404

http://localhost:6500/AAAA

Main Page in the browser window:

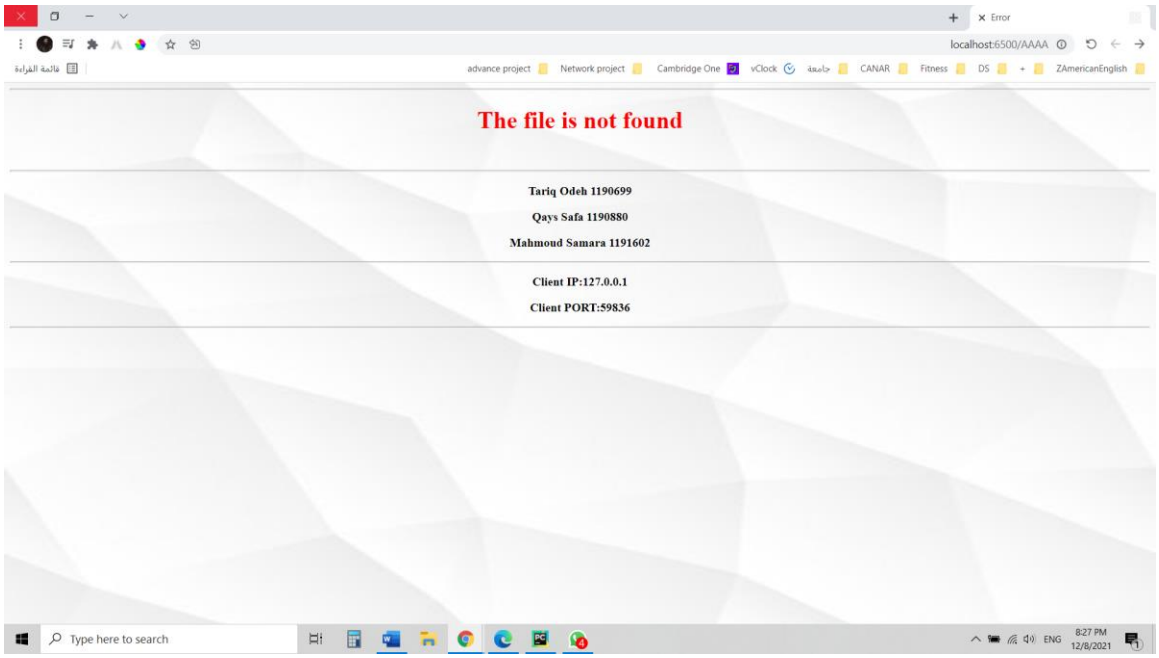


Figure 35: localhost:6500/AAAA browser window

Requests:

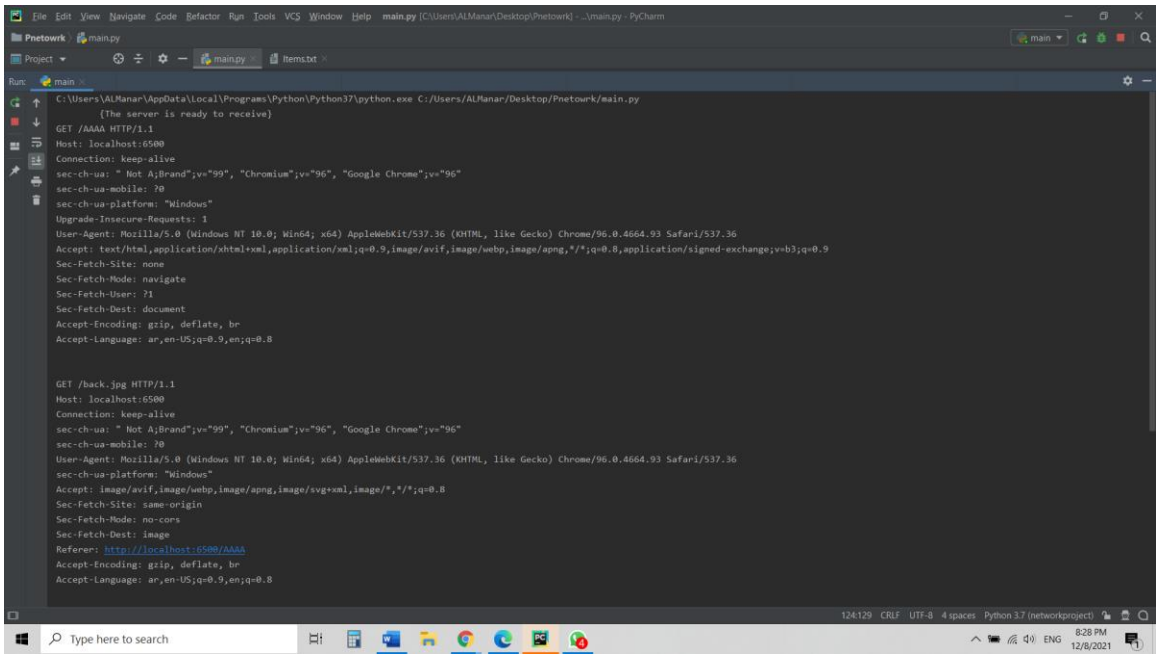


Figure 36: AAAA HTTP requests printed on command line - 1

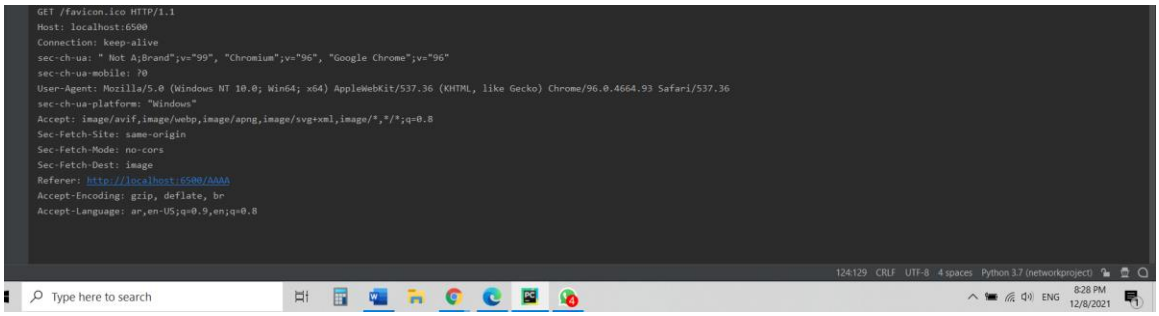


Figure 37: AAAA HTTP requests printed on command line – 2

Screenshot from another device (phone):

Figure 38: localhost:6500/AAAA browser window from phone

3.7. Full Code with comments

```
# project done by:
# Tariq Odeh (1190699)
# Qays Safa (1190880)
# Mahmoud Samara (1191602)

from socket import *
# Include Python's socket library.

items = []
# Intialise an array to put all items in it.
PORT = 6500
# Listening on port 6500.
serverSocket = socket(AF_INET, SOCK_STREAM)
# Create TCP socket for server, remote port 6500.
serverSocket.bind(("", PORT))
serverSocket.listen(1)
# Server begins listening for  incoming TCP requests.
print("\t\t{The server is ready to receive}")


# function to read the items file and cut it to items and prices

def readfile(filename):

   with open(filename) as f:
# Create inputfile to read the data in items.txt .
    item = f.readlines()
# Read the data in Items.txt line.
    for sentences in item:
# Split the data from the file and append it to a new list, then cut the data
based on ☺) .
        line = sentences.split(":")
        line[1] = str(line[1]).replace("\n", "")
        line[1] = int(line[1])
        items.append(line)
# Add item in items array.
readfile('items.txt')


while True:

    connectionSocket, address = serverSocket.accept()
# Server waits on accept() for incoming requests, new socket created on return
.
    sentence = connectionSocket.recv(1024).decode()
# Read bytes from socket.
    requestFile = sentence.split(' ')[1]

    printedfile = requestFile.lstrip('/')
# Removing the first( / )to get the requested file name .
    connectionSocket.send(f"HTTP/1.1 200 OK\r\n".encode())

    if printedfile == '' or printedfile == 'index.html':
        printedfile = 'main.html'
# Load main.html file as default so if the request is / or /index.html then
```

```
the server should send main.html file.

    try:

# Accepting different file formats
        if printedfile.endswith(".html"):
# If the request is a .html then the server should send the html file with
Content-Type: text/html.
            requestedType = 'text/html'

        elif printedfile.endswith(".css"):
# If the request is a .css then the server should send the css file with
Content-Type: text/css.
            requestedType = 'text/css'

        elif printedfile.endswith(".png"):
# If the request is a .jpg then the server should send the png image with
Content-Type: image/png.
             requestedType = 'image/png'

        elif printedfile.endswith(".jpg"):
# If the request is a .jpg then the server should send the jpg image with
Content-Type: image/jpg.
            requestedType = 'image/jpeg'

        elif printedfile == "SortByName" or printedfile == "SortByPrice":
# If the request is SortByName or SortByPrice then the server should send text
page with Content-Type: text/plain.
            requestedType = 'text/plain'

        else:
# Else then server should send text page with Content-Type: text/html.
            requestedType = 'text/html'

        if printedfile == 'SortByName' or printedfile == 'SortByPrice':

# If the user requests to sort either by name or by price for the items, it
will enter this IF condition

# to know to show the sort price page or sortname page depend on what the user
request.

                if printedfile == 'SortByName':
# To sort the items depending on the names.
                    items.sort()
                    ST = '<!DOCTYPE html><html><head><style>body {background-
image: url(' \
                         '"back.jpg");background-repeat: no-repeat;background-
attachment: fixed; background-size: ' \
                         '100% 100%;}</style></head><head><style>#Items {font-
family: Times new roman, ' \
                         'sans-serif;text-align:center;border-collapse:
collapse;width: 50%;} #Items td,' \
                         '#Items th {border: 5px solid #000000;padding: 8px;}
#Items tr:nth-child(even){' \
                         'background-color: darkgrey;} #Items tr:hover
{background-color: darkgrey;}#Items th {' \
                         'padding-top: 12px;padding-bottom: 12px;text-align:
left;text-align:center;color: ' \
```

```
                        'white;}</style></head><body><hr><center><h1>Sort By
Name</h1><table id="Items"><hr><tr ' \
                        'style="background-color: royalblue;"><th>The
Item</th><th>The Price</th></tr> '

                else:

# To sort the items depending on the pricess.
                    items.sort(key=lambda items: items[1])
                    ST = '<!DOCTYPE html><html><head><style>body {background-
image: url(' \
                        '"back.jpg");background-repeat: no-repeat;background-
attachment: fixed; background-size: ' \
                        '100% 100%;}</style></head><head><style>#Items {font-
family: Times new roman, ' \
                        'sans-serif;text-align:center;border-collapse:
collapse;width: 50%;} #Items td,' \
                        '#Items th {border: 5px solid #000000;padding: 8px;}
#Items tr:nth-child(even){' \
                        'background-color: darkgrey;} #Items tr:hover
{background-color: darkgrey;}#Items th {' \
                        'padding-top: 12px;padding-bottom: 12px;text-align:
left;text-align:center;color: ' \
                        'white;}</style></head><body><hr><center><h1>Sort By
Price</h1><table id="Items"><hr><tr ' \
                        'style="background-color: royalblue;"><th>The
Item</th><th>The Price</th></tr> '


                for OurItems in items:
# To fill the table with items.

                 ST += '<td>' + OurItems[0] + '</td><td>' + str(OurItems[1]) +
'$</td></tr>'
                ST += "</table></center></body></html>"
# End of html code.

                printedfile = 'SortItems.html'
# Set the requested file name is SortItems.html.
                Sortfile = open("SortItems.html","w")
# Create SortItems.html to write the html code after added sorrted item.
                Sortfile.write(ST)
                Sortfile.close()

        requestFile = open(printedfile,'rb')  # Open and read the requested
file in byte format.
        ST = requestFile.read()
        requestFile.close()

        header = 'Content-Type: ' + str(requestedType) + '\r\n\r\n'

    except Exception as e:

# If the request is wrong or the file doesn’t exist the server should return a
simple HTML webpage with our

# names and IDs and IP and port number of the client
        header = 'HTTP/1.1 404 Not Found\n\n'
        ST = ('<!DOCTYPE html><head><title>Error</title><style
```

```
type="text/css">h1 {text-align: center;}li {font-weight: '
                    'bold;}</style></head><head><style type="text/css">p
{text-align: '
                    'center;}li</style></head><head><style> body {background-
image: url("back.jpg"); '
                    'background-repeat: no-repeat;background-attachment:
fixed; background-size: 100% '
                    '100%;}</style></head><hr><body><h1 style="color:red">The
file is not found</h1><br><hr><p '
                    'style="color:black"><b>Tariq Odeh 1190699</b></p><p
style="color:black"><b>Qays Safa '
                    '1190880</b></p><p style="color:black"><b>Mahmoud Samara
1191602</b></p><hr><p><b>Client '
                    'IP:'+str(address[0])+'</b></p><p><b>Client
PORT:'+str(address[1])+'</b></p><hr> '
               '</body></html>').encode('utf-8')

    connectionSocket.send(f"\r\n".encode())
    connectionSocket.send(ST)
# Send the final response with all parts of header.
    connectionSocket.close()
# To closes a connectionSocket socket.
    print(sentence)
#Print the HTTP request on the terminal window.
```

3.8. HTML Code

```
<!DOCTYPE html>
<html>

<head>
  <title>ENCS3320-Simple Webserver</title>
  <link rel="stylesheet" href="style.css" type="text/css">
</head>

<head>
  <style>
  body {
    background-image: url('back.jpg');
    background-repeat: no-repeat;
    background-attachment: fixed;
    background-size: 100% 100%;
  }
  </style>
</head>

<body>
  <div class="header">
    <h1>
    <ins> Welcome to our course <span style="color: #0070C0">Computer Networks </ins>
    </span>
    </h1>
  </div>

  <img class ="b-img" src="birzeit.png" alt="Birzeit" >
  <br><br>
```

```
<div class="informationBox">
  <div>
    <h1>Tariq Odeh</h1>
    <h2>1190690</h2>
    <img align = "right" src="tariq.jpg" alt="Tariq Odeh">

    <div align = "left">
      <h1 style="color: #0070C0" >Projects </h1>
      <ul>
        <li>An 8-bit Comparator for signed 2s complement representation numbers.</li>
        <li>An educational application that uses augmented reality technology.</li>
        <li>System to manage patients data in a hospital.</li>
      </ul>

      <h1 style="color: #0070C0" >Skills</h1>
      <ul>
        <li>quick mathematical and physical analysis.</li>
        <li>scientific research.</li>
        <li>Planing.</li>
      </ul>

      <h1 style="color: #0070C0" >Hobbies</h1>
      <ul>
        <li>Reading.</li>
        <li>Football.</li>
        <li>Squash.</li>
      </ul>
    </div>
  </div>
</div>
```

```
 <br><br><br><br>


<div class="informationBox">
    <div>
        <h1>Qays Safa</h1>
        <h2>1190880</h2>
        <img align = "right" src="qays.jpg" alt="Qays Safa">

        <div align = "left">
            <h1 style="color: #0070C0" >Projects </h1>
            <ul>
                <li>A Java program for managing patient information in a hospital.</li>
                <li>Establishing a company specializing in health food products.</li>
                <li>Making a simple calculator in 8086 program in Orga course.</li>
            </ul>

            <h1 style="color: #0070C0" >Skills</h1>
            <ul>
                <li>Marketing.</li>
                <li>Teamwork.</li>
                <li>Problem Solving.</li>
            </ul>

            <h1 style="color: #0070C0" >Hobbies</h1>
            <ul>
                <li>Programming.</li>
                <li>Volleyball</li>
                <li>cycling.</li>
            </ul>
        </div>
```

```
    </div>
  </div>

  <br><br><br><br>



  <div class="informationBox">
    <div>
      <h1>Mahmoud Samara</h1>
      <h2>1191602</h2>
      <img align = "right" src="mahmoud.png" alt="Mahmoud Samara">

      <div align = "left">
        <h1 style="color: #0070C0" >Projects </h1>
        <ul>
          <li>Making a simple calculator in 8086 program in Orga course.</li>
          <li>A mathematical application that find the perfect number from group of
numbers.</li>
          <li>System to choose the shortest route from place to other using dijkstra
program.</li>
        </ul>

        <h1 style="color: #0070C0" >Skills</h1>
        <ul>
          <li>Team work.</li>
          <li>Java programming.</li>
          <li>Good speaker.</li>
        </ul>

        <h1 style="color: #0070C0" >Hobbies</h1>
        <ul>
```

```
            <li>Football.</li>
            <li>Swimming.</li>
            <li>Programming.</li>
          </ul>
        </div>
    </div>
  </div>

  <br><br><br><br>

  <div align = "center">

    <h3 style="color: #0070C0" >To go to the online HTML file press the following button </h3>
    <form action="https://www.w3schools.com/tags/att_img_src.asp" target="_blank"
method="post">
      <input type="submit" value="Online HTML file">
    </form>

    <h3 style="color: #0070C0" >To go to the local HTML file press the following button </h3>
    <form action="main.html" target="_blank" method="post">
      <input type="submit" value="Local HTML file">
    </form>
  </div>


</body>

</html>
```

3.9. CSS Code

```
.header}
    text-align: center;
    height: 85px;
    width: 100%;
    padding: 10px 0 0 20px;
    border: 0px;
    border-radius: 0px;
{



h1}
   color: black;
{



li}
   font-size: 120%;
{



.informationBox}
    text-align: center ;
    height: 600px;
    width: 100%;
    border: 5px solid #0070C0;
    border-radius: 10px;
    display: inline-block   ;
{
```

```
.image}
    align : right;
    height: 100%;
{




img}
    height: 400px;
    position: relative;
    top: 10px;
    border-radius: 20px;
    align : right;
{



.b-img}
    width: 50;
    height: 50;
    display: block;
    margin-left: auto;
    margin-right: auto  ;
}
```

4. References

[1] CSS tutorial. (2019, April 8). Retrieved December 9, 2021, from https://www.w3schools.com/css/.

[2] HTML tutorial. (2020, May 7). Retrieved December 9, 2021, from https://www.w3schools.com/html/.

[3] *Python get current time*. Programiz. (2019, May 4). Retrieved December 9, 2021, from https://www.programiz.com/python-programming/datetime/current-time.

[4] Team, P. (2019, May 7). *Send get request python socket - pretag*. Pretag development team. Retrieved December 9, 2021, from https://pretagteam.com/question/send-get-request-python-socket?fbclid=IwAR1P5xcT_GNiONtJiv2ak2SEeKMn7t1hkibXK0R-PfMtU59m__wbKlIR8lY.