Faculty of Engineering and Technology
Electrical and Computer Engineering Department

Computer Network

ENCS3320

Project 1 Report

Prepared by:

Tala Abahra 1201002

Maha Mali 1200746

Section: 1

Instructor: Dr. Abdalkarim Awad

Date: 5/1/2023

Table of Contents

Table of Figure

1. Part 1

1.1 What Is Ping, Tracert, Nslookup?

Ping: This command sends data over the network to another party, the other party receives the data, it returns back to the sending party, the command is frequently used to check network errors and identify their problems.

Tracert: This command is used to display several details about the path a packet takes from your computer or device to the destination you specify.

Nslookup: This command is used to obtain information about Internet servers. Also, it is a Tool to check if DNS is working.

1.2 Run Some Commands

1.2.1 Ping A Device in The Same Network

As we can see in Figure1, this is the result of executing the command Ping. In the first line it is stated that the size of the sent packet is 32 bytes. Shows that four data packets have been sent, and the result of each transmission is one line. Also, each line of the result started with the reply coming from the IP number 192.168.1. 103.In addition to that byte=32 The size of the packet sent to the other party in this attempt. And the time=xxx ms represents the time spent by the data packet in the round trip. Keep in mind that the distinction of time is in milliseconds. TTL mean time to live, which refers to the amount of time a packet spends on the network before being destroyed.

The ping command is particularly useful in testing since it displays the number of packets transmitted and received as well as checks for any lost packets. It also indicates the time it takes to send and receive data, which measures the connection's speed. when connecting the laptop to our local network with an IP address of: 192.168.1.103 and sending 4 packets, where all packets have the same TTL (time to live), all packets are received with different delays and the average is 69ms.

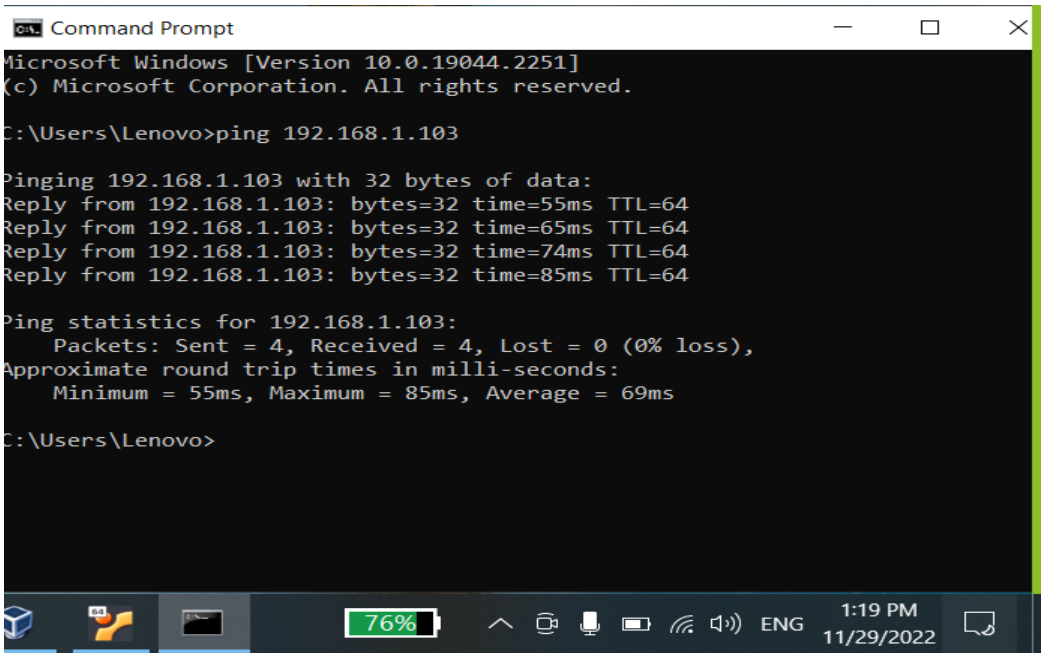


Figure 1: Ping Device in Same Network

1.2.2 Ping www.yale.edu

That response shows the URL you're pinging, the IP address associated with that URL, and the size of the packets being sent on the first line. The next four lines show the replies from each individual packet, including the time (in milliseconds) it took for the response and the time-to-live (TTL) of the packet, which is the amount of time that must pass before the packet is discarded.

At the bottom, you'll see a summary that shows how many packets were sent and received, as well as the minimum, maximum, and average response time. As shown in figure 2.

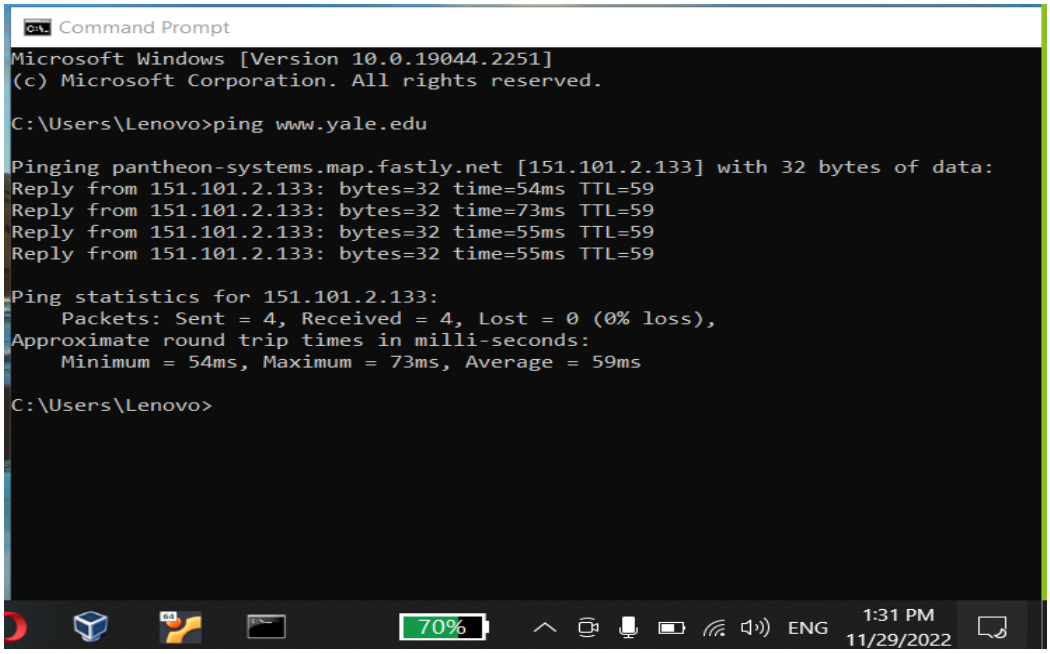


Figure 2: Ping www.yale.edu

1.2.3 Tracert www.yale.edu

The Tracert diagnostic utility determines the route to a destination by sending Internet Control Message Protocol (ICMP) echo packets to the destination, and every router involved in transferring the data gets these packets. The ICMP packets provide information about whether the routers used in the transmission are able to effectively transfer the data. so here, 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. We can see from the figure 3 that there are 3 measurements from each router. When we go down in the lines, we will see the 3 measurements increases because the next router goes further. When the request timed out and the measurements are *, then the packet is prevented from reaching the destination because there is a problem at that location or the route is incorrect.

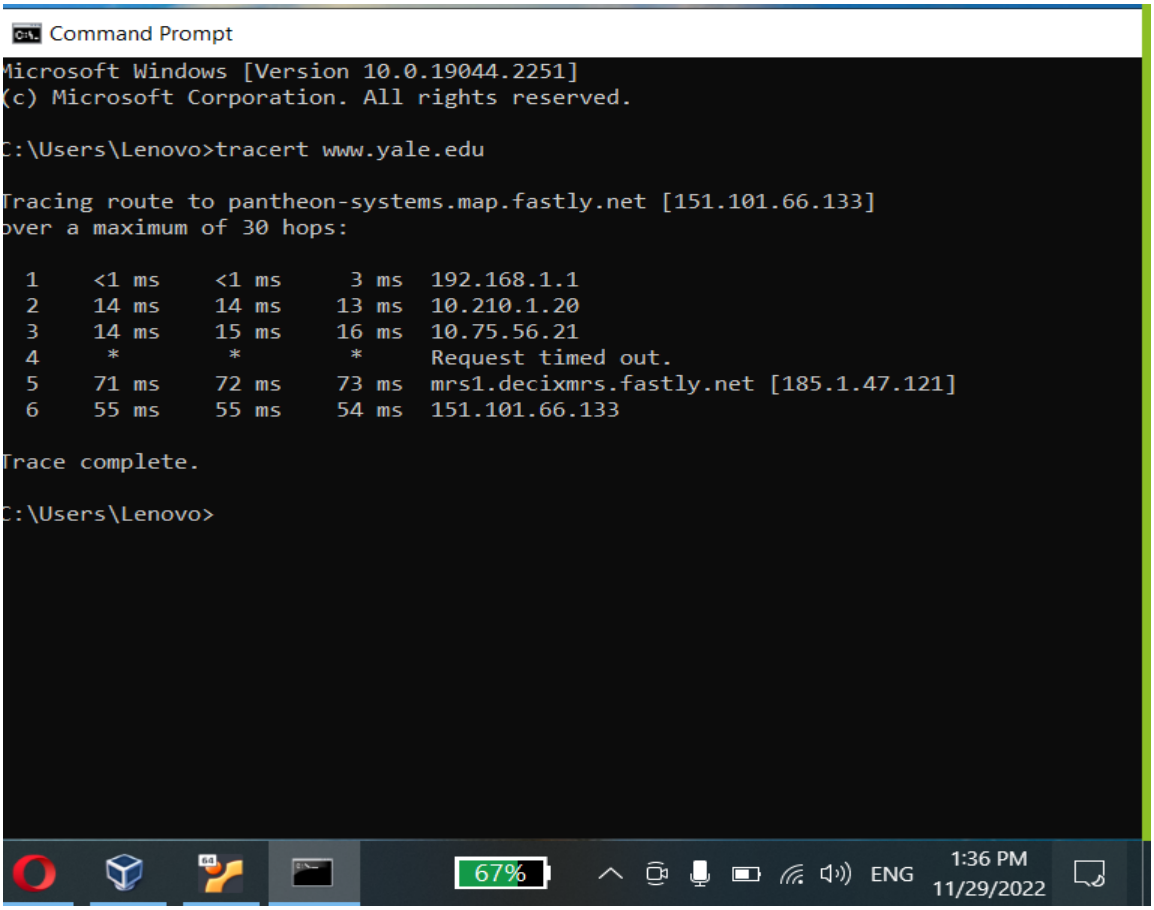


Figure 3: Tracert www.yale.edu

1.2.3 Nslookup www.yale.edu

Nslookup command allows you to change the nameserver you query, to ensure you query a nameserver from which you are guaranteed to get an accurate result.

If you query the nameserver listed against the domain name you will receive an authoritative answer, because the nameserver has authority over the DNS for the domain name can see that it prints the IP address corresponding to the host which is my laptop's IP address, and prints the name and addresses of the server which is the host that we sent a probe.

So nslookup sends DNS query to the specified DNS server, then receives a DNS reply from that same DNS server, and display results.

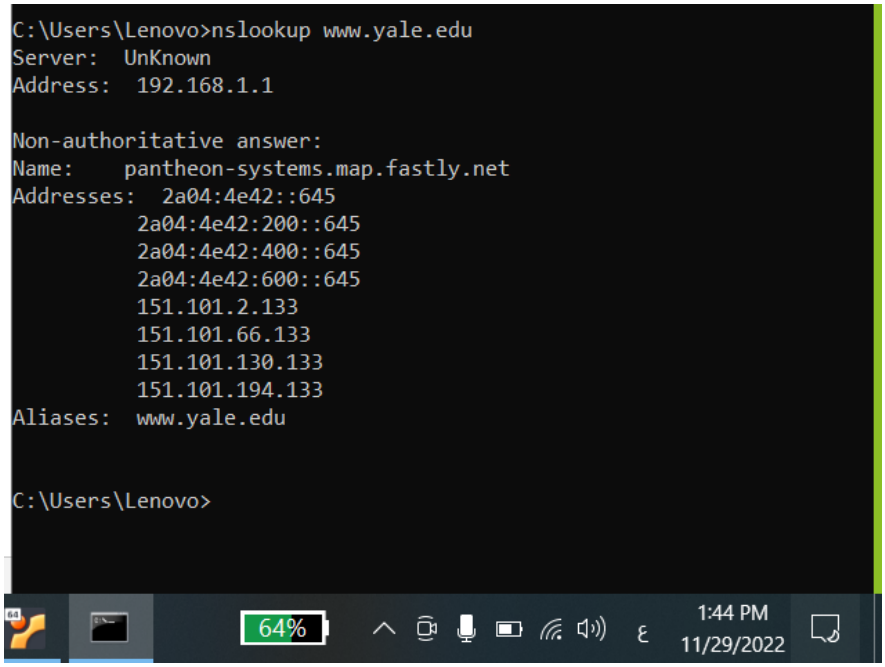


Figure 4: Tracert www.yale.edu

2. Part 2

2.1 Required

Implement the following server and client application both for TCP and for UDP: A client continuously sends the numbers from 0 to 1000,000 to a server listening on port 5566. The server counts the received messages.

2.2 Run the programs using TCP client and server

TCP is reliable because the protocol ensures that all data is fully transmitted and can be assembled by the receiver in the correct order.

2.2.1 Run on Same Computer

TCP socket server and client applications can be executed on the same Computer. TCP Client and server programs are two different and independent application. Just tcp client has to connect to correct address of the tcp server .TCP client just need to know the tcp server program address that is IP address and port to which it wants to connect.

Total time required to send the packets and to receive the packets is :555 sec which equal 9 min , we can see from next figure that counter equal the data , this mean no data loss and all object send from client to server .

2.2.1.1 Client code TCP:

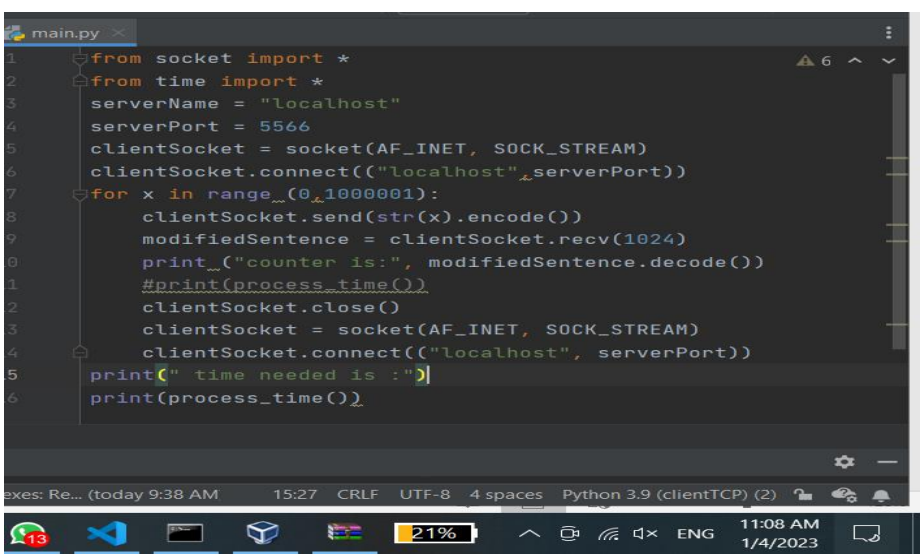


Figure 5 : Client code TCP

2.2.1.2 Client Result: Client send the number from 1 to 1000000

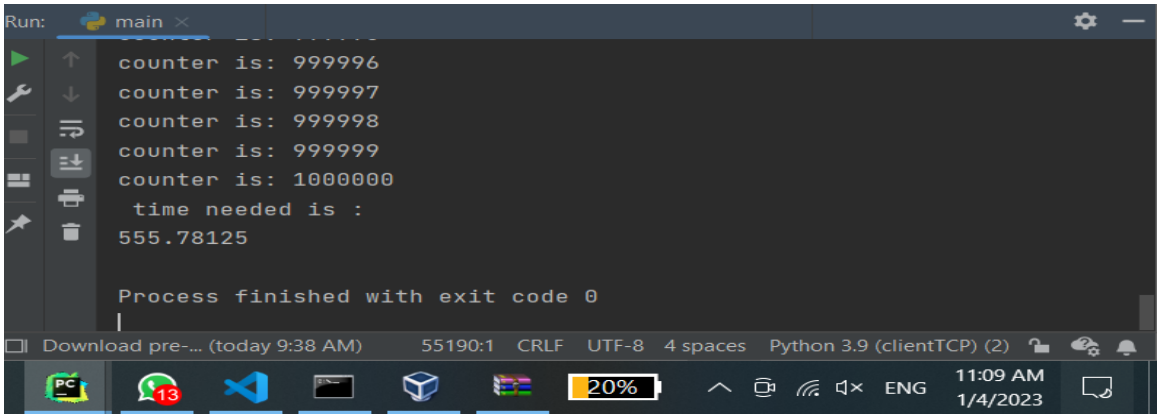


Figure 6: Client Result

2.2.1.3 Server Code in TCP:

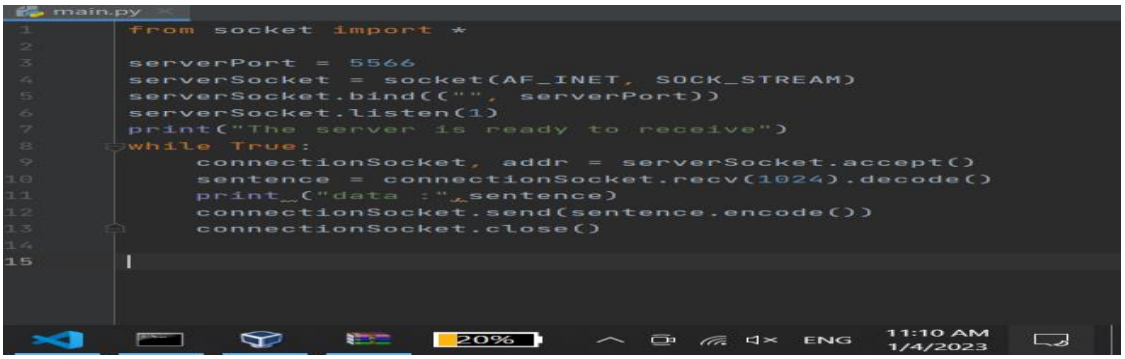


Figure 7: Server Code in TCP

2.2.1.4 Server Result in TCP

When run the program in same computer: server counts the received messages from client

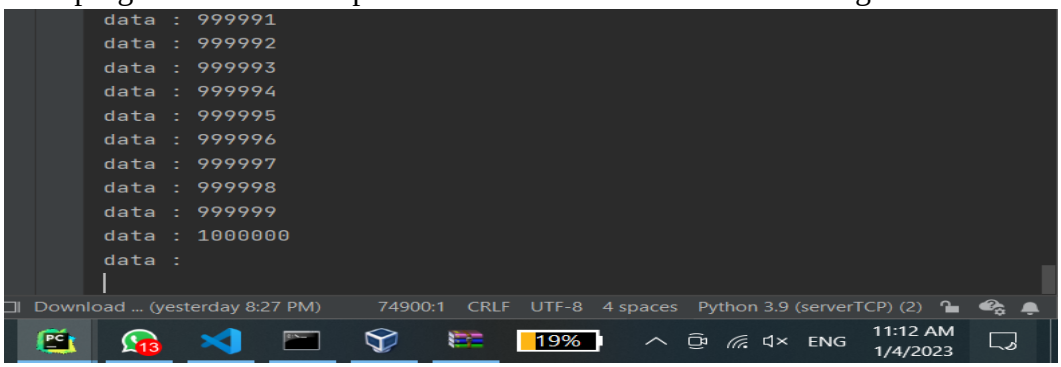


Figure 8: Server Result

2.2.2 Run on 2 different computers connected by a cable directly or through a switch

Total time required to send the packets and to receive the packets is: 286 sec, and the figure 9 show when we connected two computer using Ethernet cable.

Figure 9: connected two computer using Ethernet cable

2.2.2.1 Client Result in Ethernet Cable:

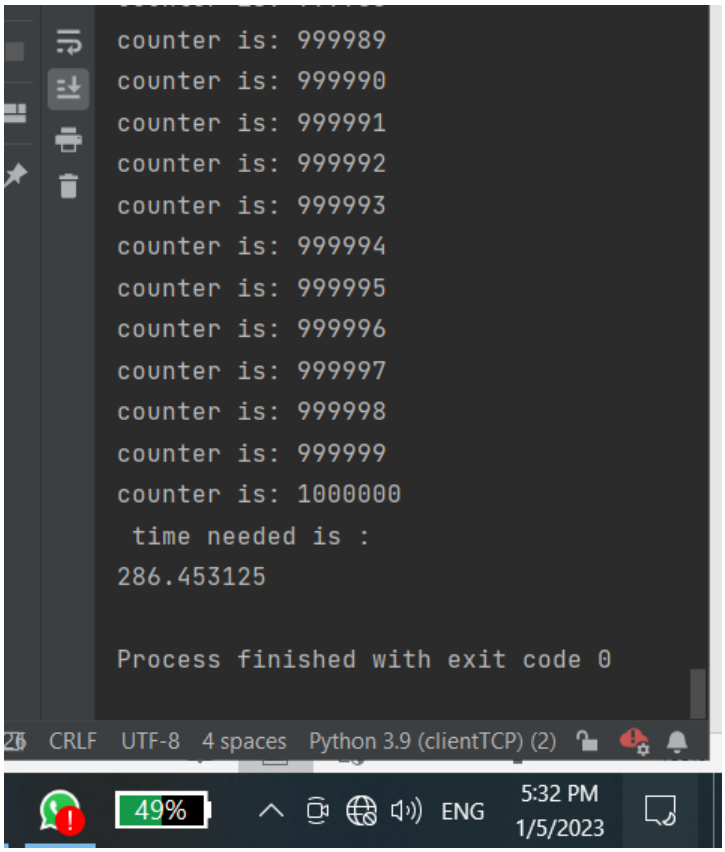


Figure 10: Client Result in Ethernet Cable

2.2.2.2 Server Result in Ethernet Cable:

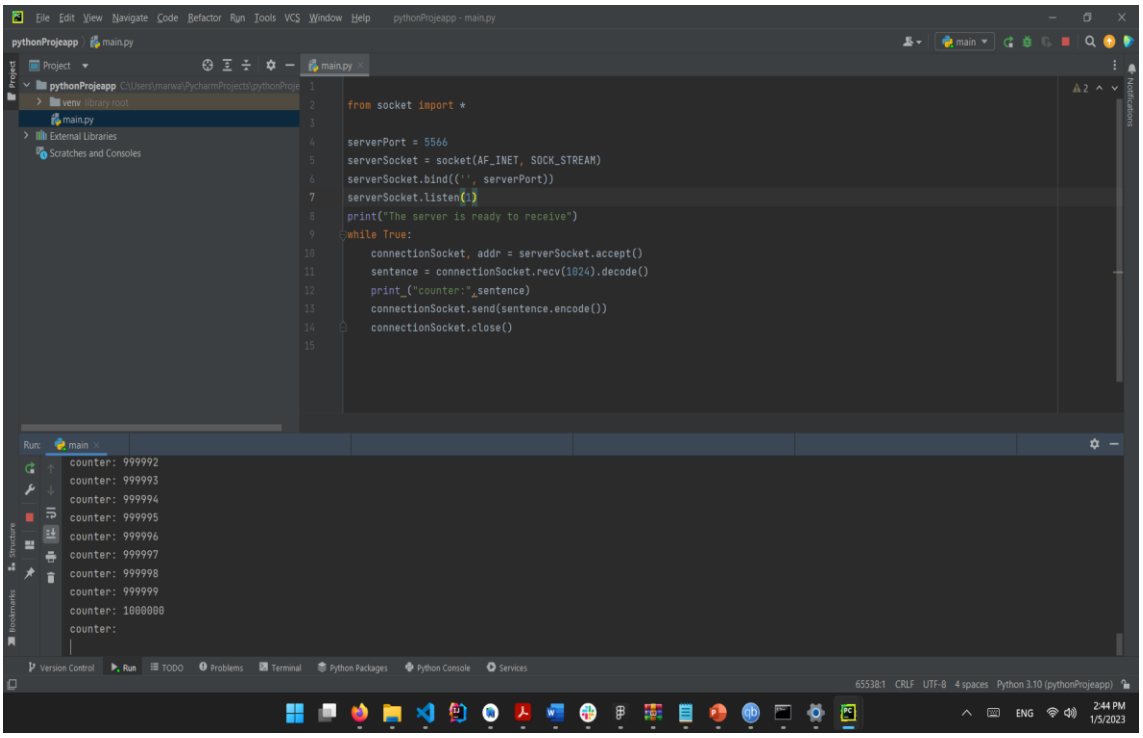


Figure 11: Server Result in Ethernet Cable

2.2.3 Run on 2 different computers connected Through WiFi

It's just TCP. The transport layer WiFI is irrelevant.
Each side needs a unique IP address and (for simplicity) they should be on the same subnet.
It is sufficient if the sever port is known to the client, the local client port can be ephemeral.
The server can be configured to just reply to the source port of the incoming connection.
ServerName : ”192.168.0.105”

```
from socket import *
from time import *
serverName = "192.168.0.105"
serverPort = 5566
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName,serverPort))
for x in range (0,1000001):
    clientSocket.send(str(x).encode())
    modifiedSentence = clientSocket.recv(1024)
    print ("counter is:", modifiedSentence.decode())
    clientSocket.close()
    clientSocket = socket(AF_INET, SOCK_STREAM)
    clientSocket.connect((serverName, serverPort))
print(" time needed is :")
print(process_time())
```

2.2.3.1 Client Result in WiFi:

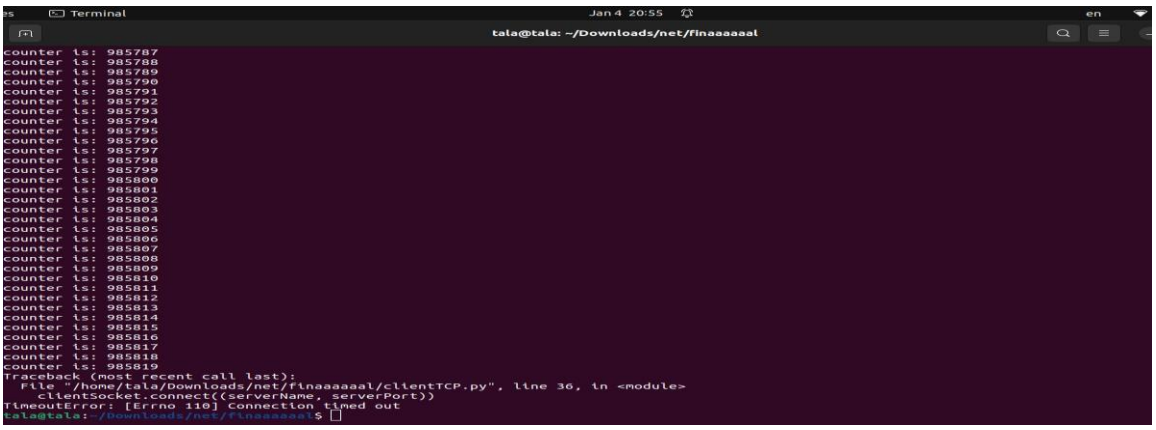


Figure 12: Client Result in WiFi

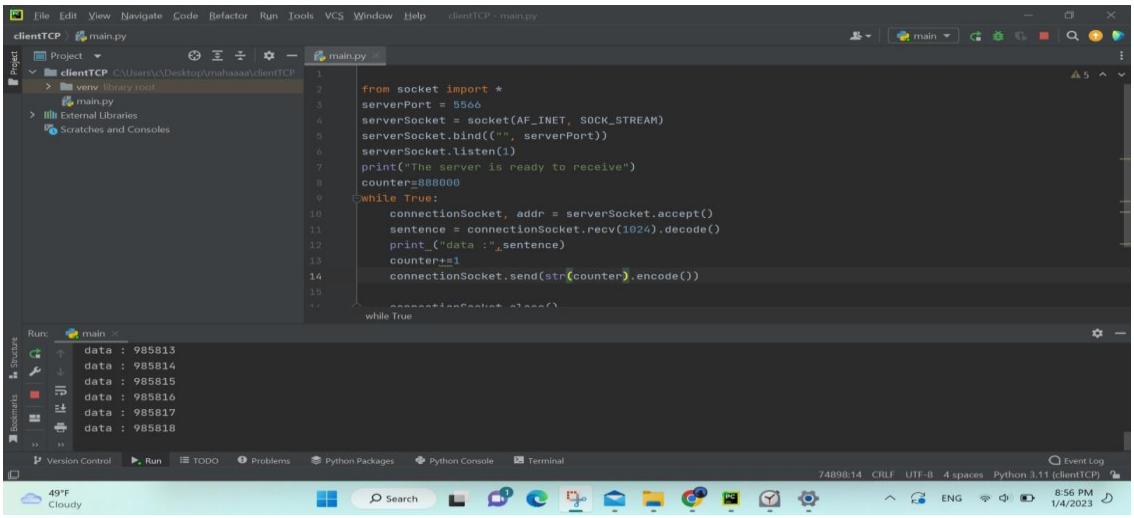


Figure 13: Server Result in WiFi

We notice that he did not complete the transmission process, because the time was relatively long, but at the same time the data was correct and sent correctly.

One of the problems of the TCP protocol in transferring data is: “Silly window syndrome TCP”: It is a problem that arises in controlling the flow of “TCP”, and in this the sender window shrinks to a very low value because the data that is sent on each trip Even smaller than the TCP header, the TCP protocol becomes very inefficient.

2.3 Run the Programs Using UDP Client and Server

UDP (User Datagram Protocol) is an unconnected transport layer protocol in the OSI (Open System Interconnection) reference model, providing a transaction-oriented simple unreliable information transfer service. It is called "unreliable" because there is no handshake or verification of the data transfer.

2.3.1 Run on same computer

Total time required to send the packets and to receive the packets is: 48sec, Thus, the time is much less than the time taken by TCP, but there is a large loss of transmitted data.

2.3.1.1 Client Code UDP:

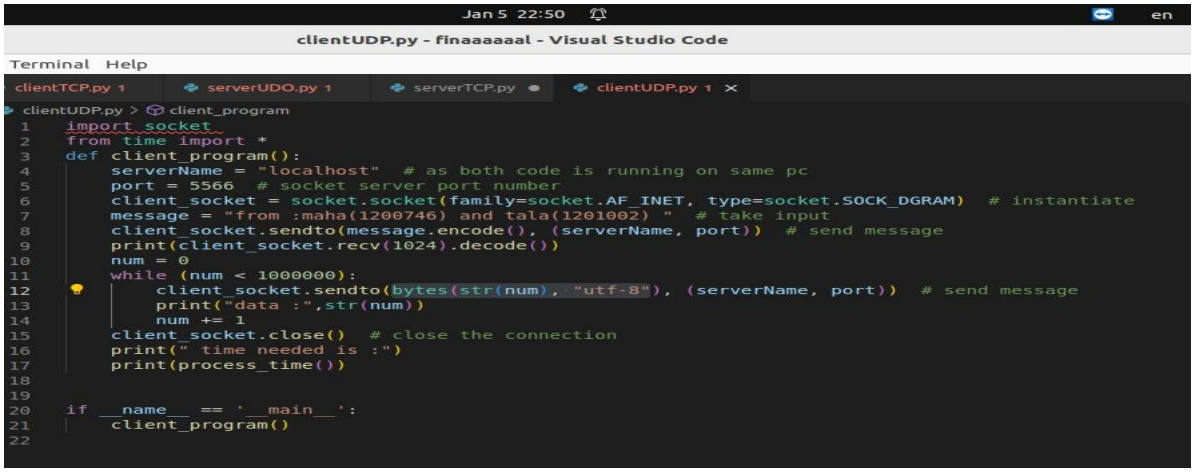


Figure 14: Client Code UDP

2.3.1.2 Client Result in UDP: Client send the number from 1 to 1000000

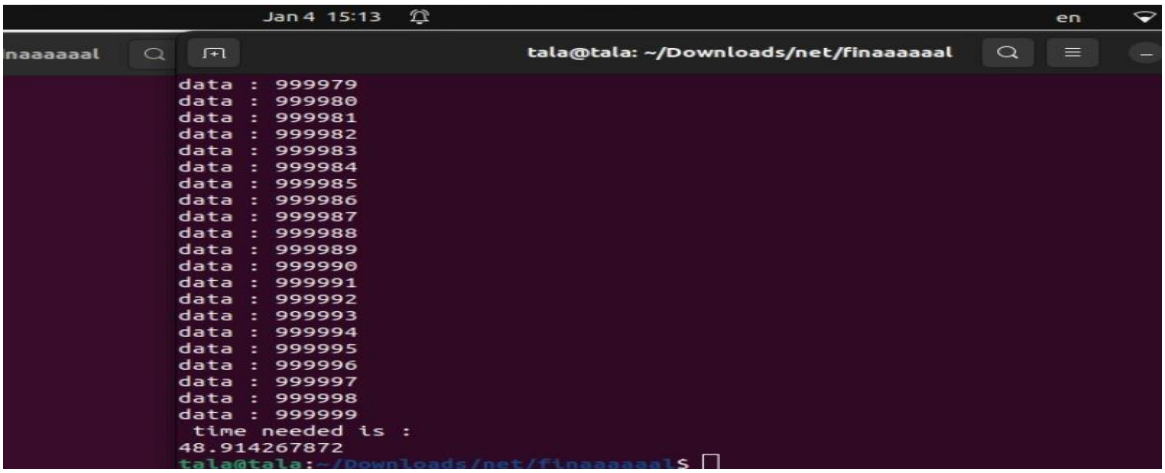


Figure 15: Client Result in UDP

2.3.1.3 Server Code in UDP:

```python
import socket

from time import *
def server_program():
    # get the hostname
    port = 7788  # initiate port no above 1024
    server_socket = socket.socket(family=socket.AF_INET, type=socket.SOCK_DGRAM)  # get instance
    # look closely. The bind() function takes tuple as argument
    server_socket.bind(("", port))  # bind host address and port together
    # configure how many client the server can listen simultaneously
    message, address = server_socket.recvfrom(2048)
    # accept new connection
    print("Connection from: " + str(address))
    print(str(message))
    server_socket.sendto(str("this connection is ok ").encode(), address)
    counter = 0
    while True:
        data, address = server_socket.recvfrom(2048)
        if not data:
            break
        print("couter : ", str(counter))
        counter += 1
    server_socket.close()
    print(" time needed is :")
    print(process_time())

if __name__ == '__main__':
    server_program()
```

Figure 16: Server Code in UDP

2.2.1.4 Server Result in UDP: server counts the received messages from client

```
couter :  997103
couter :  997104
couter :  997105
couter :  997106
couter :  997107
couter :  997108
couter :  997109
couter :  997110
couter :  997111
couter :  997112
couter :  997113
couter :  997114
couter :  997115
couter :  997116
couter :  997117
couter :  997118
couter :  997119
couter :  997120
couter :  997121
couter :  997122
couter :  997123
couter :  997124
couter :  997125
```

```
data : 999979
data : 999980
data : 999981
data : 999982
data : 999983
data : 999984
data : 999985
data : 999986
data : 999987
data : 999988
data : 999989
data : 999990
data : 999991
data : 999992
data : 999993
data : 999994
data : 999995
data : 999996
data : 999997
data : 999998
data : 999999
 time needed is :
48.914267872
tala@tala:~/Downloads/net/finaaaaaal$
```

Figure 17: Server Result in UDP

2.3.2 Run on 2 different computers connected by a cable directly or through a switch

2.3.2.1 Client Result in Ethernet Cable:

When transferring data through during ethernet cable, there is a loss of data at the client.

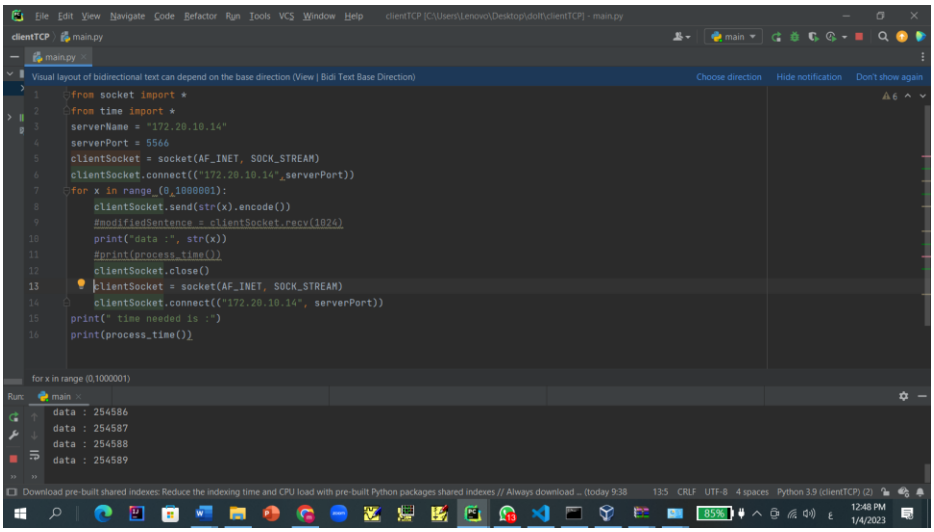


Figure 18: Client Result in Ethernet Cable

2.3.2.2 Server Result in Ethernet Cable:

Not all packages arrived at the server, because in UDP there is data loss.

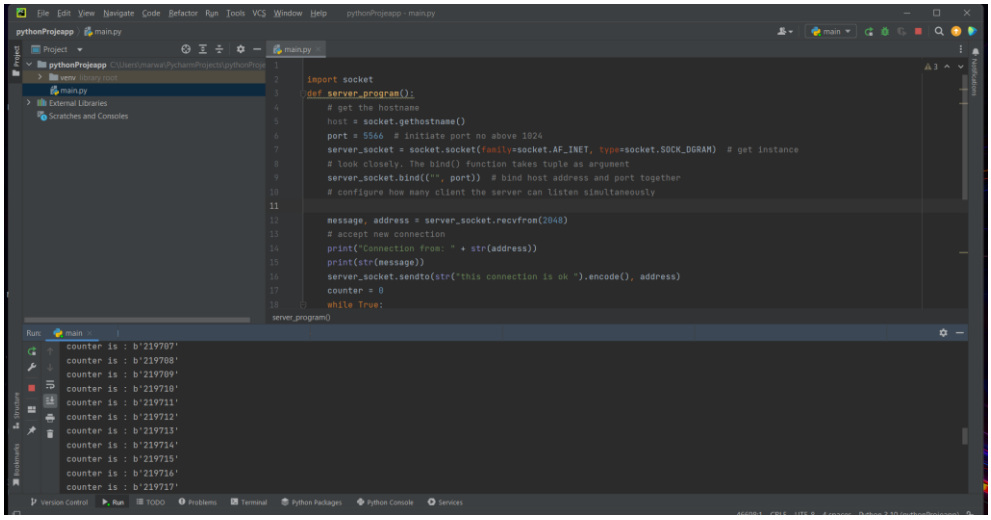


Figure 19: Server Result in Ethernet Cable

2.3.3 Run on 2 different computers connected through WiFi

Total time required to send the packets and to receive the packets is: 58 sec.

2.3.3.1 Client Result in WiFi

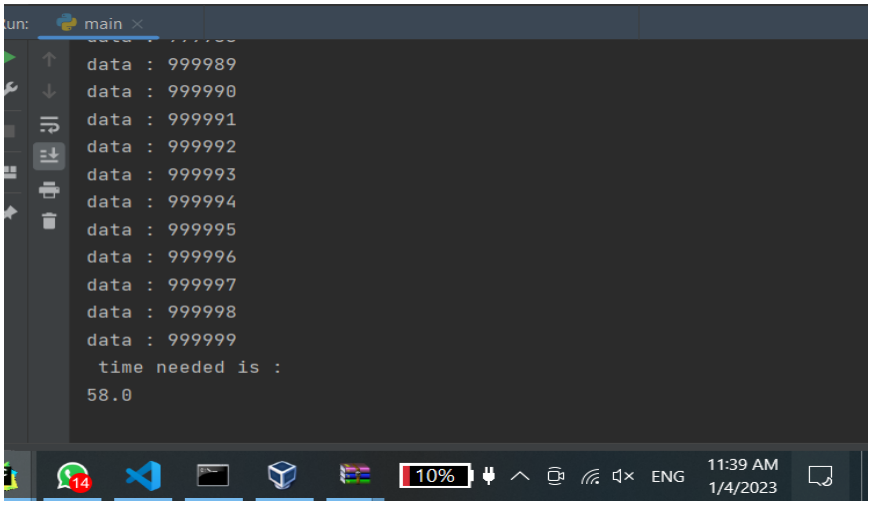


Figure 20: Client Result in WiFi

2.3.3.1 Server Result in WiFi:

There is data loss.

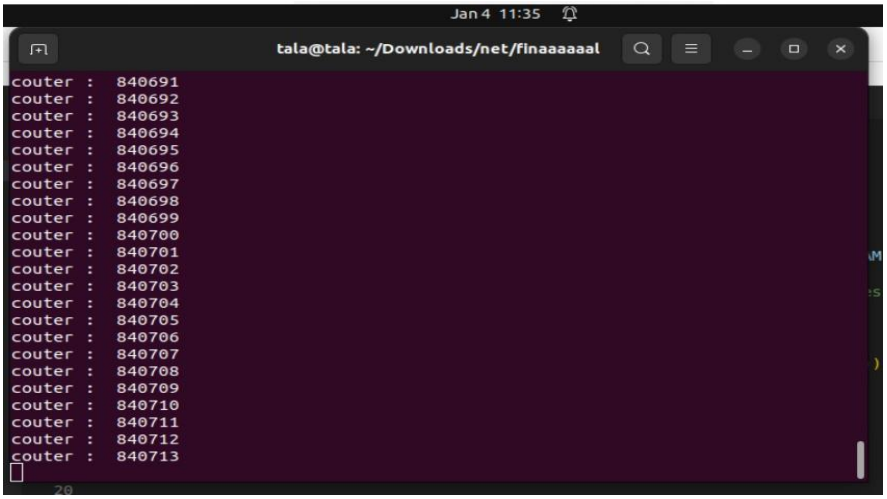


Figure 21:Server Result in WiF

3. Part 3

In this part, we used socket programming to create a complete web server on port 7788 using the Python programming language, as well as HTML, CSS.

First, we defined a server port which is 7788, then we created a socket instance and pass it two parameters, the first one is AF_INET which means to use IPv4 protocol, and the second one is SOCK_STREAM which means connection-oriented TCP protocol.As shown in figure 22.

After creating the socket, we associated it with its local address, allowing clients toconnect to the server using that address using bind(). Then we put the socket into listening mode to make it ready for the requests.

```python
from socket import *

import os.path
# reserve port 7788 on the computer
serverPort = 7788
# *The SOCK_STREAM: connection-oriented TCP protocol.
serverSocket = socket(AF_INET, SOCK_STREAM)
# Associates the socket with its local address, allowing clients to connect to the server using that a
serverSocket.bind(("", serverPort))
# Set the socket to listening mode, where 1 represents the number of incoming connections.
serverSocket.listen(1)
print("The Server is Ready!")
# Infinite loop until it interrupted or an error occurs
```

Figure 22: create the socket

The server will accept and complete the connection by using accept(). Then we will receive the http request from the server and decode it to store it in the "sentence" variable. We will get the request by splitting "sentence" and taking what is after GET /…
As shown in figure

```python
# Infinite loop until it interrupted or an error occurs
while True:
    # accept() is called by the server to accept or complete the connection.
    connection_Socket, address = serverSocket.accept()
    # Receive data from the server and decode it to obtain the string.
    sentence = connection_Socket.recv(1024).decode()
    print(address)
    print(sentence)
    ip = address[0]
    port = address[1]
    request = sentence.split()[1]
    request = request.lower()
    request = request.lstrip('/')
```

Figure 23: Accept and complete the connection

3.1 Request The Main Html File

If the request is / or /index.html or /main_en.html or /en then the server should send main_en.html file. If the request is / or main_ar html then the main html file will be sent to the client. To send the html file, the server will first open that file then read it. Then it will send the header which contains of the encoded response status. Finally, the server will send the encoded file that it read previously. As shown in figure24 .

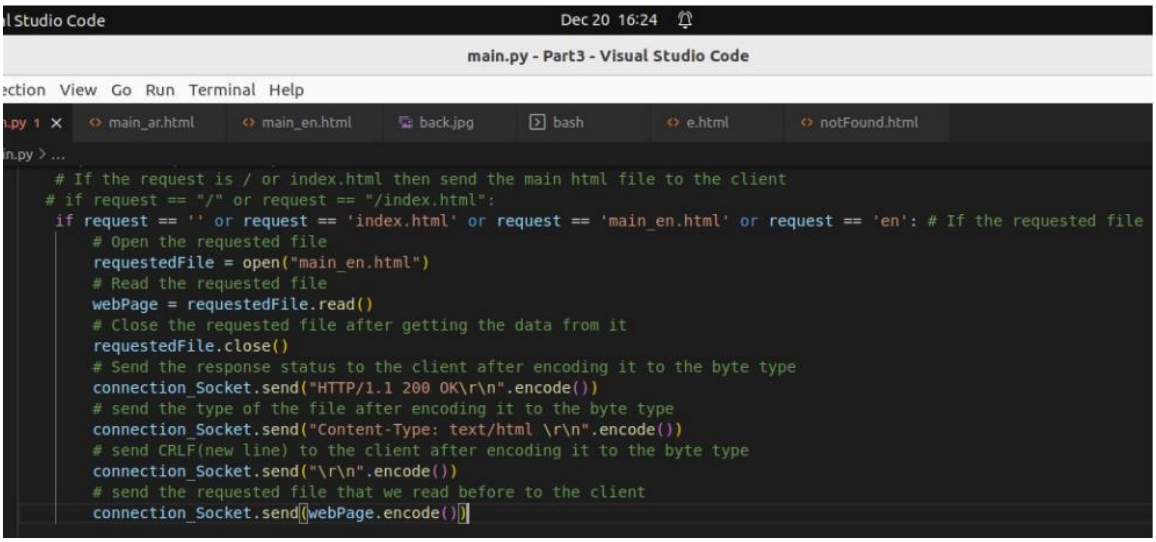


Figure 24: Request the main html file

The local HTML file :

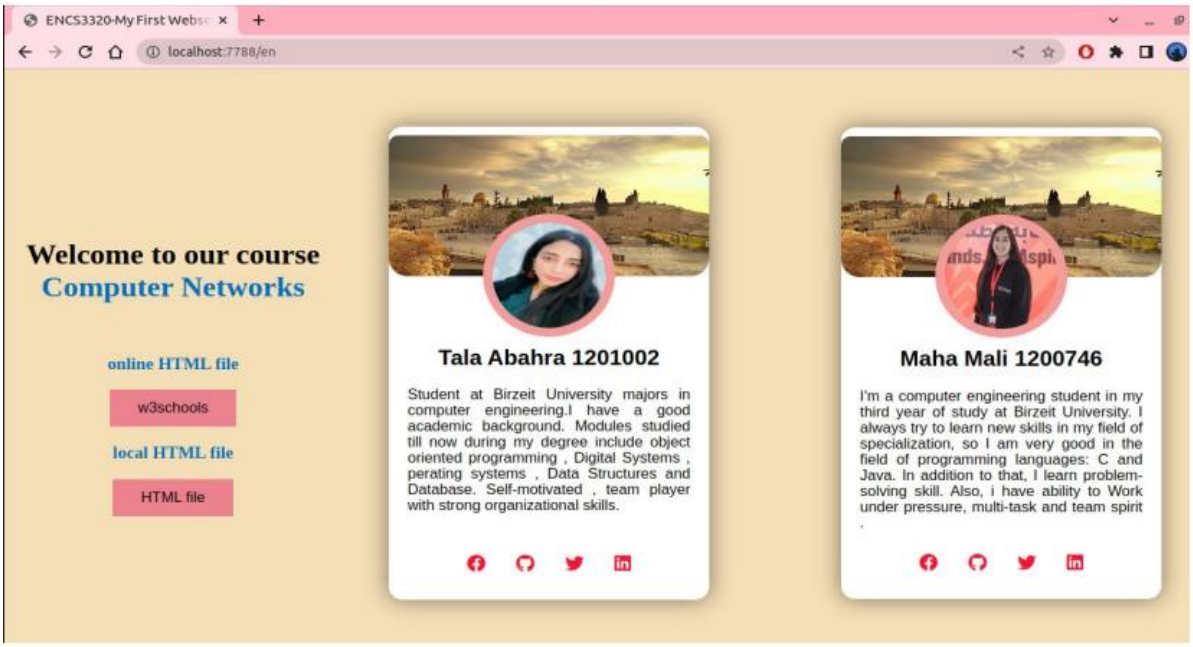


Figure 25:local html file

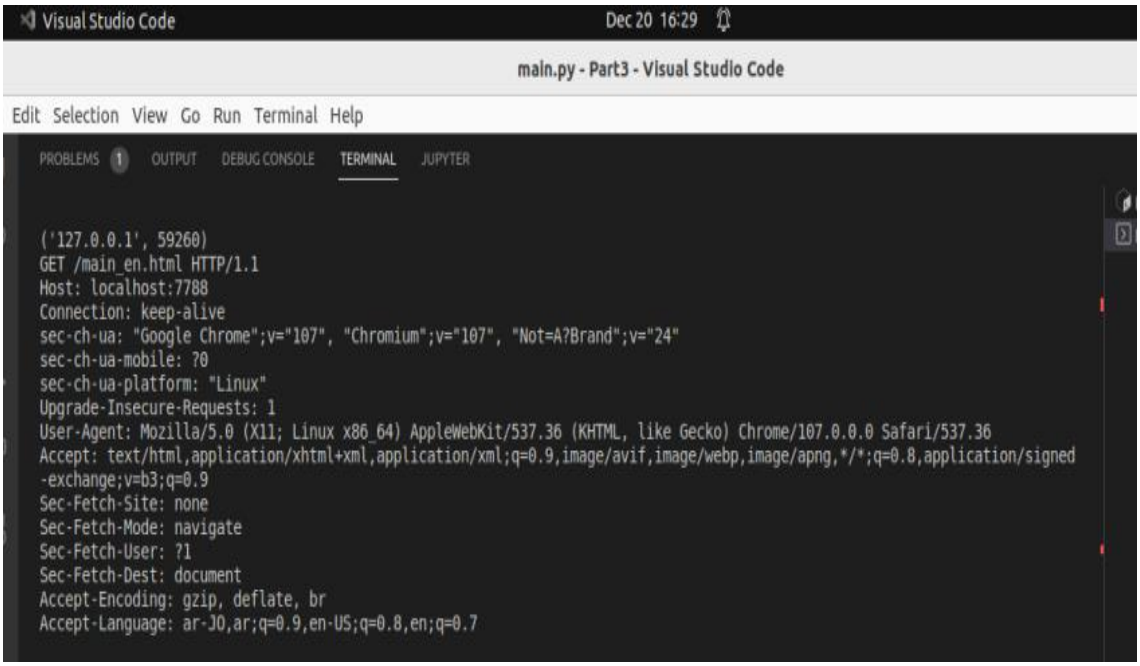


Figure 26: Response of local html file

3.2 Request Arabic version of main_en.html

If the request is ar or main_ar.html file, then the server will send the **main_ar.html** file by doing the same as the previous steps.

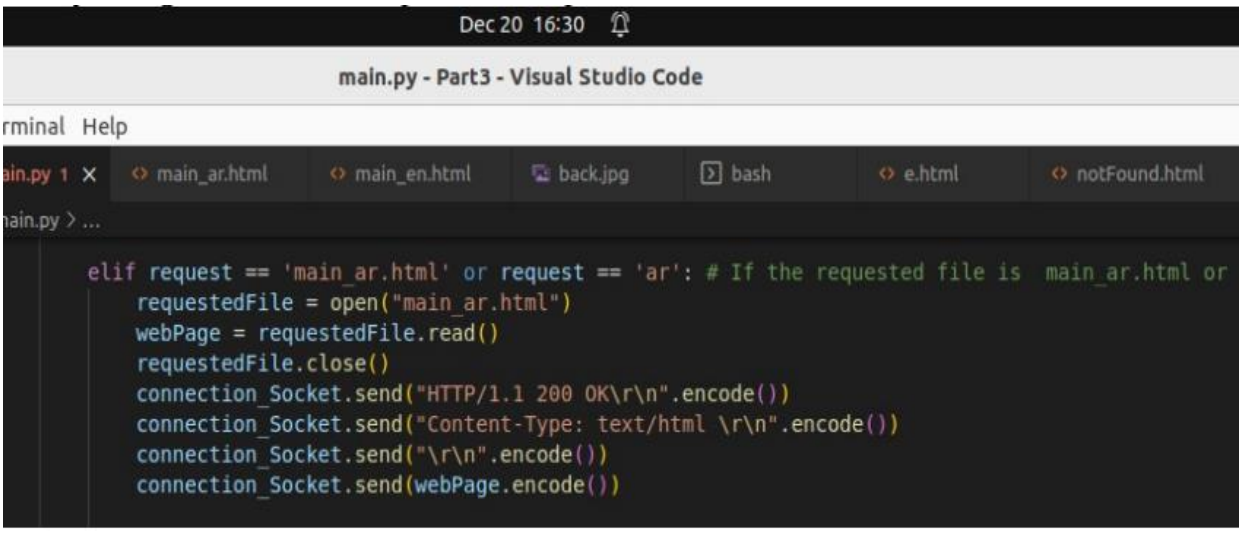


Figure 27: Request Arabic version of main_en.html

The main_ar.html :

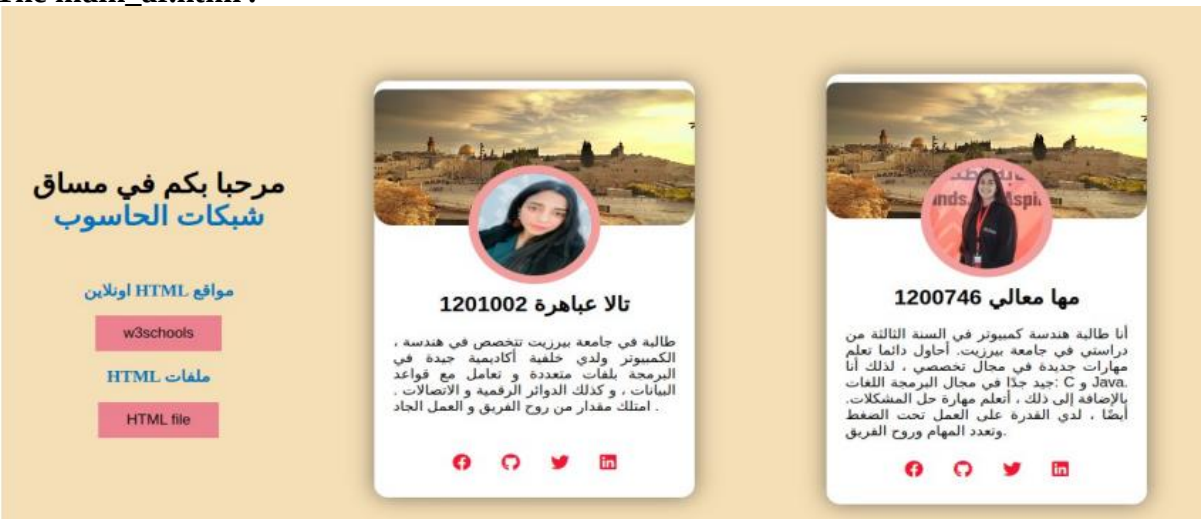


Figure 28: The main_ar.html

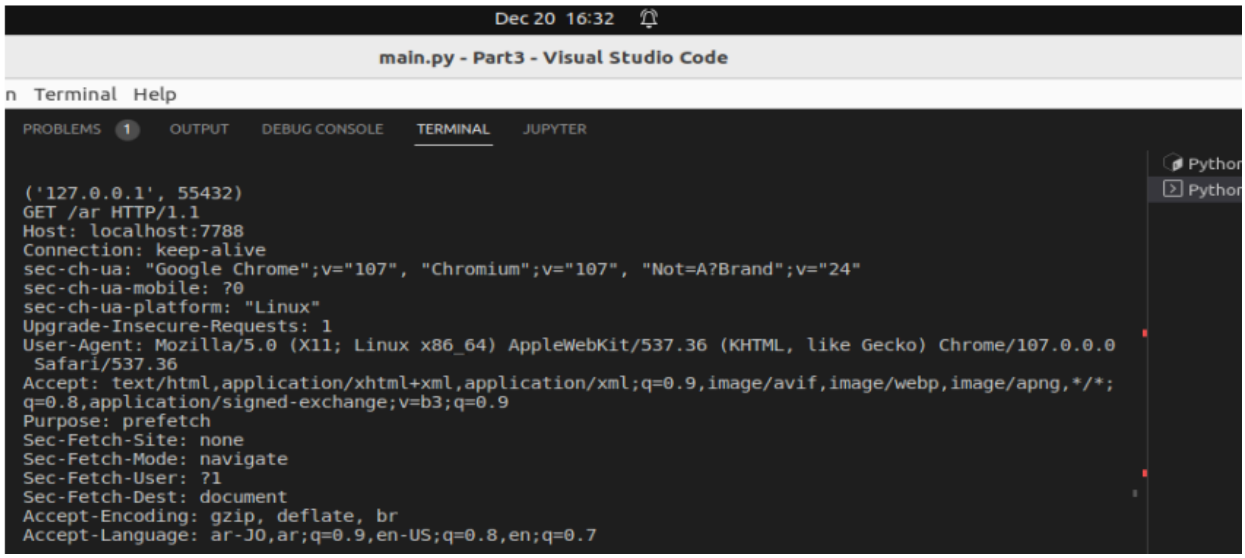


Figure 29: Response of main_ar.html

3.3 Request the html file

If the request is /html then the main html file will be sent to the client. To send the html file, the server will first open that file then read it. Then it will send the header which contains of the

encoded response status (which is OK), the encoded content type, and the encoded CRLF. Finally, the server will send the encoded file that it read previously.

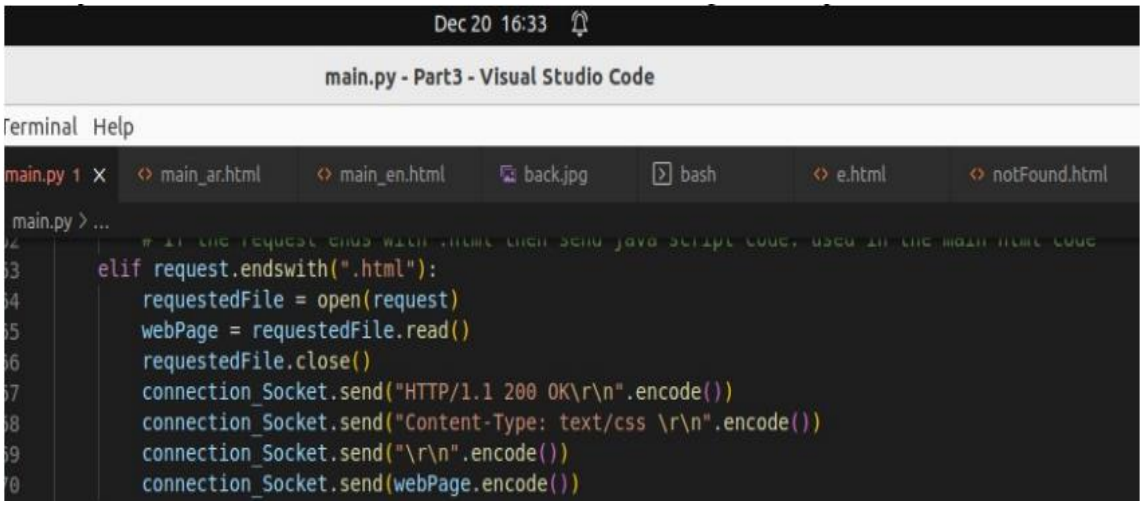
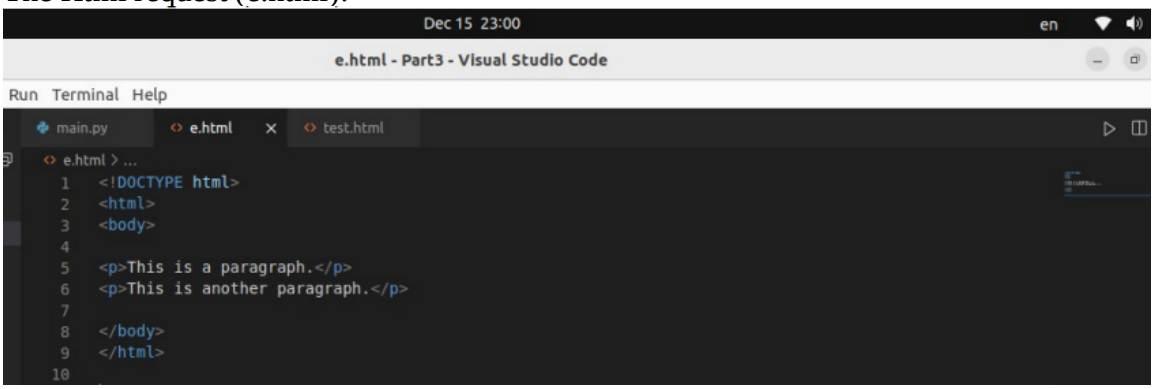


Figure 30: Request the html file

The Html request (e.html):



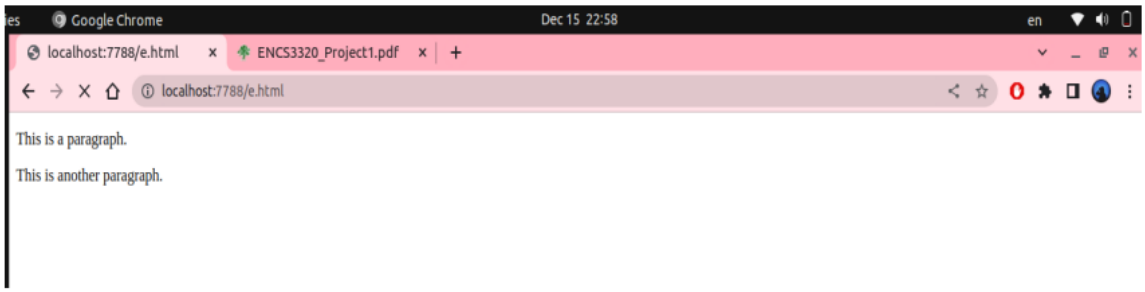


Figure 31: The Html request (e.html)

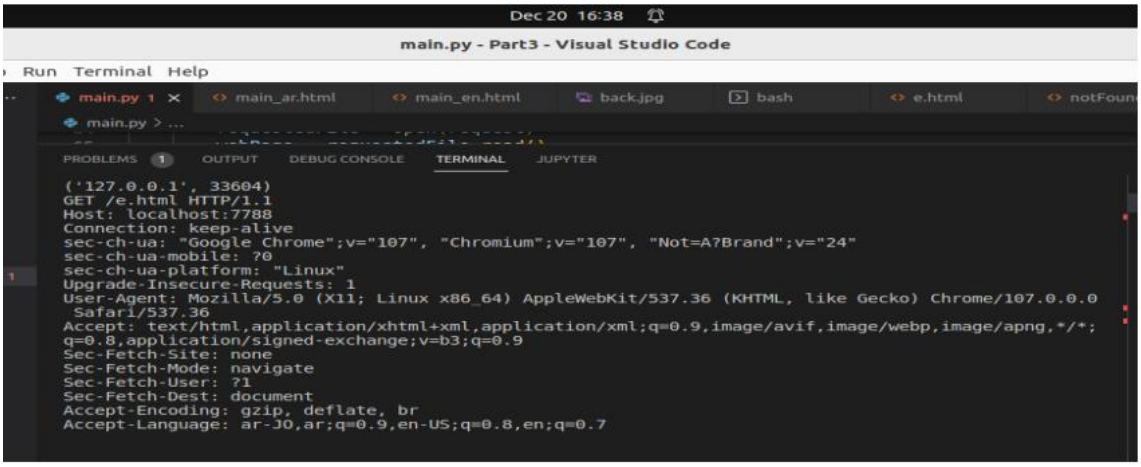


Figure 32: The Html Response (e.html)

3.4 Request the CSS file

If the request is a CSS file, then the server will send the “style.css” file by doing the same as the previous steps, but changing the type of content to text/css.

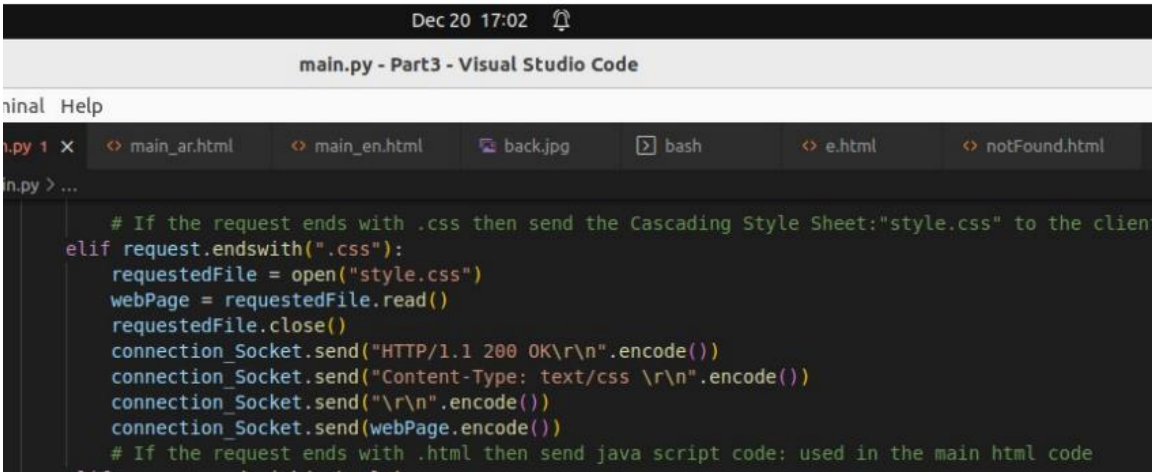
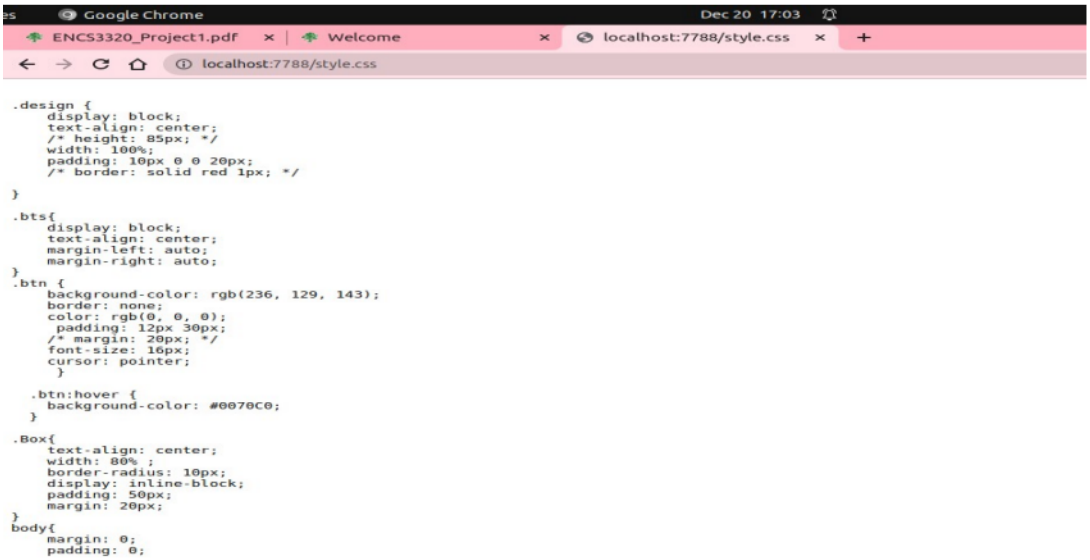


Figure 33: Request the CSS file



```
('127.0.0.1', 33606)
GET /style.css HTTP/1.1
Host: localhost:7788
Connection: keep-alive
sec-ch-ua: "Google Chrome";v="107", "Chromium";v="107", "Not=A?Brand";v="24"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Linux"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/107.0.0.0
 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;
q=0.8,application/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate, br
Accept-Language: ar-JO,ar;q=0.9,en-US;q=0.8,en;q=0.7
```

Figure 34: Response the CSS file

3.5 Request an image with jpg, jpeg, or png format

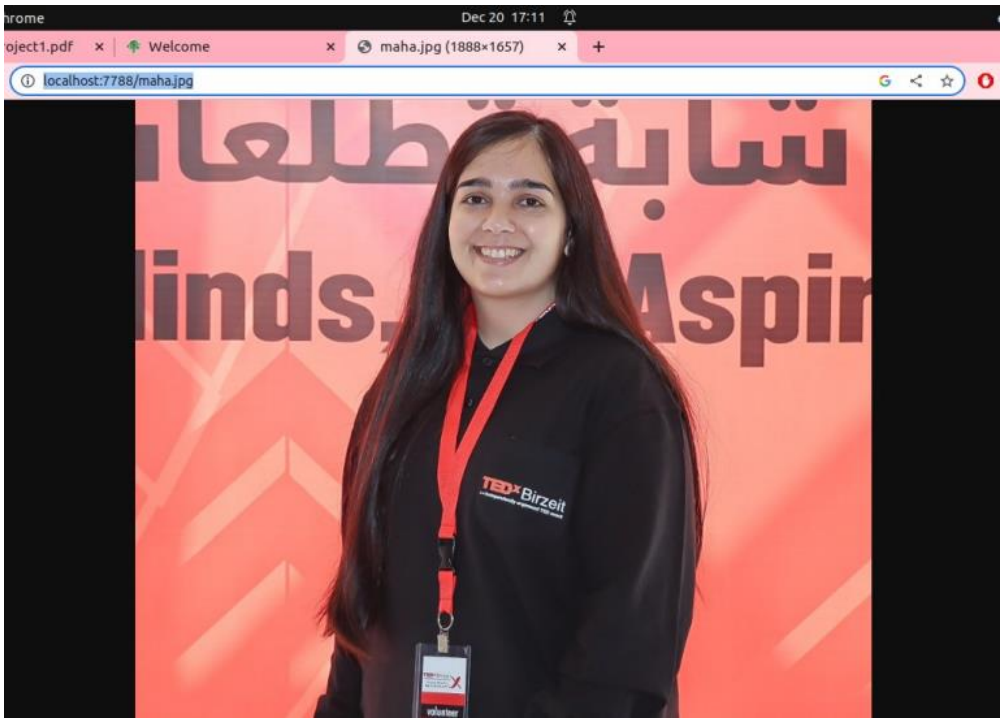
If the request is an image with jpg, jpeg, or png format, we will first see if that image is existed to send it. If not, to avoid any errors, we will send a specific image with the requested format.

```
Dec 20 17:08
main.py - Part3 - Visual Studio Code
Run  Terminal  Help
main.py 1 ×   main_ar.html   main_en.html   back.jpg   bash   e.html   notFound.html
main.py > ...
    elif request.endswith(".jpg"):
        requestedFile = open(request , "rb")
        webPage = requestedFile.read()
        requestedFile.close()
        connection_Socket.send("HTTP/1.1 200 OK\r\n".encode())
        connection_Socket.send("Content-Type: image/jpg \r\n".encode())
        connection_Socket.send("\r\n".encode())
        connection_Socket.send(webPage)
```

```
        # If the request ends with .jpeg then send image code to the client
    elif request.endswith(".jpeg"):
        requestedFile = open(request, "rb")
        webPage = requestedFile.read()
        requestedFile.close()
        connection_Socket.send("HTTP/1.1 200 OK\r\n".encode())
        connection_Socket.send("Content-Type: image/jpeg \r\n".encode())
        connection_Socket.send("\r\n".encode())
        connection_Socket.send(webPage)
```

Figure 35: Request an image with jpg, jpeg, or png format

Output:




```
('127.0.0.1', 55462)
GET /maha.jpg HTTP/1.1
Host: localhost:7788
Connection: keep-alive
sec-ch-ua: "Google Chrome";v="107", "Chromium";v="107", "Not=A?Brand";v="24"
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/107.0.0.0
 Safari/537.36
sec-ch-ua-platform: "Linux"
Accept: application/signed-exchange;v=b3;q=0.7,*/*;q=0.8
Purpose: prefetch
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: no-cors
Sec-Fetch-Dest: image
Referer: http://localhost:7788/ar
Accept-Encoding: gzip, deflate, br
Accept-Language: ar-JO,ar;q=0.9,en-US;q=0.8,en;q=0.7
```

Figure 36: ***Response an image with jpg, jpeg, or png format***

3.6 The status code 307 Temporary Redirect

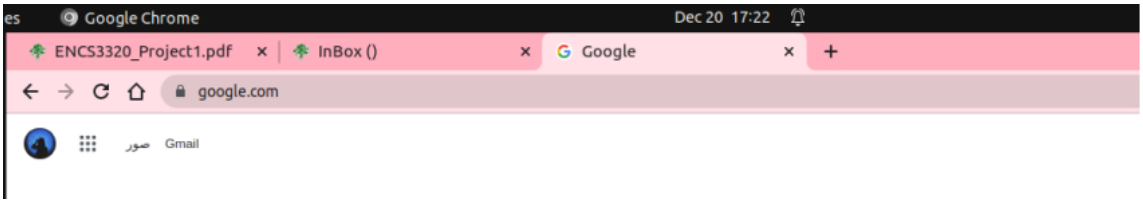
Code:

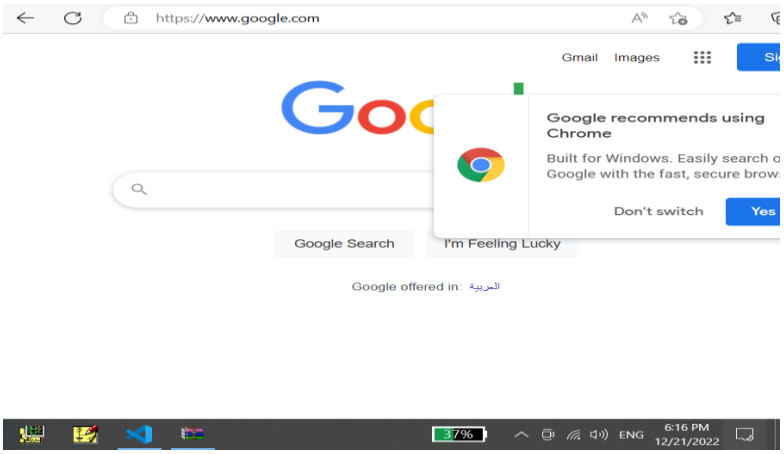
```
elif request == "go":
    connection_Socket.send("HTTP/1.1 307 Temporary Redirect \r\n".encode())
    connection_Socket.send("Content-Type: text/html \r\n".encode())
    connection_Socket.send("Location:https://www.google.com \r\n".encode())
    connection_Socket.send("\r\n".encode())
elif request == "so":
    connection_Socket.send("HTTP/1.1 307 Temporary Redirect \r\n".encode())
    connection_Socket.send("Content-Type: text/html \r\n".encode())
    connection_Socket.send("Location:https://www.stackoverflow.com \r\n".encode())
    connection_Socket.send("\r\n".encode())
elif request == "bzu":
    connection_Socket.send("HTTP/1.1 307 Temporary Redirect \r\n".encode())
    connection_Socket.send("Content-Type: text/html \r\n".encode())
    connection_Socket.send("Location:https:/ritaj.birzeit.edu/ \r\n".encode())
    connection_Socket.send("\r\n".encode())
```

Figure 37: The status code 307 Temporary Redirect

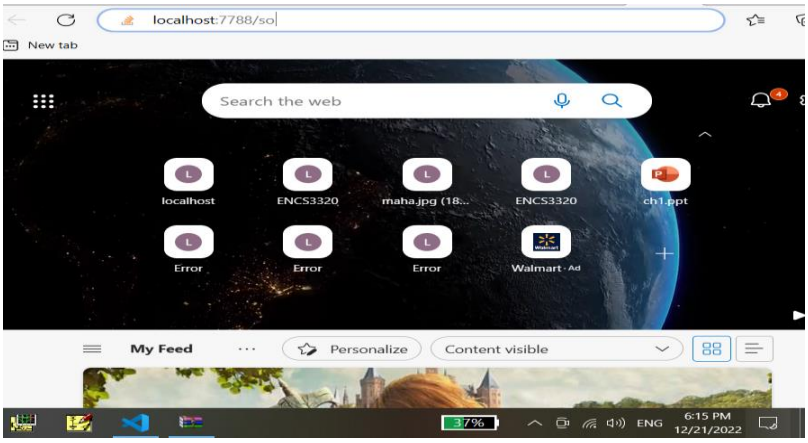
A. If the request is /go then redirect to google website.

Directly transmitted to Google after sending.

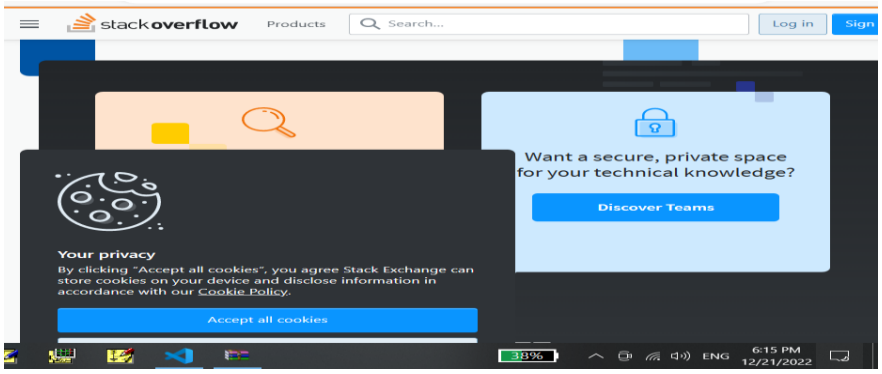




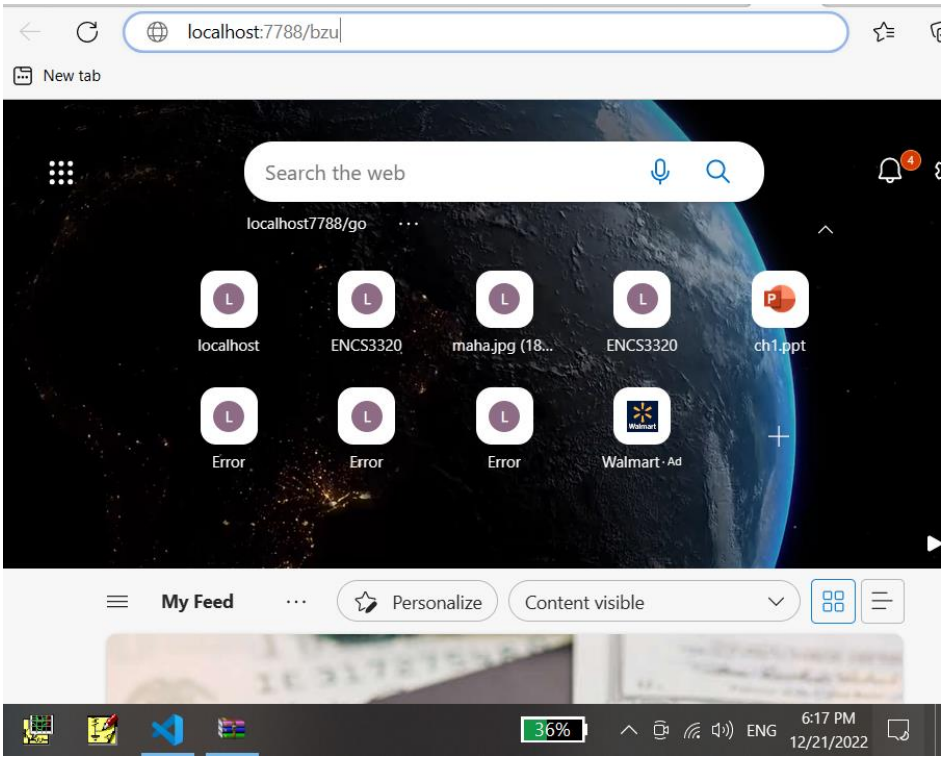
B. If the request is /so then redirects to stackoverflow.com website.



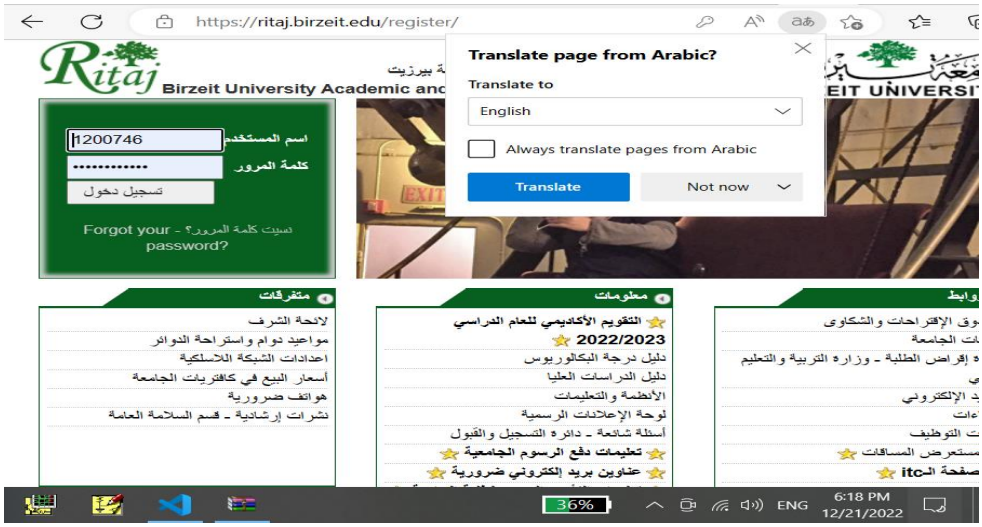
Directly transmitted to stackoverflow after sending.



C. If the request is /bzu then redirect to birzeit university website.



Directly transmitted to Birzeit website after sending.



3.7 Wrong request

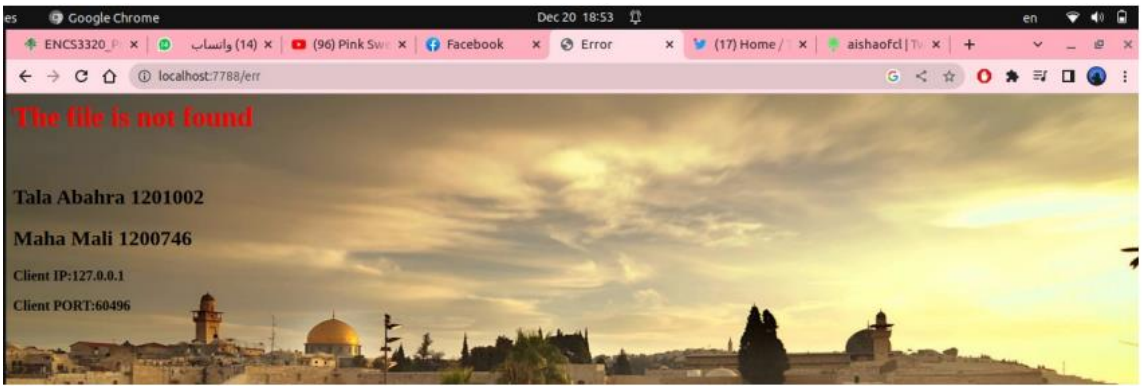
If the request is wrong, the server will return an “notfound.html” file. To add the IP address and the port number, there is two # symbol in the html file. The first one is to replace it by the IP address for the client, the second one is to replace it by the port number of the client.

Code:

```
# If the request doesn't end with the mentioned above, then send the notfound file
else:
    notFoundFile = open("notFound.html")
    errorPage = notFoundFile.read()
    notFoundFile.close()
    connection_Socket.send("HTTP/1.1 404 Not Found\r\n".encode())
    connection_Socket.send("Content-Type: text/html \r\n".encode())
    connection_Socket.send("\r\n".encode())
    connection_Socket.send(errorPage.encode())
    # Close the connection
connection_Socket.close()
```

Figure 38: Wrong Request

output:



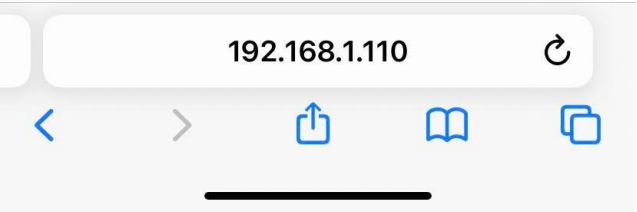
```
tala@tala:~/Desktop/Part3$ /bin/python3 /home/tala/Desktop/Part3/main.py
The Server is Ready!
('127.0.0.1', 60750)
GET /errro HTTP/1.1
Host: localhost:7788
Connection: keep-alive
Cache-Control: max-age=0
sec-ch-ua: "Google Chrome";v="107", "Chromium";v="107", "Not=A?Brand";v="24"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Linux"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/107.0.0.0 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed
-exchange;v=b3;q=0.9
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate, br
Accept-Language: ar-JO,ar;q=0.9,en-US;q=0.8,en;q=0.7
```

Figure 39:Response of Wrong request

3.8 Screenshot from Another Device (phone)

- We used the following IP address to open the figure: 192.168.1.110: 7788/maha.jpg

- We used the following IP address to open the figure: 192.168.1.110: 7788/tala.jpeg



- We used the following IP address to open the figure: 192.168.1.110:7788/jk

4. References

[1] https://www.geeksforgeeks.org/nslookup-command-in-linux-with-examples . Accessed on 11-12-2022 at 6:22PM.

[2] https://www.ibm.com/docs/en/zos/2.2.0?topic=command-nslookup-examples . Accessed on 11-12-2022 at 6:30PM.

[3] https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/ping . Accessed on13-12-2022 at 6:38PM.

[4] https://www.howtogeek.com/355664/how-to-use-ping-to-test-your-network/ . Accessed on 20-12-2022 at 6:44PM.

[5] https://www.lifewire.com/tracert-command-2618101 . Accessed on 20-12-2022 at 7:04PM.

5. Appendix

5.1 part 2

5.1.1 Client code TCP

```
from socket import *
from time import *
serverName = "localhost"
serverPort = 5566
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect(("localhost",serverPort))
for x in range (0,1000001):
    clientSocket.send(str(x).encode())
    modifiedSentence = clientSocket.recv(1024)
    print ("counter is:", modifiedSentence.decode())
    #print(process_time())
    clientSocket.close()
    clientSocket = socket(AF_INET, SOCK_STREAM)
    clientSocket.connect(("localhost", serverPort))
print(" time needed is :")
print(process_time())
```

5.1.2 Server Code TCP

```
from socket import *

serverPort = 5566
serverSocket = socket(AF_INET, SOCK_STREAM)
serverSocket.bind(("", serverPort))
serverSocket.listen(1)
print("The server is ready to receive")
while True:
    connectionSocket, addr = serverSocket.accept()
    sentence = connectionSocket.recv(1024).decode()
    print ("data :",sentence)
    connectionSocket.send(sentence.encode())
    connectionSocket.close()
```

5.1.3 Client Code UDP

```
import socket
from time import *


def client_program():
    host = socket.gethostname()  # as both code is running on same pc
    port = 5566  # socket server port number
    client_socket = socket.socket(family=socket.AF_INET,
type=socket.SOCK_DGRAM)  # instantiate
    message = "from :maha(1200746) and tala(1201002) "  # take input
    client_socket.sendto(message.encode(), (host, port))  # send message
    print(client_socket.recv(1024).decode())
    num = 0
    while (num < 1000000):
        client_socket.sendto(str(num).encode(), (host, port))  # send message
        data, address = client_socket.recvfrom(2048)
        print("number is: " + str(data))
        num += 1
    print(process_time())
    client_socket.close()  # close the connection
if __name__ == '__main__':
    client_program()
```

5.1.4 Server Code UDP

```
import socket
def server_program():
    # get the hostname
    host = socket.gethostname()
    port = 5566  # initiate port no above 1024
    server_socket = socket.socket(family=socket.AF_INET,
type=socket.SOCK_DGRAM)  # get instance
    # look closely. The bind() function takes tuple as argument
    server_socket.bind((host, port))  # bind host address and port together
    # configure how many client the server can listen simultaneously

    message, address = server_socket.recvfrom(2048)
    # accept new connection
    print("Connection from: " + str(address))
    print(str(message))
    server_socket.sendto(str("this connection is ok ").encode(), address)
    counter = 0
    while True:
        data, address = server_socket.recvfrom(2048)
        if not data:
            break
        print("counter is : " + str(data))
        counter += 1
        server_socket.sendto(str(counter).encode(), address)
    server_socket.close()
if __name__ == '__main__':
    server_program()
```

5.2 part 3

5.2.1-part 3 Python Code

```
from socket import *

import os.path
# reserve port 7788 on the computer
serverPort = 7788
# *The SOCK_STREAM: connection-oriented TCP protocol.
serverSocket = socket(AF_INET, SOCK_STREAM)
# Associates the socket with its local address, allowing clients to connect to
the server using that address.
serverSocket.bind(("", serverPort))
# Set the socket to listening mode, where 1 represents the number of incoming
connections.
serverSocket.listen(1)
print("The Server is Ready!")
# Infinite loop until it interrupted or an error occurs
while True:
    # accept() is called by the server to accept or complete the connection.
    connection_Socket, address = serverSocket.accept()
    # Receive data from the server and decode it to obtain the string.
    sentence = connection_Socket.recv(1024).decode()
    print(address)
    print(sentence)
    ip = address[0]
    port = address[1]
    request = sentence.split()[1]
    request = request.lower()
    request = request.lstrip('/')
    # If the request is / or index.html then send the main html file to the
client
   # if request == "/" or request == "/index.html":
    if request == '' or request == 'index.html' or request == 'main_en.html' or
request == 'en': # If the requested file is index.html or index or empty then
send the index.html file to the client.
        # Open the requested file
```

```
        requestedFile = open("main_en.html")
        # Read the requested file
        webPage = requestedFile.read()
        # Close the requested file after getting the data from it
        requestedFile.close()
        # Send the response status to the client after encoding it to the byte
type
        connection_Socket.send("HTTP/1.1 200 OK\r\n".encode())
        # send the type of the file after encoding it to the byte type
        connection_Socket.send("Content-Type: text/html \r\n".encode())
        # send CRLF(new line) to the client after encoding it to the byte type
        connection_Socket.send("\r\n".encode())
        # send the requested file that we read before to the client
        connection_Socket.send(webPage.encode())

    elif request == 'main_ar.html' or request == 'ar': # If the requested file is
 main_ar.html or index or empty then send the index.html file to the client.
        requestedFile = open("main_ar.html")
        webPage = requestedFile.read()
        requestedFile.close()
        connection_Socket.send("HTTP/1.1 200 OK\r\n".encode())
        connection_Socket.send("Content-Type: text/html \r\n".encode())
        connection_Socket.send("\r\n".encode())
        connection_Socket.send(webPage.encode())

        # If the request ends with .css then send the Cascading Style
Sheet:"style.css" to the client
    elif request.endswith(".css"):
        requestedFile = open("style.css")
        webPage = requestedFile.read()
        requestedFile.close()
        connection_Socket.send("HTTP/1.1 200 OK\r\n".encode())
        connection_Socket.send("Content-Type: text/css \r\n".encode())
        connection_Socket.send("\r\n".encode())
        connection_Socket.send(webPage.encode())
        # If the request ends with .html then send java script code: used in the
main html code
    elif request.endswith(".html"):
        requestedFile = open(request)
        webPage = requestedFile.read()
        requestedFile.close()
        connection_Socket.send("HTTP/1.1 200 OK\r\n".encode())
        connection_Socket.send("Content-Type: text/css \r\n".encode())
        connection_Socket.send("\r\n".encode())
```

```
        connection_Socket.send(webPage.encode())
    elif request.endswith(".jpg"):
        requestedFile = open(request , "rb")
        webPage = requestedFile.read()
        requestedFile.close()
        connection_Socket.send("HTTP/1.1 200 OK\r\n".encode())
        connection_Socket.send("Content-Type: image/jpg \r\n".encode())
        connection_Socket.send("\r\n".encode())
        connection_Socket.send(webPage)
        # If the request ends with .jpeg then send image code to the client
    elif request.endswith(".jpeg"):
        requestedFile = open(request, "rb")
        webPage = requestedFile.read()
        requestedFile.close()
        connection_Socket.send("HTTP/1.1 200 OK\r\n".encode())
        connection_Socket.send("Content-Type: image/jpeg \r\n".encode())
        connection_Socket.send("\r\n".encode())
        connection_Socket.send(webPage)
            # If the request ends with .png then send image code to the client
    elif request == "go":
        connection_Socket.send("HTTP/1.1 307 Temporary Redirect \r\n".encode())
        connection_Socket.send("Content-Type: text/html \r\n".encode())
        connection_Socket.send("Location:https://www.google.com \r\n".encode())
        connection_Socket.send("\r\n".encode())
    elif request == "so":
        connection_Socket.send("HTTP/1.1 307 Temporary Redirect \r\n".encode())
        connection_Socket.send("Content-Type: text/html \r\n".encode())
        connection_Socket.send("Location:https://www.stackoverflow.com
\r\n".encode())
        connection_Socket.send("\r\n".encode())
    elif request == "bzu":
        connection_Socket.send("HTTP/1.1 307 Temporary Redirect \r\n".encode())
        connection_Socket.send("Content-Type: text/html \r\n".encode())
        connection_Socket.send("Location:https:/ritaj.birzeit.edu/
\r\n".encode())
        connection_Socket.send("\r\n".encode())

    # If the request doesn't end with the mentioned above, then send the notfound
file
    else:
        ST = ('<head> <title>Error</title></head><head><style>body { background-
image: url("back.jpg"); background-repeat: no-repeat;'
'background-attachment: fixed;background-size: 100% 100%;}</style></head><body>
<h1 style="color:red">The file is not found</h1><br>'
```

```
'<h2 style="color:black">Tala Abahra 1201002</h2><h2 style="color:black">Maha
Mali 1200746</h2><p style="color:black">'
'<p><b>Client IP:'+str(address[0])+'</b></p><p><b>Client
PORT:'+str(address[1])+'</b></p></body></html>').encode('utf-8')
        connection_Socket.send("HTTP/1.1 404 Not Found\r\n".encode())
        connection_Socket.send("Content-Type: text/html \r\n".encode())
        connection_Socket.send("\r\n".encode())
        connection_Socket.send(ST)
        # Close the connection
    connection_Socket.close()
```

5.2.2-part 3 Main Html Code

```
<!DOCTYPE html>
<html lang="en" dir="ltr">
<head>
    <title>ENCS3320-My First Webserver</title>
    <link rel="stylesheet" href="style.css">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link href="https://fonts.googleapis.com/icon?family=Material+Icons"
rel="stylesheet">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/5.15.3/css/all.min.css"
        integrity="sha512-
iBBXm8fW90+nuLcSKlbmrPcLa0OT92xO1BIsZ+ywDWZCvqsWgccV3gFoRBv0z+8dLJgyAHIhR35VZc2oM
/gI1w=="
        crossorigin="anonymous" />
    <link rel="stylesheet" href="style.css">
</head>

<body>
    <div class="design">
        <h1>Welcome to our course <span style="color: #0070C0">Computer
Networks</h1>
        <br>
        <div class="bts">
            <h3 style="color: #0070C0"> online HTML file</h3>
            <!-- <button class="btn btn1"> w3schools </button> -->
            <form
action="https://www.w3schools.com/python/gloss_python_multi_line_strings.asp">
                <button class="btn" type="submit">
                    w3schools
                </button>
            </form>
            <h3 style="color: #0070C0">local HTML file  </h3>
            <form action="main_en.html" target="_blank" method="post">
                <button class="btn" type="submit">
                    HTML file
                </button>
            </form>
        </div>
    </div>
```

```
    <div class="Box">
        <div class="card">
            <div class="card-image">
                <img src="back.jpg" alt="">
            </div>

            <div class="profile-image">
                <img src="tala.jpeg" alt="">
            </div>

            <div class="card-content">
                <h2>Tala Abahra 1201002 </h2>
                <p> Student at Birzeit University majors in computer
engineering.I have a good academic background.
                    Modules studied till now during my degree include object
oriented programming , Digital Systems ,
                    perating systems , Data Structures and Database.
                    Self-motivated , team player with strong organizational
skills.
                <p>
            </div>

            <div class="icons">
                <a href="https://www.facebook.com/tala.abahra/" class="fab fa-
facebook"></a>
                <a href="#" class="fab fa-github"></a>
                <a href="https://twitter.com/TAbahra" class="fab fa-twitter"></a>
                <a href="https://www.linkedin.com/in/tala-abahra-6906b5222/"
class="fab fa-linkedin"></a>

            </div>
        </div>

    </div>
    <div class="Box">
        <div class="card">
            <div class="card-image">
                <img src="back.jpg" alt="">
            </div>

            <div class="profile-image">
                <img src="maha.jpg" alt="">
            </div>
```

```
            <div class="card-content">
                <h2>Maha Mali 1200746 </h2>
                <p>I'm a computer engineering student in my third year of study
at Birzeit University. I always try to
                    learn new skills in my field of specialization, so I am very
good in the field of programming
                    languages: C and Java. In addition to that, I learn problem-
solving skill.
                    Also, i have ability to Work under pressure, multi-task and
team spirit .

                </p>
            </div>

            <div class="icons">
                <a href="https://www.facebook.com/profile.php?id=100008196248374"
class="fab fa-facebook"></a>
                <a href="#" class="fab fa-github"></a>
                <a href="#" class="fab fa-twitter"></a>
                <a href="https://www.linkedin.com/in/maha-mali-04a98a222/"
class="fab fa-linkedin"></a>
            </div>
        </div>
    </div>


    <script>

    </script>
</body>
```

5.2.3-part 3 CSS Code

```
.design {
    display: block;
    text-align: center;
    /* height: 85px; */
    width: 100%;
    padding: 10px 0 0 20px;
    /* border: solid red 1px; */

}

.bts{
    display: block;
    text-align: center;
    margin-left: auto;
    margin-right: auto;
}
.btn {
    background-color: rgb(236, 129, 143);
    border: none;
    color: rgb(0, 0, 0);
     padding: 12px 30px;
    /* margin: 20px; */
    font-size: 16px;
    cursor: pointer;
     }

  .btn:hover {
    background-color: #0070C0;
  }

.Box{
    text-align: center;
    width: 80% ;
    border-radius: 10px;
    display: inline-block;
    padding: 50px;
    margin: 20px;
}
body{
    margin: 0;
    padding: 0;
    height: 100vh;
```

```
    justify-content: center;
    align-items: center;
    display: flex;
    background: rgb(245, 223, 182);
}

.card{
    font-family: "Candara", sans-serif;
    width: 340px;
    overflow: hidden;
    background: #fff;
    border-radius: 15px;
    box-shadow: 0 0 25px rgba(0,0,0,0.5);
    display: flex;
    flex-direction: column;
}

.card-image img{
    width: 100%;
    height: 160px;
    border-top-left-radius: 10px;
    border-top-right-radius: 10px;
    object-fit: cover;
}

.profile-image img{
    z-index: 1;
    height: 120px;
    width: 120px;
    position: relative;
    margin-top: -75px;
    display: block;
    margin-left: auto;
    margin-right: auto;
    border-radius: 100px;
    border: 10px solid rgb(243, 157, 157);
    transition-duration: 0.4s;
    transition-property: transform;
}

.profile-image img:hover{
    transform: scale(1.1);
}
```

```
.card-content h3{
    font-size: 25px;
    text-align: center;
    margin: 0;
}

.card-content p{
    font-size: 16px;
    text-align: justify;
    padding: 0 20px 5px 20px;
}

.icons{
    text-align: center;
    padding-top: 5px;
    padding-bottom: 30px;
}

.icons a{
    text-decoration: none;
    font-size: 20px;
    color: rgb(247, 19, 49);
    padding: 0 14px;
    transition-duration: 0.4s;
    transition-property: transform;
}

.icons a:hover{
    color: black;
    transform: scale(1.5);
}
```

5.2.4-part 3 Second Html Code

```
<!DOCTYPE html>
<html>
<body>

<p>This is a paragraph.</p>
<p>This is another paragraph.</p>

</body>
</html>
```

5.2.5-part 3 Notfound Html Code

```
<!DOCTYPE html>

<head> <title>Error</title></head><head><style>body { background-image:
url("back.jpg"); background-repeat: no-repeat;
background-attachment: fixed;background-size: 100% 100%;}</style></head><body>
<h1 style="color:red">The file is not found</h1><br>
<h2 style="color:black">Tala Abahra 1201002</h2><h2 style="color:black">Maha Mali
1200746</h2><p style="color:black">
<p><b>Client IP:'+str(address[0])+'</b></p><p><b>Client
PORT:'+str(address[1])+'</b></p></body></html>
```

5.2.6-part 3 Arabic Html Code

```
<!DOCTYPE html>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<html lang="ar" dir="ltr">
<head>
    <title>ENCS3320-My First Webserver</title>
    <link rel="stylesheet" href="style.css">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link href="https://fonts.googleapis.com/icon?family=Material+Icons"
rel="stylesheet">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/5.15.3/css/all.min.css"
        integrity="sha512-
iBBXm8fW90+nuLcSKlbmrPcLa0OT92xO1BIsZ+ywDWZCvqsWgccV3gFoRBv0z+8dLJgyAHIhR35VZc2oM
/gI1w=="
        crossorigin="anonymous" />
    <link rel="stylesheet" href="style.css">
</head>

<body>
    <div class="design">
        <h1> مرحبابكم في مساق<span style="color: #0070C0">شبكات الحاسوب</h1>
        <br>
        <div class="bts">
            <h3 style="color: #0070C0"> اونلاين HTML مواقع</h3>
            <!-- <button class="btn btn1"> w3schools </button> -->
```

```
            <form
action="https://www.w3schools.com/python/gloss_python_multi_line_strings.asp">
                <button class="btn" type="submit">
                    w3schools
                </button>
            </form>
            <h3 style="color: #0070C0"> HTML  ملفات</h3>
            <form action="main_en.html" target="_blank" method="post">
                <button class="btn" type="submit">
                    HTML file
                </button>
            </form>
        </div>
    </div>



    <div class="Box">
        <div class="card">
            <div class="card-image">
                <img src="back.jpg" alt="">
            </div>

            <div class="profile-image">
                <img src="tala.jpeg" alt="">
            </div>

            <div class="card-content">
                <h2> 1201002 تالا عباهرة</h2>
                <p>    طالبة في جامعة بيرزيت تتخصص في هندسة الكمبيوتر ولدي خلفية أكاديمية جيدة في البرمجة بلغات ،
متعددة و تعامل مع قواعد البيانات ، و كذلك الدوائر الرقمية و الاتصالات .
                امتلك مقدار من روح الفريق و العمل الجاد .
                <p>
            </div>

            <div class="icons">
                <a href="https://www.facebook.com/tala.abahra/" class="fab fa-
facebook"></a>
                <a href="#" class="fab fa-github"></a>
                <a href="https://twitter.com/TAbahra" class="fab fa-twitter"></a>
                <a href="https://www.linkedin.com/in/tala-abahra-6906b5222/"
class="fab fa-linkedin"></a>

            </div>
        </div>
```

```
    </div>
    <div class="Box">
        <div class="card">
            <div class="card-image">
                <img src="back.jpg" alt="">
            </div>

            <div class="profile-image">
                <img src="maha.jpg" alt="">
            </div>

            <div class="card-content">
                <h2> 1200746 مها معالي</h2>
                <p>أنا طالبة هندسة كمبيوتر في السنة الثالثة من دراستي في جامعة بيرزيت. أحاول دائما
                    تعلم مهارات جديدة في مجال تخصصي ، لذلك أنا جيد جدًا في مجال البرمجة
                    بالإضافة إلى ذلك ، أتعلم مهارة حل المشكلات. Java و C : اللغات
                    أيضًا ، لدي القدرة على العمل تحت الضغط وتعدد المهام وروح الفريق.

                </p>
            </div>

            <div class="icons">
                <a href="https://www.facebook.com/profile.php?id=100008196248374"
class="fab fa-facebook"></a>
                <a href="#" class="fab fa-github"></a>
                <a href="#" class="fab fa-twitter"></a>
                <a href="https://www.linkedin.com/in/maha-mali-04a98a222/"
class="fab fa-linkedin"></a>
            </div>
        </div>
    </div>


    <script>

    </script>
</body>
```