Department of Computer Science

ENCS3320, COMPUTER NETWORKS

Project#1: Socket Programming

G9 :

Amro Deek 1221642 , Section 2

Karmel Bader 1221473 , Section 2

Shaden Hamda 1220169 , Section 2

Date : 15/8/2024

Contents

Theory and Procedure Part

1. Introduction to Socket Programming

Socket programming is a method of enabling communication between two or more devices over a network. It involves creating endpoints (sockets) that allow data to be sent and received over the network. This project focuses on understanding and implementing basic concepts of socket programming using TCP (Transmission Control Protocol) and UDP (User Datagram Protocol).

2. TCP and UDP Sockets

- **TCP (Transmission Control Protocol):** TCP is a connection-oriented protocol that ensures reliable communication between devices. It guarantees that data sent from one socket will arrive at the other socket in the correct order and without loss. TCP is used for applications where data integrity is crucial (e.g., web browsing, email).
    - **TCP Socket Programming:** Involves creating a client-server architecture where the server waits for connection requests from clients. Once the connection is established, data is exchanged reliably between the two sockets.
- **UDP (User Datagram Protocol):** UDP is a connectionless protocol that allows data to be sent without establishing a connection between sender and receiver. It is faster than TCP but does not guarantee the delivery or order of data packets. It is used for applications where speed is more critical than reliability (e.g., streaming video, online gaming).

3. Commands in Networking

Socket programming requires a deep understanding of basic networking commands and protocols. These commands help in troubleshooting and testing network connectivity, resolving domain names, and diagnosing network-related issues. Some essential networking commands that might be used in this project are:

4. Basics of HTML and Web Server

The project also requires familiarity with basic HTML and web server concepts. This involves understanding how to serve HTML pages over the network using a web server, which listens for HTTP requests and responds with appropriate content, such as web pages, images, or files.

- **HTML:** The standard markup language for creating web pages. It structures the content that a web server sends to a client's web browser.**Web Server**: A server that processes HTTP requests and serves HTML content to clients (browsers). In this project, you would create a basic web server using sockets, which listens for incoming HTTP requests and responds with HTML pages

Task1:

Q1:

Ping

is like sending a quick message to another device on a network to check if it's available and how long it takes to respond. Imagine you're asking, and waiting for a back. If the device answers, it means it's online and reachable. If not, it might be offline or unreachable. It's a simple way to see if a website or server is up and running or if your computer has a connection to it.

Tracert

is like tracking the route your data takes across the internet. Every time you visit a website, your data hops between several devices (called routers) before it gets to its destination. Tracert shows you each "hop" along the way and how long it takes to get from one device to the next. This is helpful if something's slowing down your connection—you can pinpoint exactly where the slowdown is happening.

Nslookup

helps translate a website name (like example.com) into its numerical IP address, which is what computers actually use to find websites. It's like looking up a phone number for a contact. You can also use nslookup to dig into a website's DNS records (like the list of servers that handle its email). It's a handy tool for troubleshooting if a website isn't loading properly or you need to check its domain settings.

Telnet

lets remotely connect to another computer over a network, giving you a way to control it from afar, usually by typing commands. However, telnet isn't very secure because it doesn't encrypt the data it sends, which means someone could intercept sensitive information like passwords. These days, most people use a more secure option like SSH, but telnet can still be useful in specific situations for configuring devices.

Q2:

*Ping a device in the same network from laptop to smartphone

As the figure shows that the IP address 192.168.0.228, sent 4 packets. Each packet was received at different times, with an average delay of 115 ms. ‘There was no packet loss’.

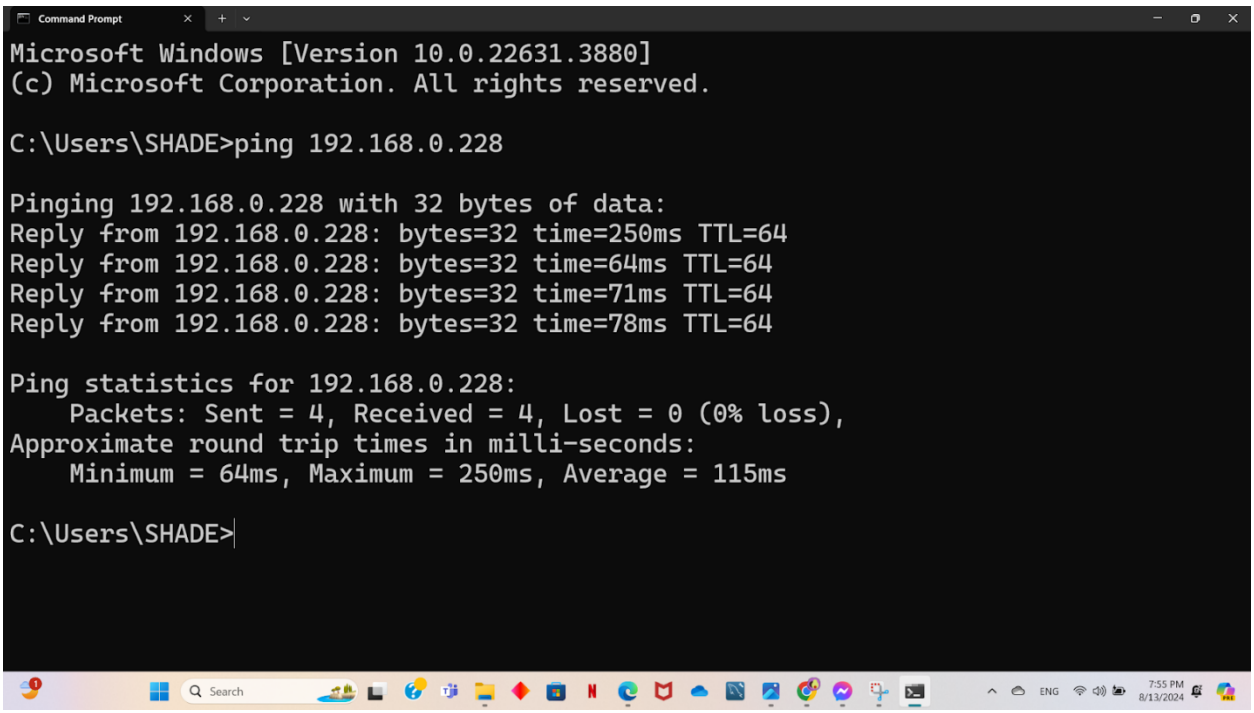


Figure 1:Task1_Q2_partA

*Ping http://www.ox.ac.uk.

The figure shows that the IP response address is 104.22.49.74, and we sent 4 packets. Each packet was received at different times, with an average delay of 3ms. ‘There was no packet loss’.notice that the time to response from my ip takes more than a response from a ac.uk

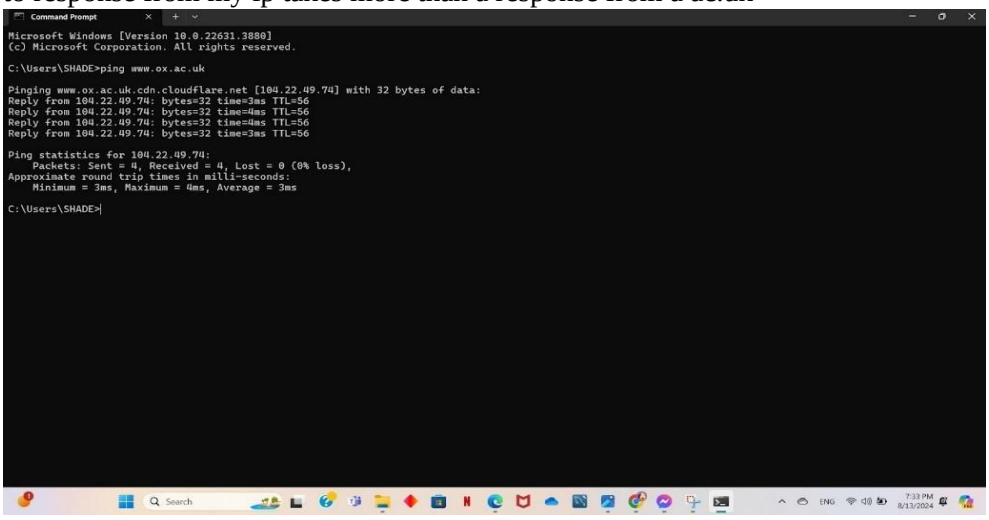


Figure 2:Task1_Q2_partB

*tracert www.ox.ac.uk

The tracert sent 3 messages per every hop(router) and the number of hops is 9 so the total is 27 , the (*) means the packet didn't receive a response within the timeout period , and as we see there is no delays or dropped packets along the route, which suggests a good network path.

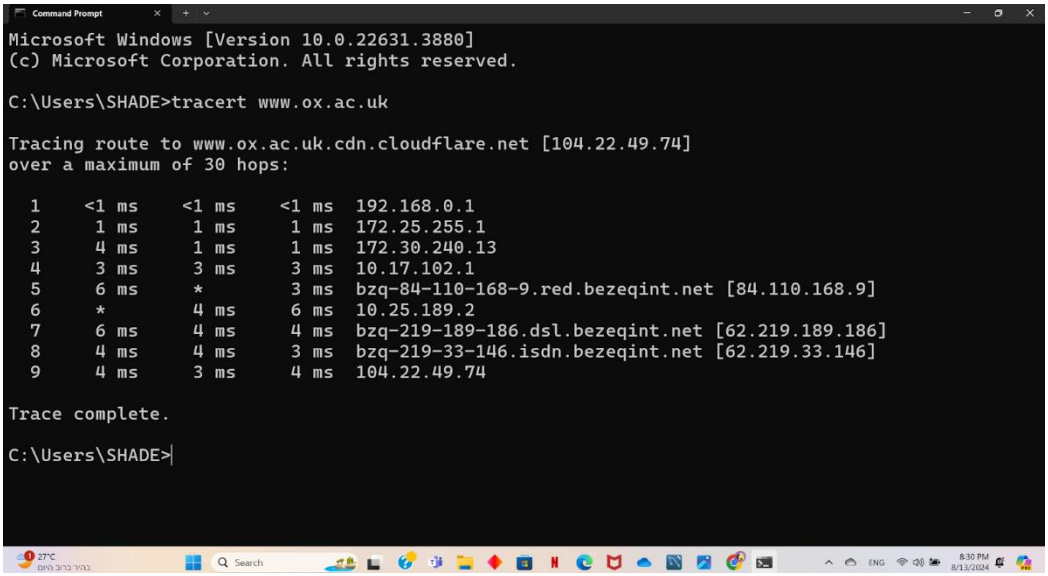


Figure 3:Task1_Q2_partC

*nslookup www.ox.ac.uk

As the figure shows , it prints my router address , and prints the name ,aliases name , and addresses of the server which is the host that we sent a messages.

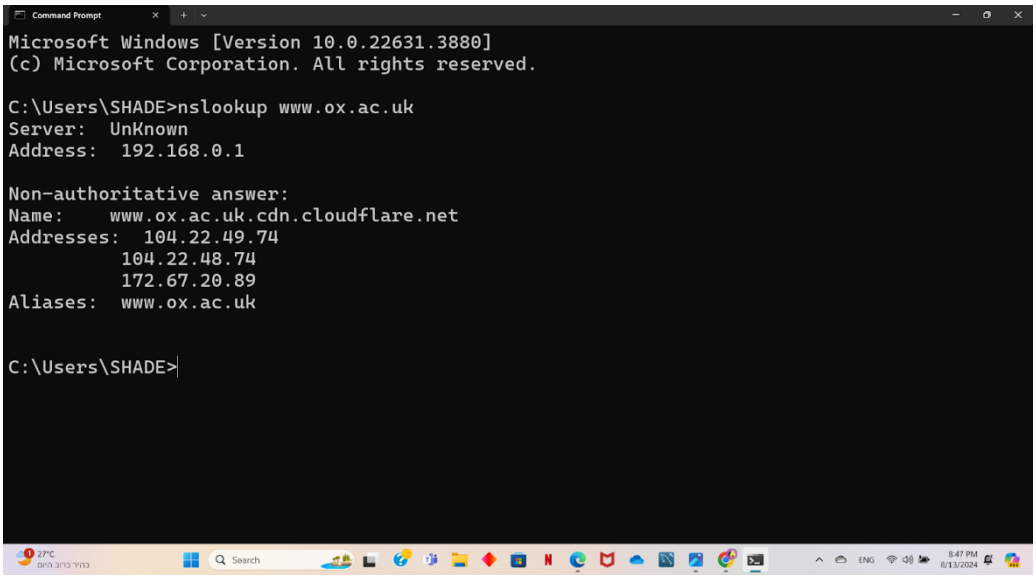


Figure 4:Task1_Q2_partD

telnet www.ox.ac.uk

When I run the command it give me that does not have this port open because Telnet is an outdated and insecure protocol.

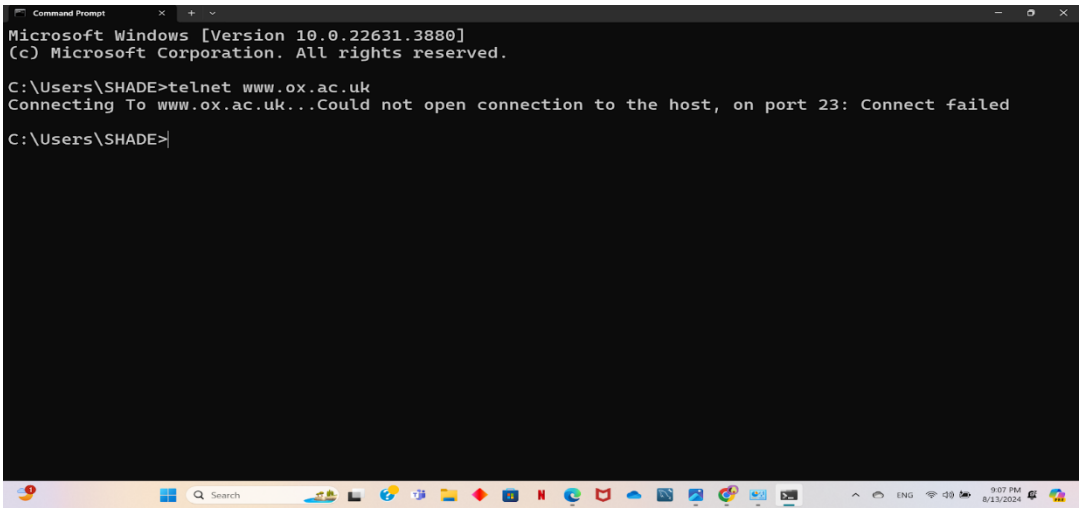


Figure 5:Task1_Q2_partE

Then I change the port to 80

That allow me to communicate with the server (give me bad request because i didn't give the server any specific request)

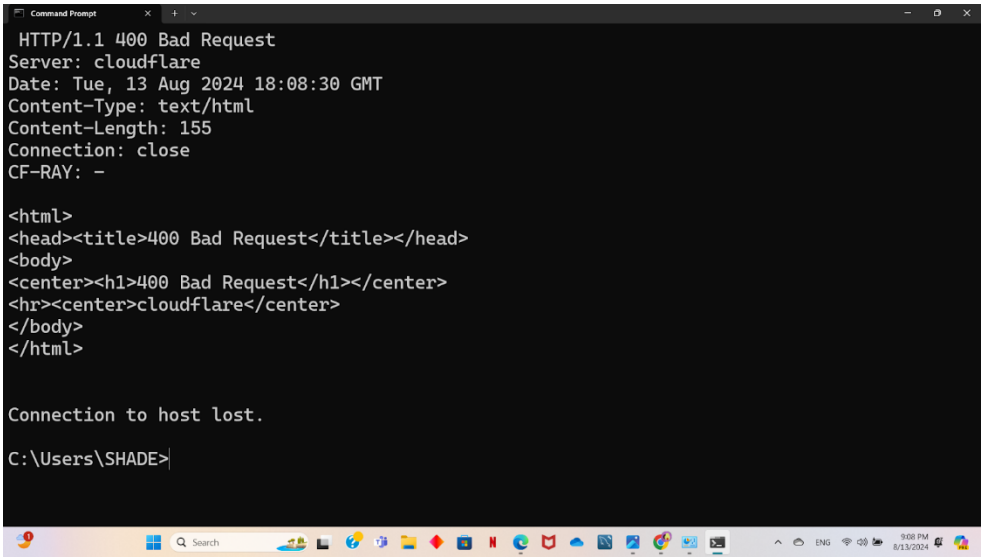


Figure 6:Task1_Q2_partF

Q3:

(we tried to use www.bgpview.io but it didn't work so we use bgp.he.net)

Take the IP address by using nslookup command then by using Hurricane Electric BGP Toolkit (he.net) and find the AS which is AS13335

104.22.49.74

Announced By		
Origin AS	**Announcement**	**Description**
AS13335	104.16.0.0/12	Cloudflare, Inc.
AS13335	104.22.48.0/20	Cloudflare, Inc.

Address has 2 hosts associated with it.

Figure 7:Task1_Q3_AC

The prefixes:

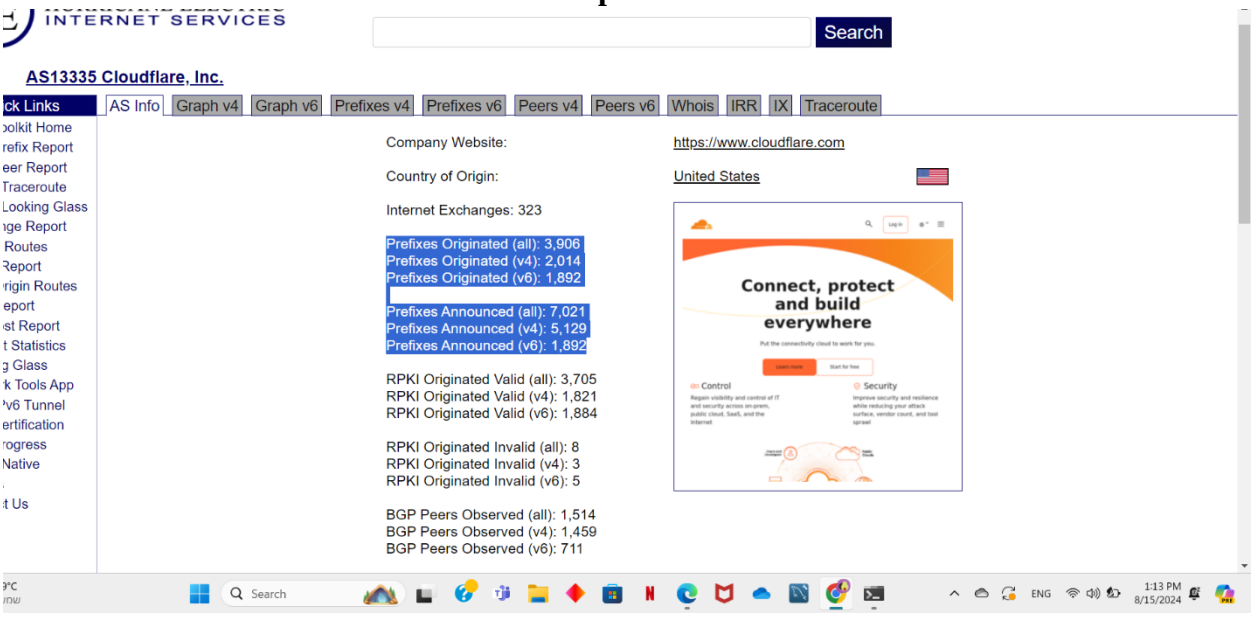


Figure 8:Task1_Q3_prefies

The IPs:

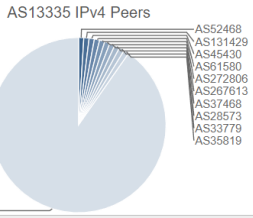
RPKI Originated Valid (v4): 1,821
RPKI Originated Valid (v6): 1,884

RPKI Originated Invalid (all): 8
RPKI Originated Invalid (v4): 3
RPKI Originated Invalid (v6): 5

BGP Peers Observed (all): 1,514
BGP Peers Observed (v4): 1,459
BGP Peers Observed (v6): 711

IPs Originated (v4): 1,698,304
AS Paths Observed (v4): 135,043
AS Paths Observed (v6): 117,100

Average AS Path Length (all): 4.970
Average AS Path Length (v4): 4.990
Average AS Path Length (v6): 4.946



ASN	Name
AS52468	UFINET PANAMA S.A.
AS131429	MOBIFONE Corporation
AS45430	SBN-IIG/AWN-IIG transit provider
AS61580	N◆cleo de Inf. e Coord. do Ponto BR - NIC.BR
AS272806	PIT COLOMBIA SAS
AS267613	ELETRONET S.A.
AS37468	Angola Cables
AS28573	Claro NXT Telecomunicacoes Ltda
AS33779	Wataniya Telecom Algerie
AS35819	Etihad Etisalat, a joint stock company

Figure 9:Task1_Q3_IP

The peers:

AS13335 Cloudflare, Inc.

Links | AS Info | Graph v4 | Graph v6 | Prefixes v4 | Prefixes v6 | Peers v4 | Peers v6 | Whois | IRR | IX | Traceroute

Rank	Description	IPv6	Peer
1	UFINET PANAMA S.A.	X	AS52468
2	MOBIFONE Corporation	X	AS131429
3	SBN-IIG/AWN-IIG transit provider	X	AS45430
4	N◆cleo de Inf. e Coord. do Ponto BR - NIC.BR	X	AS61580
5	PIT COLOMBIA SAS	X	AS272806
6	ELETRONET S.A.	X	AS267613
7	Angola Cables	X	AS37468
8	Claro NXT Telecomunicacoes Ltda	X	AS28573
9	Wataniya Telecom Algerie		AS33779
10	Etihad Etisalat, a joint stock company	X	AS35819
11	COMCEL GUATEMALA S.A.	X	AS23243
12	Liquid Telecommunications Ltd	X	AS30844
13	Mobile Telecommunication Company Saudi Arabia Joint-Stock company	X	AS43766
14	Telconet S.A	X	AS27947
15	Vocus Connect International Backbone	X	AS4826

Figure 10:Task1_Q3_peers

The Tier 1 ISPs name:

Lumen Technologies (formerly known as Level 3)

Q4:

*Use wireshark (free tool and available online) to capture some DNS messages.

It show a lot of packet that I received with details like the port and IP address to source and destination

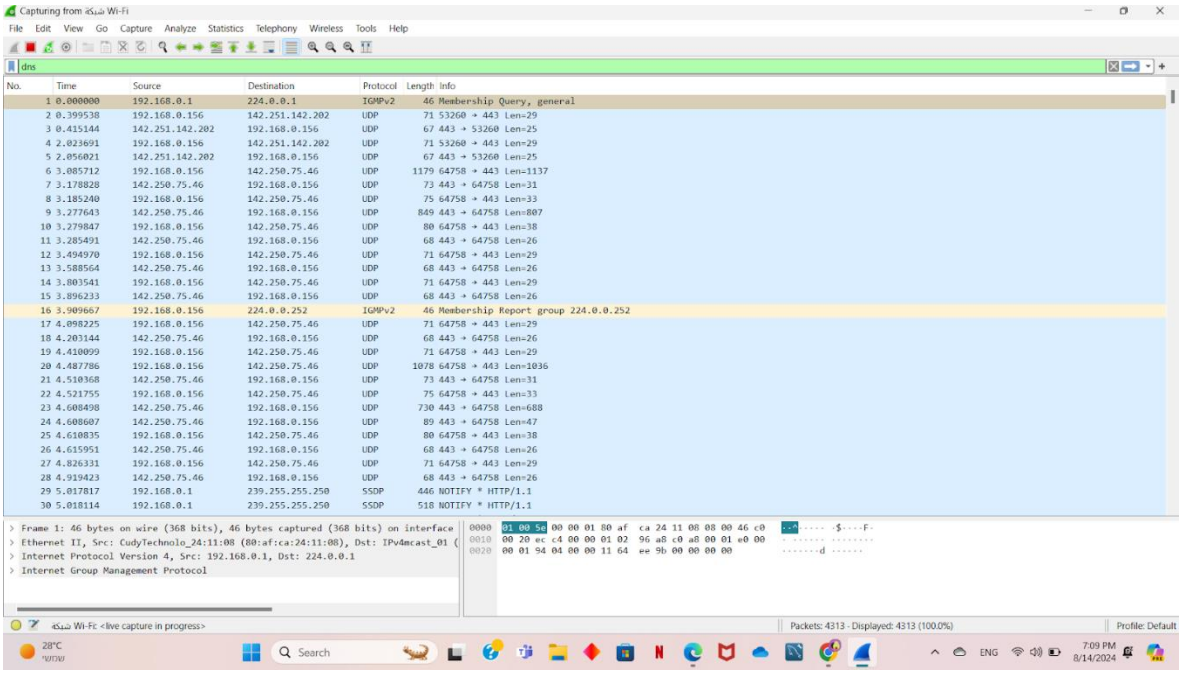


Figure 11:Task1_Q4

Task 2:

Q1:

Using socket programming, with TCP client server python applications , we open a connection between client and server so the client can send data to the server and replace all the vowels (aeiou/AEIOU) with'#' and return the string to the client and print it. We used a port # 1642 for this communication.

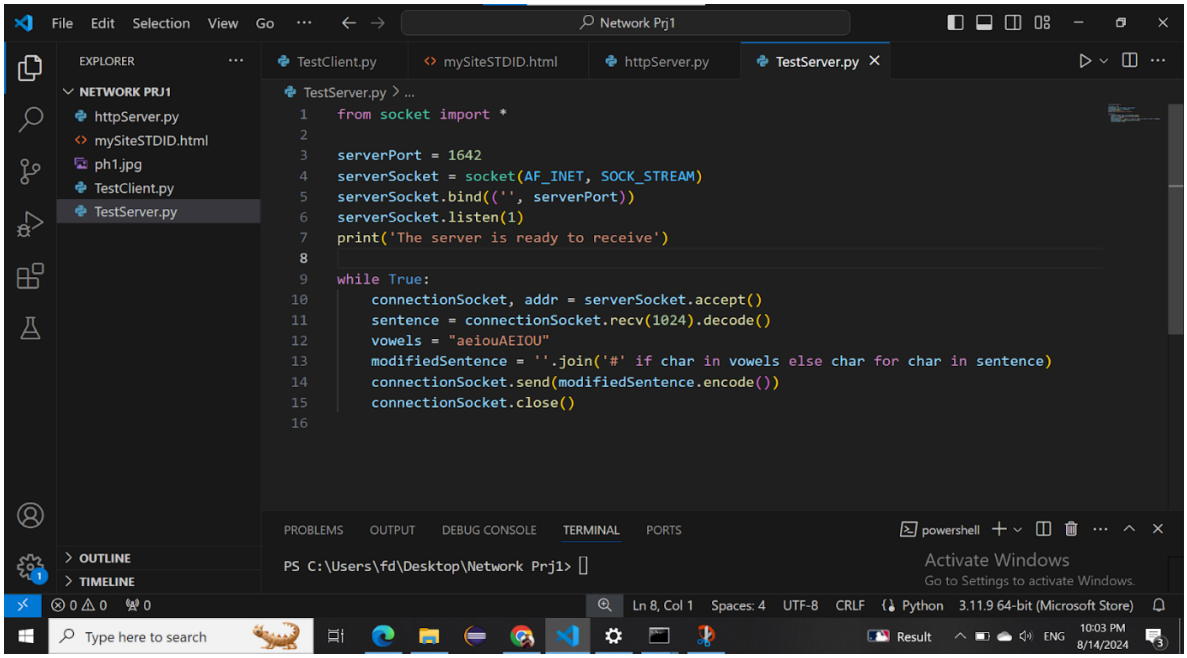


Figure 12:Task2_Q1_server code

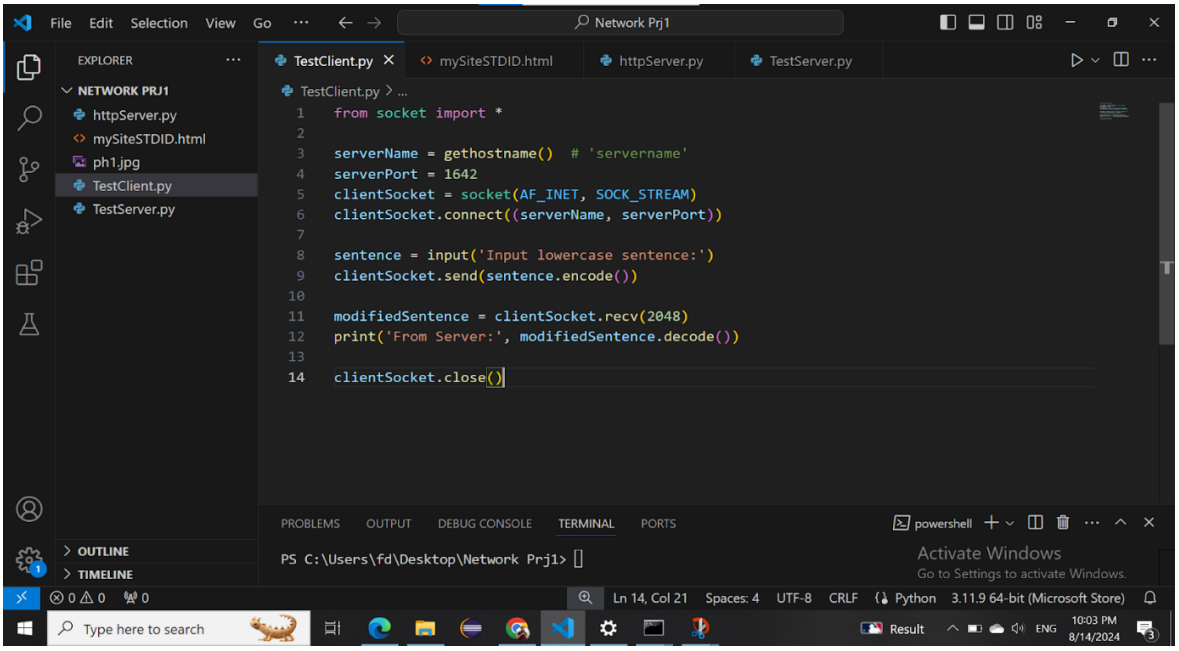


Figure 13:Task2_Q1_client code

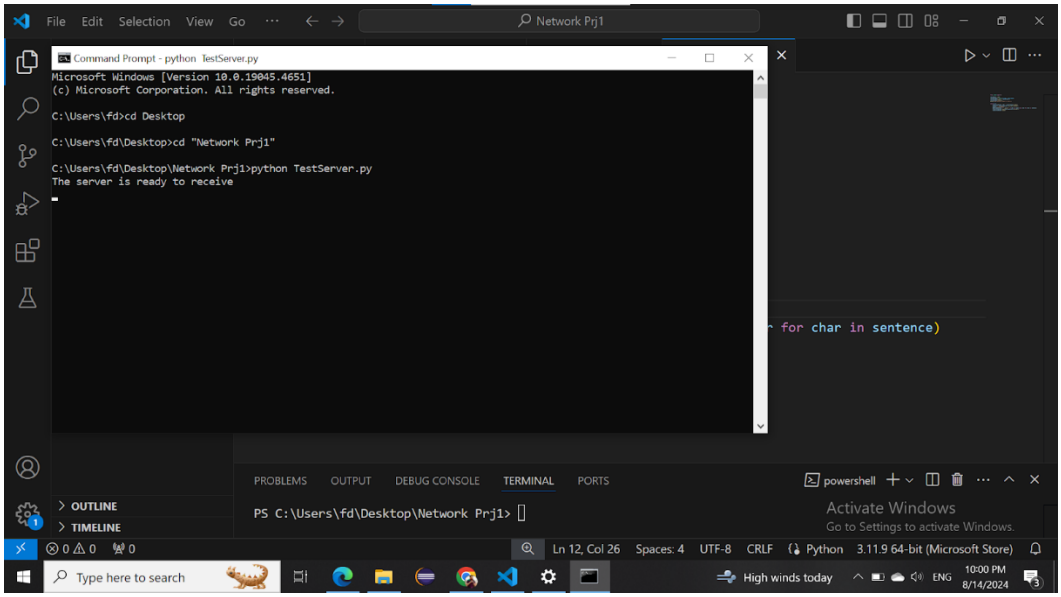


Figure 14:Task2_Q1_server side (run time)

Client Side (run time): after opening connection server becomes ready to receiver input from client and replace all vowels into # sign .

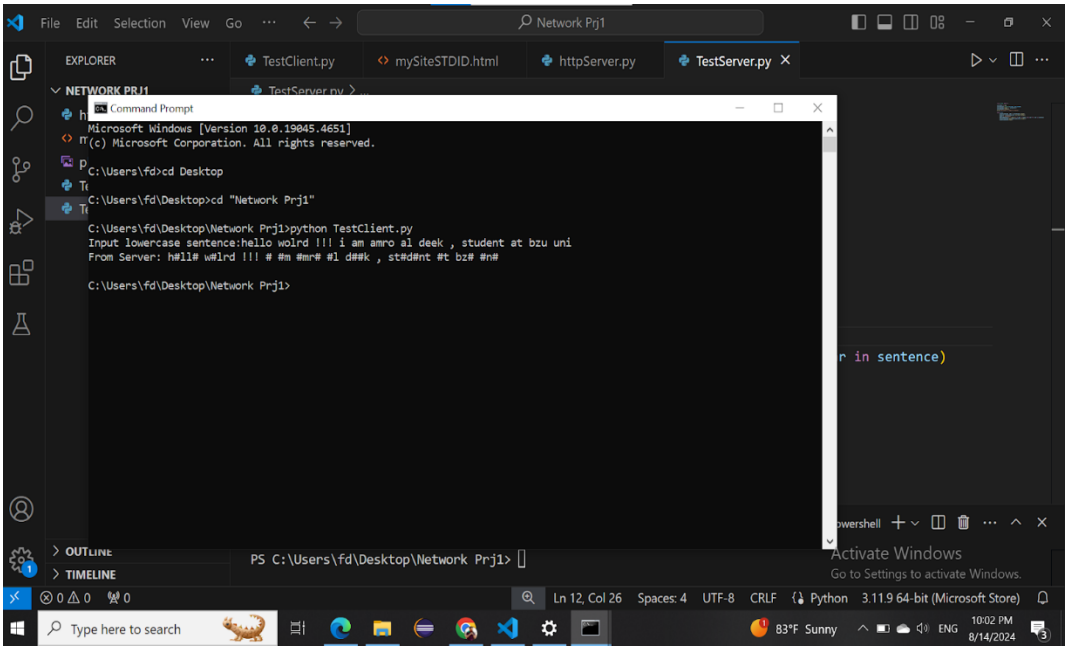


Figure 15: Task2_Q1_client side (run time)

Q2:

*note: my studID is 1220169 so my port should 0169 but the number cant start with 0 so I put it in the end to be 1690 (I told you in person)

This code allows the server to communicate with multiple clients over UDP, assigning each a unique identifier and print messages back. (each client has a unique name)

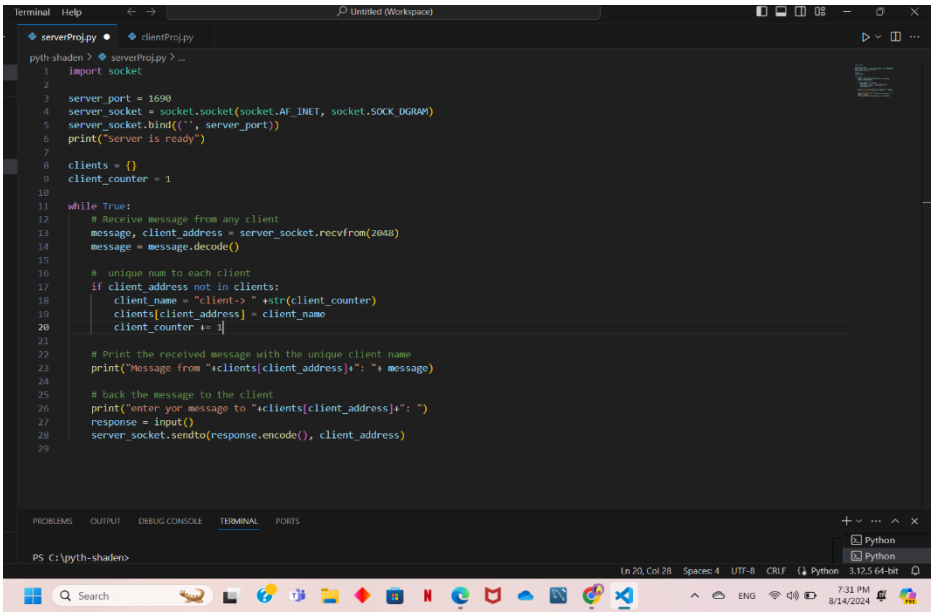


Figure 16:Task2_Q2_code with comments1

This code enables the client to send messages to the server and receive responses

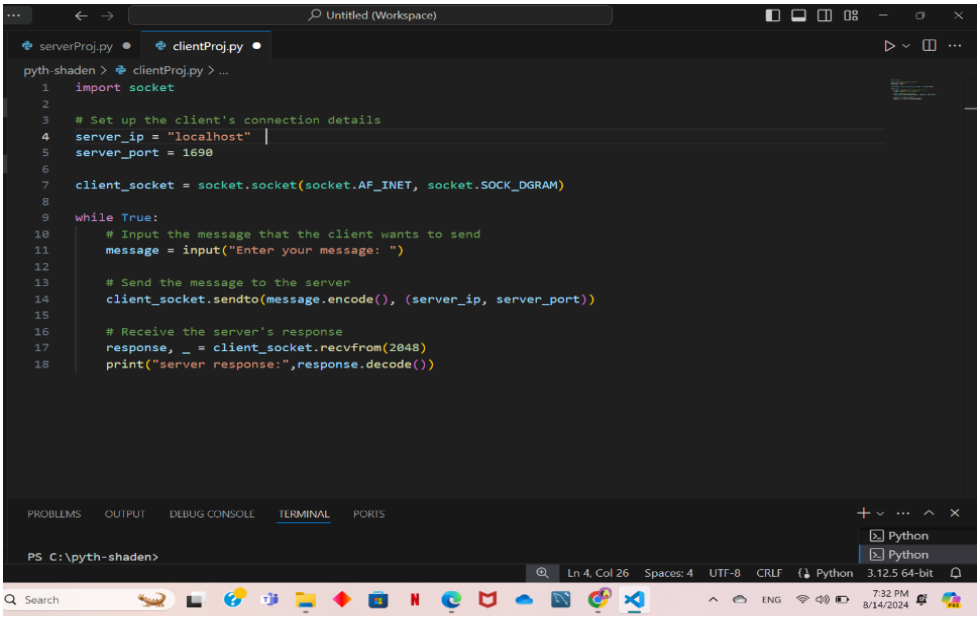


Figure 17: Task2_Q2_code with comments2

There is the result after running both sides , the server(client) and two clients(as a example) , first run the server side to be ready to receive any messages , then run the client side and send a message to the server (client) and wait the answer

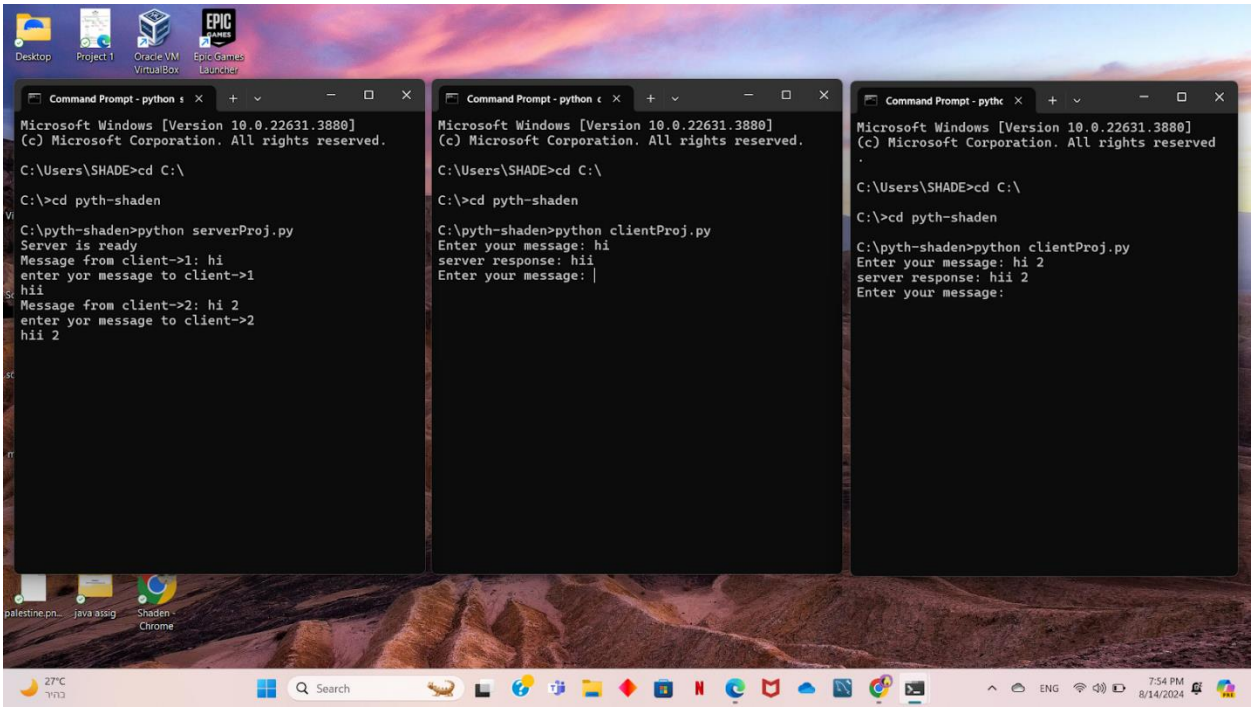


Figure 18: Task2_Q2_output

Task3:

Here is the main_en.html file which the server should send in some cases:

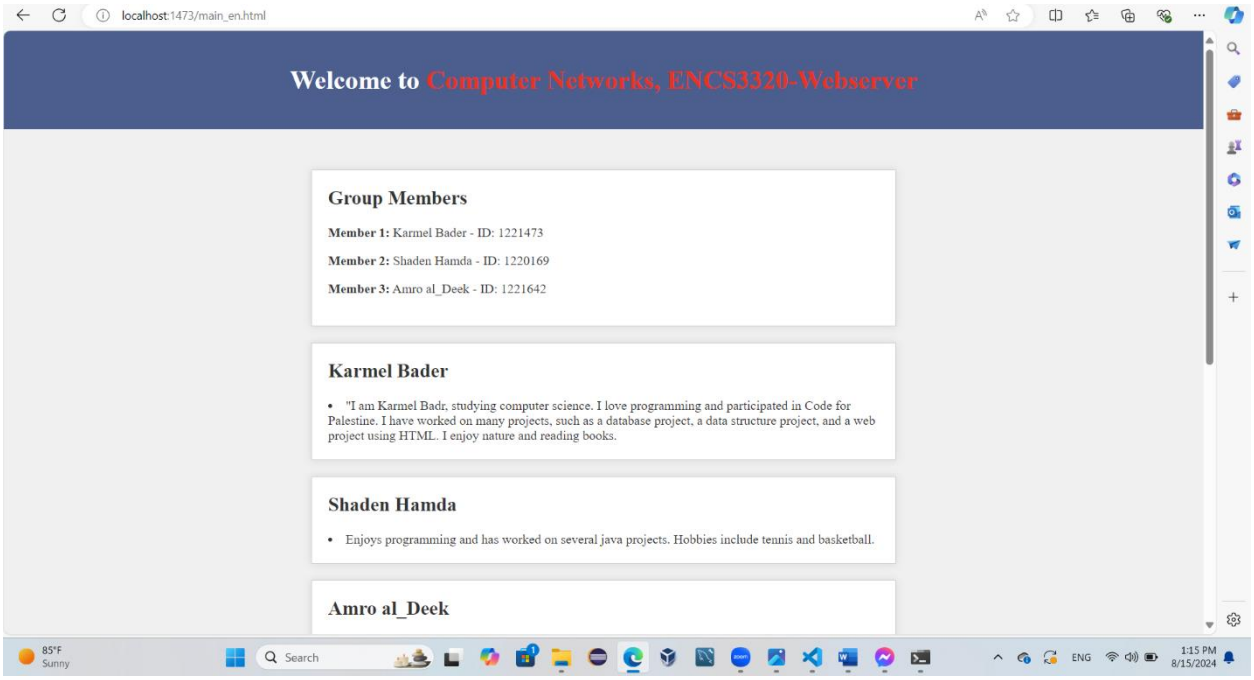


Figure 19: Task3_main_en.html run output1

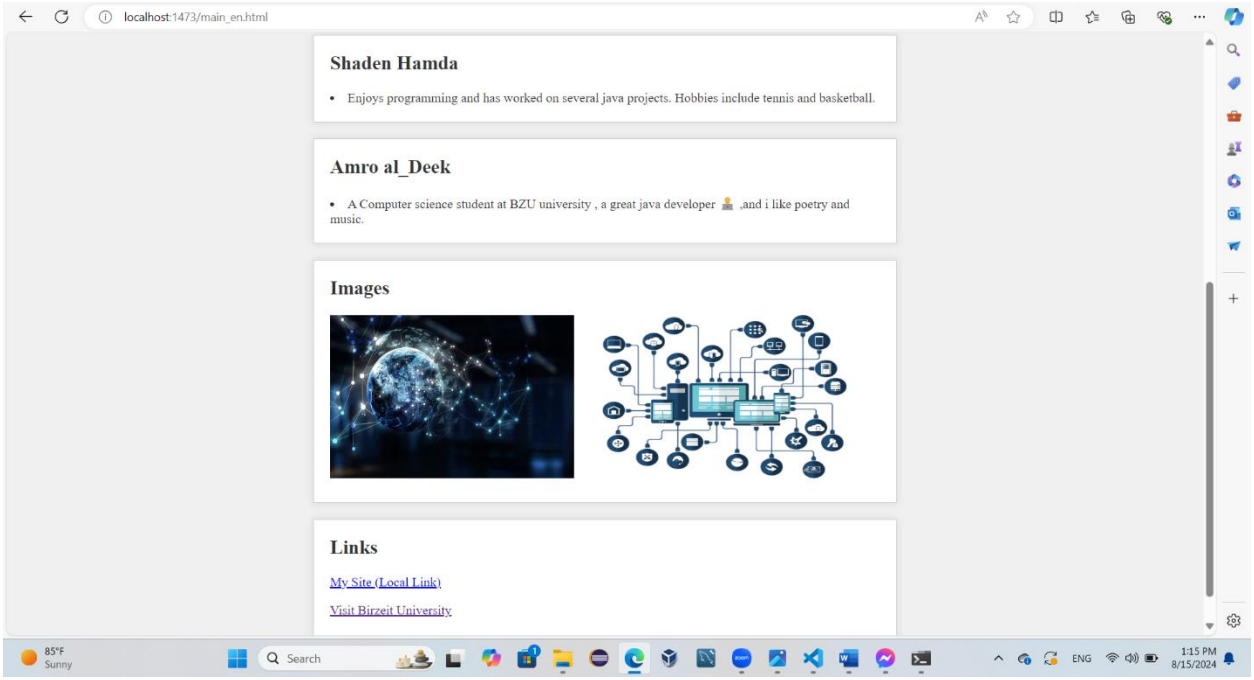


Figure 20: Task3_main_en.html run output2

The main_ar.html file:

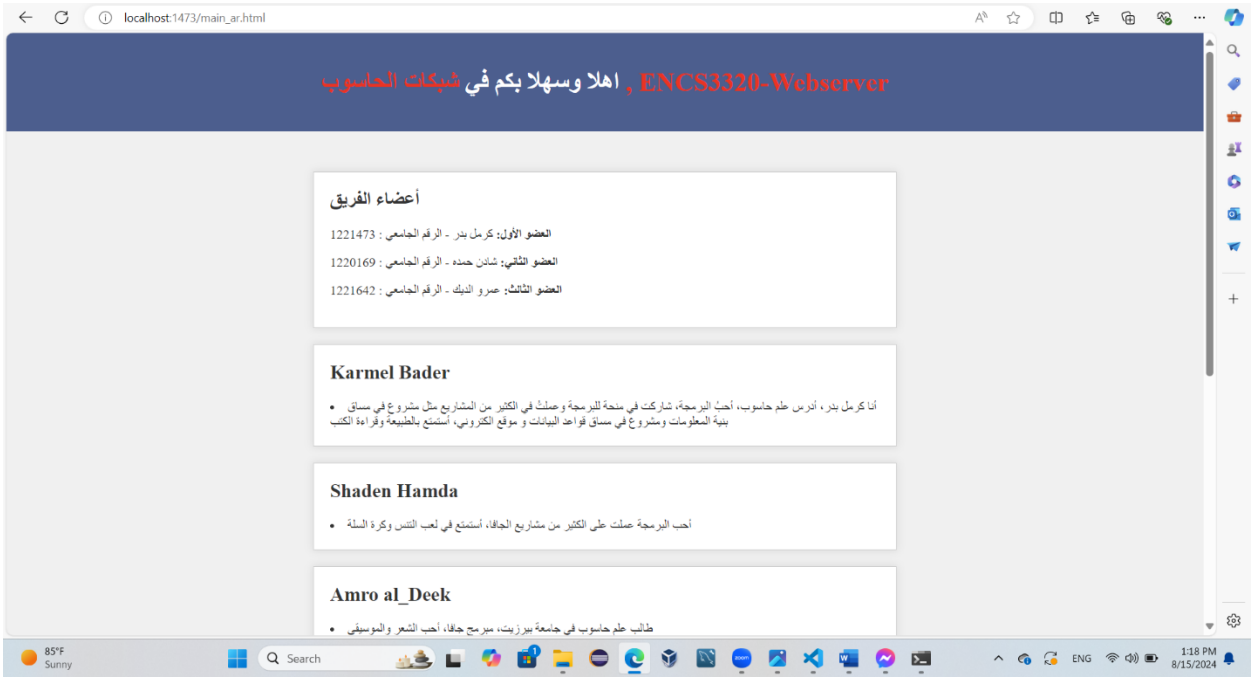


Figure 21: Task3_main_ar.html run output1

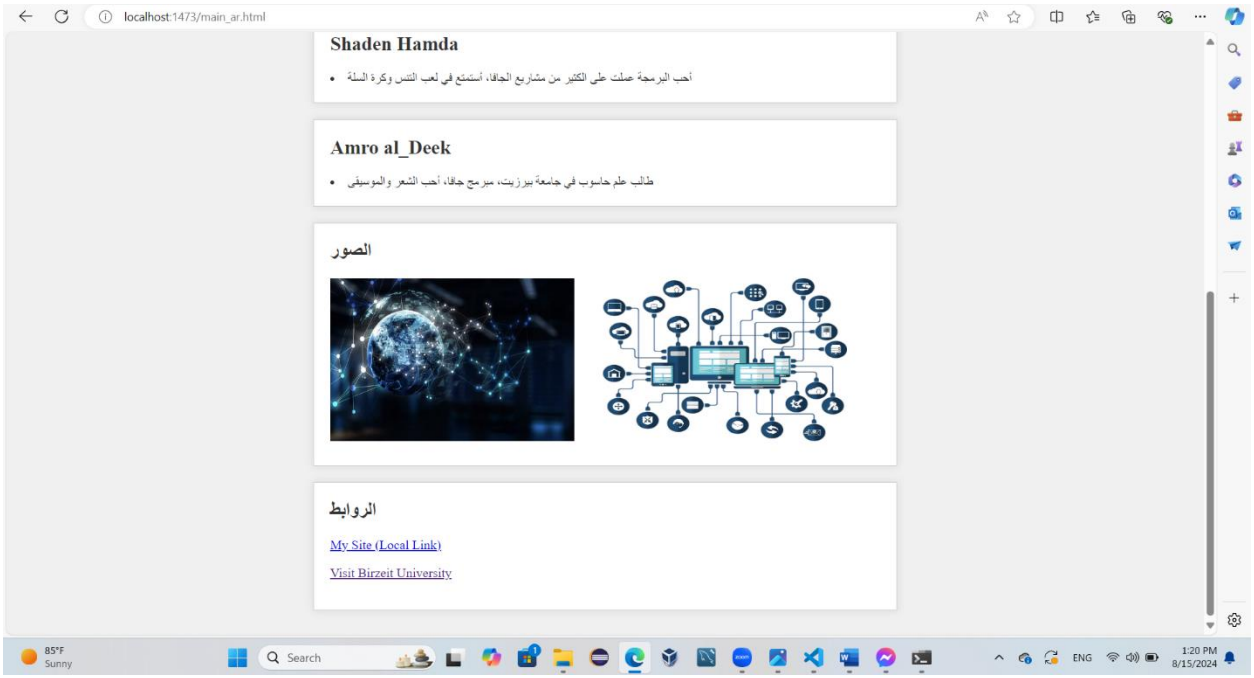


Figure 22: Task3_main_ar.html run output2

Here is the cases of the request:

Case1: when the request localhost:1473/ the server should send the main_en.html file

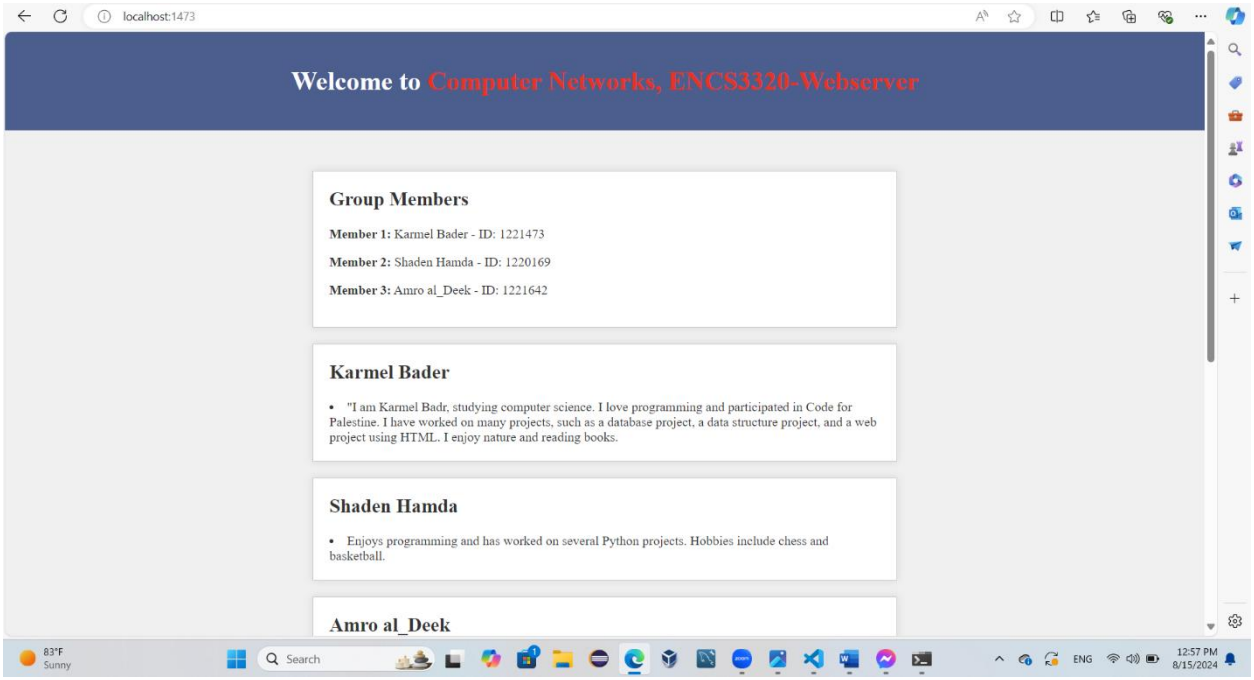


Figure 23: case1 request

The HTTP request printed on the command Line

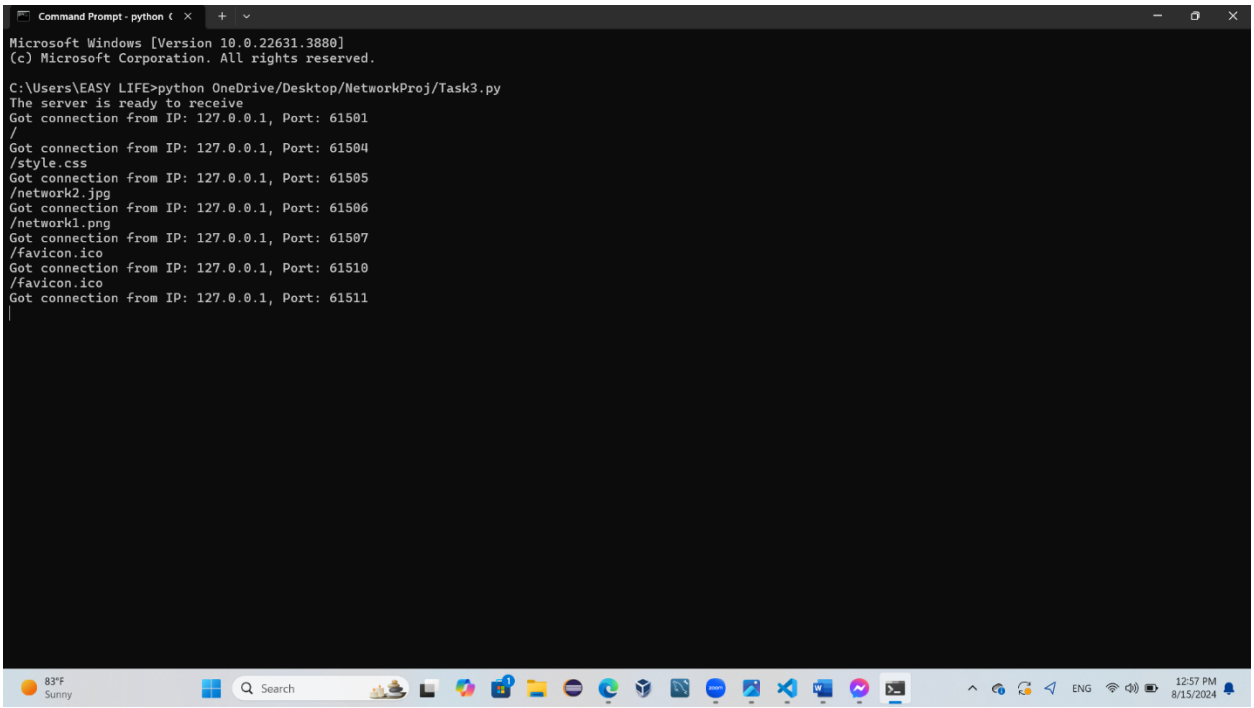


Figure 24: case1 request on command line

Case2: when the request localhost:1473/en the server should send the main_en.html file

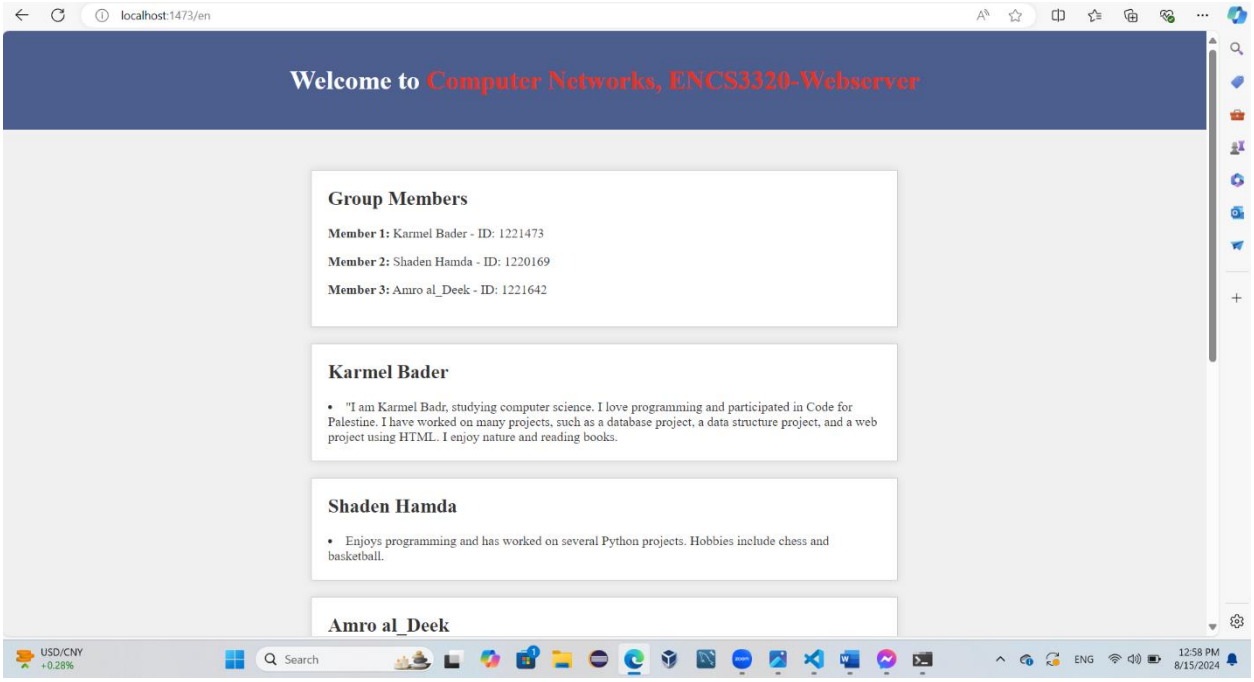


Figure 25: case 2 request

The HTTP request printed on the command Line

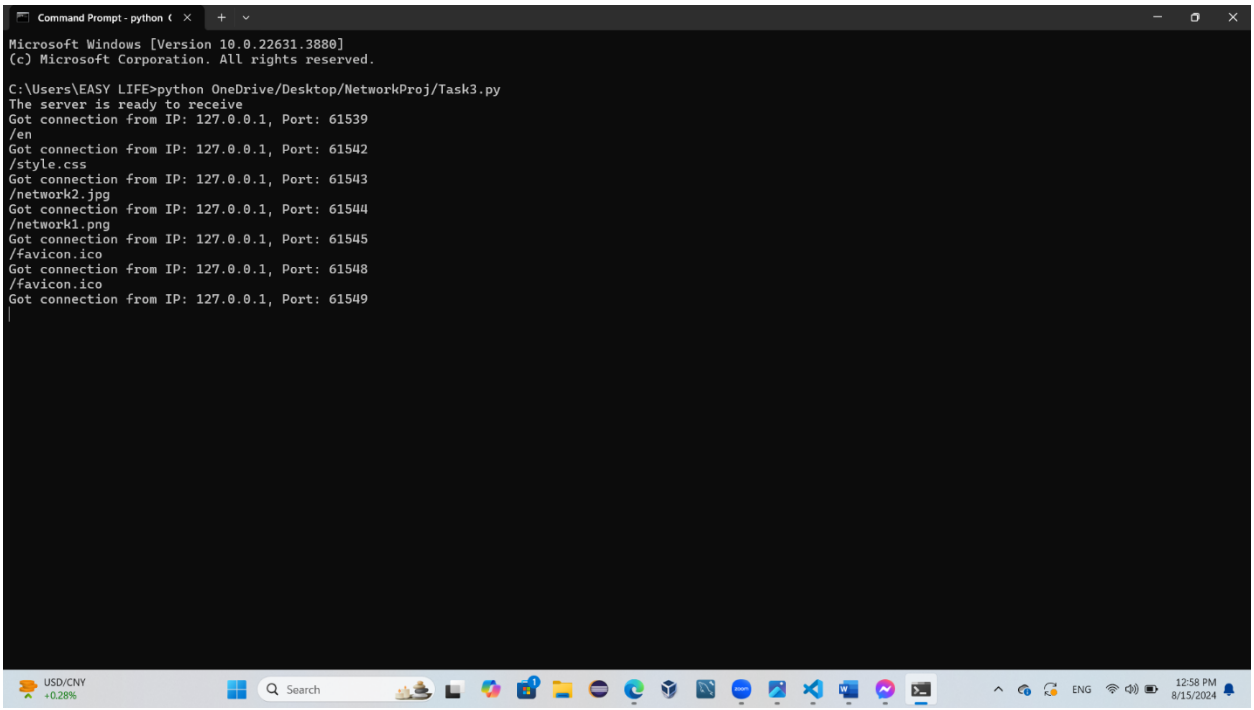


Figure 26: case 2 request on command line

Case3: when the request localhost:1473/main_en.html the server should send the main_en.html file

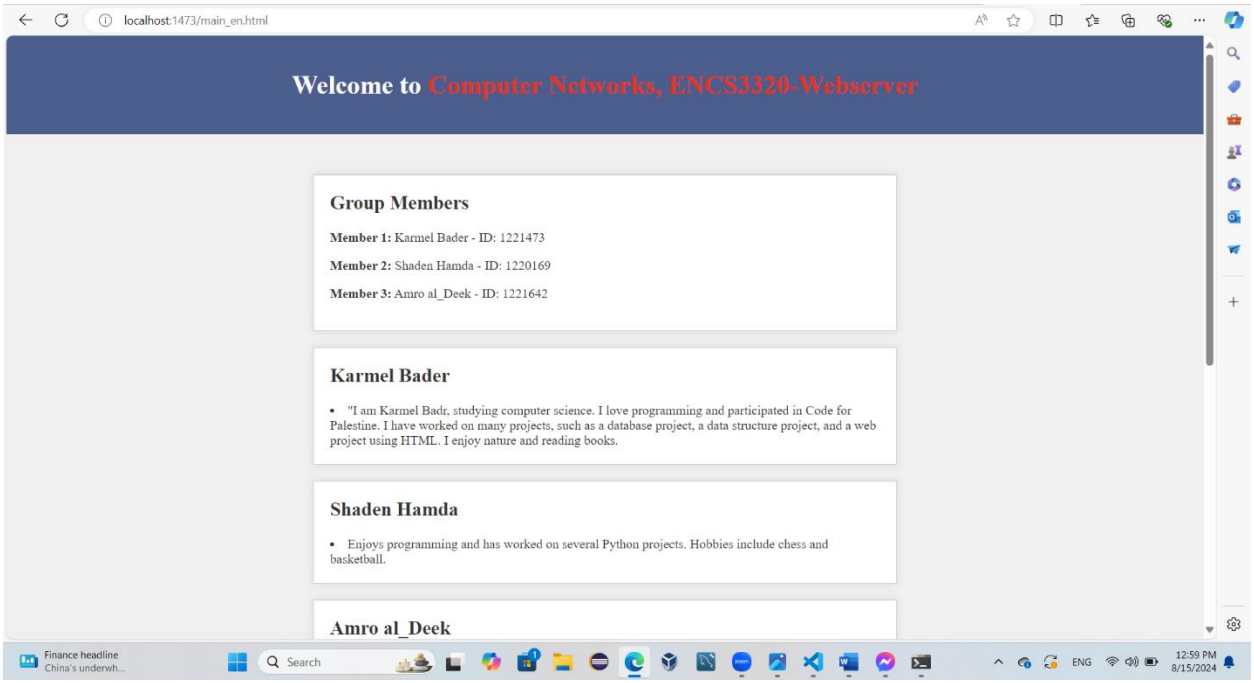


Figure 27: case 3 request

The HTTP request printed on the command Line

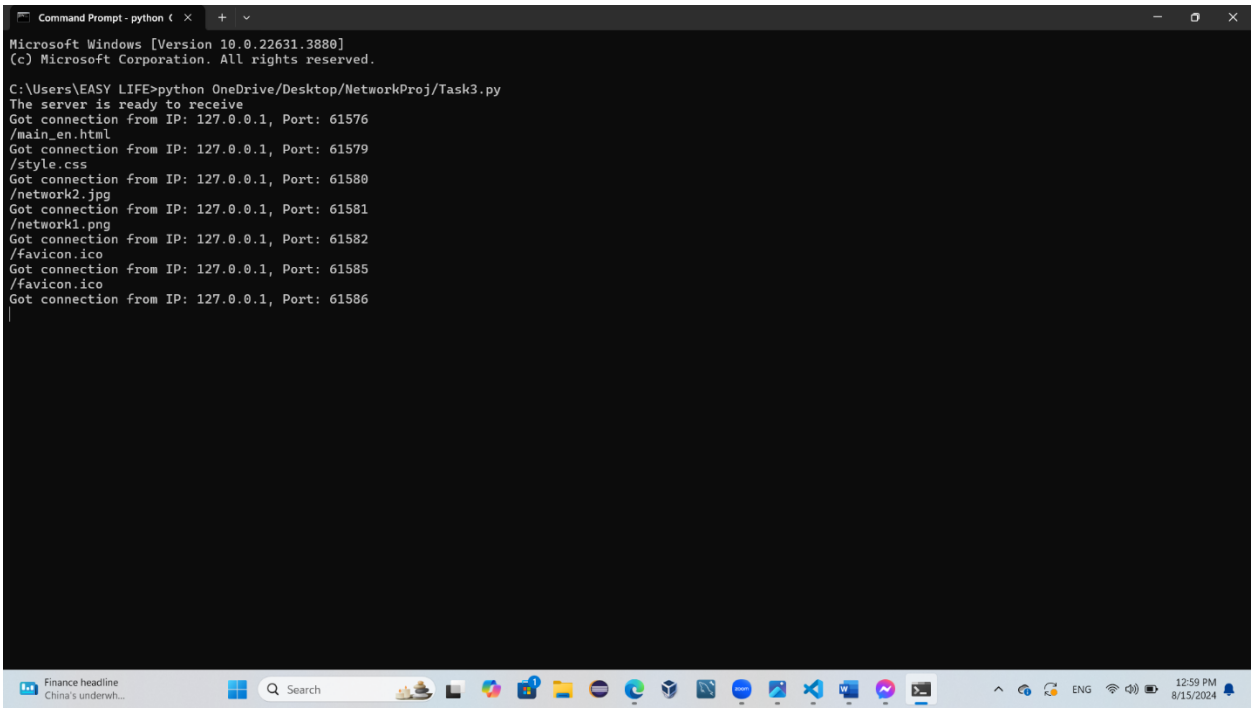


Figure 28: case 3 request on command line

Case4: when the request localhost:1473/index.html the server should send the main_en.html file

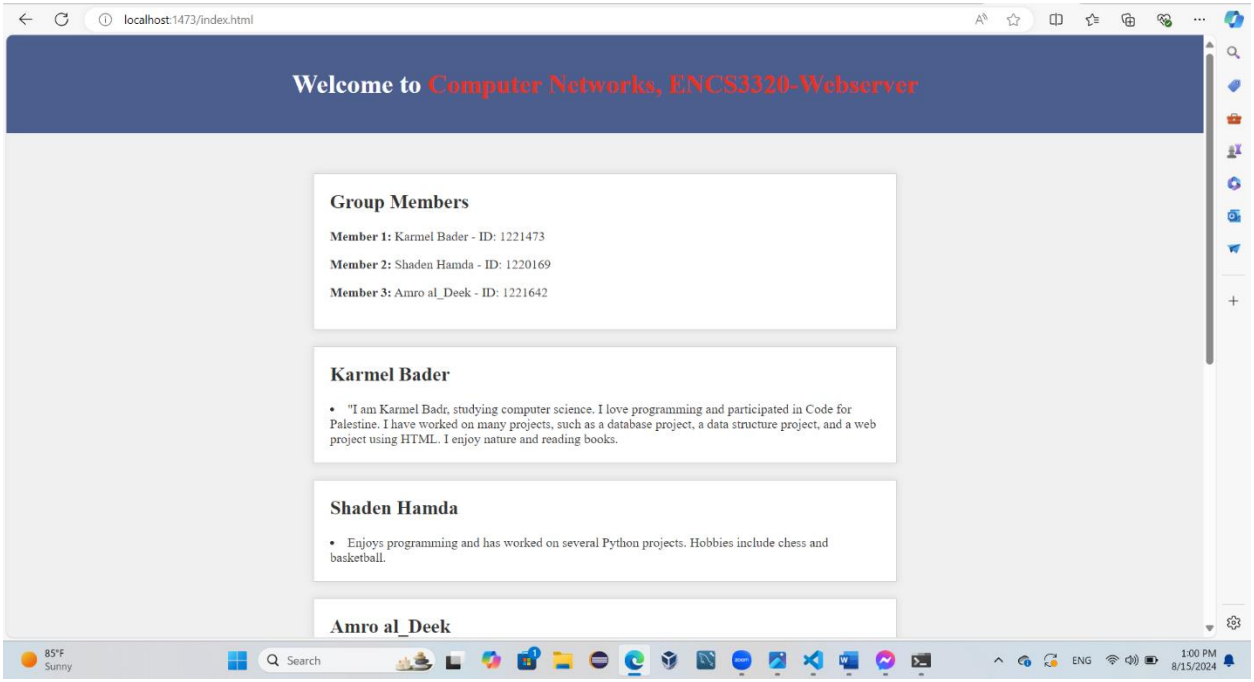


Figure 29: case 4 request

The HTTP request printed on the command Line

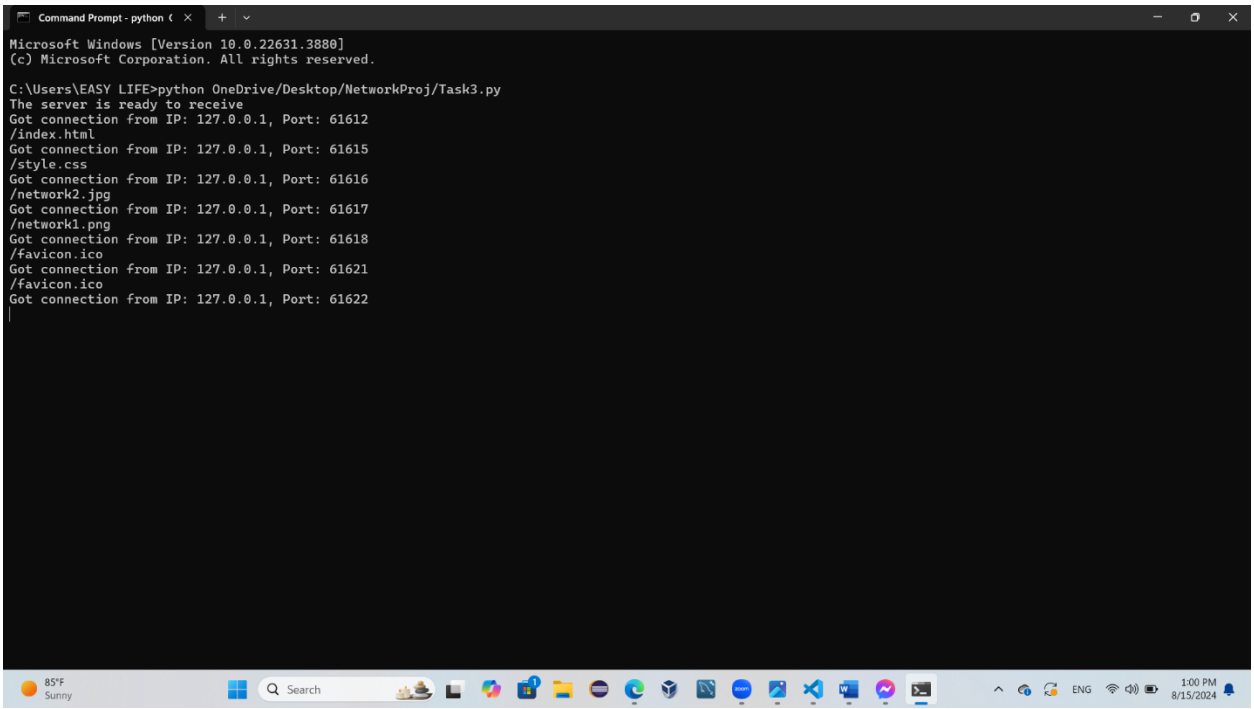


Figure 30: case 4 request on command line

Case5: when the request localhost:1473/main_ar.html the server should send the main_ar.html file

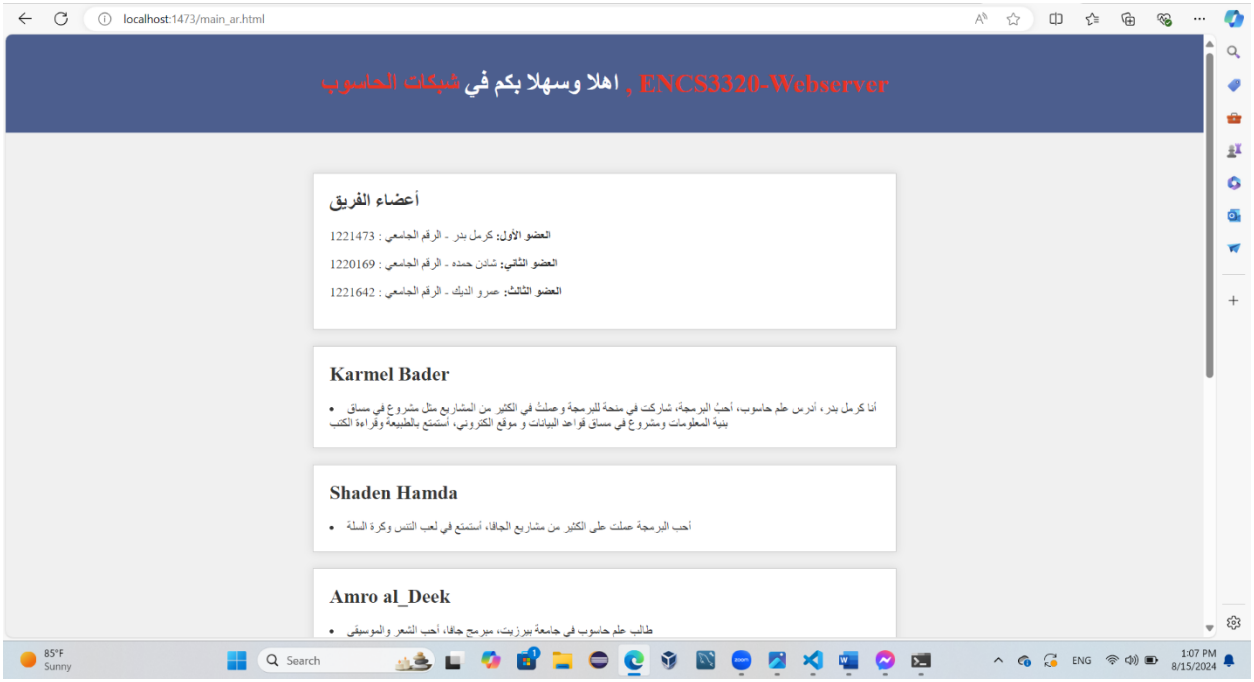


Figure 31: case 5 request

The HTTP request printed on the command Line

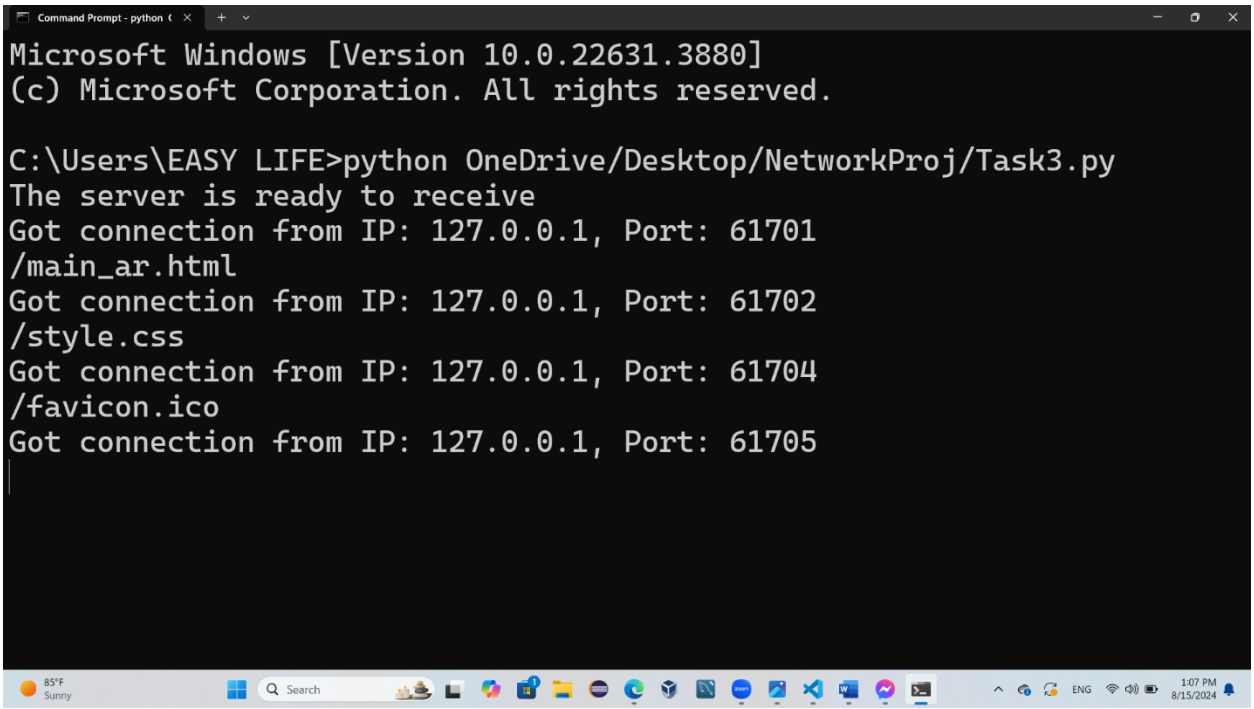


Figure 32: case 5 request on command line

Case6:

when the request localhost:1473/network1.png the server should send the network1.png

Note: network1.png is just an example any another image will work on it.

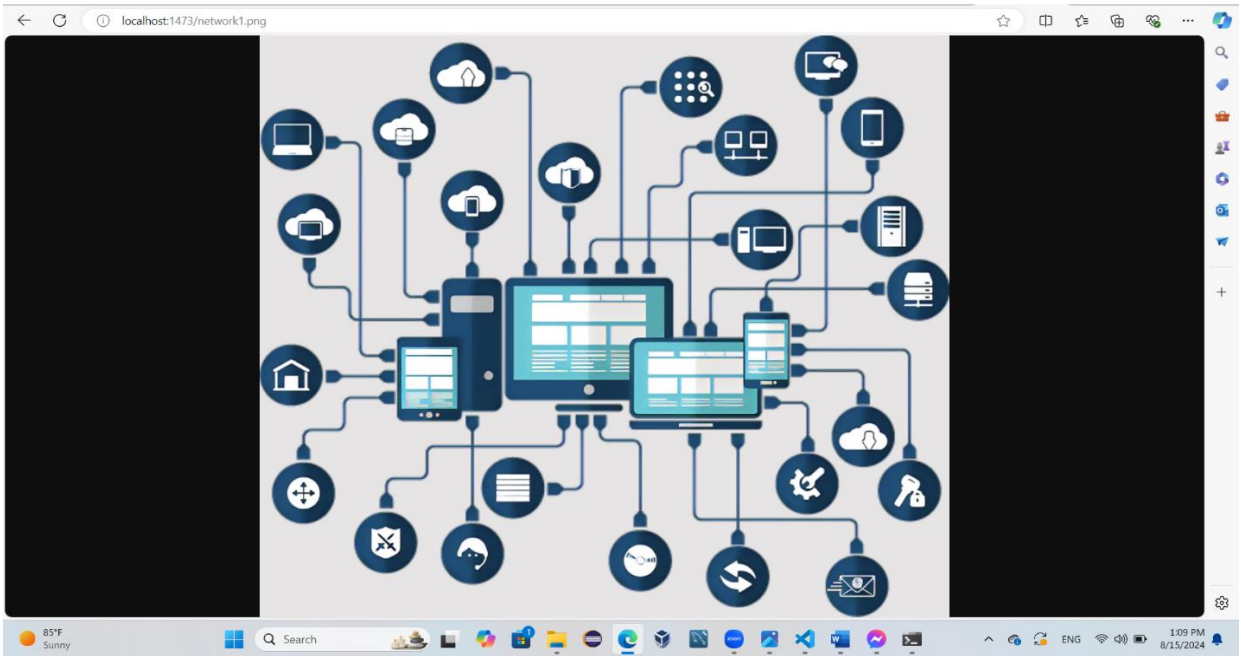


Figure 33: case 6 request

The HTTP request printed on the command Line

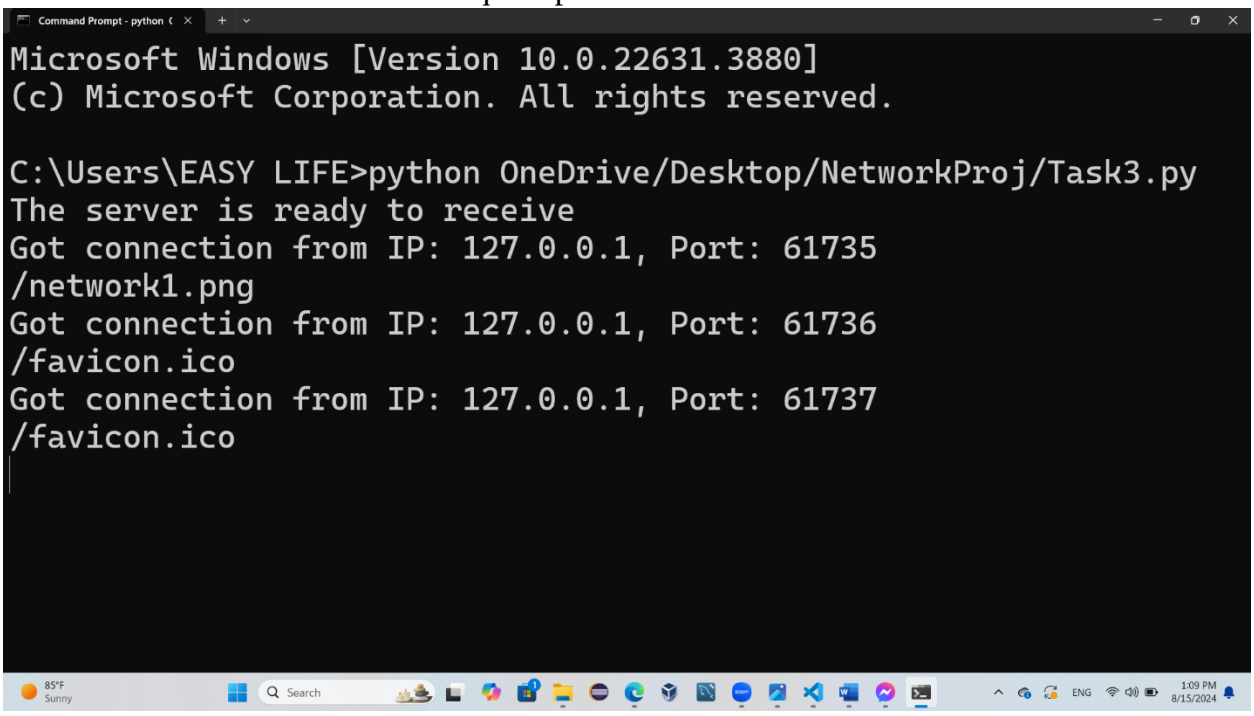


Figure 34: case 6 request on command line

Case7:

when the request localhost:1473/network2.jpg the server should send the network2.jpg

Note: network2.jpg is just an example any another image will work on it.

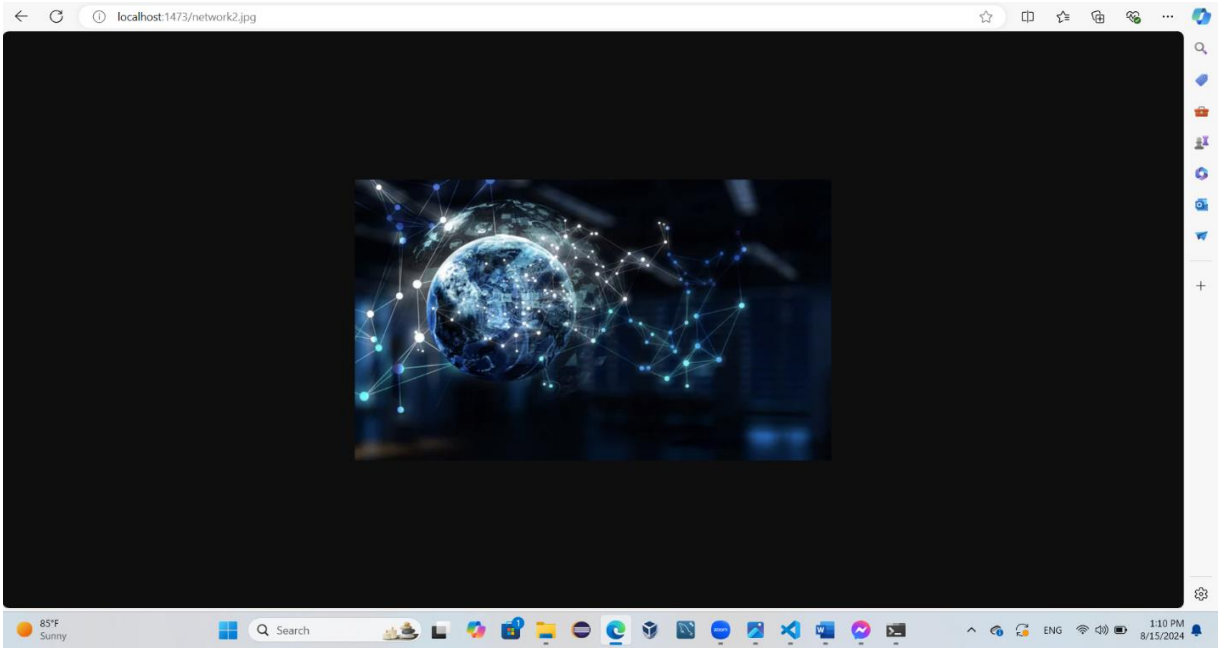


Figure 35: case 7 request

The HTTP request printed on the command Line

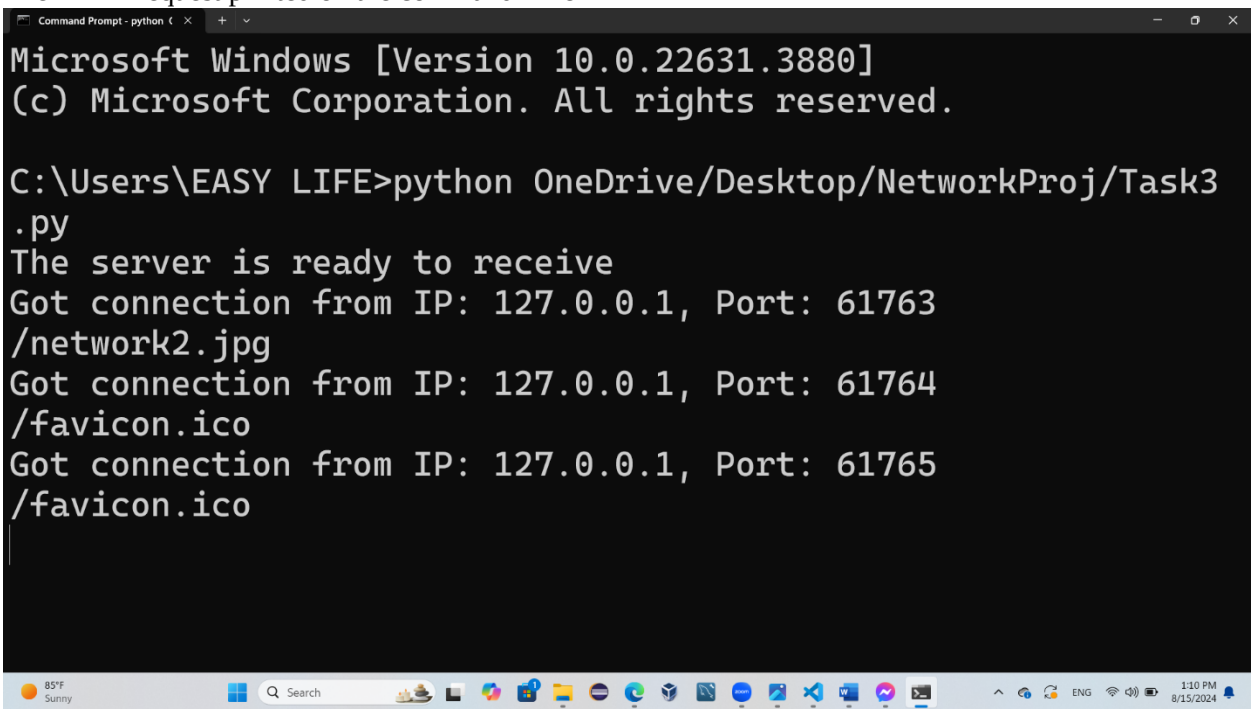


Figure 36: case 7 request on command line

Case8:

when the request localhost:1473/style.css the server should send the style.css

Note: style.css is just an example any another file .css will work on it.

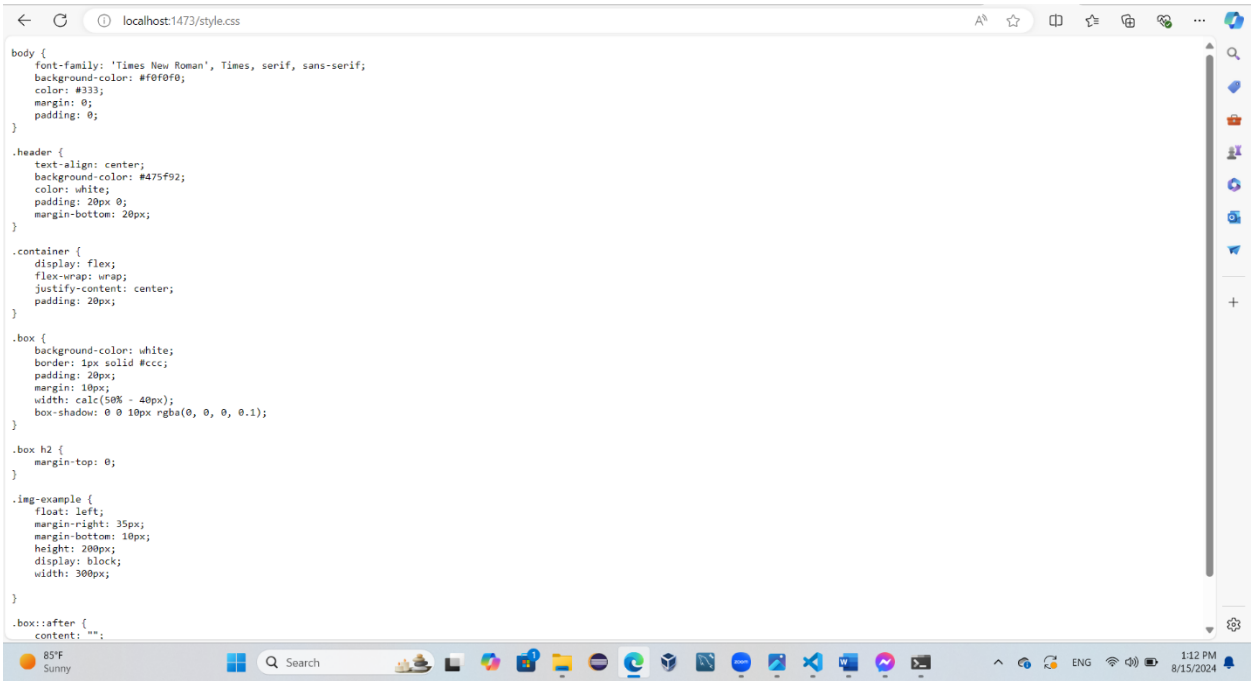


Figure 37: case 8 request

The HTTP request printed on the command Line

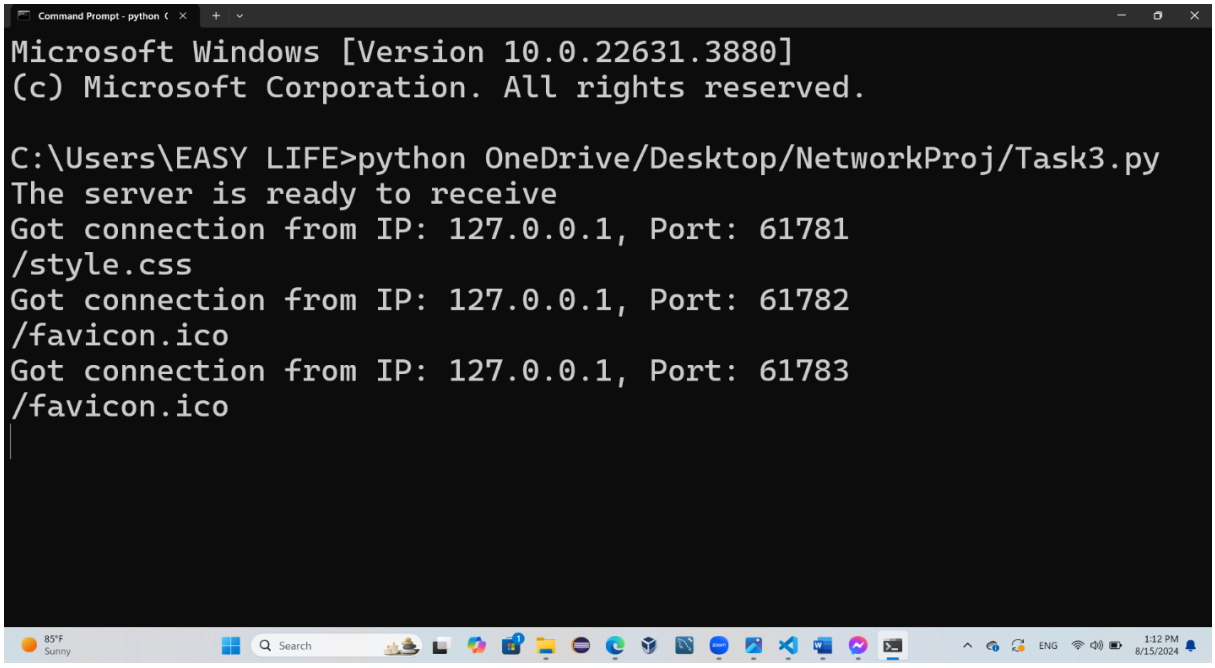


Figure 38: case 8 request on command line

Case9:

when the request localhost:1473/main_en.html the server should send the main_en.html

Note: main_en.html is just an example any another file .html will work on it.

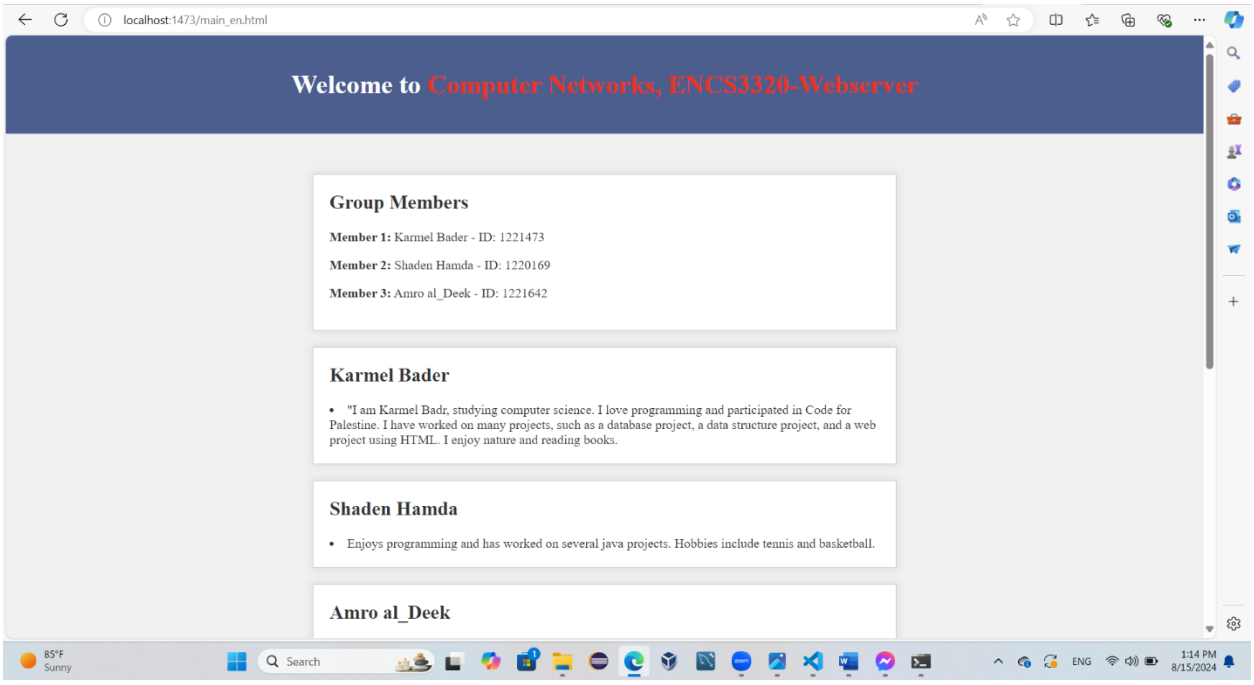


Figure 39: case 9 request

The HTTP request printed on the command Line

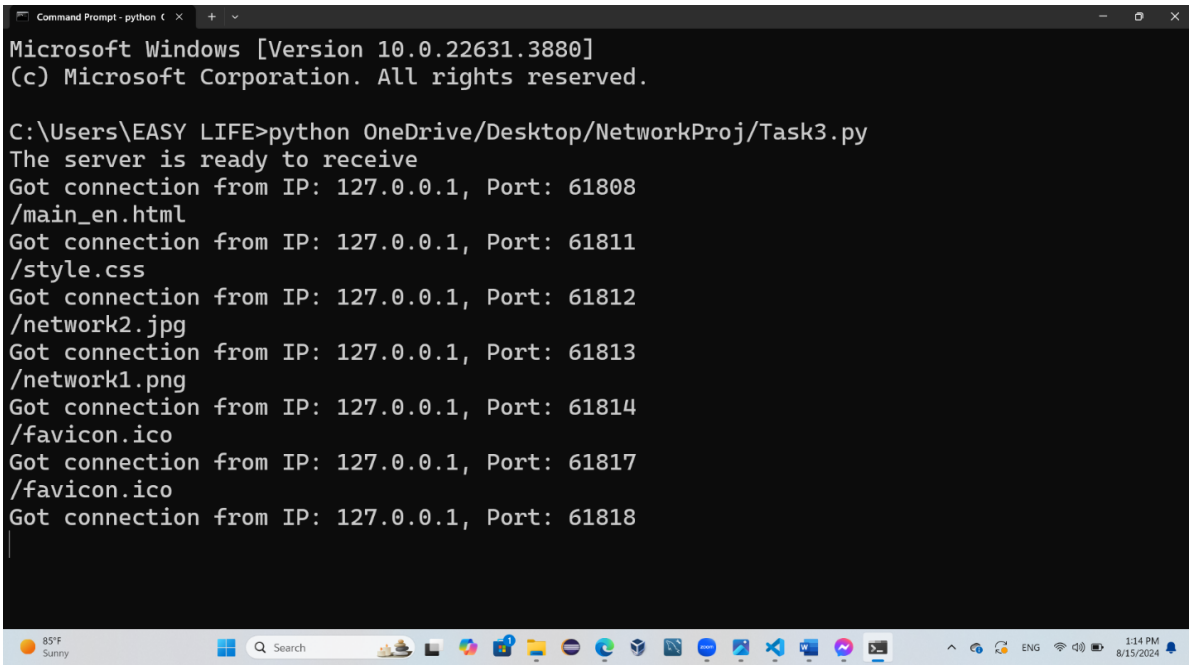


Figure 40: case 9 request on command line

mySiteSTDID.html file to get an image if the path is correct :

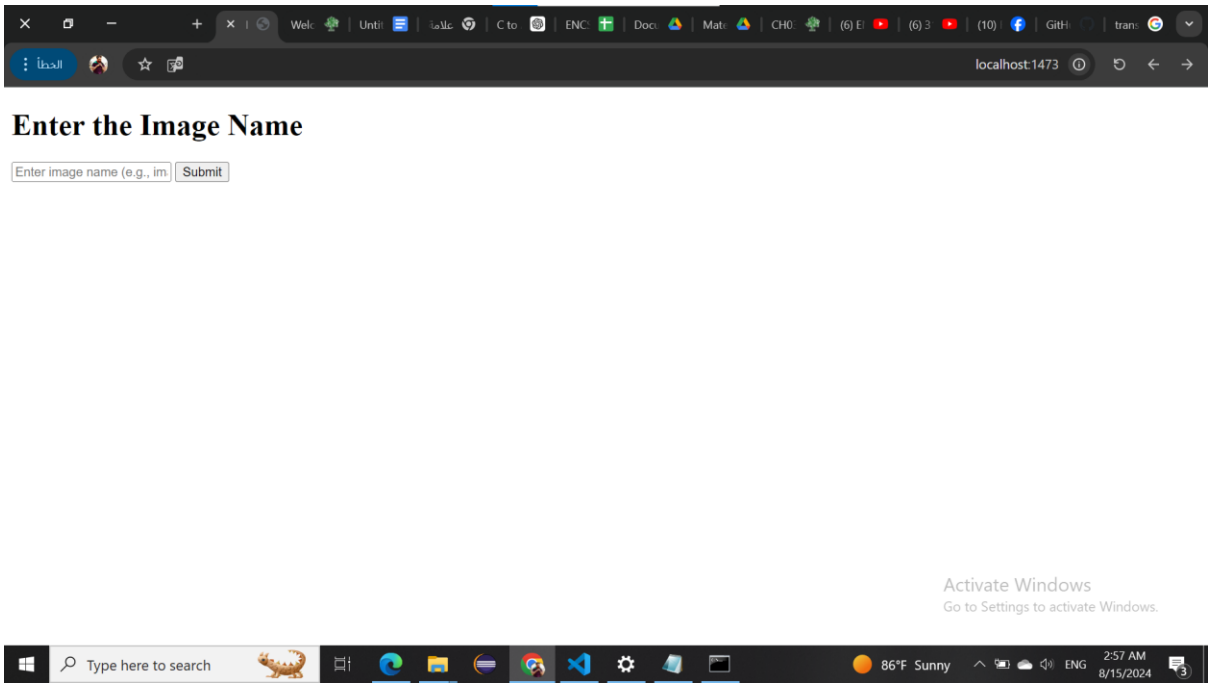


Figure 41: mySiteSTDID.html running output

Enter a correct path to show the image :

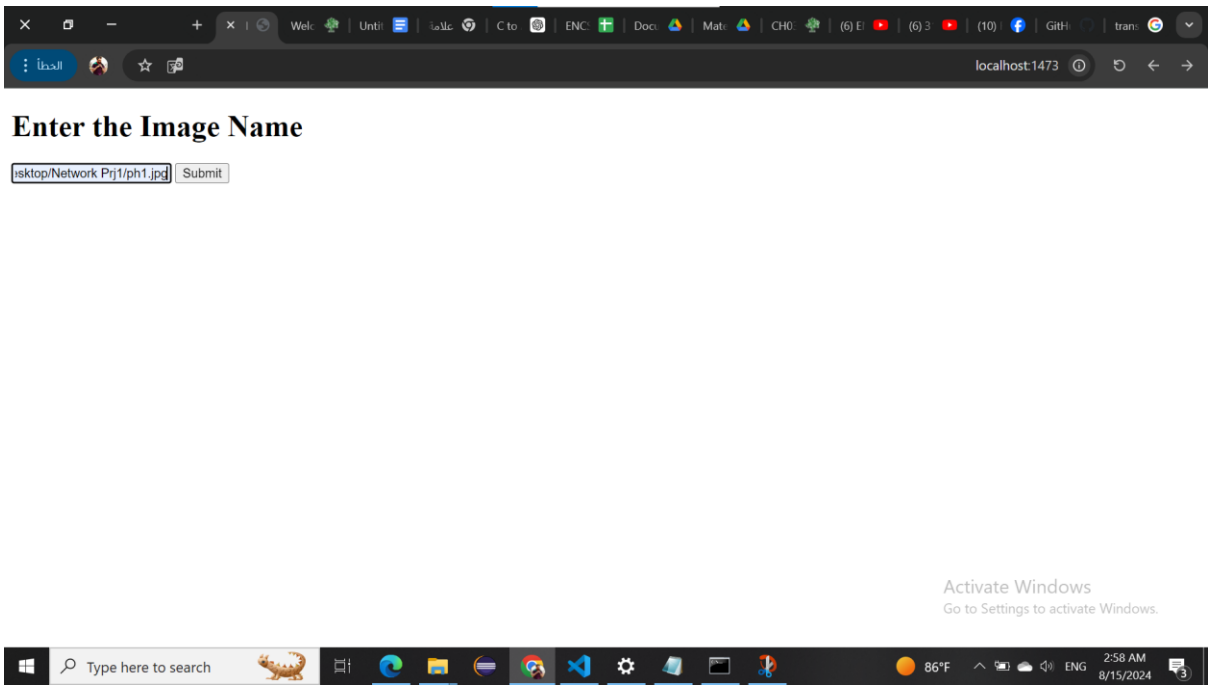


Figure 42: mySiteStDID.html running output with correct path

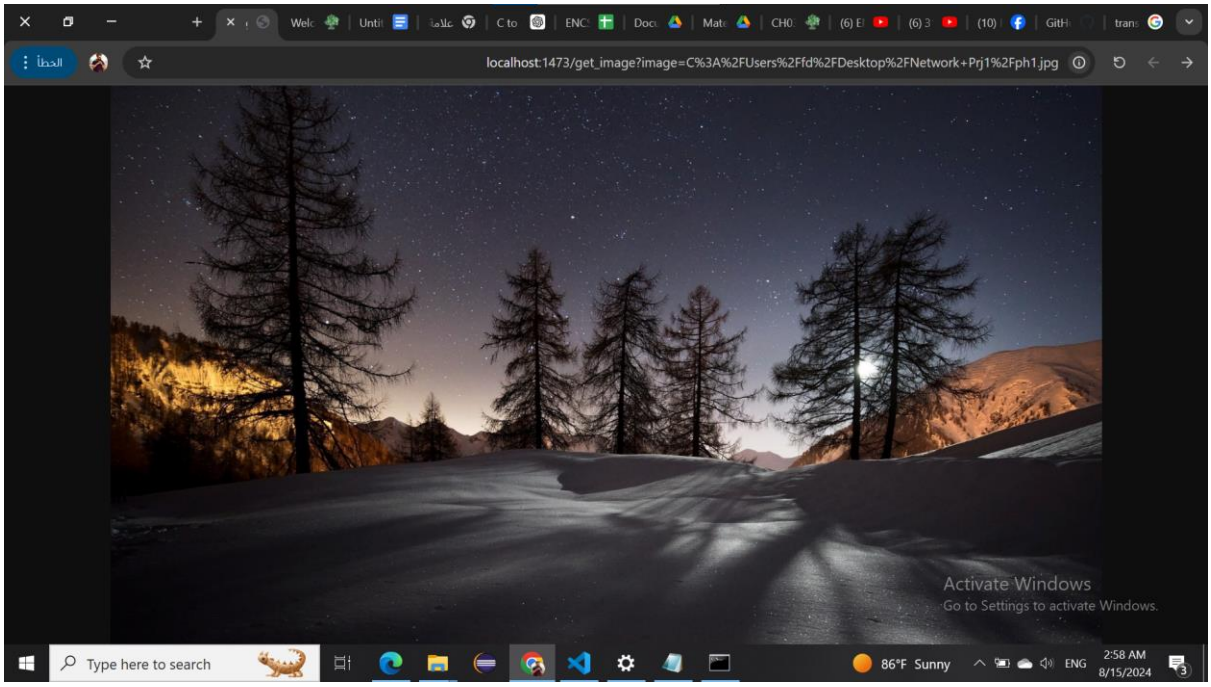


Figure 43: output of correct path

Enter a **wrong** path :

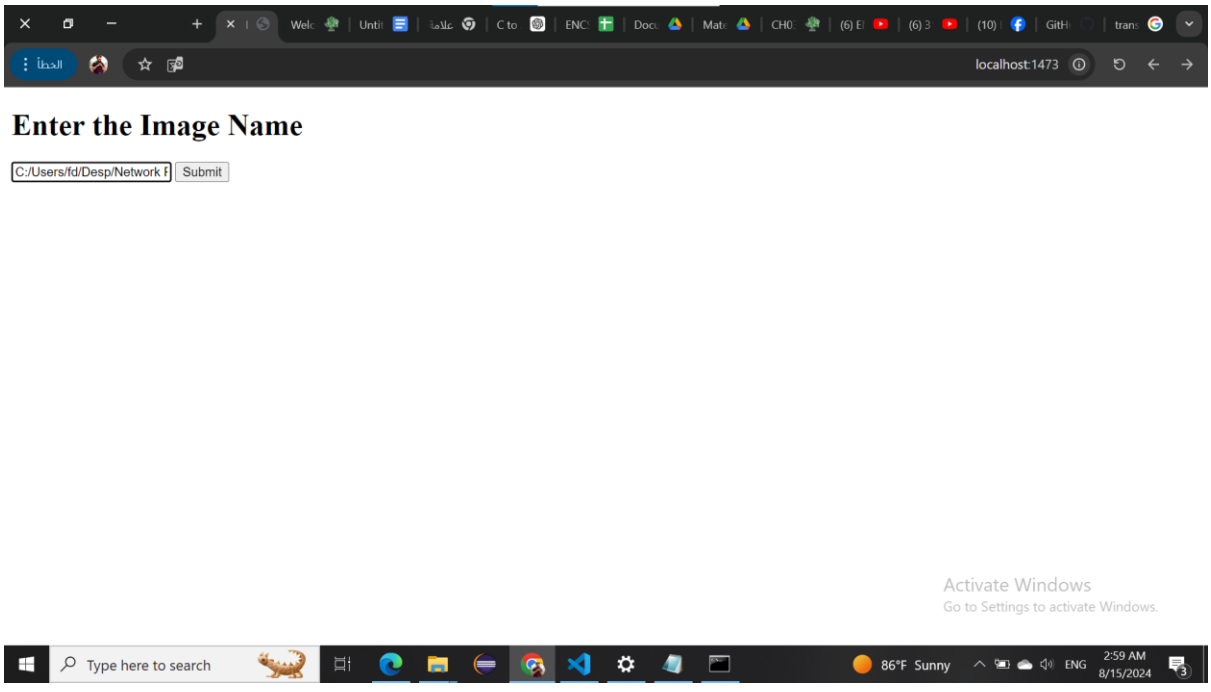


Figure 44: mySiteStDID.html running output with wrong path

Because the path is **wrong** or the image is **not exists** is shows a simple html file like :

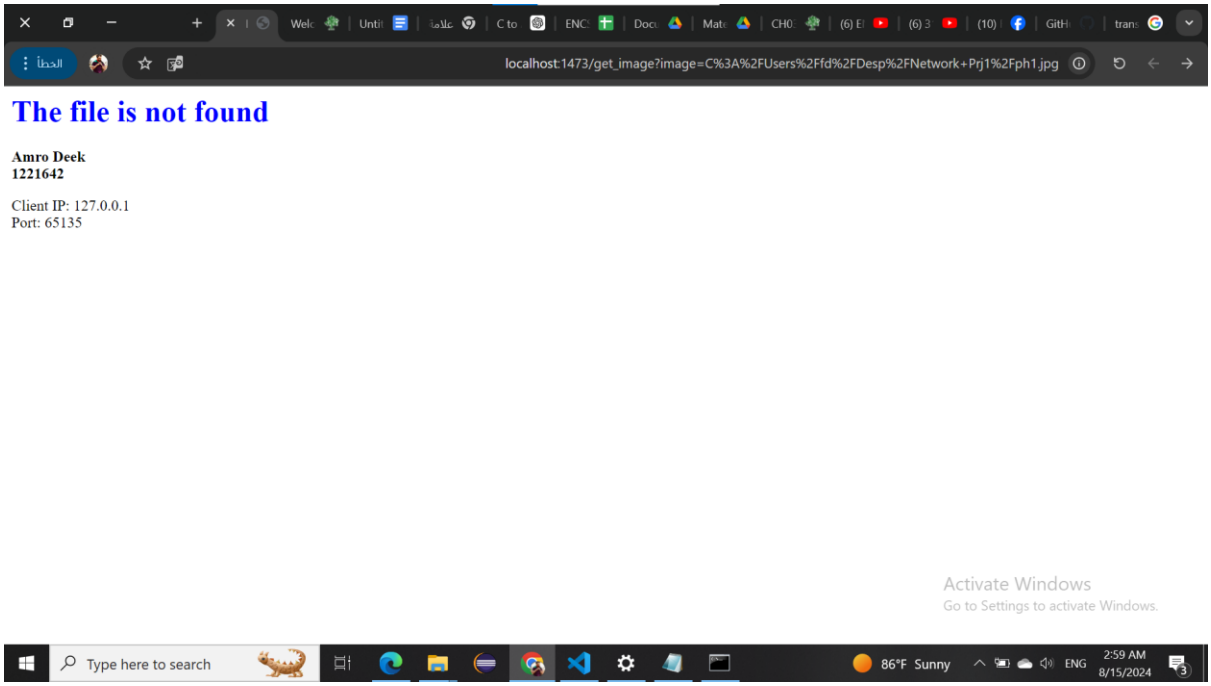


Figure 45: output of wrong path

the HTTP requests on the terminal window (command line window):

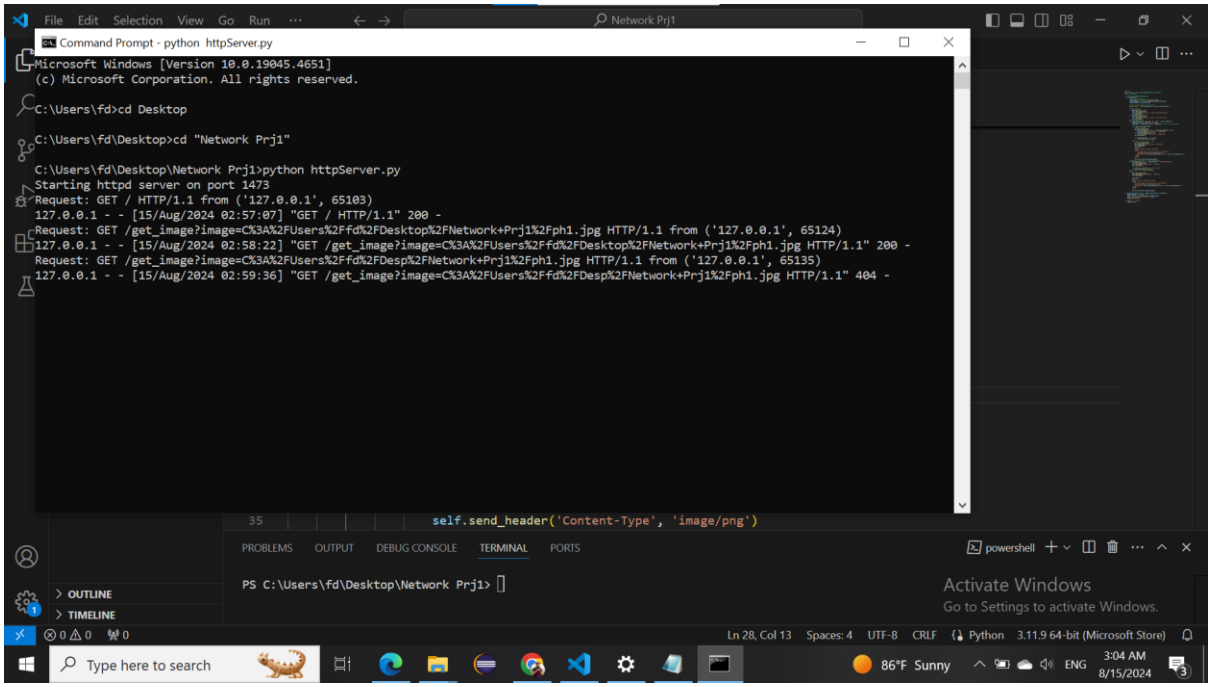


Figure 46: http request on command line

If the request is /so then redirect to stackoverflow.com website :

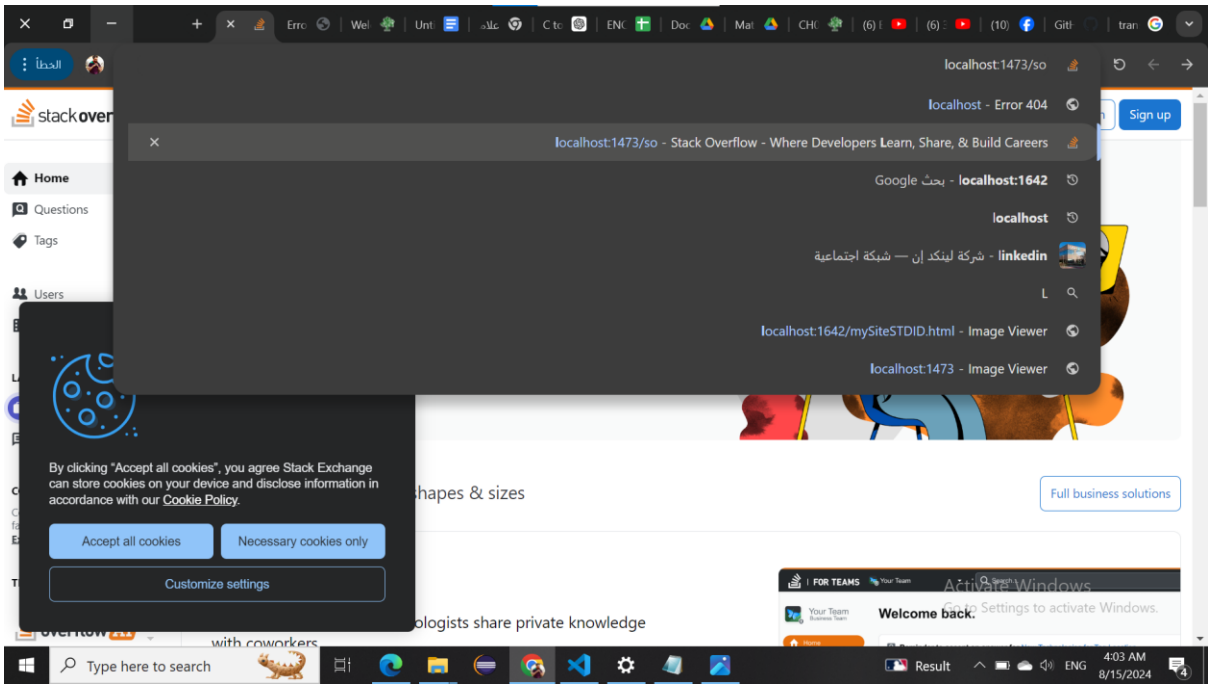


Figure 47: output of /so request

If the request is /itc then redirect to itc.birzeit.edu website:

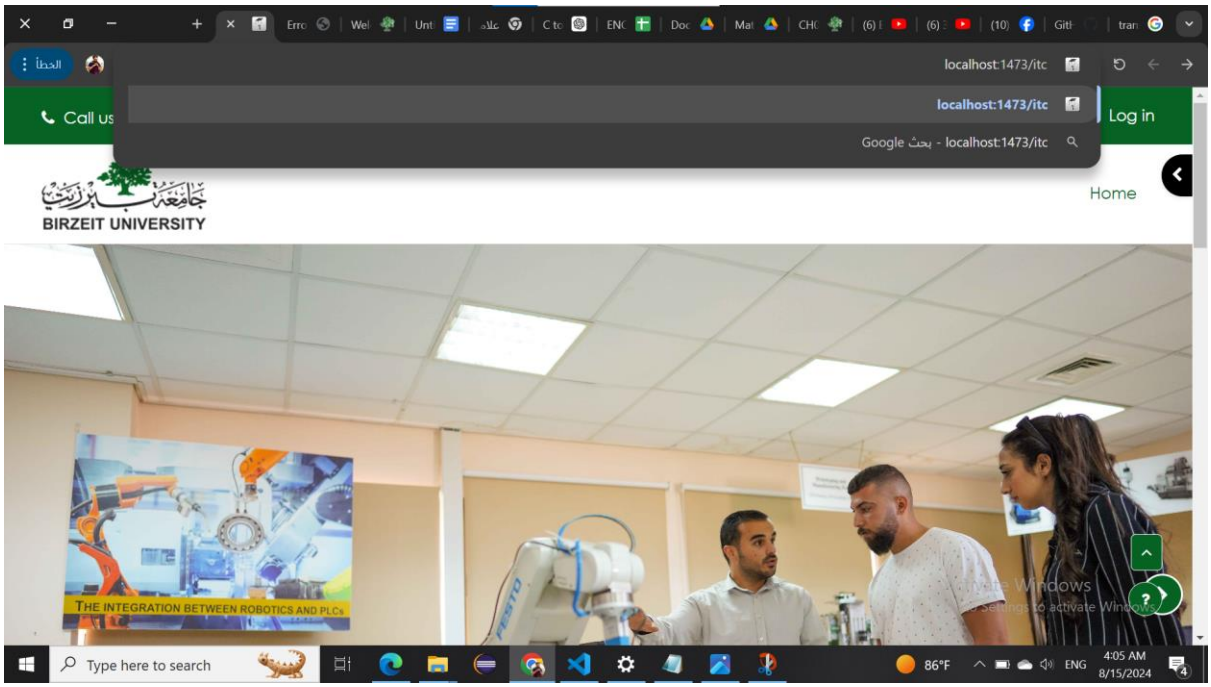


Figure 48: output of /itc request

Test the project from a browser on a different computer

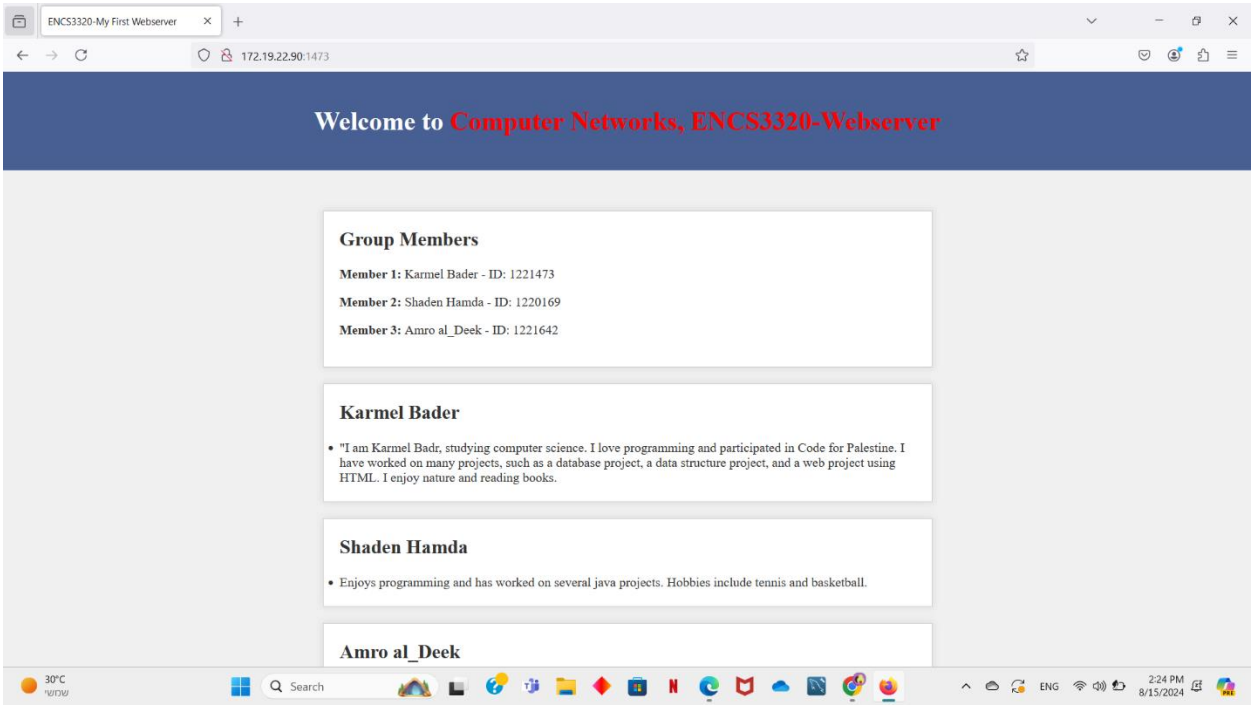
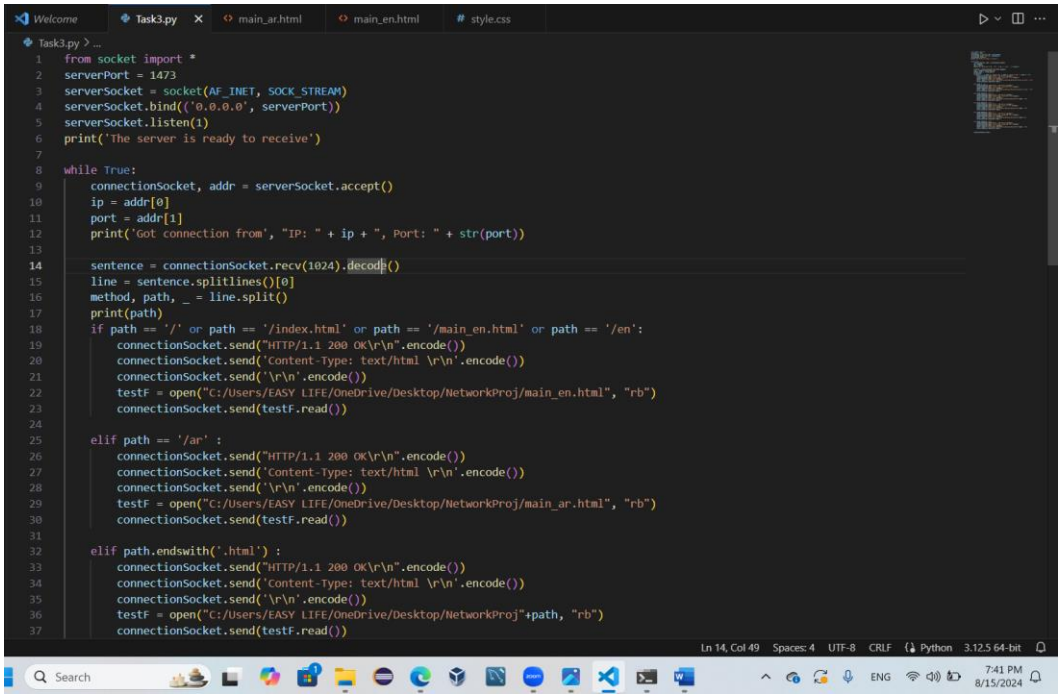


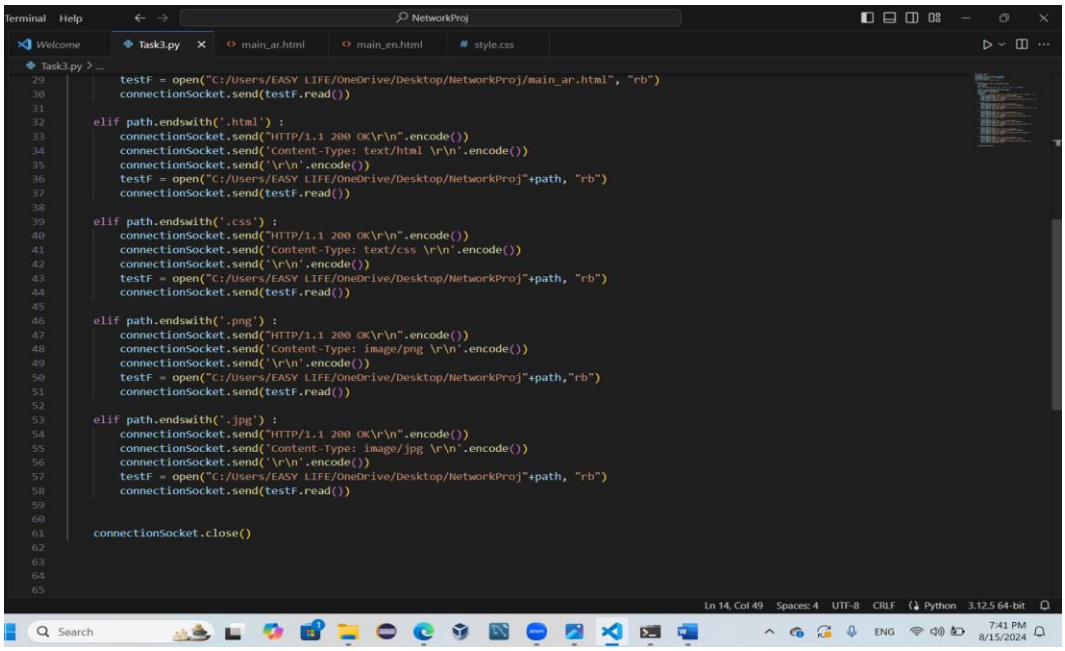
Figure 49: output of testing the project on other computer

Task3 Code: here is Task3.py code which handle the cases of the request, the requested path is checked by taking the first line of the request then splitting it and then comparing the different cases of the path to output the required one (.html, .jpg … etc).



```python
from socket import *
serverPort = 1473
serverSocket = socket(AF_INET, SOCK_STREAM)
serverSocket.bind(('0.0.0.0', serverPort))
serverSocket.listen(1)
print('The server is ready to receive')

while True:
    connectionSocket, addr = serverSocket.accept()
    ip = addr[0]
    port = addr[1]
    print('Got connection from', "IP: " + ip + ", Port: " + str(port))

    sentence = connectionSocket.recv(1024).decode()
    line = sentence.splitlines()[0]
    method, path, _ = line.split()
    print(path)
    if path == '/' or path == '/index.html' or path == '/main_en.html' or path == '/en':
        connectionSocket.send("HTTP/1.1 200 OK\r\n".encode())
        connectionSocket.send('Content-Type: text/html \r\n'.encode())
        connectionSocket.send('\r\n'.encode())
        testF = open("C:/Users/EASY LIFE/OneDrive/Desktop/NetworkProj/main_en.html", "rb")
        connectionSocket.send(testF.read())

    elif path == '/ar' :
        connectionSocket.send("HTTP/1.1 200 OK\r\n".encode())
        connectionSocket.send('Content-Type: text/html \r\n'.encode())
        connectionSocket.send('\r\n'.encode())
        testF = open("C:/Users/EASY LIFE/OneDrive/Desktop/NetworkProj/main_ar.html", "rb")
        connectionSocket.send(testF.read())

    elif path.endswith('.html') :
        connectionSocket.send("HTTP/1.1 200 OK\r\n".encode())
        connectionSocket.send('Content-Type: text/html \r\n'.encode())
        connectionSocket.send('\r\n'.encode())
        testF = open("C:/Users/EASY LIFE/OneDrive/Desktop/NetworkProj"+path, "rb")
        connectionSocket.send(testF.read())
```

Figure 50: Task3.py code_1



```python
        testF = open("C:/Users/EASY LIFE/OneDrive/Desktop/NetworkProj/main_ar.html", "rb")
        connectionSocket.send(testF.read())

    elif path.endswith('.html') :
        connectionSocket.send("HTTP/1.1 200 OK\r\n".encode())
        connectionSocket.send('Content-Type: text/html \r\n'.encode())
        connectionSocket.send('\r\n'.encode())
        testF = open("C:/Users/EASY LIFE/OneDrive/Desktop/NetworkProj"+path, "rb")
        connectionSocket.send(testF.read())

    elif path.endswith('.css') :
        connectionSocket.send("HTTP/1.1 200 OK\r\n".encode())
        connectionSocket.send('Content-Type: text/css \r\n'.encode())
        connectionSocket.send('\r\n'.encode())
        testF = open("C:/Users/EASY LIFE/OneDrive/Desktop/NetworkProj"+path, "rb")
        connectionSocket.send(testF.read())

    elif path.endswith('.png') :
        connectionSocket.send("HTTP/1.1 200 OK\r\n".encode())
        connectionSocket.send('Content-Type: image/png \r\n'.encode())
        connectionSocket.send('\r\n'.encode())
        testF = open("C:/Users/EASY LIFE/OneDrive/Desktop/NetworkProj"+path,"rb")
        connectionSocket.send(testF.read())

    elif path.endswith('.jpg') :
        connectionSocket.send("HTTP/1.1 200 OK\r\n".encode())
        connectionSocket.send('Content-Type: image/jpg \r\n'.encode())
        connectionSocket.send('\r\n'.encode())
        testF = open("C:/Users/EASY LIFE/OneDrive/Desktop/NetworkProj"+path, "rb")
        connectionSocket.send(testF.read())


    connectionSocket.close()
```

Figure 51: Task3.py code_2

Here is the English version of the html page, which contains the group members information and two images also two links

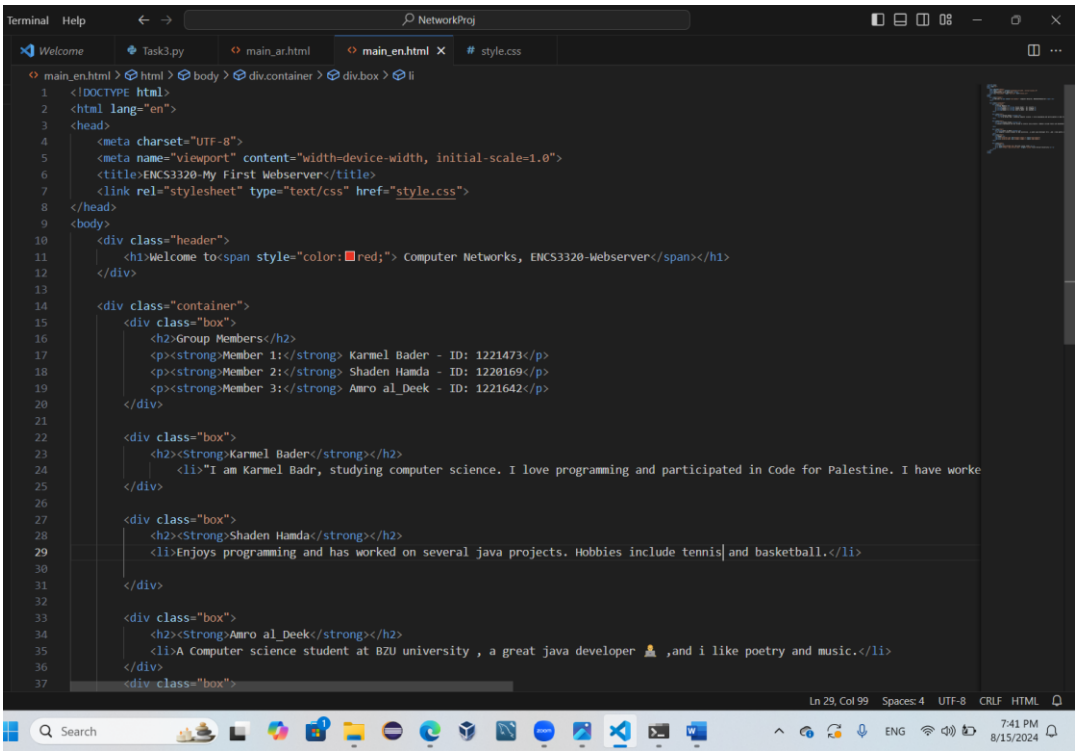


Figure 52: main_en.html code_1

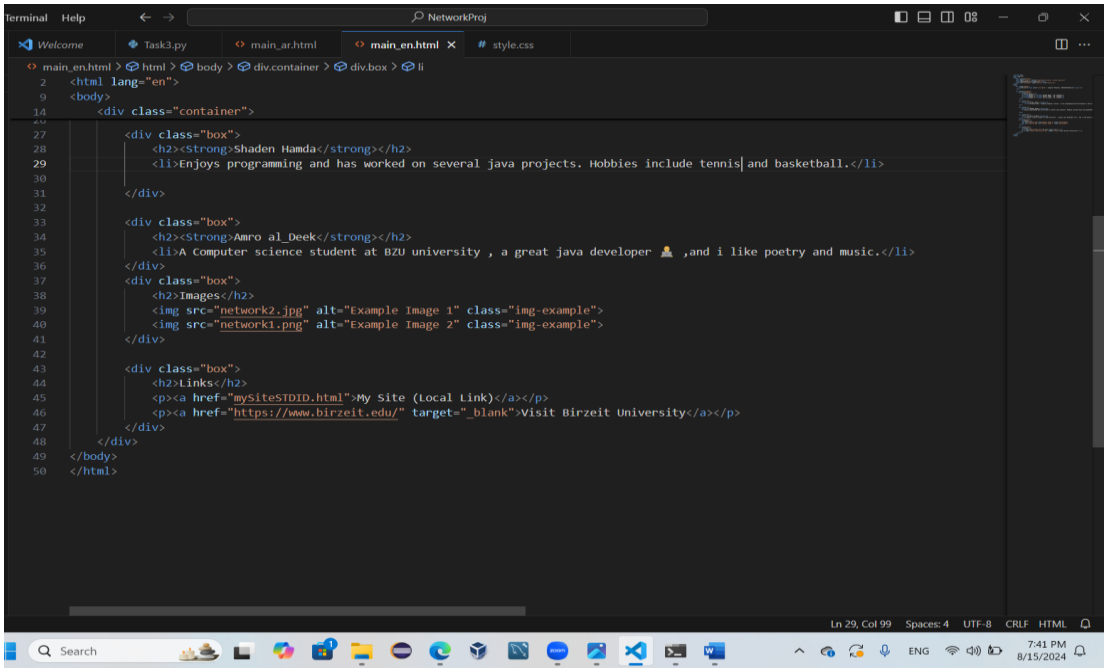


Figure 53: main_en.html code_2

Here is the css file which confirm the style of the html files (both English and Arabic versions).

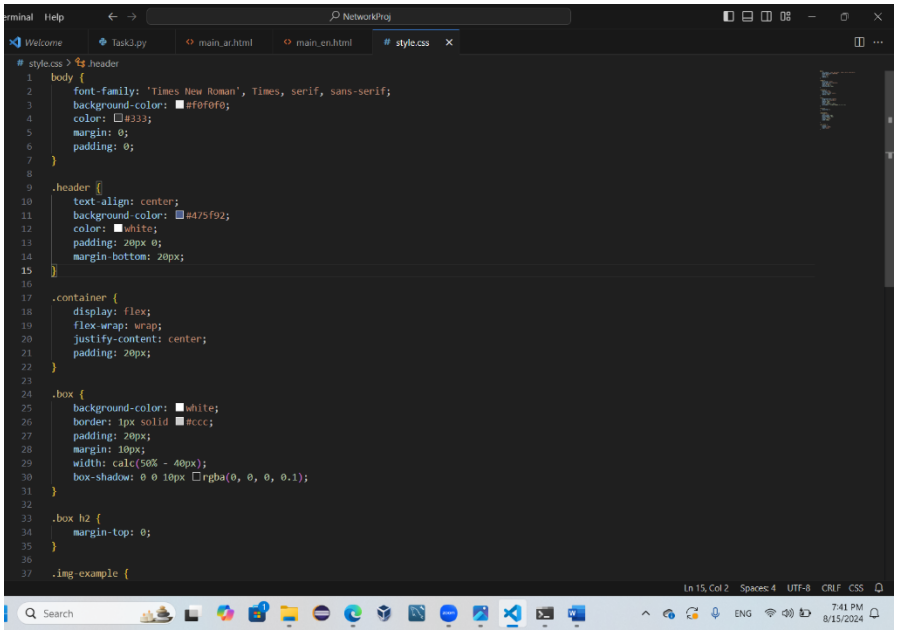


Figure 54: style.css code_1

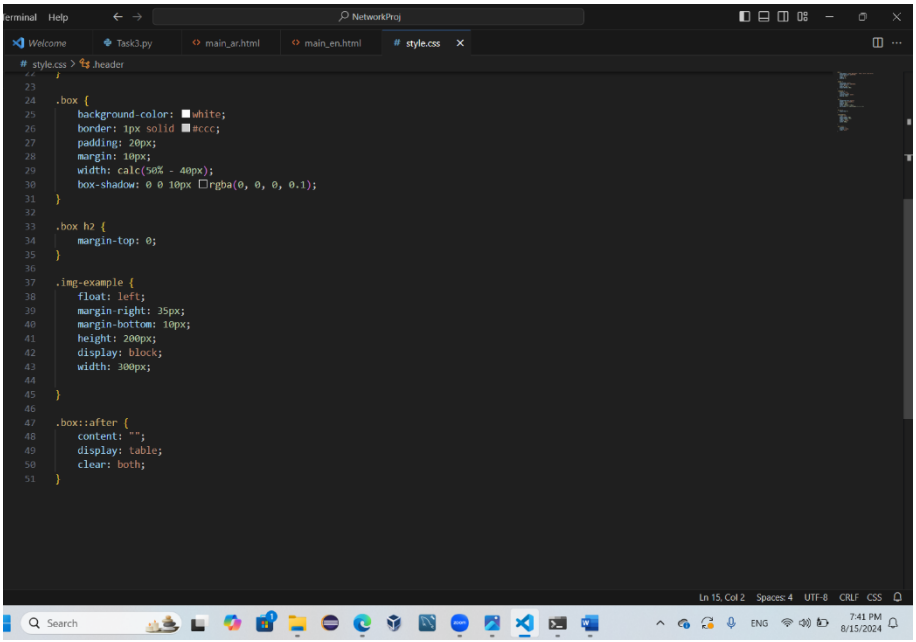


Figure 55: style.css code_2

mySiteSTDID.html code , we use the GET method (for sending data to server)

include user data in URL field of the HTTP GET request message.

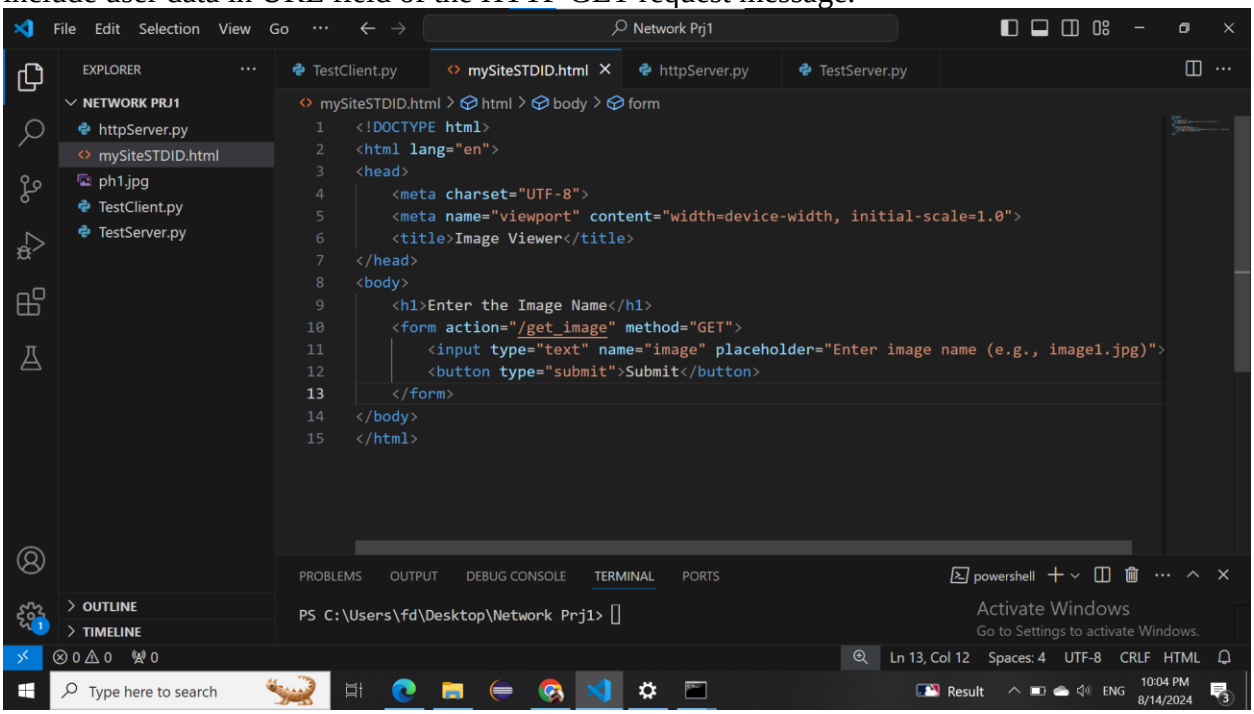


Figure 56: mySiteSTDID.html code_1

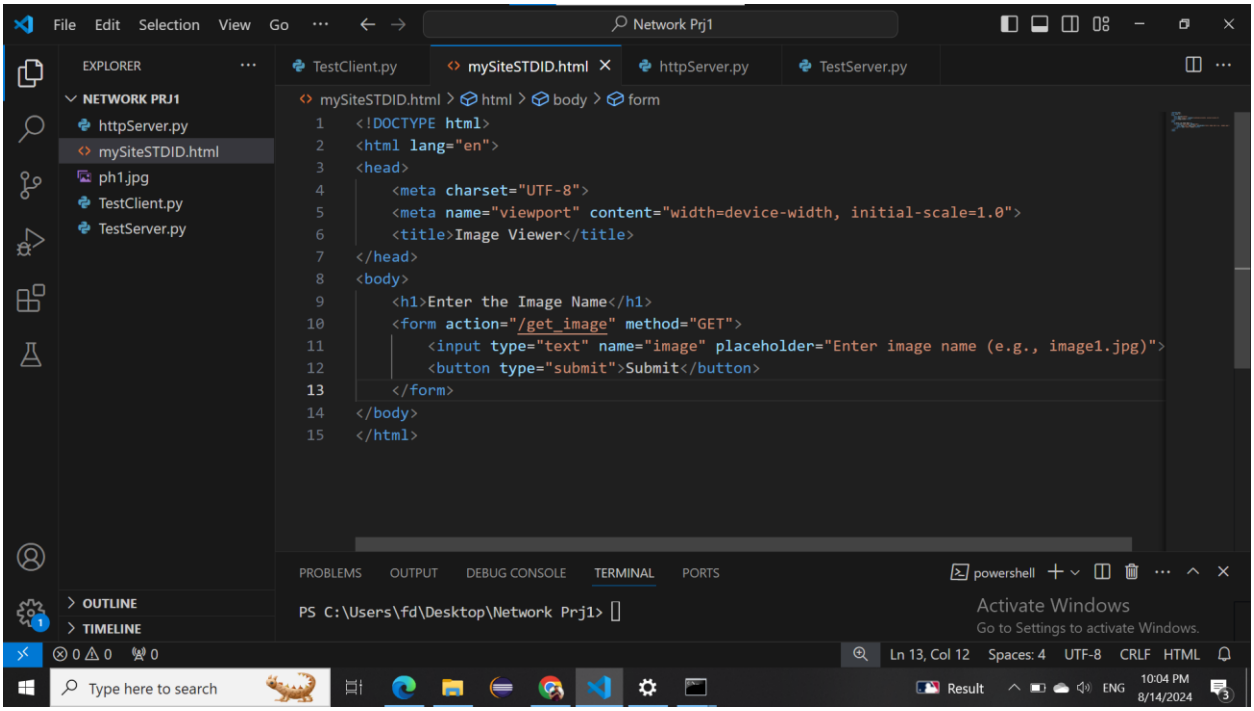


Figure 57: mySiteSTDID.html code_2

and **httpServer.py** code , This Python script sets up a basic HTTP server that handles specific GET requests. It can redirect to predefined URLs, serve images from a directory based on user input, and display a custom HTML page if the requested image is not found. The server also logs each request and handles 404 errors with a user-defined response.we used port # 1473 (karmel's ID because my ID 1642 does not work , it seems to be reserved for something on my computer).

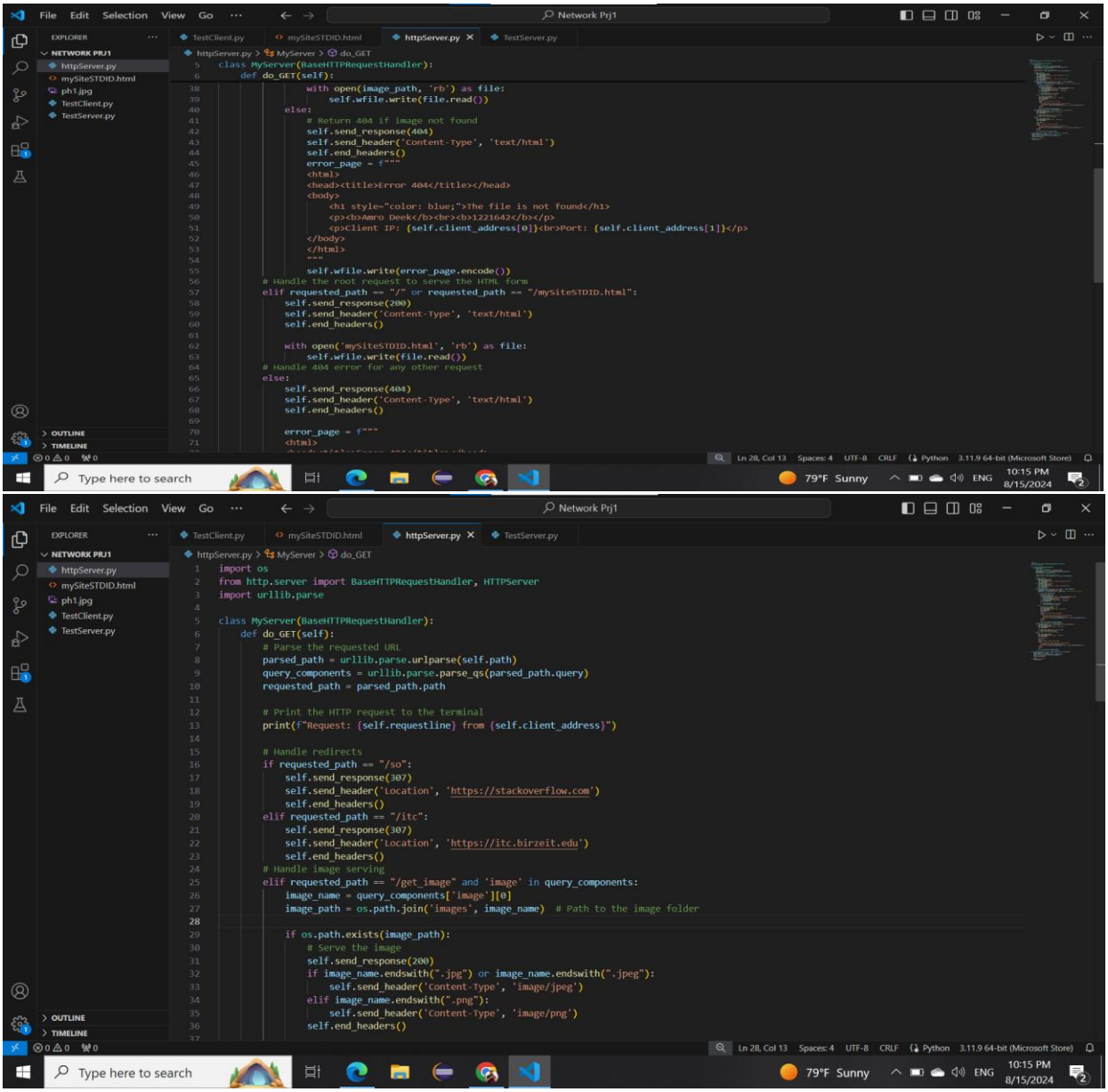


Figure 58: httpSerever.py code_1

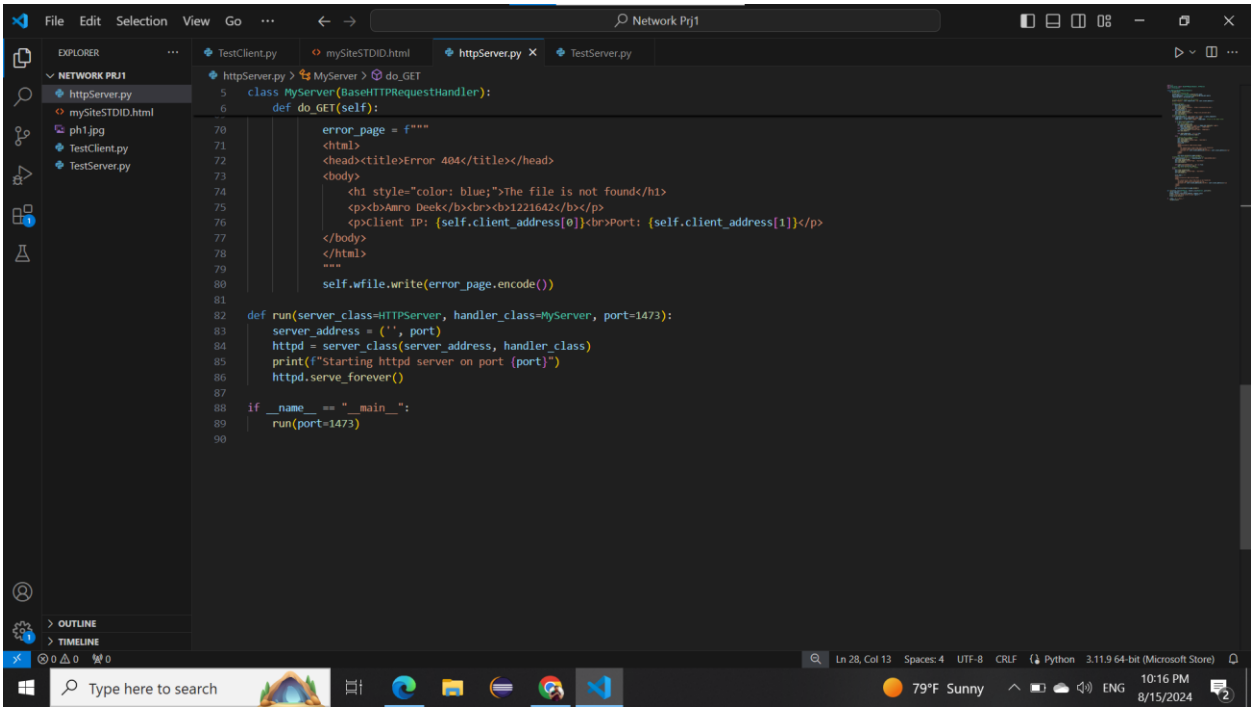
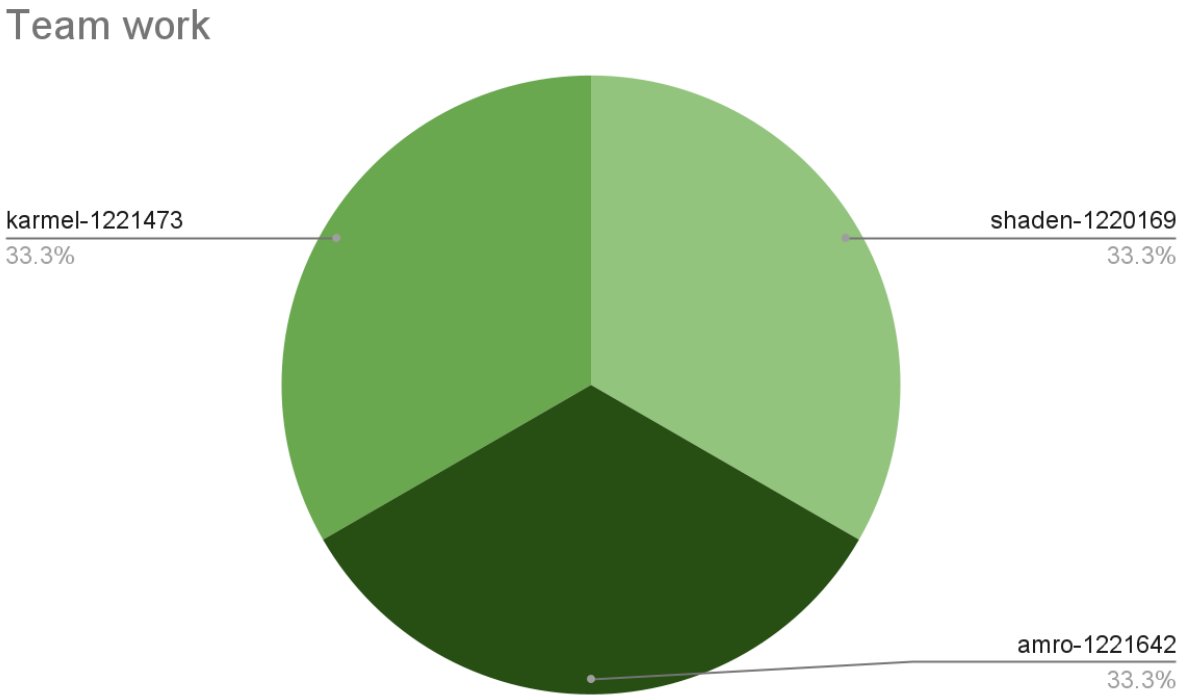


Figure 59: httpServer code_2

Team Work



Everyone's job:

Shaden: Task 1(Q2,Q4),Task 2 (Q2)
Karmel: Task 1(Q1,Q3),Task 3 (to point 6)
Amro: Task 2(Q1) ,Task 3(from 7 to 11)

Issues and Limitations Part

Web Page Not Functioning on Another Device We have encountered an issue where the web page is not functioning as expected on another device. While the page works correctly on the original device, it fails to operate or displays errors when accessed from a different one. This problem may be related to compatibility, browser settings, or device-specific configurations.

Solution

the solution was to configure the server, making sure that the firewall is set up correctly, and determining whether we want to access the server locally or globally

References :

What are Traceroute, Ping, Telnet and Nslookup commands? - Hosting - Namecheap.com

Hurricane Electric BGP Toolkit (he.net)