Faculty of Engineering and Technology

Electrical and Computer Engineering Department

ENCS3320 - Computer Network

First Project Report

Instructor Name : Ibrahim Nemer

Sections:2

Group : 36

Alaa Moqade _ 1211910

Leyan Buirat _1211439

Abstract :

This project consists of three parts. The first part is to run some commands on the terminal; ping a device in the same network, ping www.cornell.edu, tracert and nslookup it. The second part is implementing server and client application both for TCP. The server should listen on port 1439. The last part is to implement a simple but a complete web server in python that is listening on port 1910. Both html and CSS are used for the design of our website.

Table of contents:

Table of figure:

1.Theory

1.1 Overview of Network Diagnostic Commands ping

In networking diagnostics, ping is a utility that tests connectivity by sending ICMP Echo Requests to a target and measuring the response time, thereby determining reachability and latency. Tracert (or traceroute in Unix/Linux) traces the route packets take to a destination by incrementally increasing the Time-to-Live (TTL) value of packets, revealing each hop along the path and associated delays. Nslookup queries DNS servers to resolve domain names to IP addresses and vice versa, providing DNS record details essential for diagnosing domain resolution issues. **Telnet** establishes a remote command-line interface over TCP, typically used for testing connectivity to services on specific ports, though it's largely replaced by more secure protocols like SSH. To analyze network traffic, Wireshark can capture DNS messages, allowing for detailed inspection of DNS queries and responses. Autonomous System (AS) information, including AS number, IP prefixes, peers, and Tier 1 ISP, can be retrieved using online tools such as BGPView, which provides insights into the routing and administrative details of networks.

The ping command sends ICMP requests, also known as pings, to a remote device and returns the response. It is the most popular command-line tool for testing connectivity between two network devices[1].

To do a ping test, execute the command followed by the IP Address or domain name of the remote host you want to test[1].

The ping command tests connectivity between two network devices by sending an **ICMP Echo Request** to the remote device. If the remote device receives the request, it sends an **ICMP Echo Reply** confirming the connectivity[1].

If the ping is successful, you will see the ICMP Echo Reply from the remote IP, as shown in the following screenshot[1].

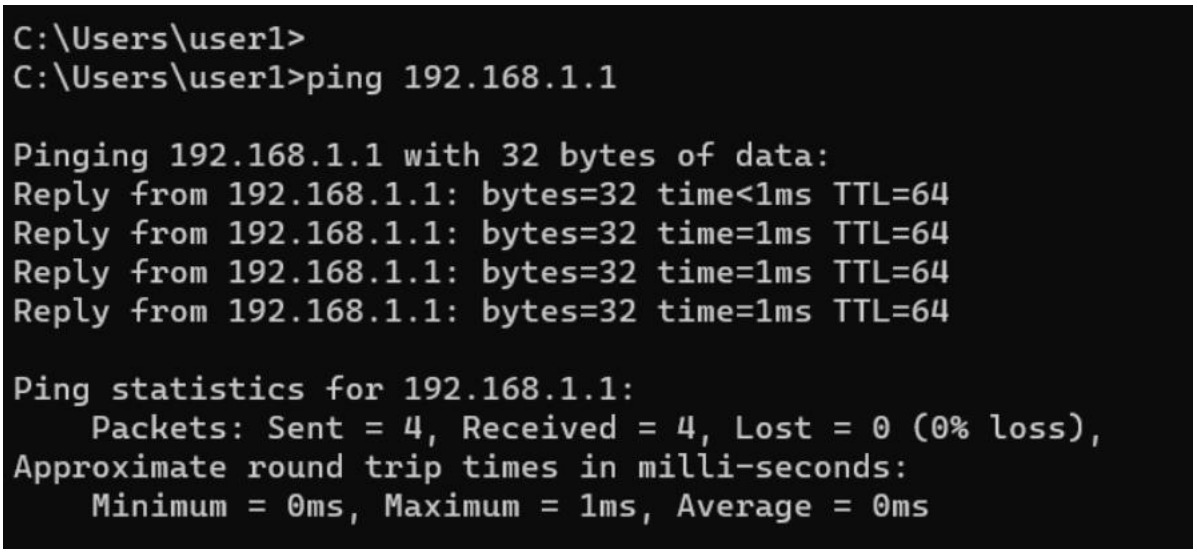


Figure 1:ping command

Under the statistics section, you will see the number of packets that were sent, received, and lost.

When the ping command fails, you will get one of several error messages like Destination host unreachable or Request timed out.

- **Destination host unreachable** - No routes to the destination. Your router does not know how to reach the remote computer you specified.
- **Ping request could not find host** - This is a DNS error. Your DNS Server can't resolve the domain name to the IP address. You probably entered an incorrect domain name.
- **Request timed out** - Your computer did not receive ICMP Echo Reply from the targeted device. This doesn't necessarily mean that the host is down. It might be because the remote computer blocks pings.

Continuous Ping Command

The `-t` is the continuous ping command option. It sends ICMP Echo Requests to the remote computer until you manually stop the command by pressing `Ctrl + C`.

```
C:\Users\user1>
C:\Users\user1>ping -t 192.168.1.1

Pinging 192.168.1.1 with 32 bytes of data:
Reply from 192.168.1.1: bytes=32 time=1ms TTL=64
Reply from 192.168.1.1: bytes=32 time<1ms TTL=64
Reply from 192.168.1.1: bytes=32 time=1ms TTL=64
Reply from 192.168.1.1: bytes=32 time<1ms TTL=64
Reply from 192.168.1.1: bytes=32 time=1ms TTL=64
Reply from 192.168.1.1: bytes=32 time<1ms TTL=64
Reply from 192.168.1.1: bytes=32 time=1ms TTL=64
```

Figure 2Continuous Ping Command

You can view the statistics while the test is running by pressing Ctrl + Pause key combination.

```
C:\Users\user1>
C:\Users\user1>ping -t 192.168.1.1

Pinging 192.168.1.1 with 32 bytes of data:
Reply from 192.168.1.1: bytes=32 time=1ms TTL=64
Reply from 192.168.1.1: bytes=32 time=1ms TTL=64
Reply from 192.168.1.1: bytes=32 time=1ms TTL=64
Reply from 192.168.1.1: bytes=32 time=1ms TTL=64

Ping statistics for 192.168.1.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1ms, Maximum = 1ms, Average = 1ms
Control-Break
Reply from 192.168.1.1: bytes=32 time=1ms TTL=64
Reply from 192.168.1.1: bytes=32 time<1ms TTL=64
Reply from 192.168.1.1: bytes=32 time=1ms TTL=64
```

Figure 3:ping -t IP

The -n option allows you to specify the number of pings to send. The Windows default is 4 ICMP requests.

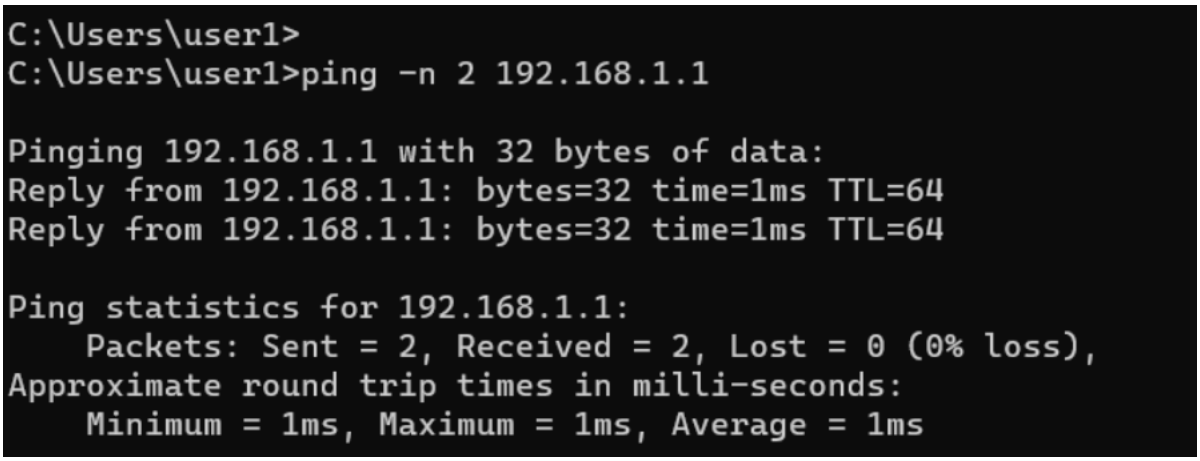


Figure 4: -n 2 IP

1.1.1. Ping a Device in the Same Network

The command `ping 192.168.0.108 is used to test the connectivity between two devices within the same local area network (LAN). By sending ICMP Echo Request packets to the device with the IP address 192.168.0.108, the command checks whether the device responds and measures the round-trip time of the packets. Successful replies confirm that both devices can communicate over the LAN without issues, indicating that the network connection is functional and properly configured.

1.1.2. Ping www.ox.ac.uk

Using the command `ping www.ox.ac.uk` tests connectivity to the Oxford University web server. This command sends ICMP Echo Requests to the domain and measures the round-trip time for each packet. The results provide insights into the network performance between the source and the remote server, including latency, which is crucial for evaluating the responsiveness and reliability of the connection to the external server.

1.1.3. Location of the Server

Ping results show the round-trip time in milliseconds (ms), reflecting the latency between the source and destination. While this time can offer a general idea of the distance to the server, it does not directly indicate the server's physical location. To pinpoint the exact geographical location of the server, additional tools such as IP geolocation databases are necessary, which map IP addresses to physical locations based on various data sources.

1.1.4_Tracert www.ox.ac.uk

The tracert command is a network diagnostic tool that is used to trace the route taken by packets from your PC to a target IP address. It shows the number of hops between your PC and the target, along with details such as IP addresses and response times for each hop. This information can be helpful in troubleshooting network connectivity issues and identifying network latency.[2]

Use case 1: Trace a route Code: tracert IP

Motivation: Tracing a route can provide valuable information about the path taken by packets between your PC and the target IP address. This can help identify any network issues or bottlenecks that may be affecting connectivity or performance.

Explanation: Replace "IP" with the IP address or hostname of the target. The 'tracert' command will send a series of packets with increasing time-to-live (TTL) values to the target and it will display the route taken by the packets[2].

Example output:

```
  1    <1 ms    <1 ms    <1 ms  192.168.1.1
  2    10 ms    10 ms    11 ms  10.10.100.1
  3    10 ms    9 ms     9 ms   203.0.113.1
  4    12 ms    11 ms    11 ms  203.0.113.254
  5    25 ms    24 ms    24 ms  203.0.112.201
  6    25 ms    24 ms    24 ms  203.0.112.202
  7    26 ms    25 ms    25 ms  203.0.112.203
  ...
```

Figure 5:Example output of use tracet IP[2]

The 'tracert' command is a useful network diagnostic tool for tracing the route between your PC and a target IP address. It provides valuable information about network connectivity, latency, and potential issues along the path. By understanding the different use cases of the 'tracert' command and the available command-line arguments, you can effectively troubleshoot network problems and optimize network performance[2].

1.1.5 Nslookup www.ox.ac.uk

When troubleshooting DNS issues, it is useful to have access to Domain Name System (DNS) records of a website. All mainstream operating systems have tools that enable users to query a web server and receive important information such as IP addresses and other pieces of domain-related information.

This article will introduce the nslookup command which is used for obtaining server records. It will also provide examples of the command's most popular options[3].

The **nslookup Syntax**: **nslookup** command can be used in two modes **interactive** and **non-interactive**. To initiate the **nslookup** interactive mode, type the command name only:

Nslookup For example, you can type a domain name and receive information about it.www.google.com

After **nslookup** outputs the information, it provides another prompt[3].

```
marko@test-main:~$ nslookup
> www.google.com
Server:         127.0.0.53
Address:        127.0.0.53#53

Non-authoritative answer:
Name:   www.google.com
Address: 142.250.180.196
Name:   www.google.com
Address: 2a00:1450:400d:80c::2004
>
```

Figure 6:After nslookup outputs[3]

1.2.6 Telnet www.ox.ac.uk

Telnet is a command-line tool that allows users to connect (TCP connection)to a remote host over a network using the telnet protocol. It is commonly used for troubleshooting network connectivity, testing network services, and accessing remote devices or services. Telnet has various use cases, and in this article, we will illustrate each of them with examples[4].

Use case 1: Telnet to the default port of a host

Motivation: This use case allows us to connect to the default port = 80 of a remote host using the telnet protocol. It is useful when we want to establish a telnet session with a host without specifying a specific port[4].

- Explanation:**telnet**: The telnet command.
- **host**: The hostname or IP address of the remote host.

Code example:telnet hosty

```
Trying 192.168.0.1...
Connected to example.com.
Escape character is '^]'.
```

Figure 7:output of talnet hosty[4]

Use case 2: Telnet to a specific port of a host

Code:

telnet ip_address port[4]

Motivation: This use case allows us to connect to a specific port of a remote host using the telnet protocol. It is useful when we want to establish a telnet session with a host on a non-default port.

Explanation[4]:

- **telnet**: The telnet command.
- **ip_address**: The IP address of the remote host.
- **port**: The port number on the remote host.

```
Trying 192.168.0.1...
Connected to example.com.
Escape character is '^]'.
```

Figure 8: output example of talent IP[4]

1.2.7 autonomous system (AS)

The Internet is a network of networks*, and autonomous systems are the big networks that make up the Internet. More specifically, an autonomous system (AS) is a large network or group of networks that has a unified routing policy. Every computer or device that connects to the Internet is connected to an AS[5].

Imagine an AS as being like a town's post office. Mail goes from post office to post office until it reaches the right town, and that town's post office will then deliver the mail within that town. Similarly, data packets cross the Internet by hopping from AS to AS until they reach the AS that contains their destination Internet Protocol (IP) address. Routers within that AS send the packet to the IP address.

Every AS controls a specific set of IP addresses, just as every town's post office is responsible for delivering mail to all the addresses within that town. The range of IP addresses that a given AS has control over is called their "IP address space."[5]

Most ASes connect to several other ASes. If an AS connects to only one other AS and shares the same routing policy, it may instead be considered a subnetwork of the first AS.

Typically, each AS is operated by a single large organization, such as an Internet service provider (ISP), a large enterprise technology company, a university, or a government agency.

A network is a group of two or more connected computers[5].

Figure 9 autonomous system (AS) is a large network or group of networks that has a unified routing policy[5]

An AS routing policy is a list of the IP address space that the AS controls, plus a list of the other ASes to which it connects. This information is necessary for routing packets to the correct networks. ASes announce this information to the Internet using the Border Gateway Protocol (BGP[5]).

IP, or the Internet Protocol, is indeed used for routing in that it specifies which destination each packet is going to. BGP is responsible for directing packets on the fastest route to their destination. Without BGP, IP packets would bounce around the Internet randomly from AS to AS, like a driver trying to reach their destination by guessing which roads to take[5].

2.1.4. Capturing DNS Messages with Wireshark

DNS or **Domain Name System abbreviated** as DNS is a system used to resolve domain names, IP addresses, different servers for e.g., FTP servers, game servers, active directories, etc., and keep

their records. Invented by Jon Postel and Paul Mockapetris in 1982, DNS has now become one of the most significant players in the modern-day web world[6].

DNS actually gives a mapping of the hostname of a network and its address. It has proved to ease human life manifold when one looks at its working and the service it offers. It helps users by translating the domain names into IP addresses, allowing them to surf the web without memorizing such complex IP codes. Coming on to **Wireshark**, which is an open-source packet analyzer and has been widely in use since its inception in the web world, to analyze packets received or sent in a network. We can use Wireshark to segment the DNS system and get a detailed look at it. The default port for DNS traffic in Wireshark is 53, and the protocol is UDP (**User Datagram Protocol**). After we start Wireshark, we can analyze DNS queries easily. We shall be following the below steps[6]:

- In the menu bar, Capture → Interfaces.
- Select a particular Ethernet adapter and click start.
- After this, browse to any web address and then return to Wireshark. Browsing would get packets captured and in Wireshark click the stop in the Capture menu to stop the capture.
- If you haven't got the packet list by now, you can access it using **Edit → Find Packets**. This will give you the packet list.
- Since we are going to analyze DNS we shall be studying only DNS packets and to get DNS packets, only you can apply DNS in the filters [6]

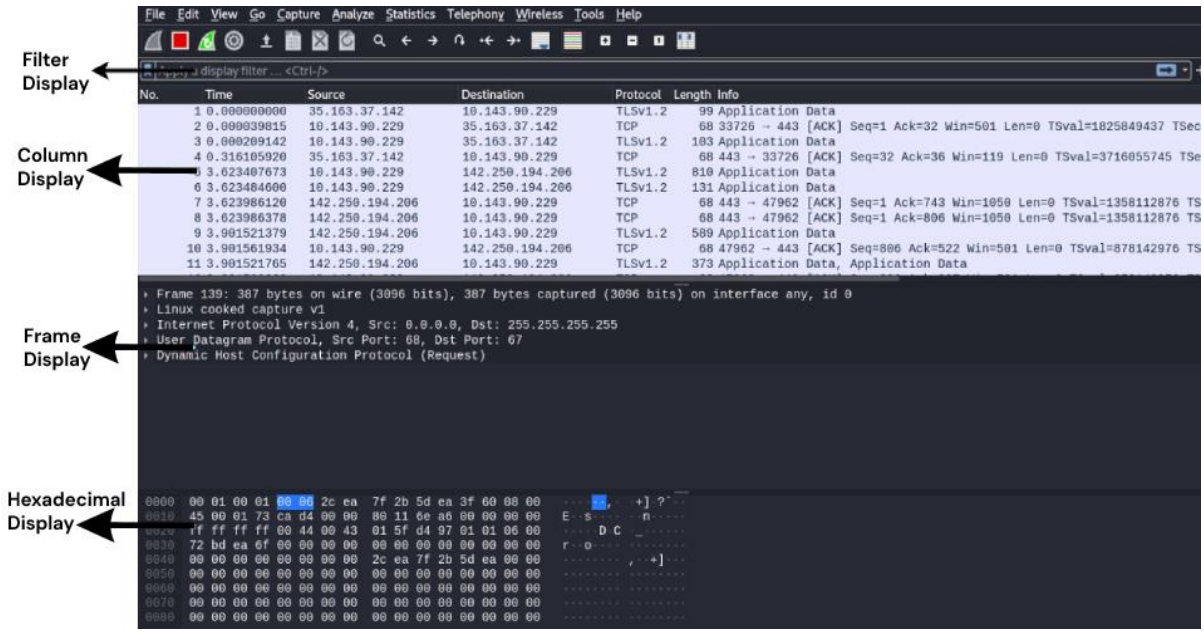


Figure 10:DNS in the filters[6]

You can have access to the DNS details of any packet by clicking the Domain Name System label in the frame detail section of the Wireshark window. You can have a look at different sections of the interface in the image above[6].

A basic **DNS response** has:

1. **Transaction Id**-for identification of the communication done.
2. **Flags**-for verification of response whether it is valid or not.
3. **Questions**-default is 1 for any request sent or received. It mainly denotes whether you have queried for something or not.
4. **Answers**-default is 0 if the response is sent, and it's 1 if received. If the received packet is viewed then the Answers section has the IP address of the desired domain name along with **Time to Live** which is basically a counter which expires after its allotted time[6].

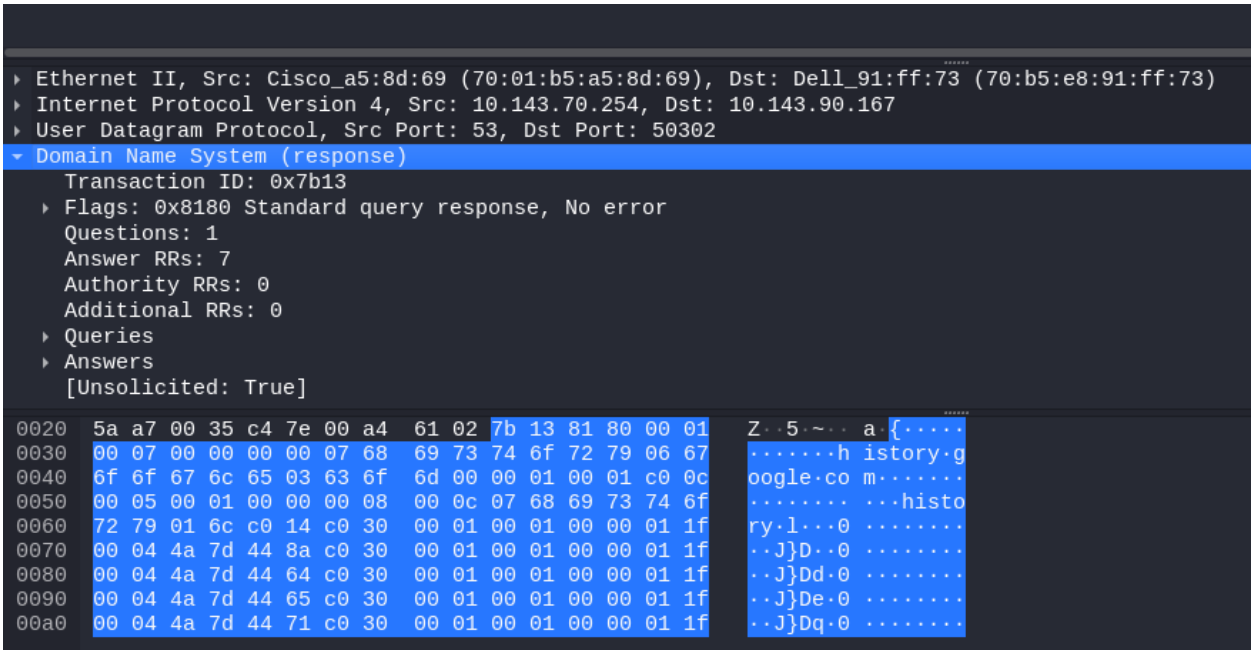


Figure 11: Domain name system

Besides, these, it has a **Queries** section which gives the subjective details of the communication. The queries section has the following:

1. **Name**: Domain name of the destination or web address to be reached or reached by in case of the received packet. This section further has its length, character by character under[**Name-Length**], and the count of words separated by separators, i.e., dot(.) under the name[**Labels**].
2. **Type**: which is '**A**' for IPv4(32 bits) and is '**AAAA**' for IPv6(128 bits).
3. **Class**: which is '**IN**' by default, which means an internet IP address has been asked for.

Captured packets are also stored in the local machine, We can also view our received packets in command prompt by typing the following instruction:

ipconfig /displaydns

1.8.part3: web part

This part involves creating a Python-based HTTP server using sockets that handles a variety of requests by dynamically serving content based on URL paths. The server responds to requests for specific paths like /, /en, and /ar by serving localized HTML files (main_en.html and main_ar.html) styled with separate CSS. It handles generic requests for HTML, CSS, and image files by

determining the correct content type and serving the appropriate files. Images are stored in a designated folder and can be retrieved via a user-input form in mySiteSTDID.html. The server also supports 307 Temporary Redirects for specified paths, ensuring the original request method and data are preserved during the redirection. If a file is not found, a custom 404 error page is displayed with specific information, including client details. Additionally, all HTTP requests are logged in the terminal, providing real-time monitoring of server activity.[8]

2. Results and Discussions Part

2.1Task 1: Commands & Wireshark

2.1.1Part 1:define each command:

1. **Ping**: This command sends ICMP Echo Request packets to a target IP address or hostname to check if it is reachable and to measure the round-trip time for messages sent from the source to the destination. It's used for testing connectivity and latency.

2. **Tracert (Trace Route)**: This command traces the path that packets take from the source to the destination. It shows each hop along the way, including the IP addresses of intermediate routers and the time taken for each hop. It's useful for diagnosing routing issues.

3. **Nslookup**: This command queries DNS servers to obtain domain name or IP address mapping. It helps in troubleshooting DNS-related issues by allowing you to check the IP address of a domain name or the domain name associated with an IP address.

4. **Telnet**: This command connects to a remote host using the Telnet protocol, allowing you to communicate with the host over a TCP connection. It's often used for testing and debugging network services, but it's less secure compared to modern alternatives like SSH.

1: 1.1. 2.1.a. Ping a Device in the Same Network

The command `ping 192.168.0.108 is used to test the connectivity between two devices within the same local area network (LAN). By sending ICMP Echo Request packets to the device with the IP address 192.168.0.108, the command checks whether the device responds and measures the round-trip time of the packets. Successful replies confirm that both devices can communicate over the LAN without issues, indicating that the network connection is functional and properly configured.

Ping a device in the same network, e.g. from a laptop to a smartphone, Note: 192.168.0.108 this is leyan Burait IP address Phone

Ping command

Ping is a command that works across all operating systems. It can be used to determine how long it will take you to accomplish your goal and whether you can reach it. Ping transmits packets using the Internet Control Message Protocol (ICMP) to the intended address. It then awaits the echo response. It can display this request's statistics, faults, and packet loss. This command will cause you to send a small number of echo requests—typically four. The results for each of them will then be sent to you, together with information on how much data was received, how long it took for a response, and TTL (Time to live). You can transmit a brief data packet to a specific IP address by using this command. After then, watch for a feedback packet. It can be used to ping a name resolution as well. A ping to an IP address that returns an answer but not to a name indicates that the two are inconsistent.

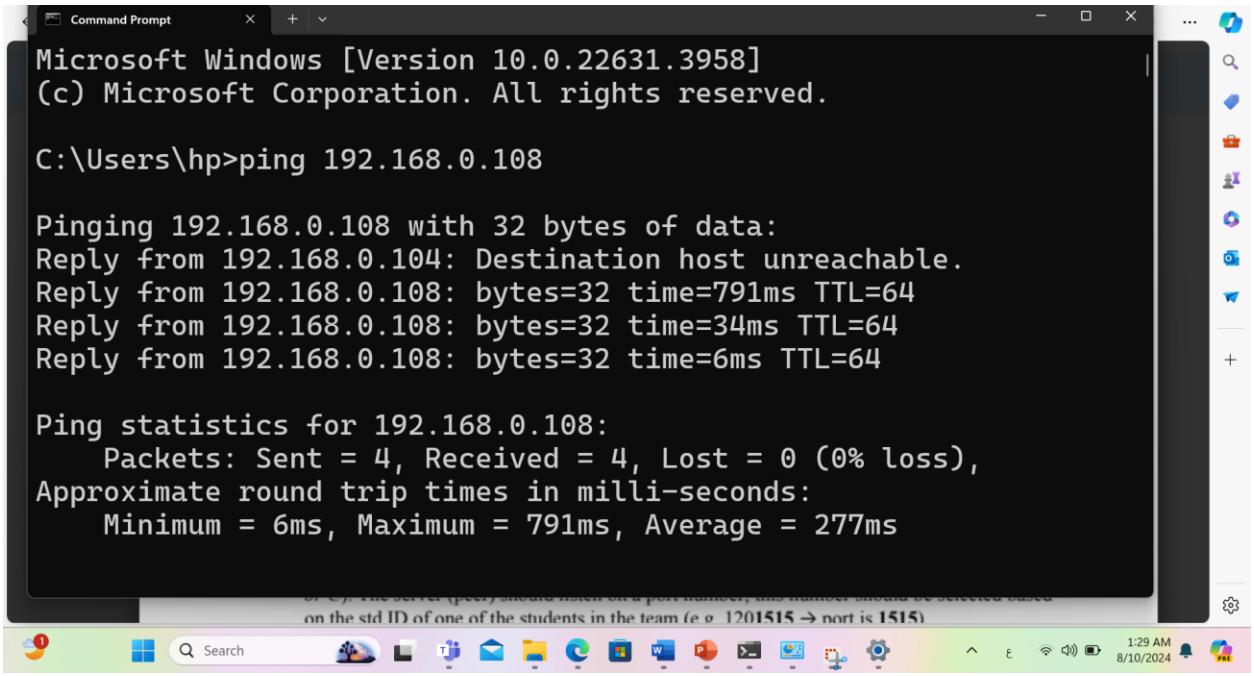


Figure 12: ping 192.168.0.108

We sent out four packets to the destination, and the destination responds back with the same four packets, so we can see four responses is reply from 192.168.0 and 108 that is for router, Packets:

Sent = 4, Received = 4, Lost = 0 (0% loss), We can note that the time (time to reach destination and back) is very small because the two devices are in the same network and there is no need to go outside of the network, Approximate round trip times in mile-seconds: Minimum = 6ms, Maximum = 791ms, Average = 277ms.

1: 1.2.b ping www.ox.ac.uk.

Using the command `ping www.ox.ac.uk` tests connectivity to the Oxford University web server. This command sends ICMP Echo Requests to the domain and measures the round-trip time for each packet. The results provide insights into the network performance between the source and the remote server, including latency, which is crucial for evaluating the responsiveness and reliability of the connection to the external server.

Minimum = 13ms, Maximum = 17ms, Average = 15ms.

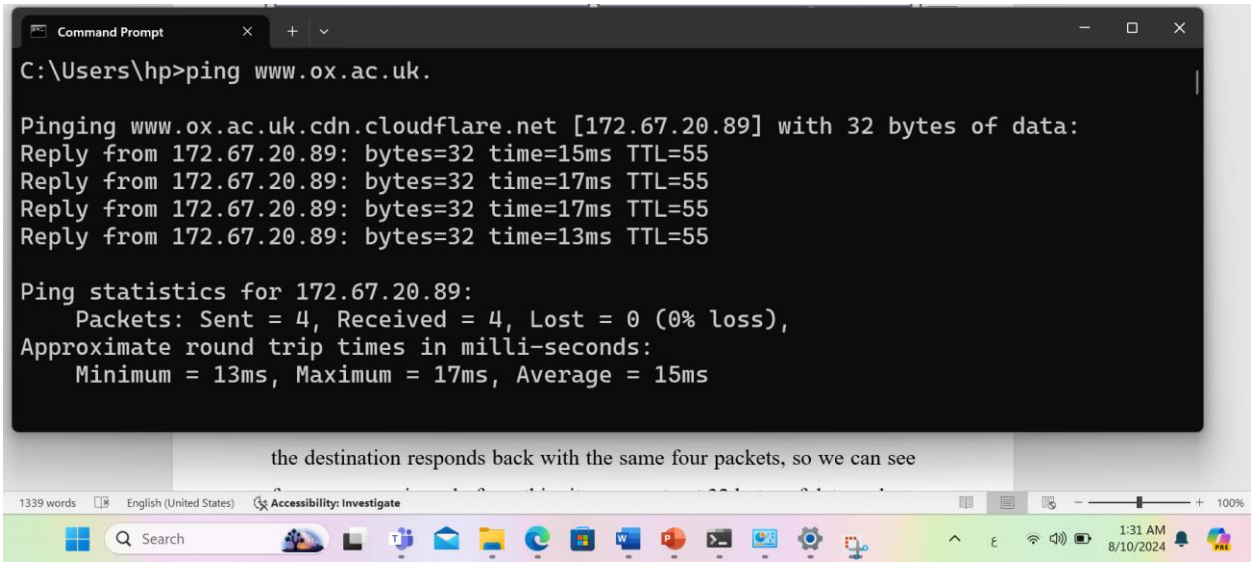


Figure 13: ping www.ox.ac.uk

Cornell.edu is a site name, we sent out four packets to the destination, and the destination responds back with the same four packets, so we can see four responses is reply from this site, we sent out 32 bytes of data and we got back 32 byte of data. Packets: Sent = 4, Received = 4, Lost = 0 (0% loss), there is an important note that the time is more than the previous one because the server and client are not in the same network and there is need to go outside of the network to find the server and create connection with it, and as seen above (TTL expired in transit) it means that the Time to-Live (TTL) value of the packet has reached zero during its journey through the network.

c. Location of the Server

Ping results show the round-trip time in milliseconds (ms), reflecting the latency between the source and destination. While this time can offer a general idea of the distance to the server, it does not directly indicate the server's physical location. To pinpoint the exact geographical location of the server, additional tools such as IP geolocation databases are necessary, which map IP addresses to physical locations based on various data sources.

1.2. Tracert command

It's command-line utility that you can use to trace the path that an Internet Protocol (IP) packet takes to its destination. The TRACERT diagnostic utility determines the route to a destination by sending Internet Control Message Protocol (ICMP) echo packets to the destination. In these packets, TRACERT uses varying IP Time-To-Live (TTL) values. Because each router along the path is required to decrement the packet's TTL by at least 1 before forwarding the packet, the TTL is effectively a hop counter. When the TTL on a packet reaches zero (0), the router sends an ICMP "Time Exceeded" message back to the source computer.

1.2.1.dtracert www.ox.ac.uk

The command `tracert www.ox.ac.uk` traces the route that packets take from the source to the Oxford University server. It reveals all intermediate hops (routers) and the time taken to reach each one. This information helps in understanding the routing path and identifying any delays or issues along the way, which can be useful for diagnosing network problems and optimizing routing paths.Tracert (known as "traceroute" on Unix/Linux systems) is a network diagnostic tool designed to trace the path that packets take from the source to a destination across an IP network, identifying all intermediate routers or hops along the route. It operates by sending packets with incrementally increasing Time-to-Live (TTL) values; each router along the path decrements the TTL and, when it reaches zero, sends an ICMP Time Exceeded message back to the source. By analyzing these ICMP messages, Tracert reveals the route taken by the packets and measures the time required for each hop, which is instrumental in identifying network bottlenecks, understanding routing paths, and diagnosing connectivity issues.This trace shows the path taken from your machine through various networks to reach the final destination. It includes both local network devices and several external network nodes.

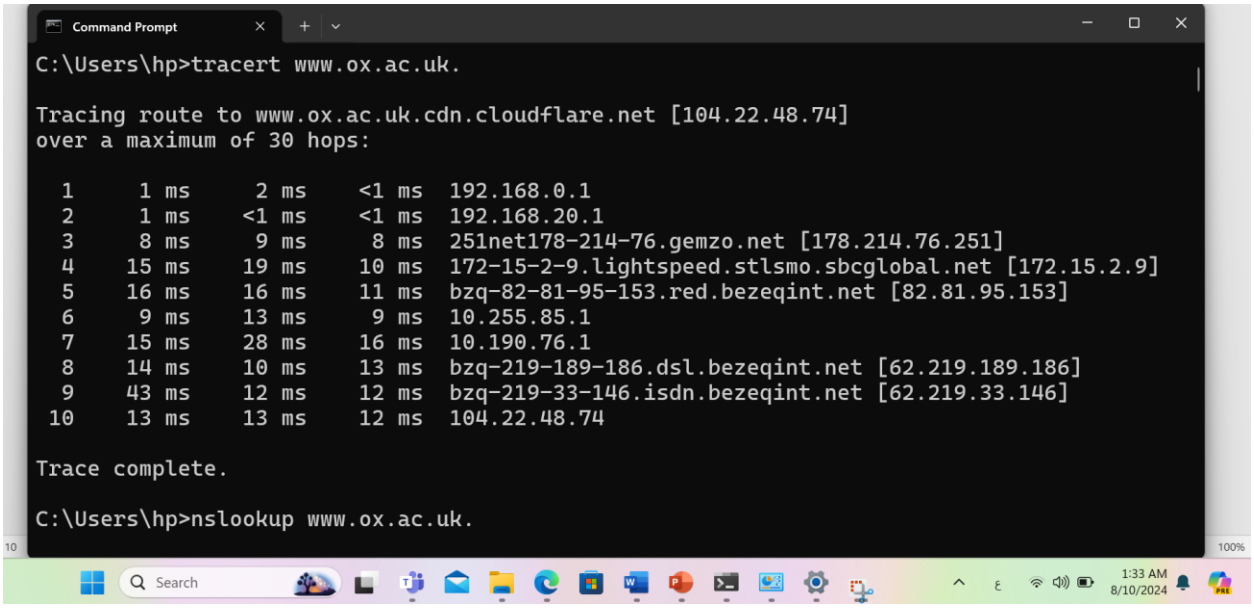


Figure 14dtracert www.ox.ac.uk result

Here the [104.22.48.74] is the IP address for www.ox.ac.uk, and as shown above it went step by step through 30 hops before www.ox.ac.uk

The tracert output shows the path packets take from your computer to www.ox.ac.uk, revealing each network hop along the way. It starts at your local router (192.168.0.1), moves through another local device (192.168.20.1), and then traverses various intermediate network nodes, including those managed by your ISP and Bezeq International (such as 178.214.76.251 and 82.81.95.153). The trace includes internal network devices (10.255.85.1 and 10.190.76.1) before reaching the final destination, www.ox.ac.uk, served via Cloudflare's CDN at IP address 104.22.48.74.

1.2.3.d Nslookup Command : nslookup www.ox.ac.uk.

This command will fetch the DNS records for a given domain name or IP address. The IP address and domain names are stores in DNS Servers, so the nslookup command lets you query the DNS Records to gather information. Using nslookup online is very simple. Enter a domain name in the search bar above and hit 'enter'. This will take you to an overview of DNS records for the domain name you specified. It allows you to view all the DNS records for a website. There are many situations where online nslookup can be a useful tool. For example, when you are configuring the DNS records of your own domain, you might want to check whether you have configured them correctly. You can do this by entering the domain name at the top of this page.

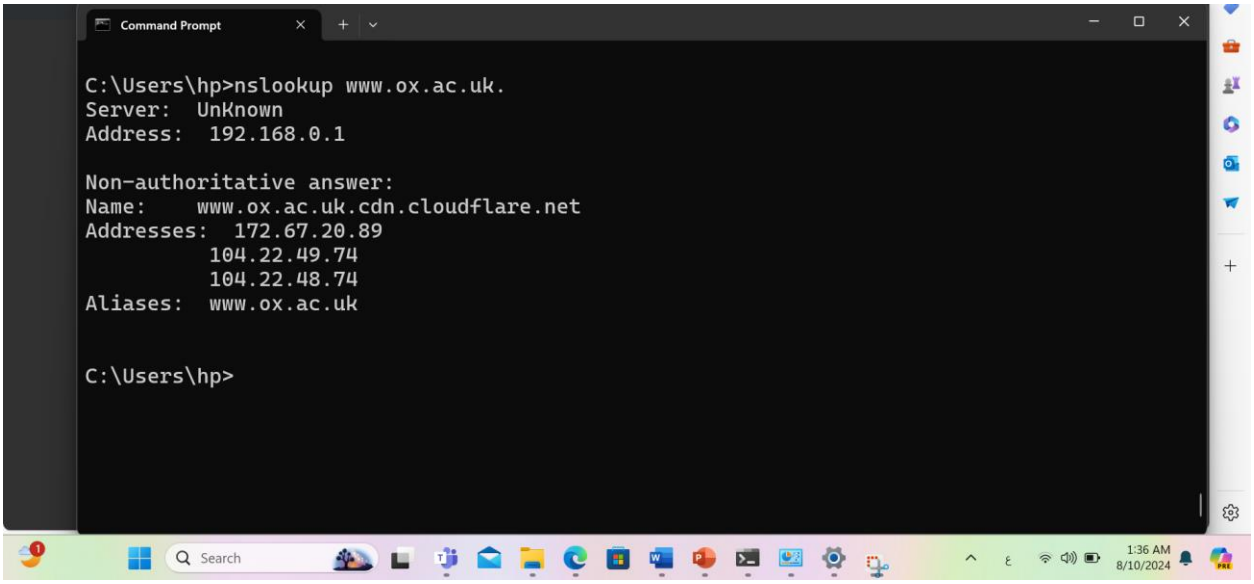


Figure 15Nslookup Command : nslookup www.ox.ac.uk.result

The `nslookup` output shows that `www.ox.ac.uk` resolves to multiple IP addresses via Cloudflare's CDN, with the addresses `104.22.49.74`, `104.22.48.74`, and `172.67.20.89`. The "Server: UnKnown" indicates that the DNS server used for the query is set to your local router (192.168.0.1), which does not have a specific name. The result also lists `www.ox.ac.uk` as an alias for `www.ox.ac.uk.cdn.cloudflare.net`, indicating that Cloudflare handles the DNS resolution and content delivery for this domain.

By executing `nslookup www.ox.ac.uk`, users query DNS servers to retrieve the IP address associated with the domain name `www.ox.ac.uk`. This command provides DNS records, such as A records, which map domain names to IP addresses. It is an essential tool for verifying domain resolution and gathering DNS-related information, which is helpful for troubleshooting and network configuration.

Nslookup (Name Server Lookup) is a network administration command-line tool used to query Domain Name System (DNS) servers for domain name or IP address mapping information. It functions by sending queries to a specified or default DNS server to retrieve DNS records such as A records, which map domain names to IP addresses, MX records for mail exchange servers, and other DNS records. Nslookup provides essential details such as the IP address associated with a

domain name or the domain name associated with an IP address, making it a valuable tool for troubleshooting DNS issues, verifying domain name resolution, and gathering comprehensive DNS information.

1.2.4.e Telnet www.ox.ac.uk

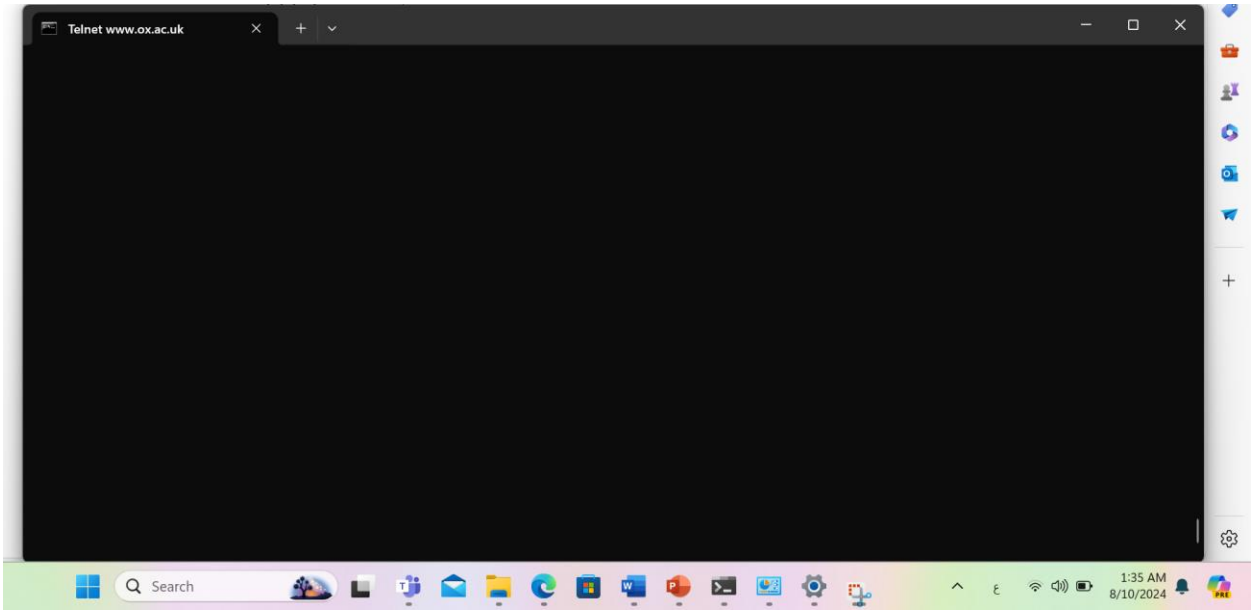


Figure 16Telnet www.ox.ac.uk result

The command `telnet www.ox.ac.uk 80` attempts to establish a Telnet connection to the Oxford University server on port 80, which is used for HTTP traffic. This connection allows users to interact with the server and manually send HTTP requests. While useful for testing and troubleshooting web servers, Telnet does not provide encryption, making it less secure than modern alternatives like SSH for secure remote communication.

Telnet is a network protocol and command-line tool that establishes a TCP connection to a remote device or server, typically using port 80, to provide a command-line interface for communication. When a Telnet command is executed, it initiates a TCP connection to the specified host and port, allowing users to interact with the remote system's command-line interface and run commands. Despite its utility for remote administration, testing network services, and troubleshooting, Telnet

is not secure as it transmits data in plaintext. Consequently, it has largely been superseded by more secure protocols like SSH (Secure Shell), which offer encrypted connections.

Explanation: This command attempts to establish a Telnet connection to the www.ox.ac.uk server on port 80, which is typically used for HTTP traffic. If successful, it provides a command-line interface to interact with the web server. Note that many modern servers disable Telnet access for security reasons, and you might see a connection error if Telnet is not supported.

1.2.1.f. AS13335 Cloudflare,Inc

To gather details about the Autonomous System (AS) associated with `www.ox.ac.uk`, one needs to find several key pieces of information. The **AS Number** is a unique identifier for the AS, which is a collection of IP networks and routers under a single administrative entity. The **Number of IPs and Prefixes** refers to the range of IP addresses and network prefixes advertised by the AS. **Peers** are other ASes with which this AS exchanges routing information. The **Tier 1 ISP** is a major ISP providing backbone connectivity to the internet without paying other ISPs for transit. This information can be obtained using online tools like BGPView, which provides detailed BGP (Border Gateway Protocol) data.

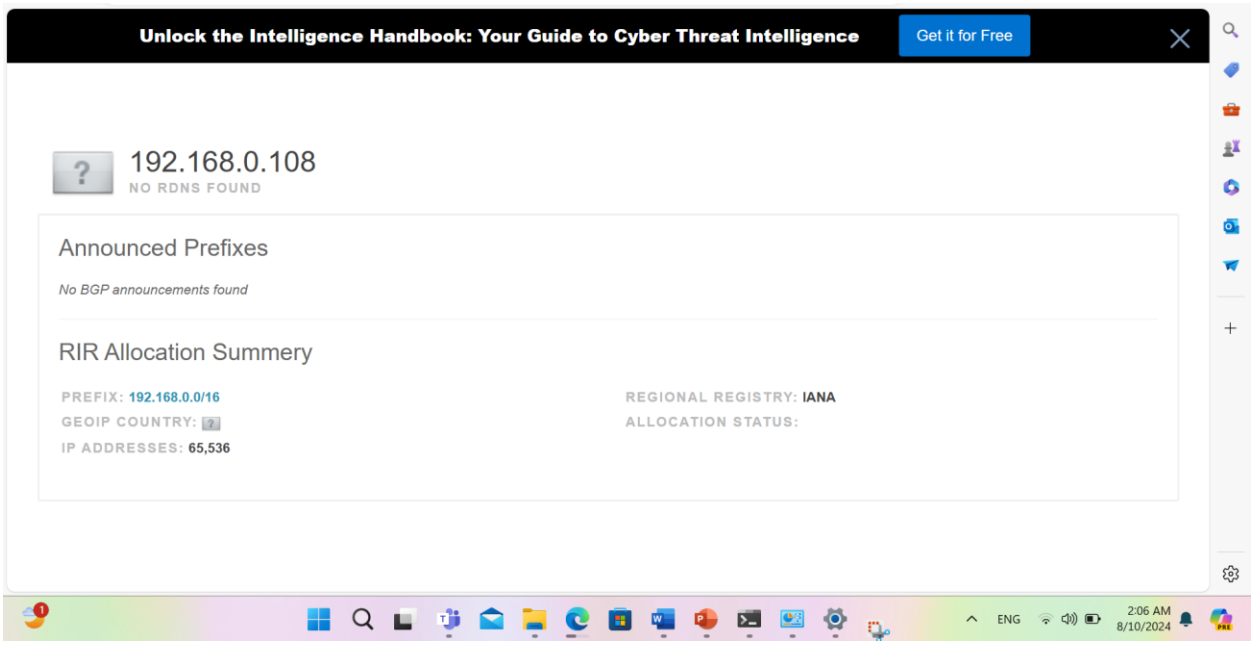


Figure 17AS result of IP:192.168.0.108

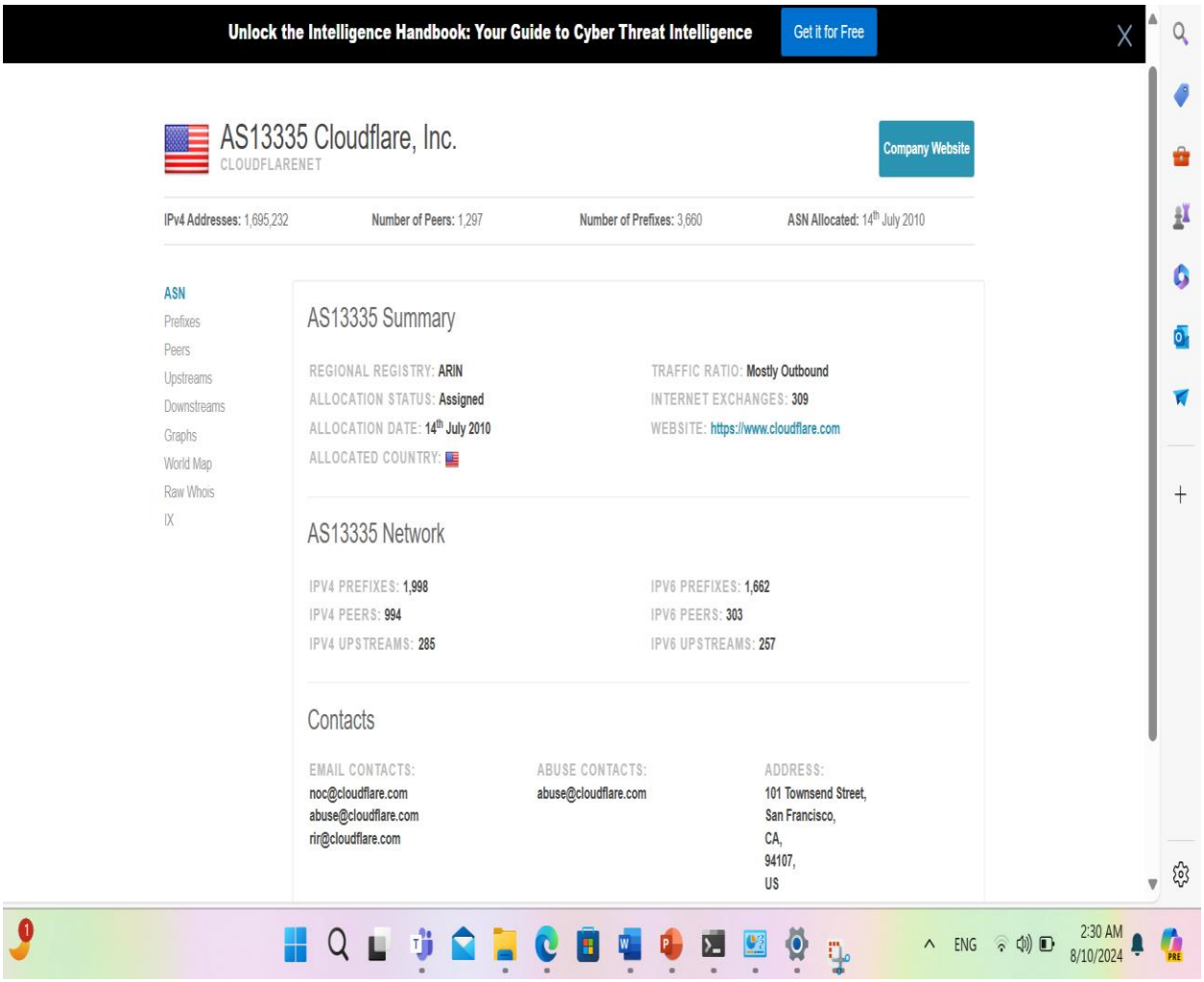


Figure 18: AS13335 Cloudflare,Inc.

Expected Result: This command tries to establish a Telnet connection to the HTTP port (80) of www.ox.ac.uk. If successful, you'll get a blank screen or prompt, which indicates that you have connected to the server. If it fails, you might see an error message such as "Could not open connection". Give some details about autonomous system (AS) number, number of IPs, prefixes, peers,name of Tier1-ISP of www.ox.ac.uk.So : IPv4 Address:1,695.232 , Number of peer 1,297 Number of prefix: 3,660 ,ASN Allocated: 14_july_2010

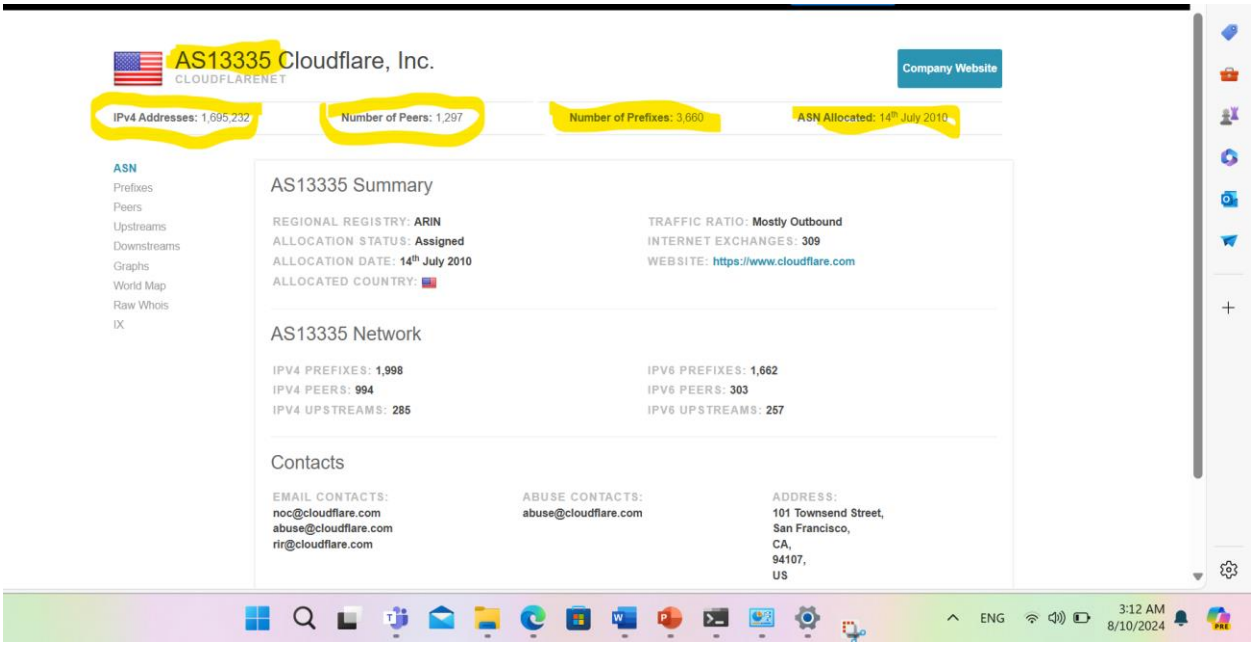


Figure 19AS13335 Cloudflare, Inc with limited all result

1.2.1.4 use wireshark to capture some DNS messages.

Wireshark is an open-source network protocol analyzer widely used for capturing, analyzing, and troubleshooting network traffic. Developed by the Wireshark community, it provides a detailed and real-time view of data moving across a network, helping users understand the communication patterns between devices. Here we have to use the wireshark to capture some DNS messages, we chose Wi-Fi the network interface that corresponds to the network we want to monitor, to focus on DNS messages, we applied a display filter (DNS).

To capture DNS traffic using Wireshark, first, download and install the software from its official website. Once installed, open Wireshark and select the appropriate network interface, such as Wi-Fi or Ethernet, to monitor. Apply a display filter using dns to isolate DNS packets from the overall traffic. Start the capture process and perform DNS queries using tools like nslookup to generate DNS traffic. After capturing the traffic, stop the capture and analyze the DNS packets, focusing on query and response messages to understand DNS resolution processes, including the details of queried domains and resolved IP addresses.

C:\Users\hp>nslookup read.com

Server: UnKnown

Address: 192.168.0.1

Non-authoritative answer:

Name: read.com

Address: 34.206.39.153

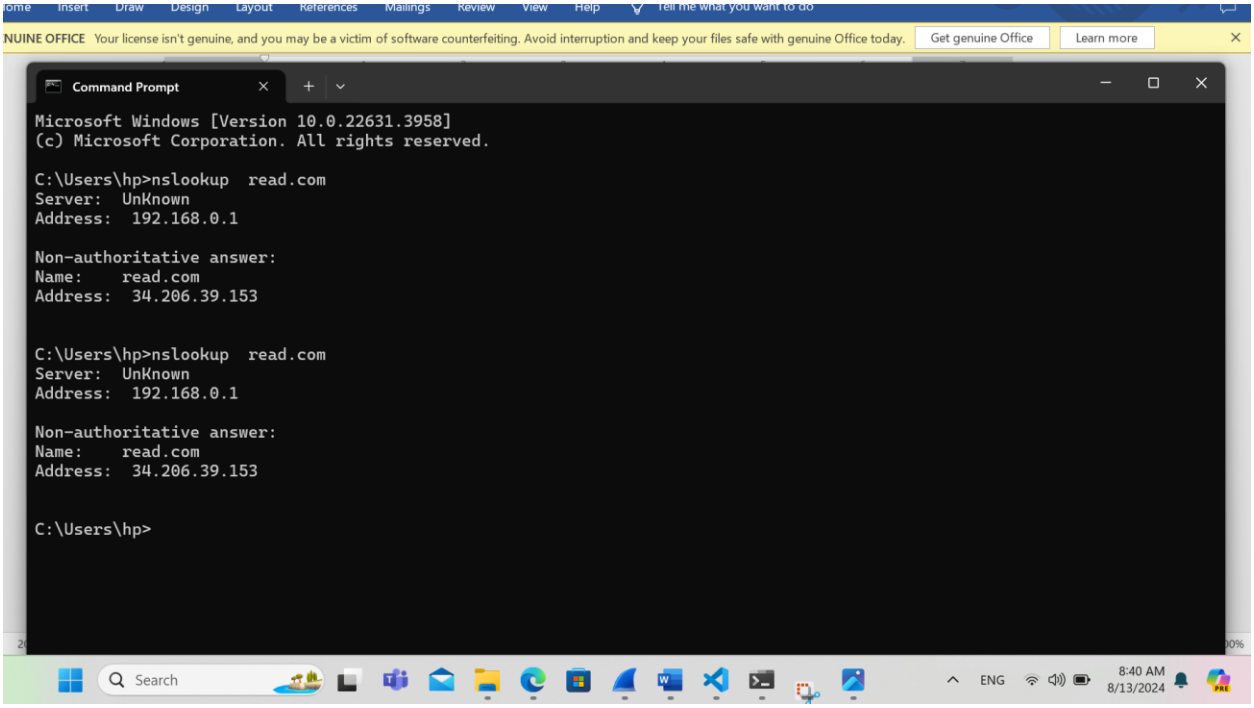


Figure 20perform DNS queries using tools like nslookup to generate DNS traffic

I use cmd to read some DNS as shown above then of Use wireshark do start captured to get some of DNS as shown bellow at first picture some of DNS & the protocol of it is UDP as shown bellow

& the last one some of DNS & the protocol of it is TCP as shown bellow

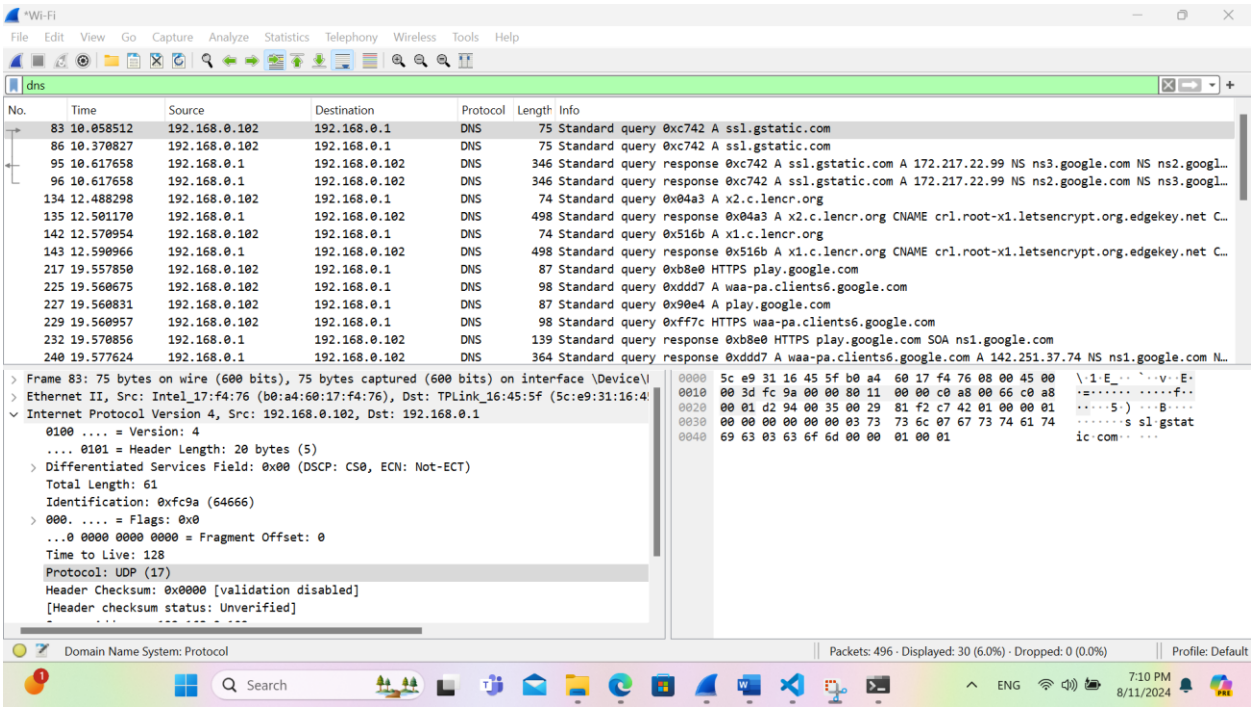


Figure 21some of DNS & the protocol of it is UDP

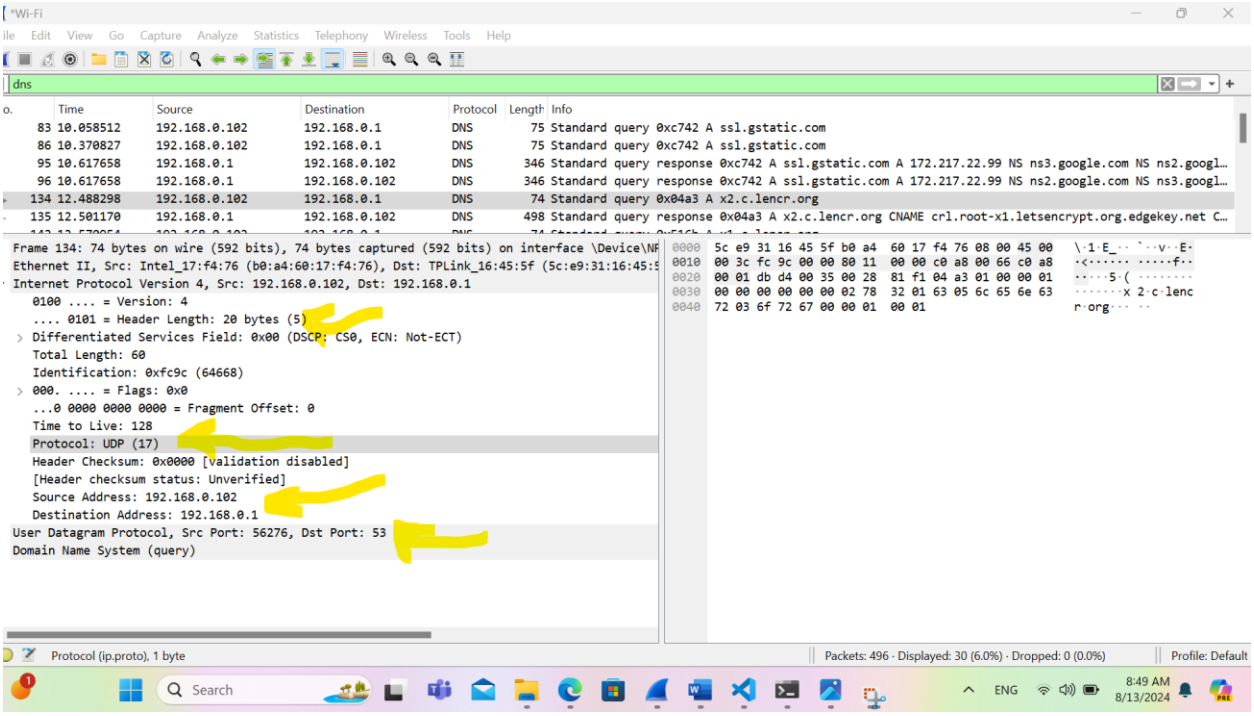


Figure 22the all other information as shown

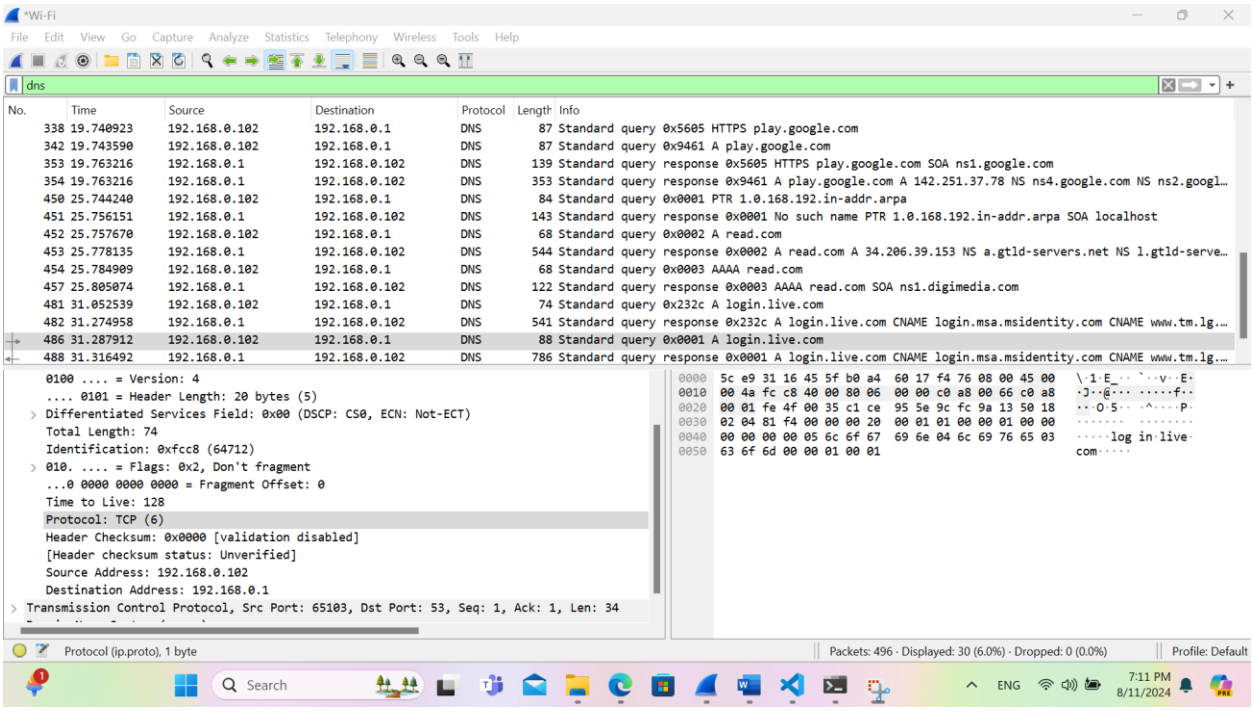


Figure 23some of DNS & the protocol of it is TCP

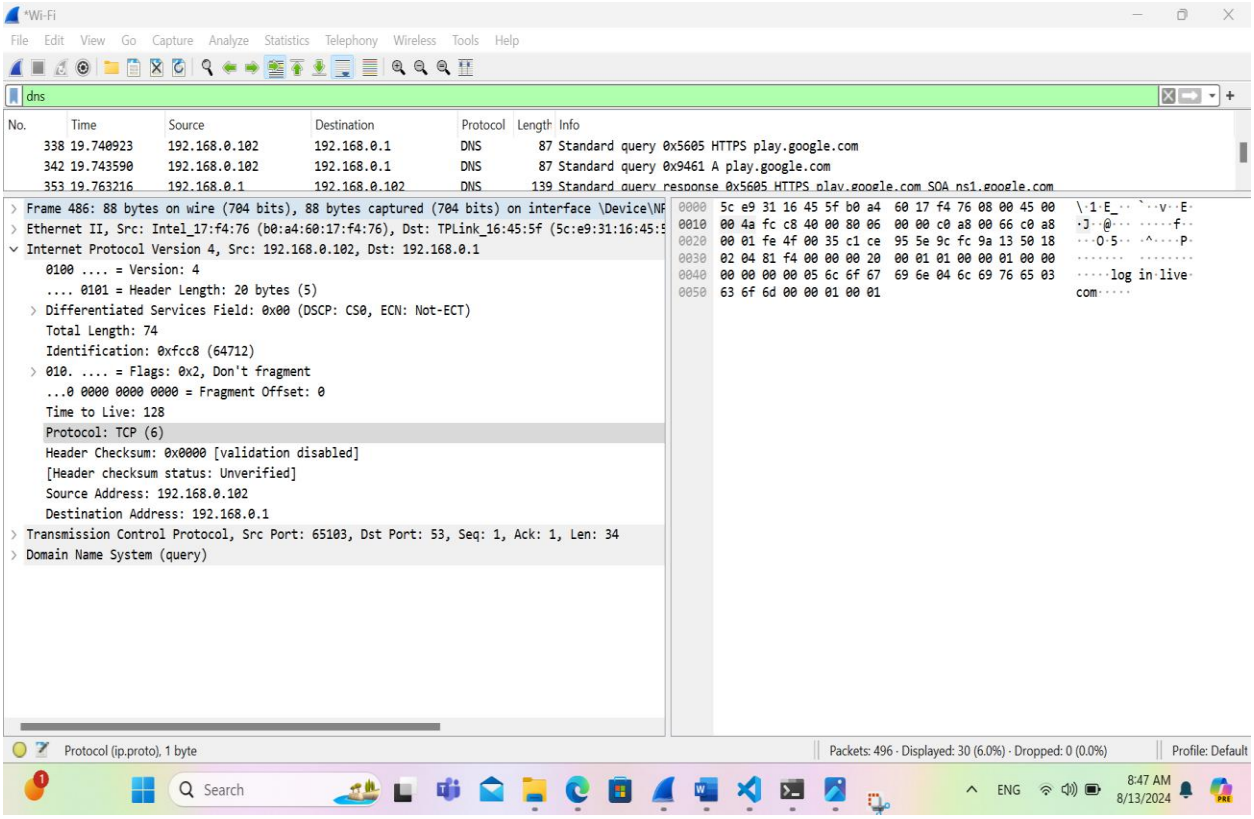


Figure 24& the all information as shown

2-Part two: Socket Programming (TCP and UDP)

Question: Using socket programming, write simple TCP client server python applications (in go, python,

java or C) to send input data from client to server, replace all the vowels (aeiou/AEIOU) with '#' and return the string to client and print it. You need to choose the port number based on the std.ID of one of the students in the team (e.g. 1201515 → port is 1515) for this communication.

I using python to do this task as shown bellow result & code of server & client

2.1 result when use cmd

I open PowerShell window or terminal.Then enter the location of file python of server directory as (C:\Users\hp\Documents\projects\network python .\tcp_client.py).

PS C:\Users\hp\Documents\projects\network> python tcp_server.py

Server listening on port 1439...

As shown bellow

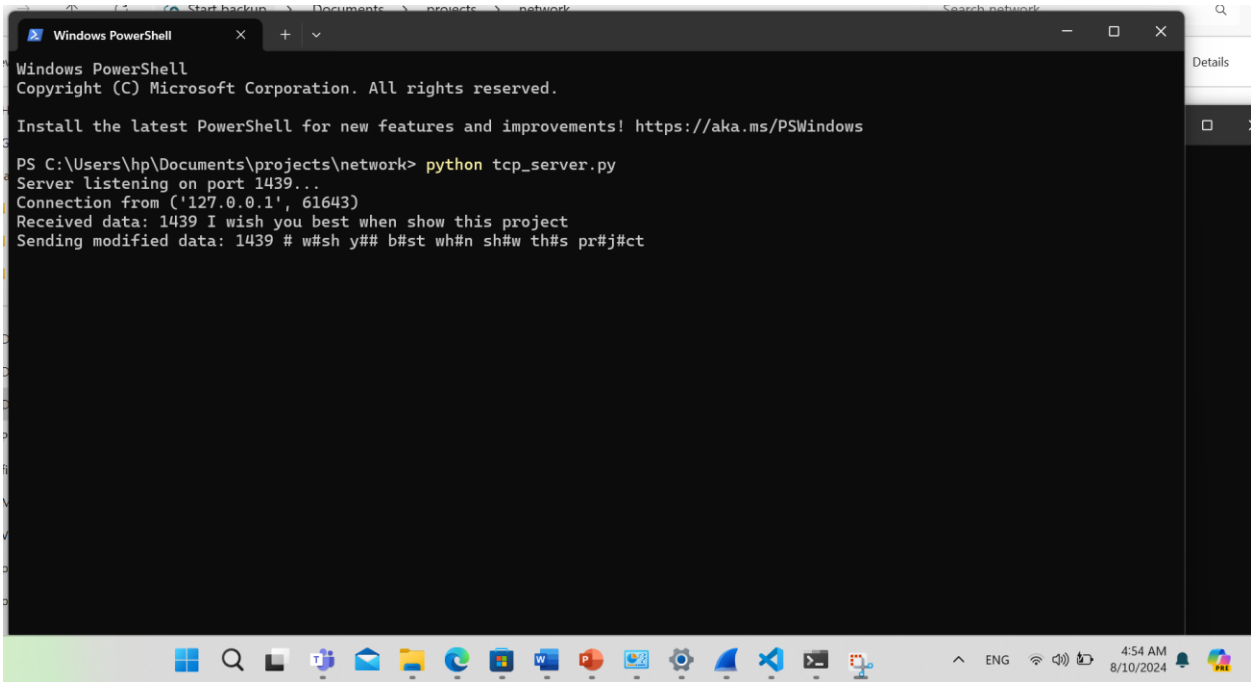


Figure 25TCP server running

TCP server is running correctly and is listening for connections. Here's what happened:

In the PowerShell output shown, the TCP server script was executed successfully. The server began by listening on port 1439. Upon establishing a connection from the client (with the IP address

127.0.0.1 and port 61643), the server received the input string "1439 I wish you best when show this project." The server then processed this string by replacing all vowels with the # character. The modified string "1439 # w#sh y## b#st wh#n sh#w th#s pr#j#ct" was sent back to the client. This output confirms that the server is correctly performing the vowel replacement and transmitting the altered data to the client, demonstrating the effective implementation of the client-server communication and processing logic.

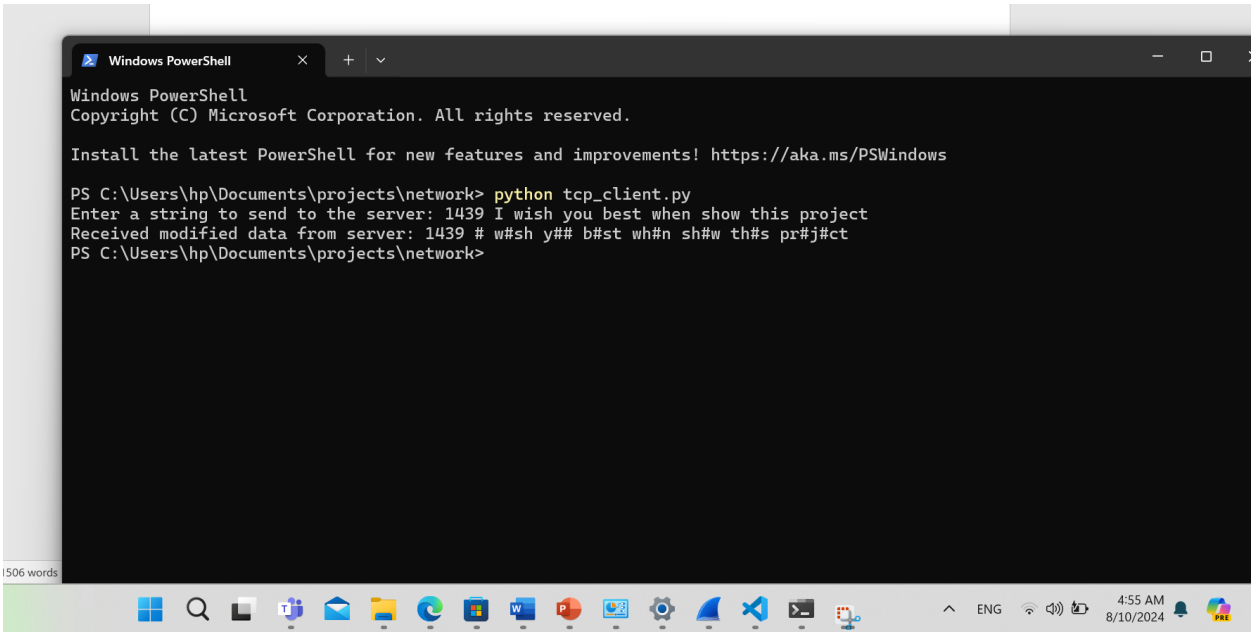


Figure 26TCP client cmd

The server is successfully replacing vowels with # and sending the modified data back to the client.

In the PowerShell output, the TCP client script was executed successfully. After launching the client, it prompted for a string to send to the server. The input provided was "1439 I wish you best when show this project." The client sent this string to the server, which then processed it by replacing all vowels with the `#` character. Upon receiving the modified data from the server, the client displayed the altered string "1439 # w#sh y## b#st wh#n sh#w th#s pr#j#ct." This output confirms that the client correctly received and displayed the vowel-replaced string from the server, indicating that the client-server communication and vowel replacement functionality are working as expected.

Server code using the python

The provided Python code sets up a simple TCP server that listens for incoming client connections on a specified port. The `replace_vowels` function replaces all vowels in a given string with the `#` character. In the `start_server` function, a TCP/IP socket is created and bound to the local address and specified port. The server listens for incoming connections and, upon accepting a connection, receives data from the client. This data is then processed by replacing all vowels with `#`, and the modified string is sent back to the client. Finally, the server closes the client connection. This script continuously operates, handling multiple client requests by repeating these steps in a loop as shown bellow.

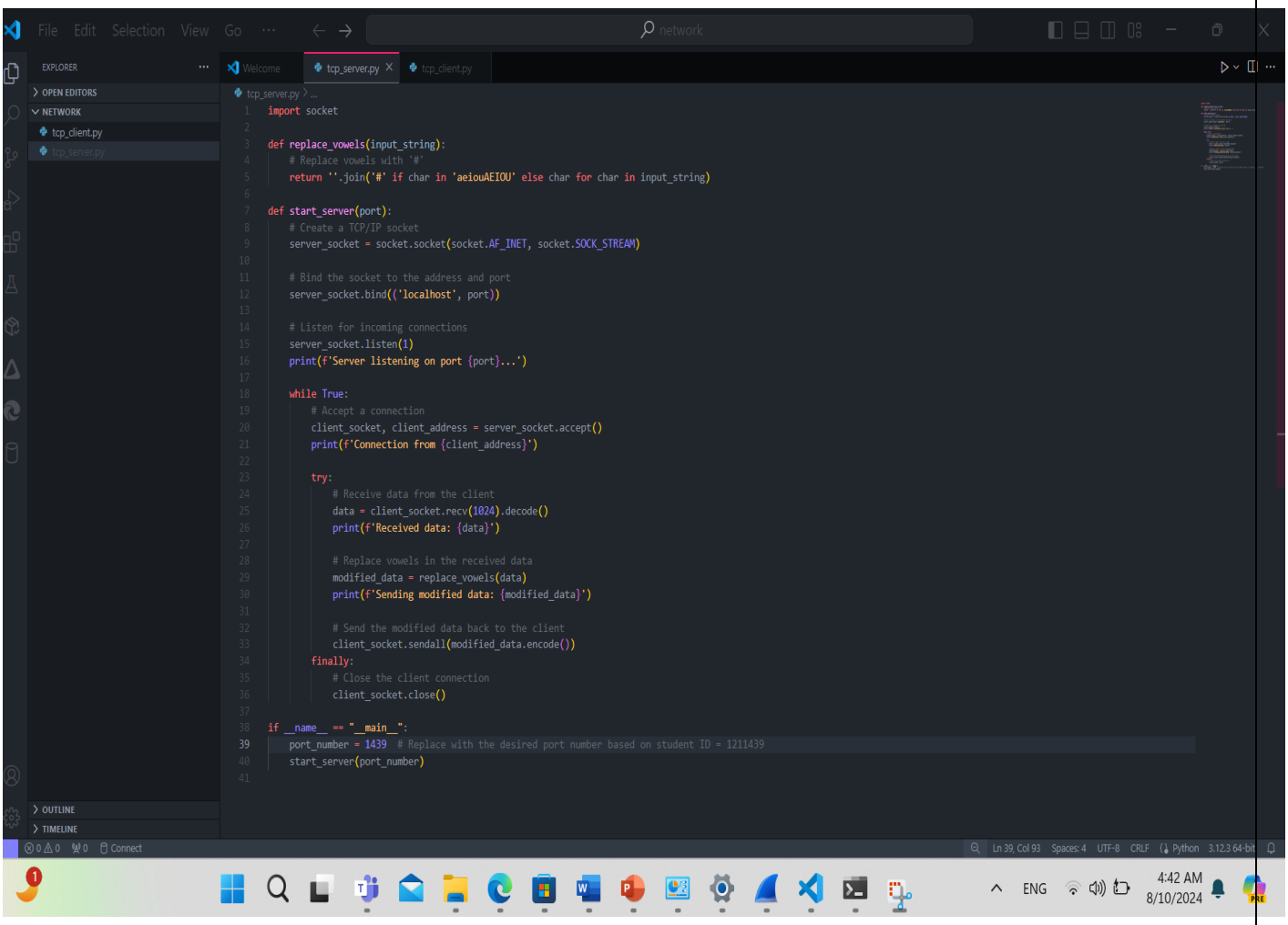


Figure 27servsr code using the python

The provided Python code is a TCP client script that communicates with a server using a specified port number. The `main` function sets up a client socket using the `socket` module and connects it to a server running on `localhost` at port 1439. The client then prompts the user to enter a string, which it sends to the server. After sending the data, the client waits for a response from the server, which is expected to be the modified string with vowels replaced by `#`. The received data is then printed to the console. Finally, the client closes the socket connection to clean up resources. This script demonstrates a simple client-side implementation that interacts with a TCP server to exchange and modify data as shown bellows.

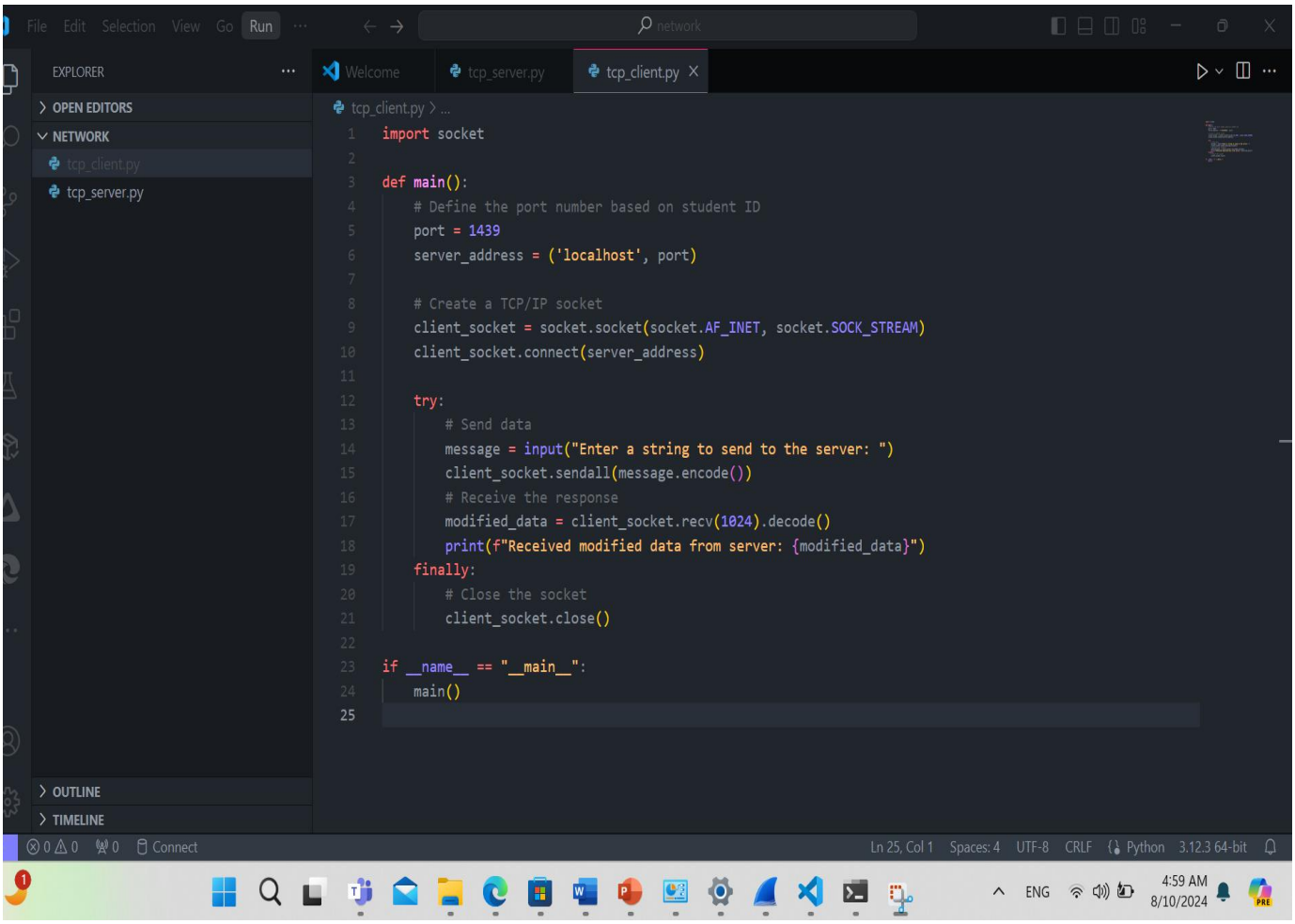


Figure 28Client code using the python

2.2 UDP Program:

In a UDP-based socket programming setup, you create client and server applications to facilitate communication between multiple peers. The server listens on a specific port number, derived from a student ID (e.g., port 1439 for ID 1211439), and handles messages from various clients. Each client sends messages to the server, which includes the client's identifier and the message content. The server processes these messages and responds to each client, displaying a log of interactions with each client

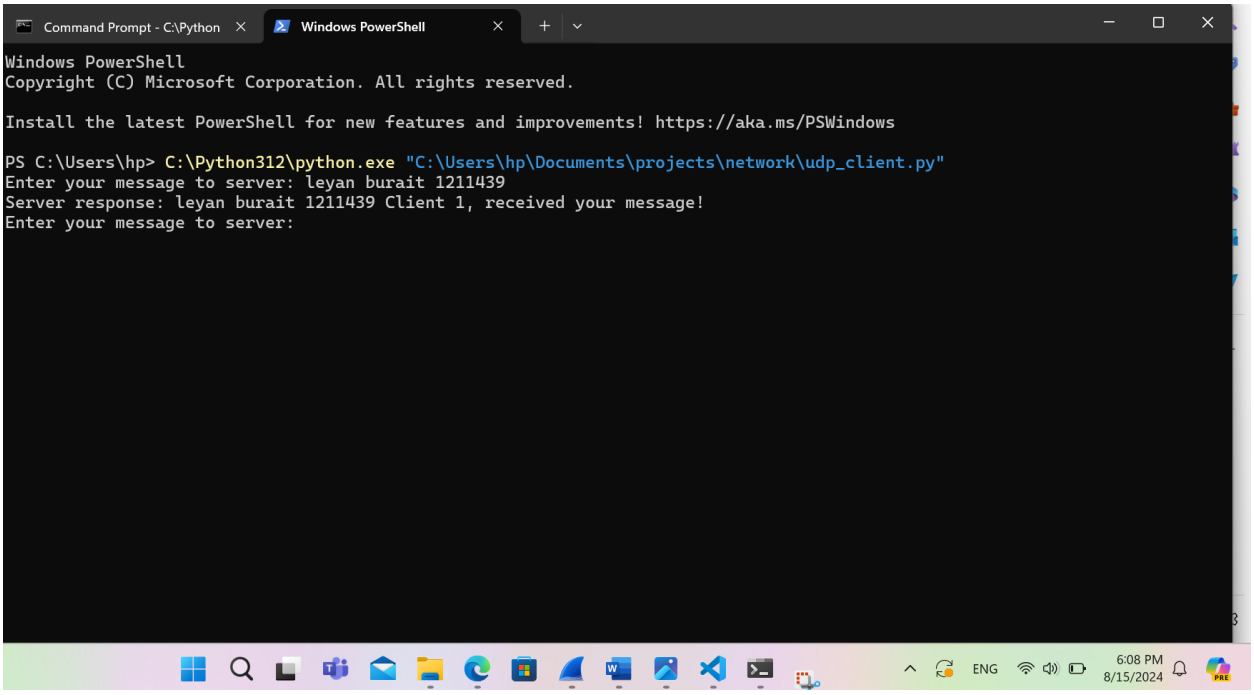


Figure 29first massage of client 1

The provided server and client Python scripts use UDP socket programming to facilitate communication between multiple clients and a server. The server listens on a specific port and assigns a unique identifier (e.g., "Client 1", "Client 2") to each client based on their first connection, managing client addresses through dictionaries to track which ID corresponds to which address. It prints messages from clients with their assigned ID and allows the server user to respond to each client. The client.

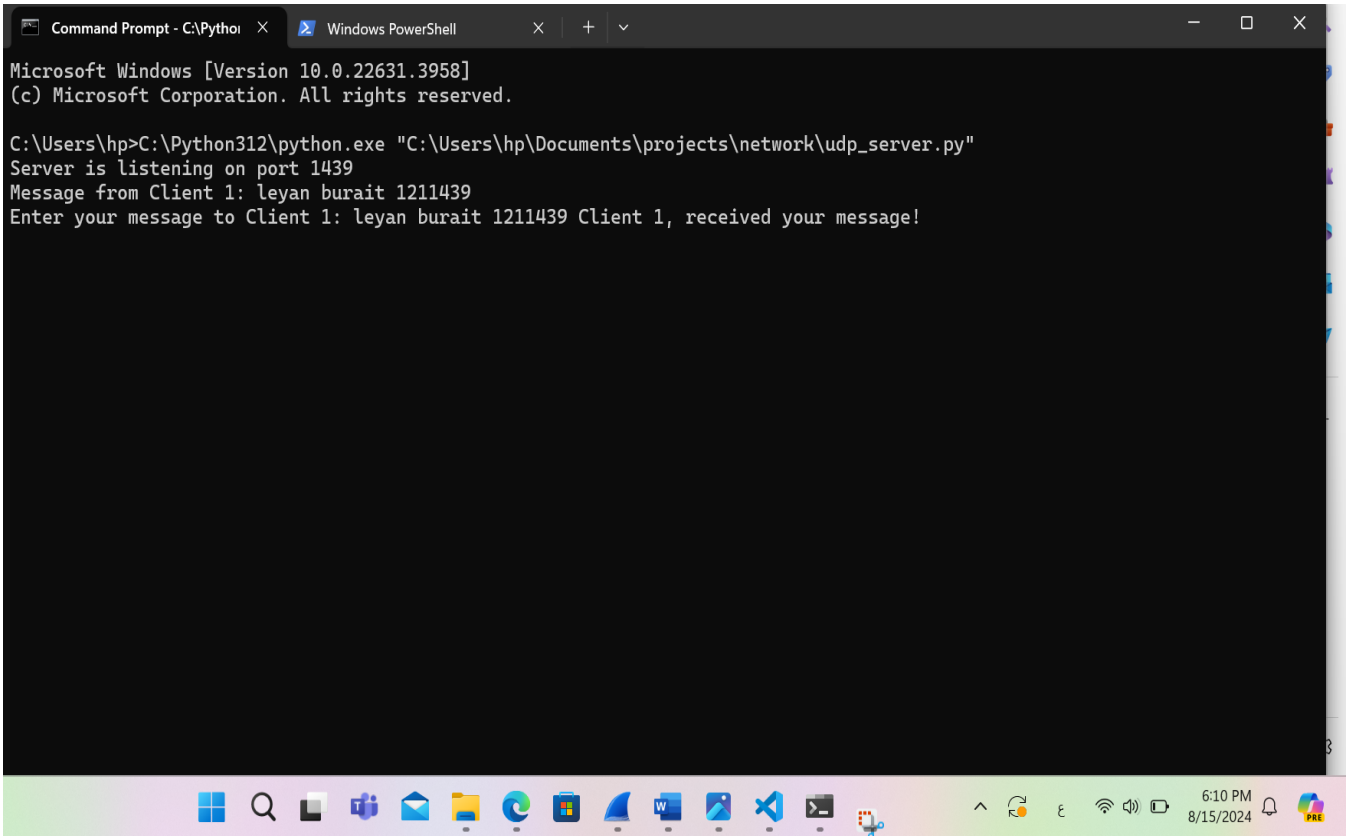


Figure 30server when lisitien to client1

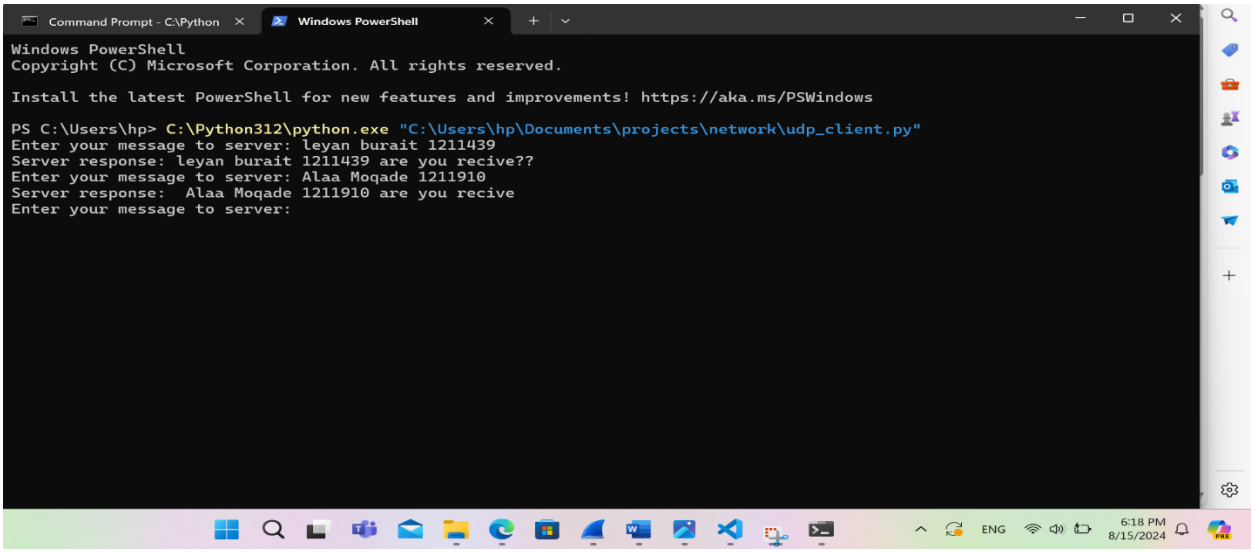


Figure 31:second massage of client 2

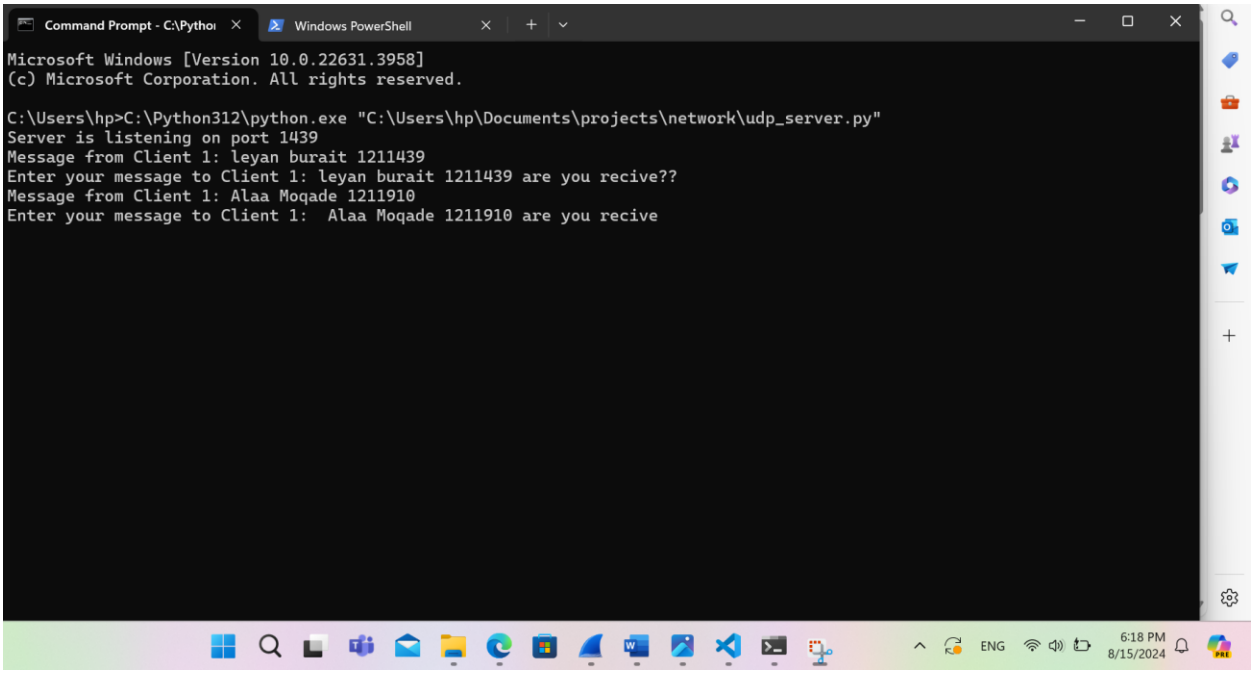


Figure 32server lisiten of client 2

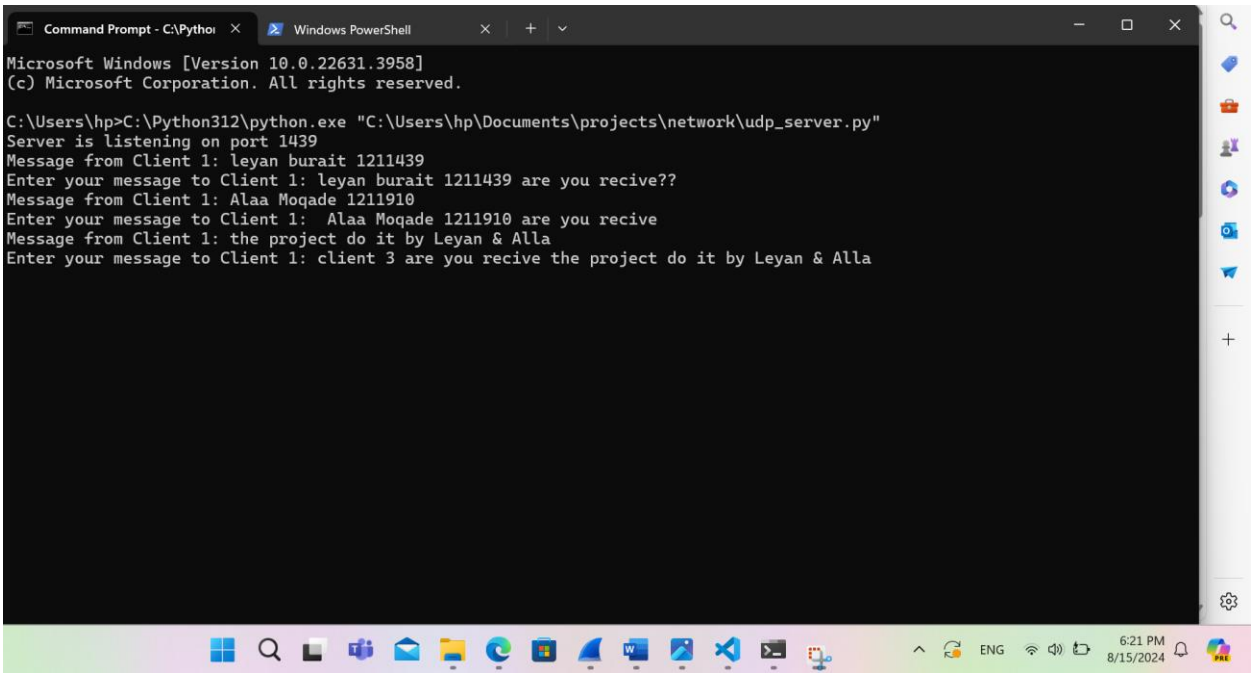


Figure 33client 3 massage was listened by server

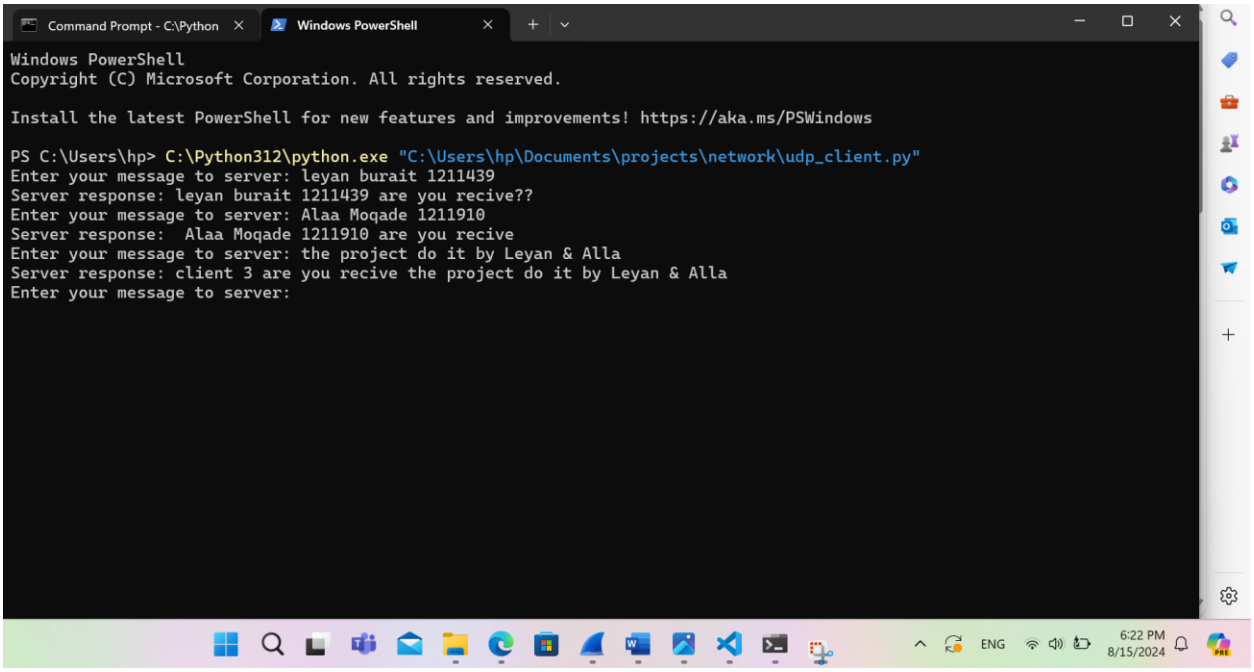


Figure 34: client 3 massage

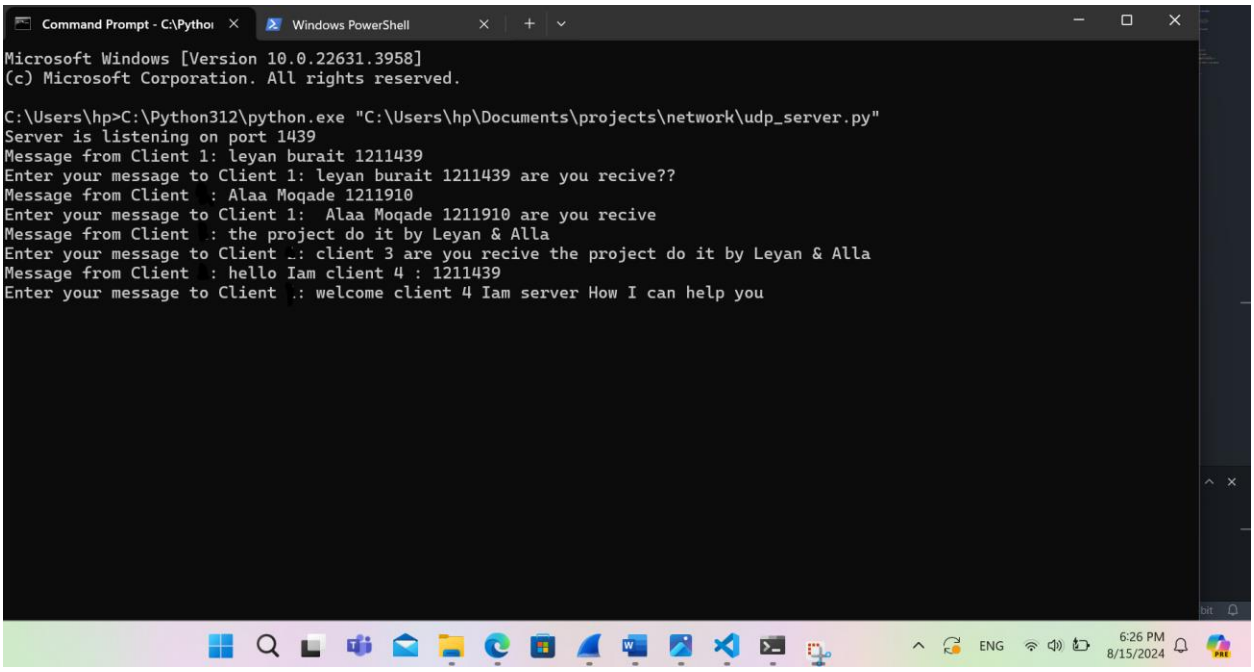


Figure 35server lisiened to client4

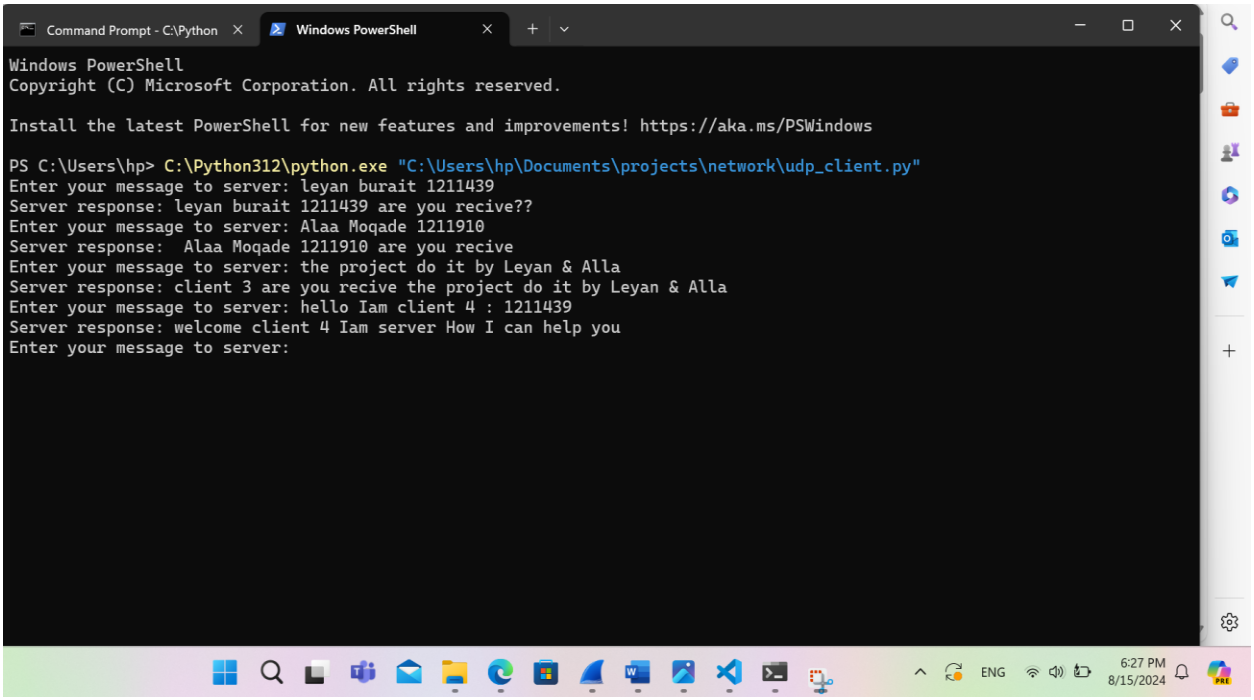


Figure 36client 4 massage

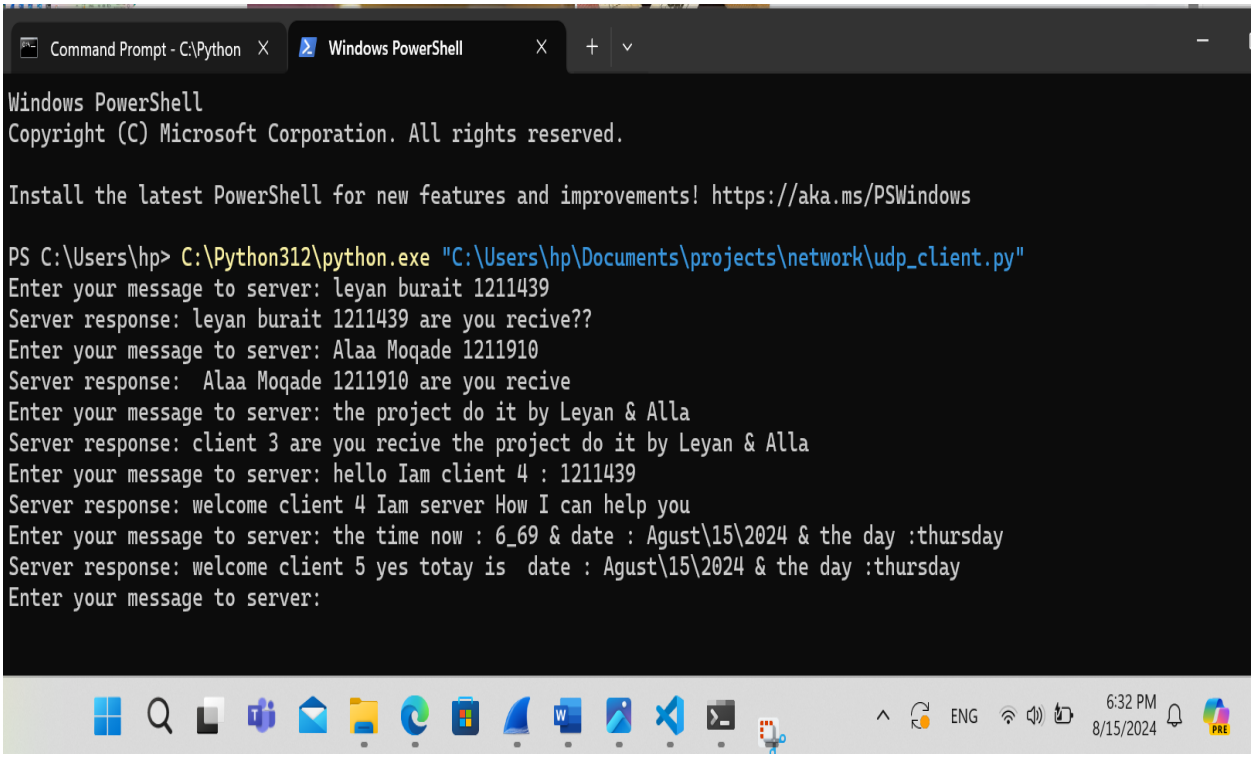


Figure 37client 5 massage

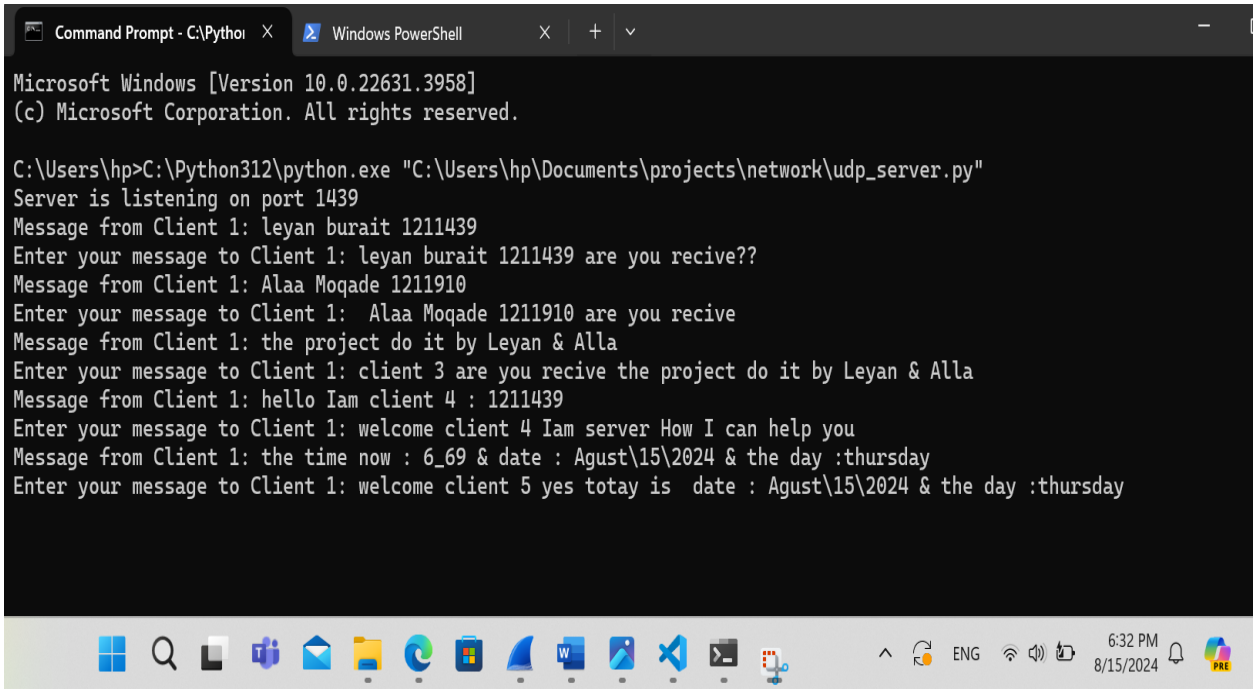


Figure 38server was lisened all massage from client 1 to client 5

Explain of Result for server:

The provided output from the Windows PowerShell session illustrates a UDP server script running and interacting with clients. The server, listening on port 1439, successfully receives and processes messages from clients. Each client message, which includes an identifier and content, is echoed back with a personalized server response acknowledging receipt and including the client's information. For example, messages like "leyan burait 1211439" and "Alaa Moqade 1211910" are responded to with confirmations and additional information, demonstrating the server's ability to track and address messages from different clients sequentially. This output showcases the server's functionality in managing and responding to multiple client interactions in real-time.

Explain of Result for client :

In the provided Windows PowerShell session, a Python UDP client script (`udp_client.py`) is executed, interacting with a UDP server. The client prompts the user to input messages, which are sent to the server, and the server responds accordingly. Each message includes a client identifier and the content, such as "leyan burait 1211439" or "Alaa Moqade 1211910". The server acknowledges each message with a response that confirms receipt and includes the client's message, demonstrating that it processes and responds to messages from multiple clients. For

instance, the server acknowledges messages like "the project do it by Leyan & Alla" and provides a unique response for each, such as "welcome client 4" or "welcome client 5". This interaction showcases how the client-server communication is handled in a UDP context, with the client sending various messages and receiving specific responses from the server.

Part Three:

In this part, we used socket programming to create a complete web server on port 1910 using the Python programming language, as well as HTML, CSS.

3.0- in the beginning, we defined a server port which is 9966, then we created a socket instance and pass AF_INET which means to use IPv4 protocol, and SOCK_STREAM which means connection-oriented TCP protocol.

After creating the socket, we associated it with its local address, allowing clients to connect to the server using that address using (bind ())

Then we put the socket into listening mode to make it ready for the requests

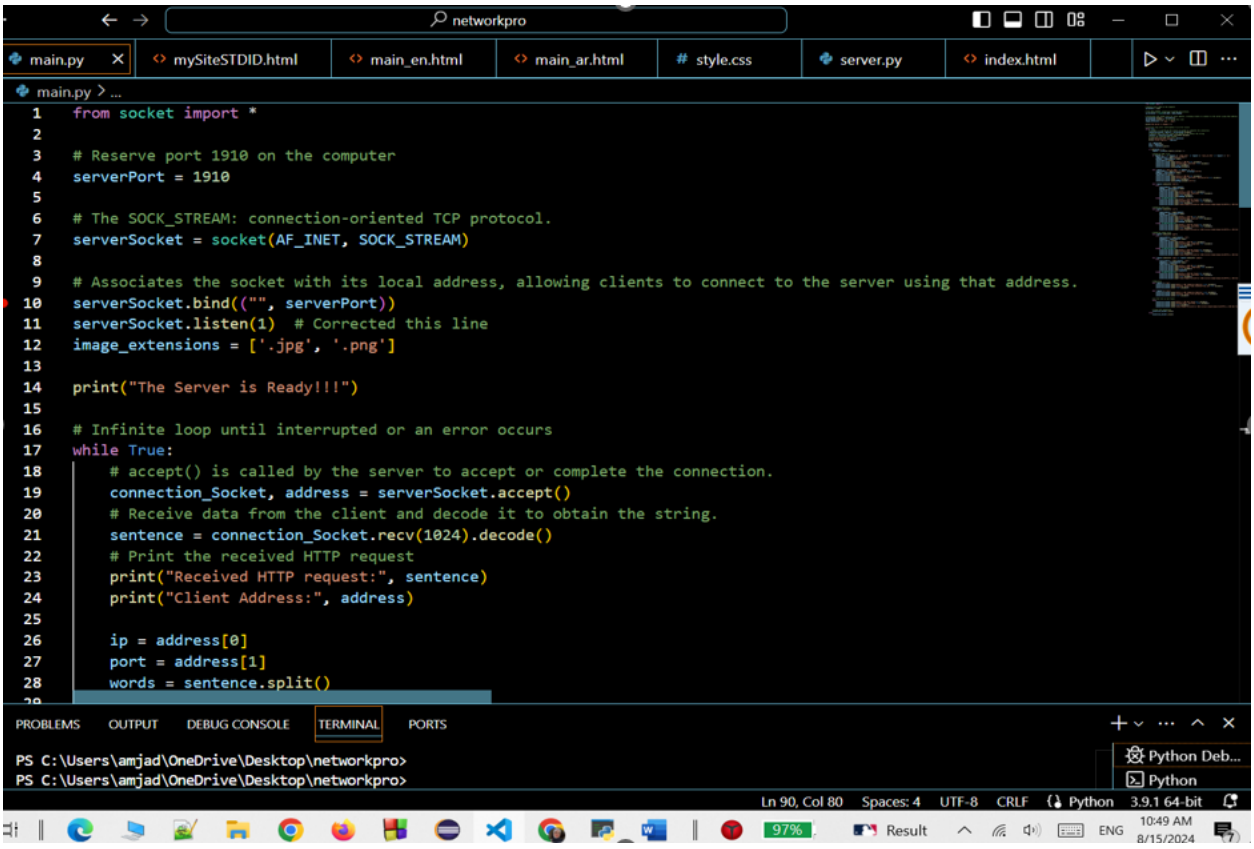


Figure 39create the socket

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 /

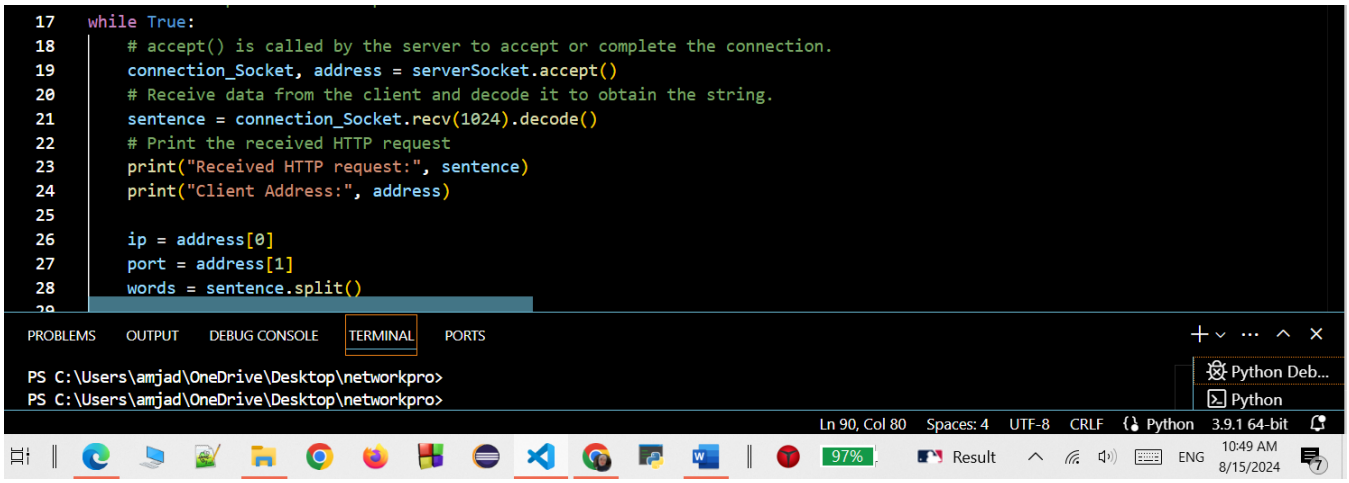


Figure 40: Accept and complete the connection

3.1. Request the main_en.Html File

Check if the request it (is / or /index.html or /main_en.html or /en):

First, the server should send main_en.html file. If the request is / or main_ar html

Second, 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.

Third, it will send the header which contains of the encoded response status.

Finally, the server will send the encoded file that it read previously.

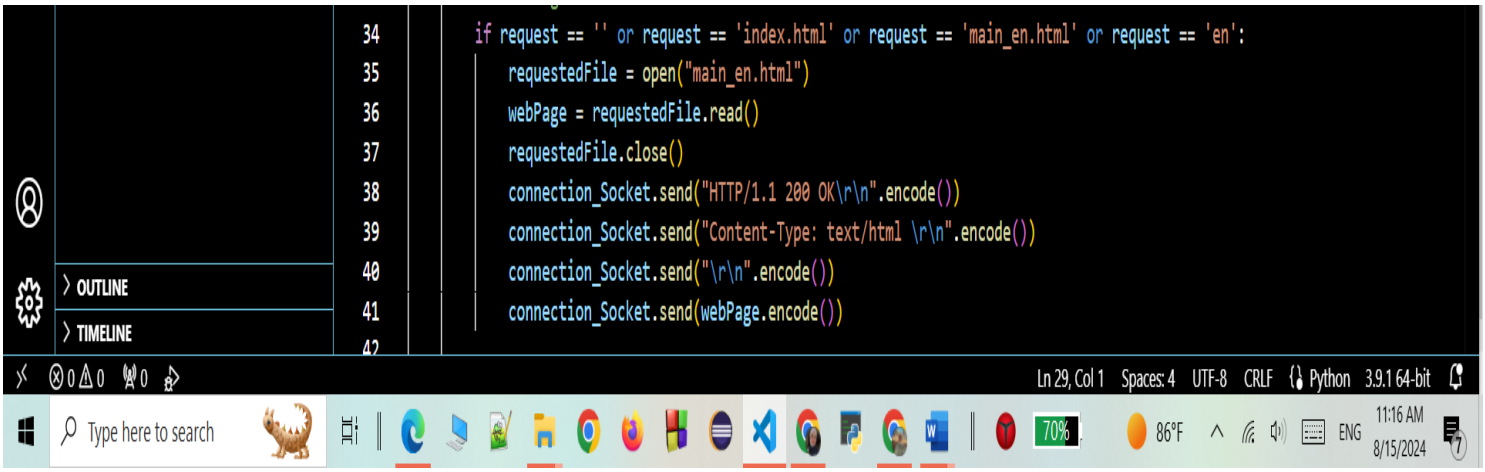


Figure 41: Request the main_en.html file

- **Response the main html file:**

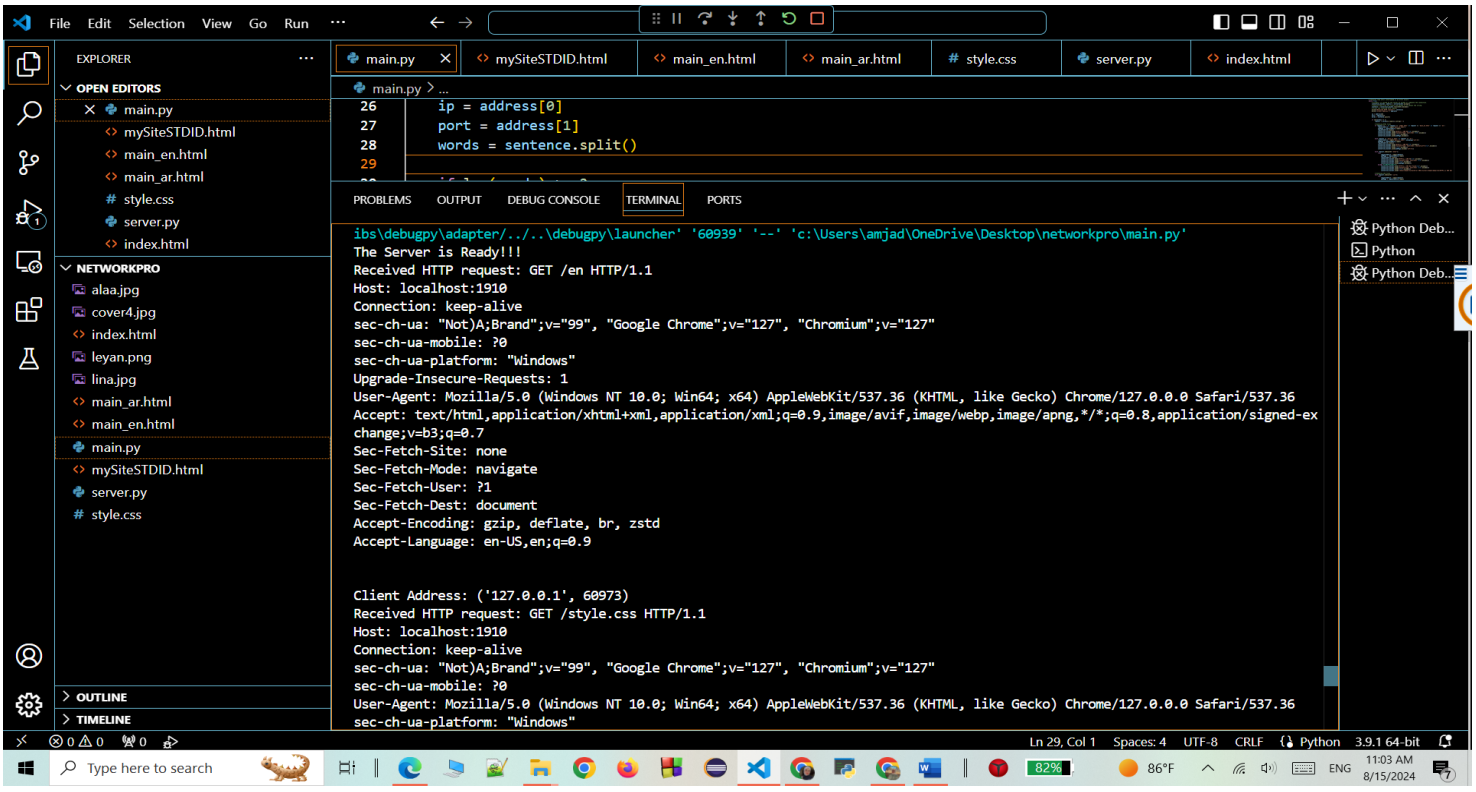


Figure 42: Response the main_en.html file

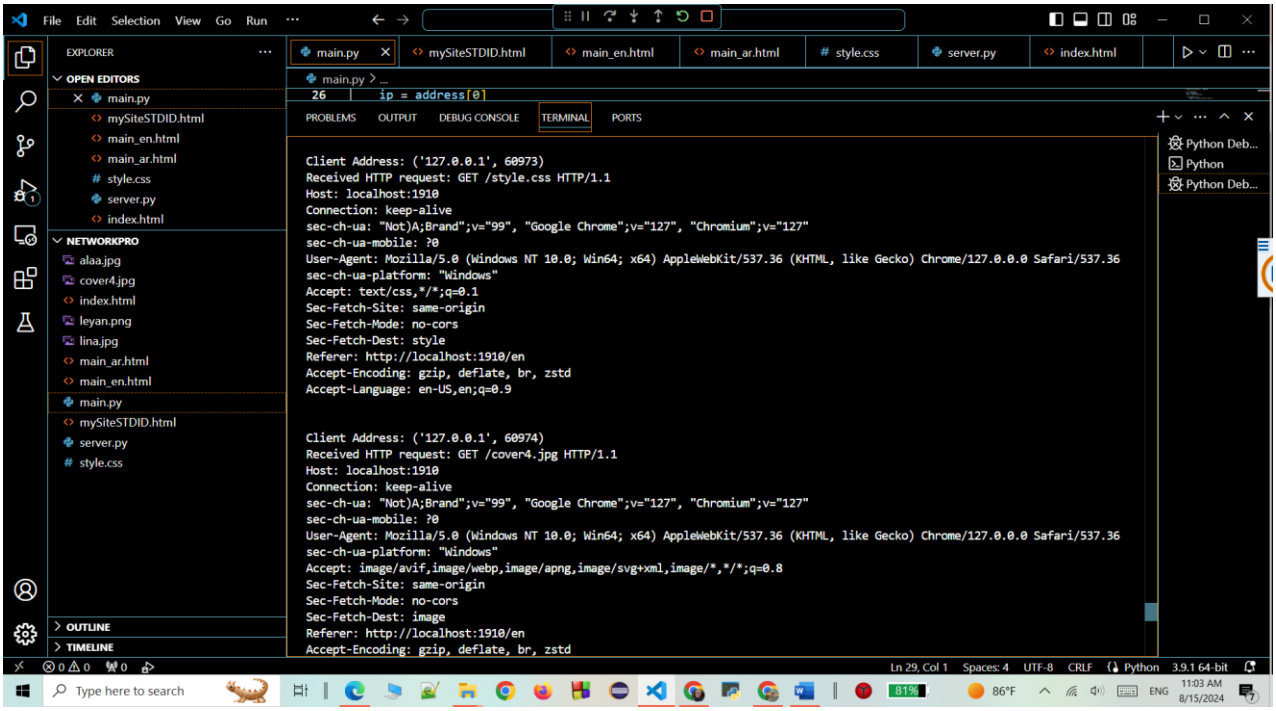


Figure 43: Response the main_en.html file1

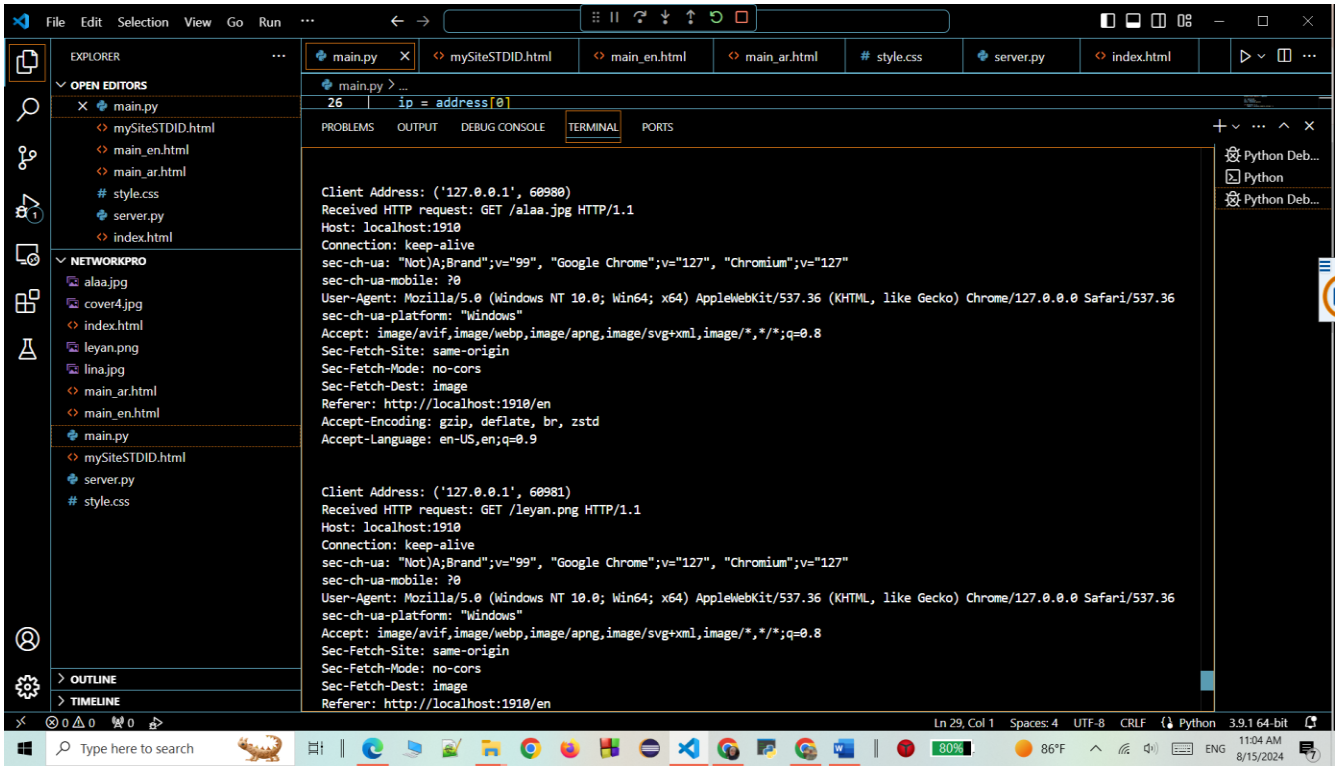


Figure 44: Response the main_en.html file

➢ ***The local HTML file:***

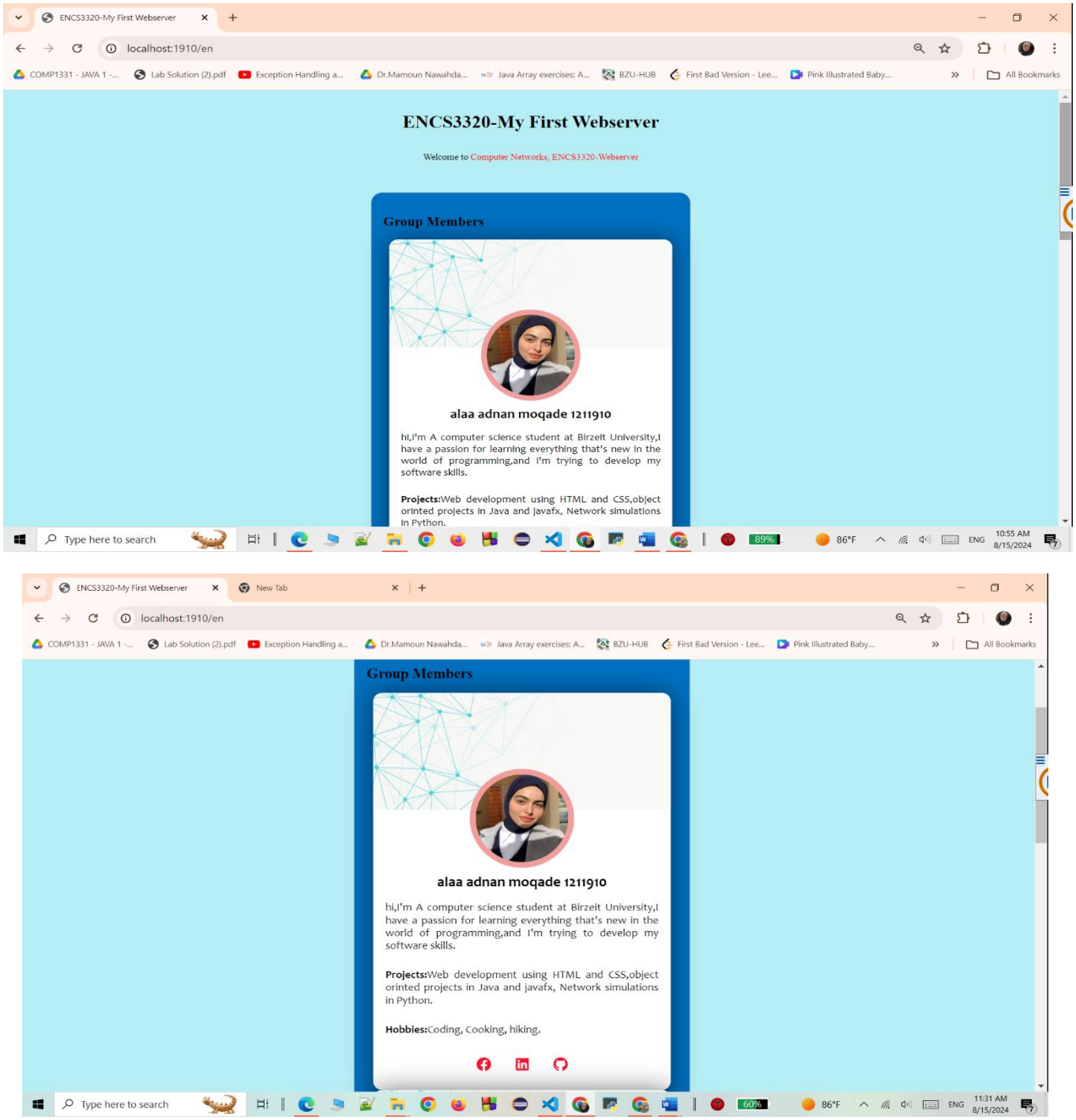


Figure 45: alaa profile

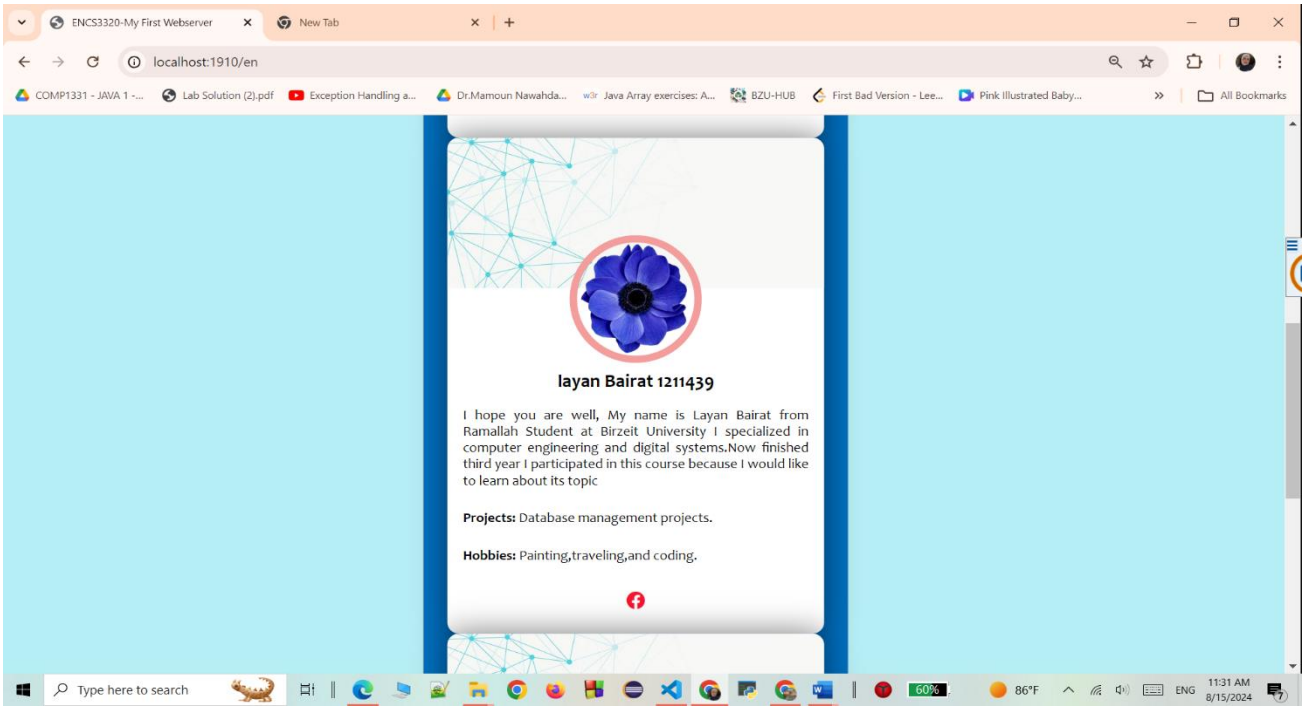


Figure 46: Leyan profile

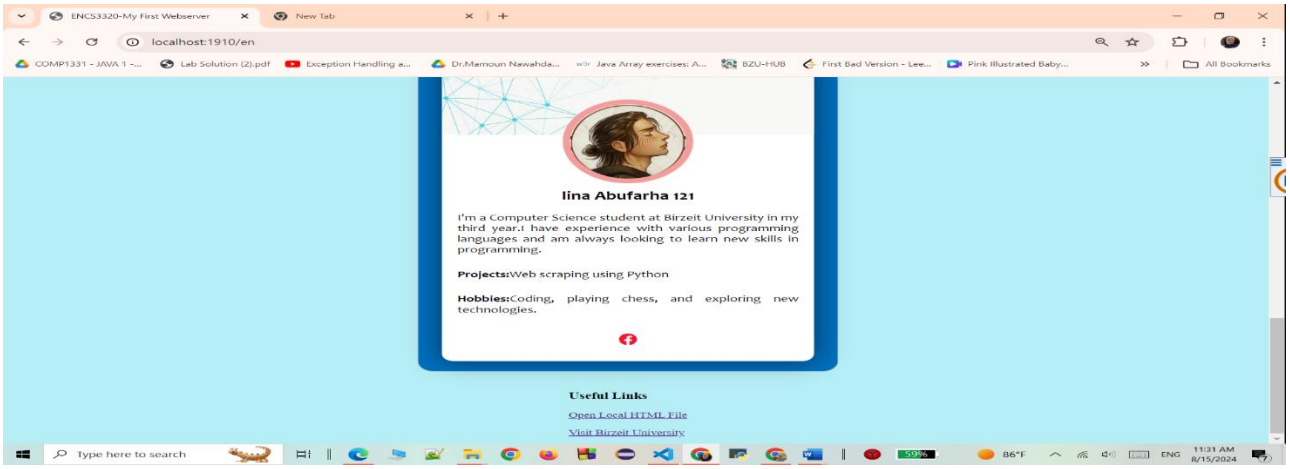


Figure 47: Leena profile

3.2 Request Arabic version*:* 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 .

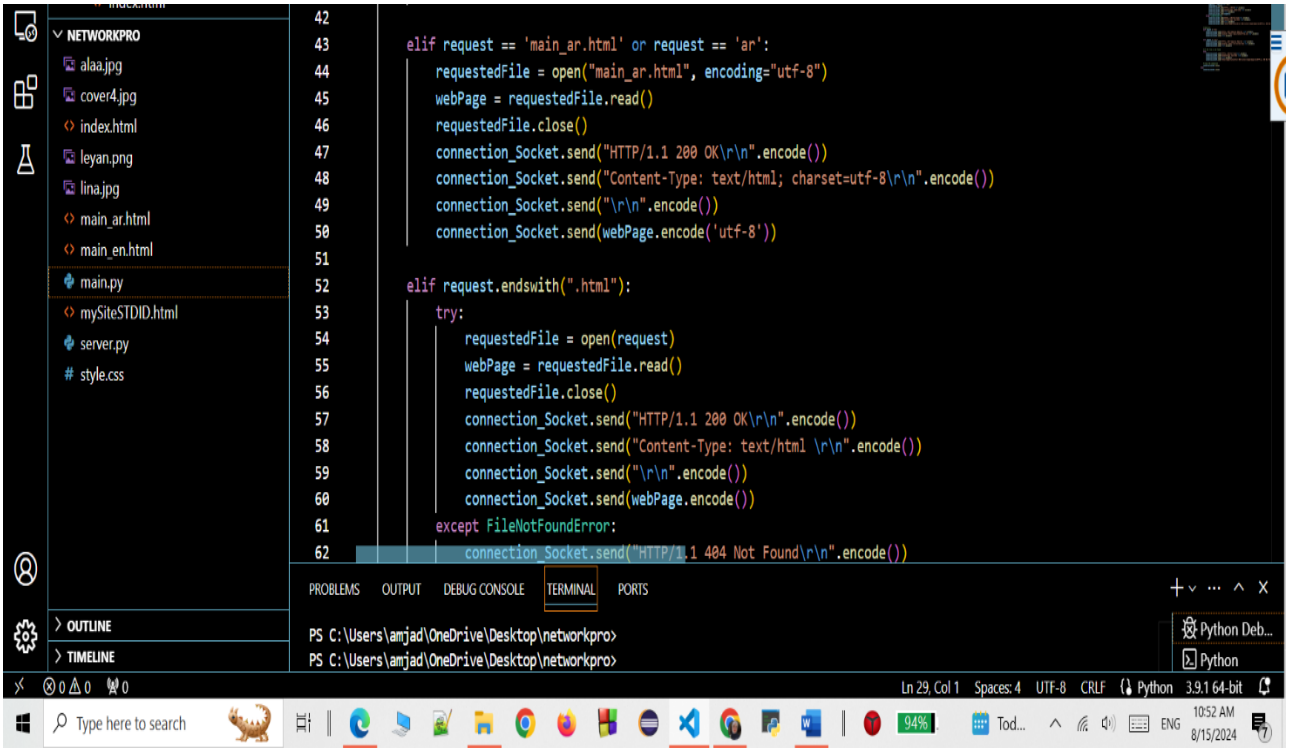


Figure 48:request in Arabic versio

response in Arabic version:

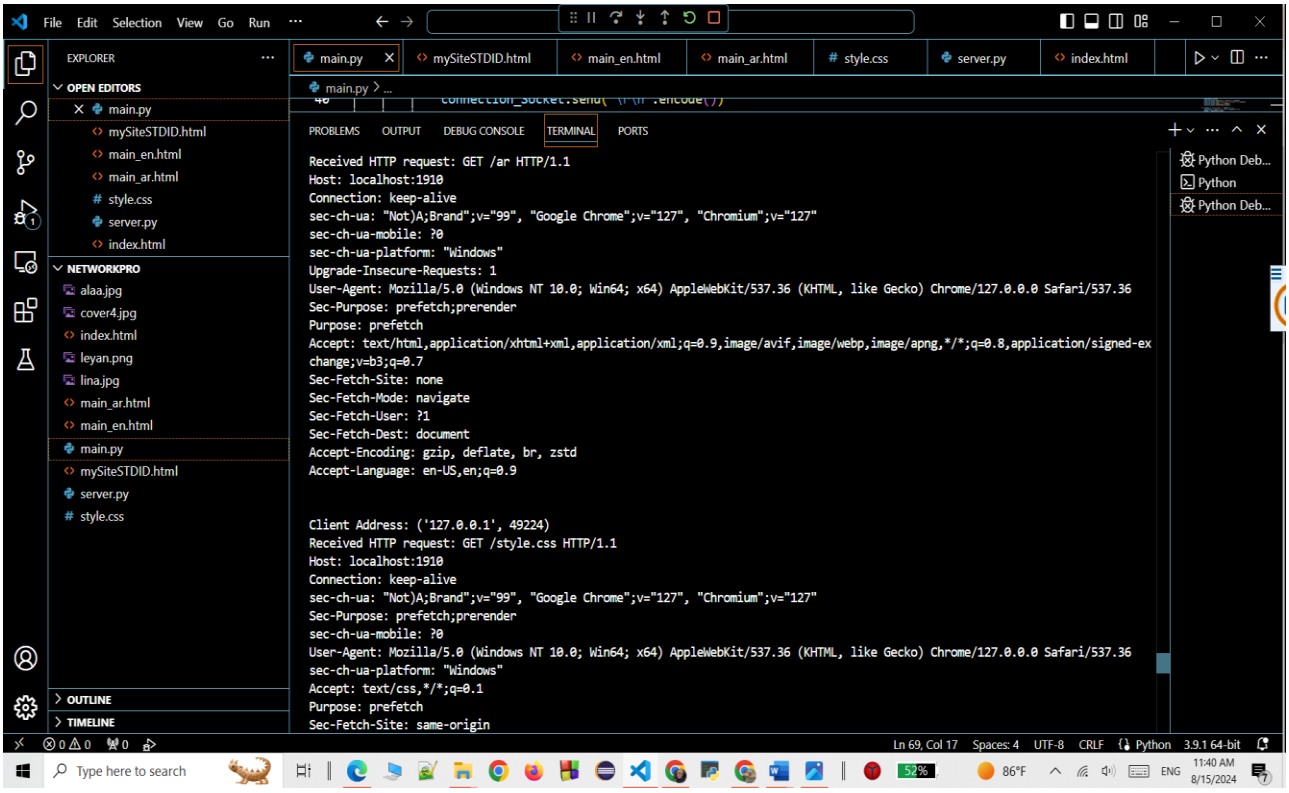


Figure 49response in Arabic version1

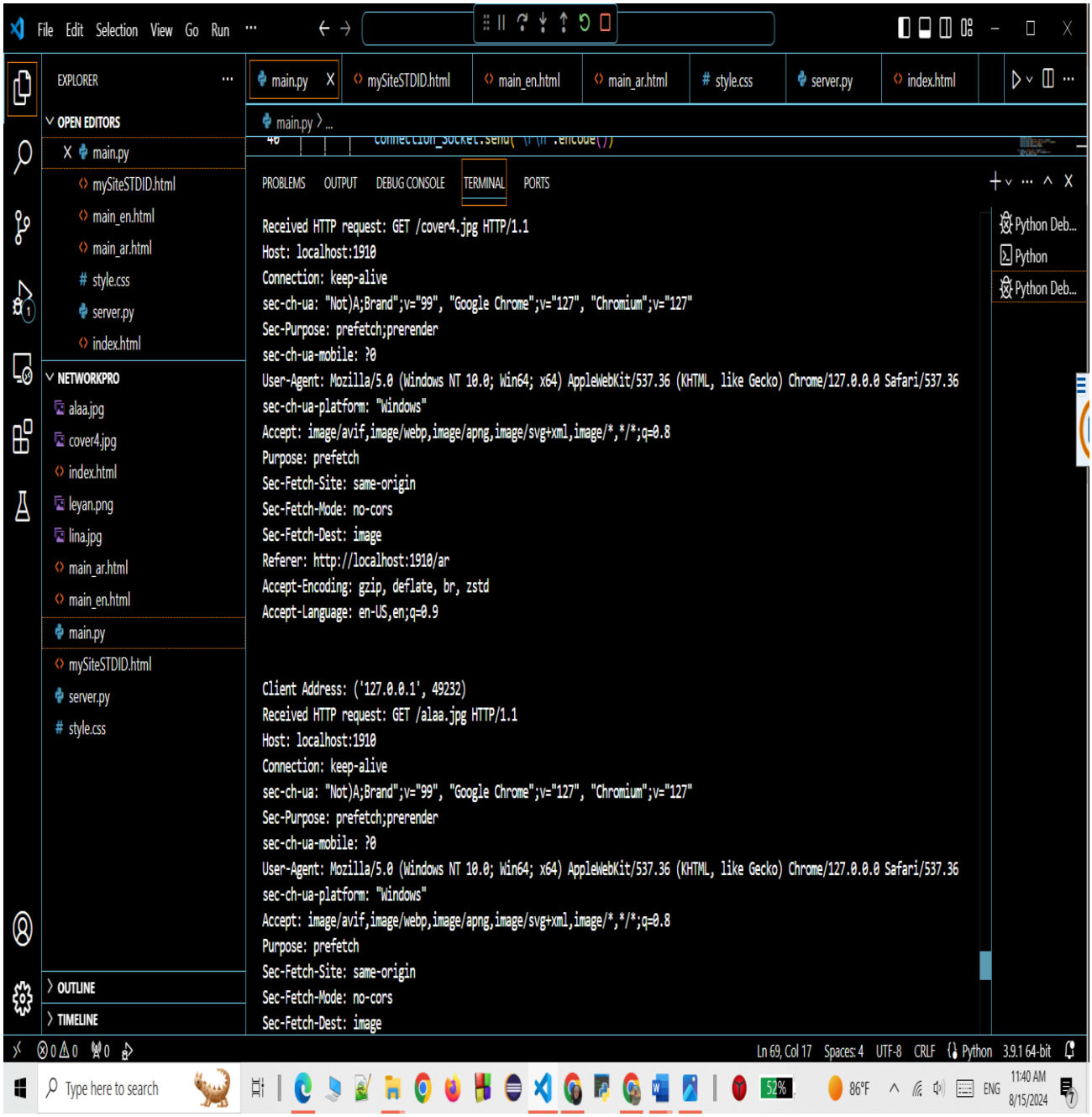


Figure 50response in Arabic version2

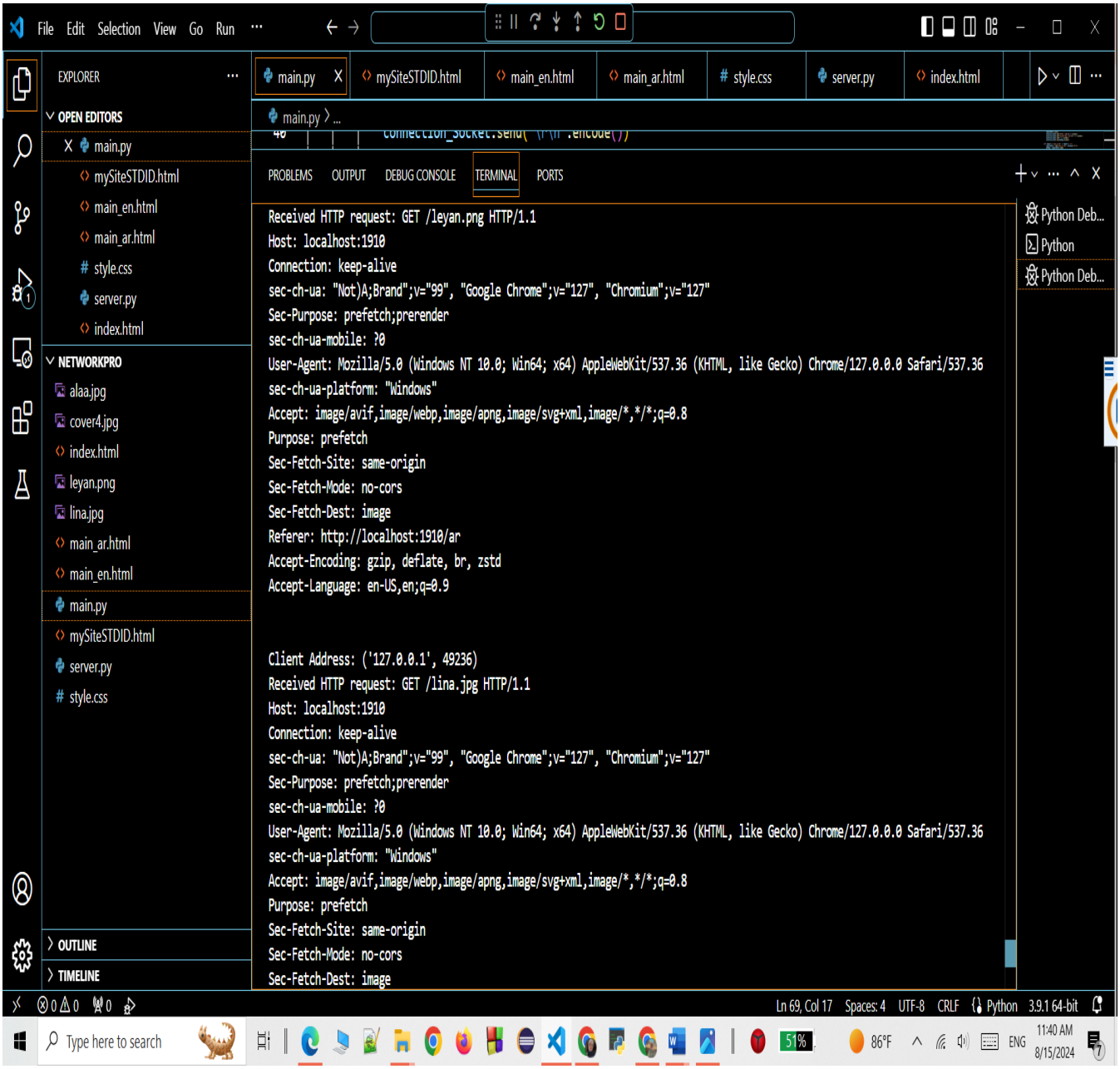


Figure 51response in Arabic version 3

The main_ar.html:

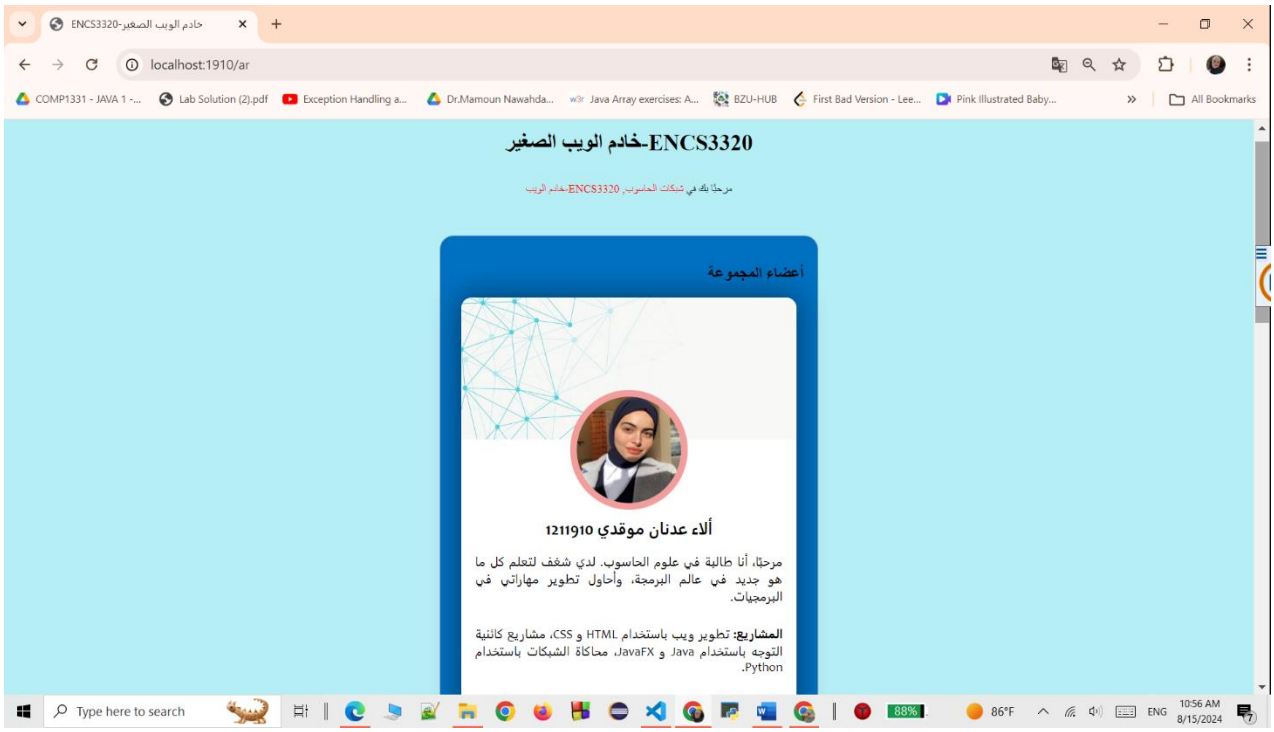


Figure 52main_ar.html alaa profile

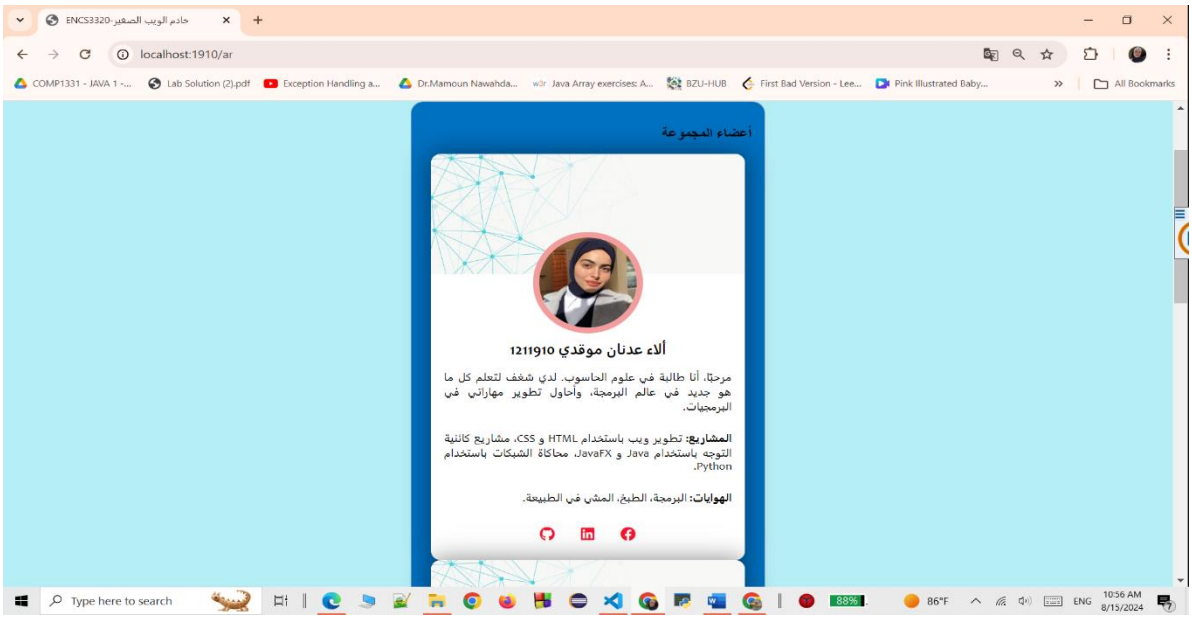


Figure 53main_ar.html

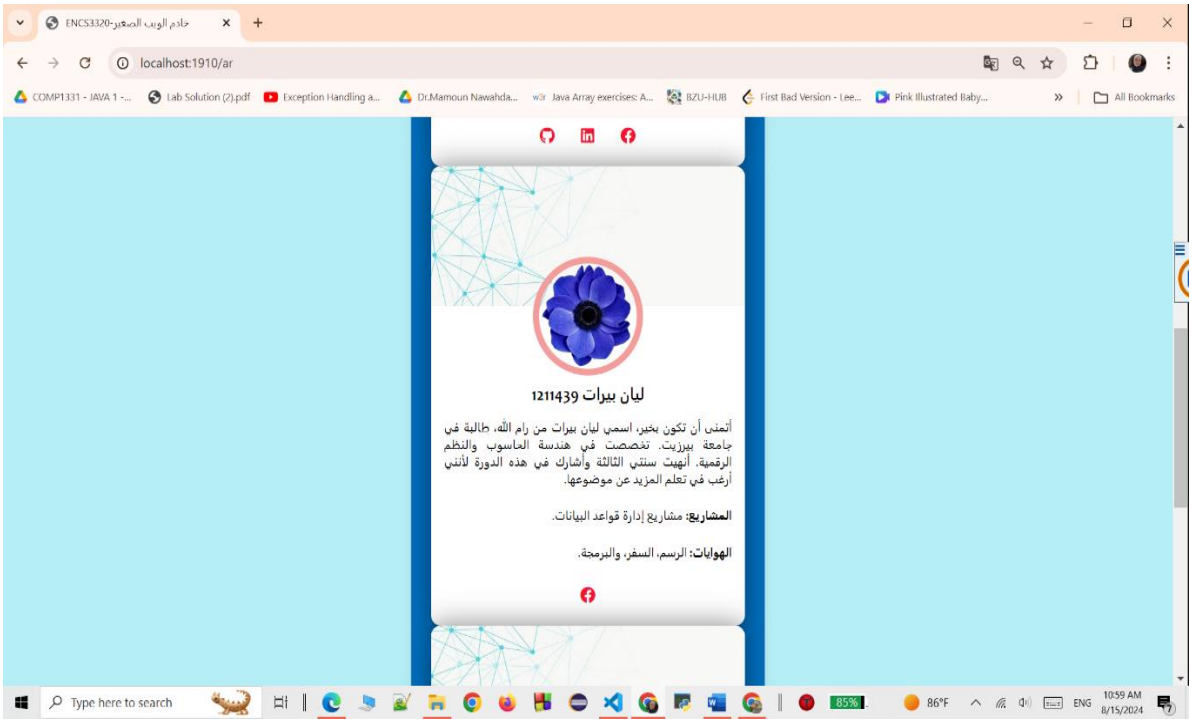


Figure 54main_ar.html leyan profile

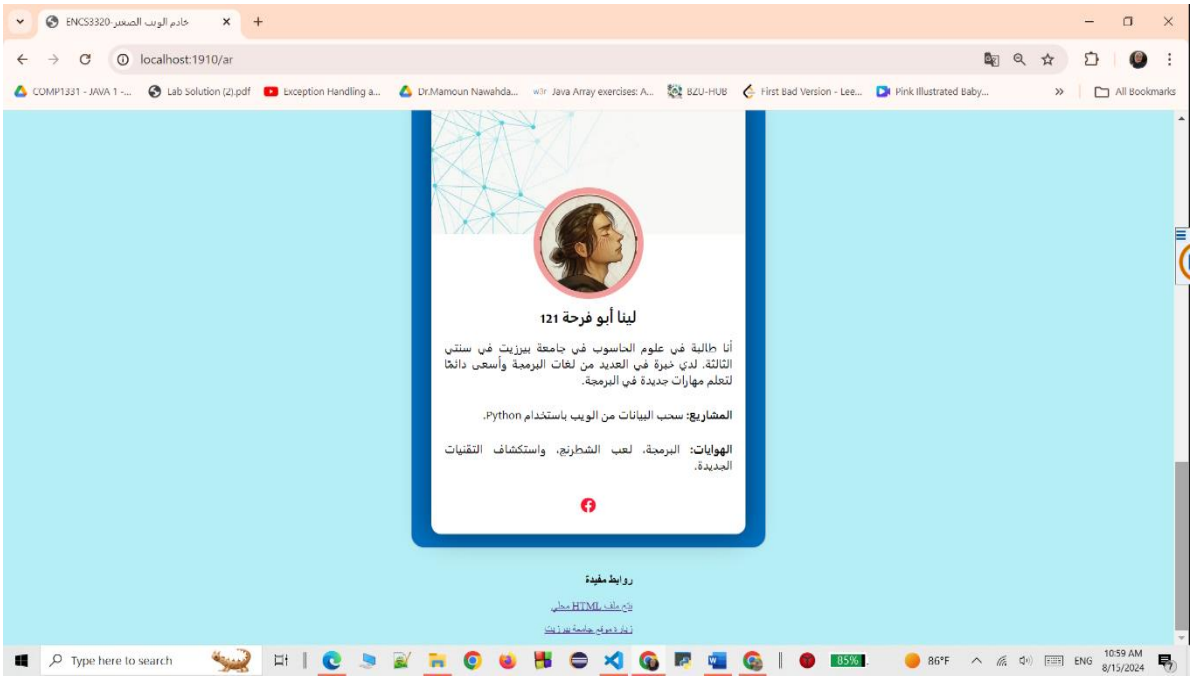


Figure 55main_ar.htmllena

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 ,Then 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**

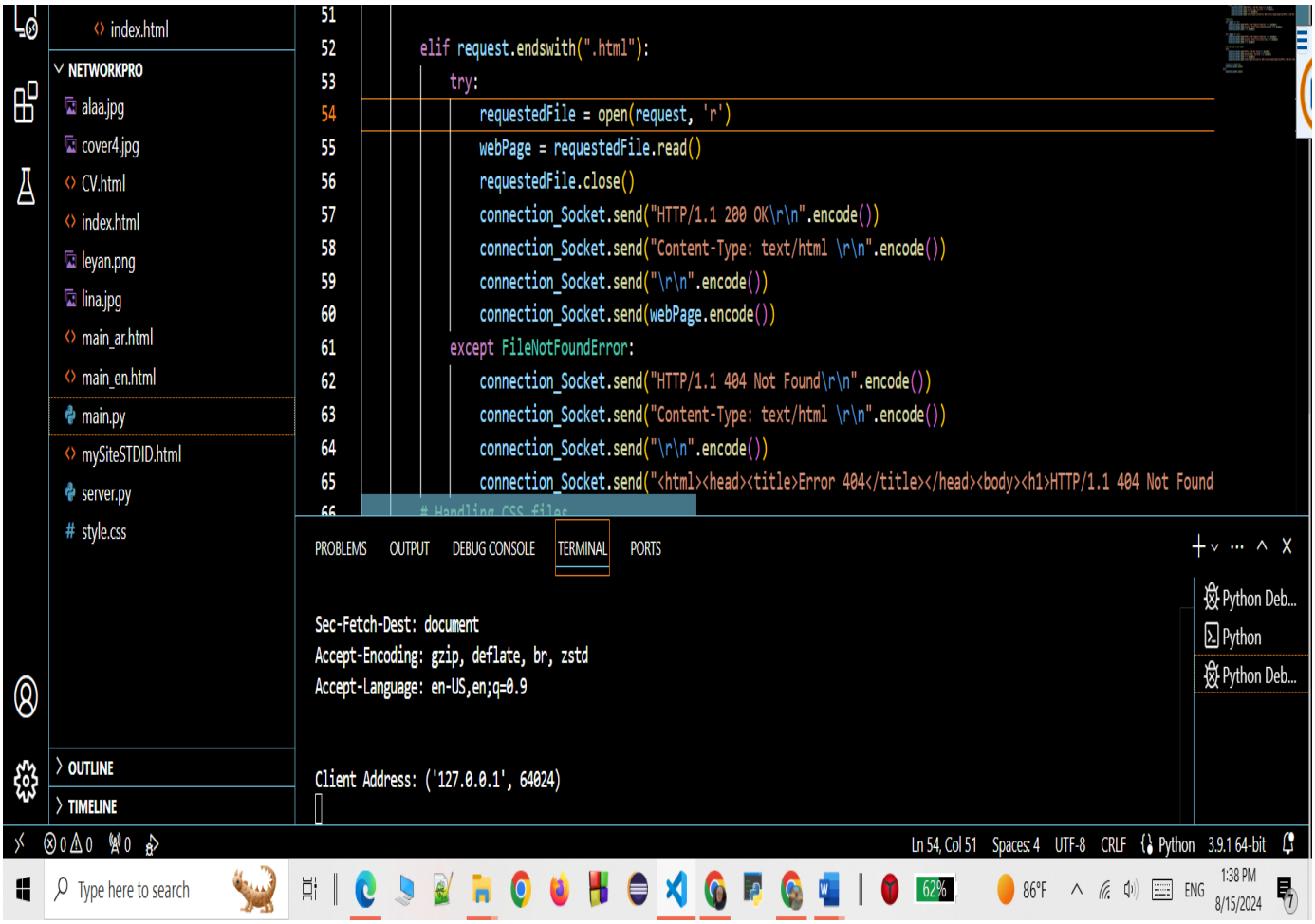
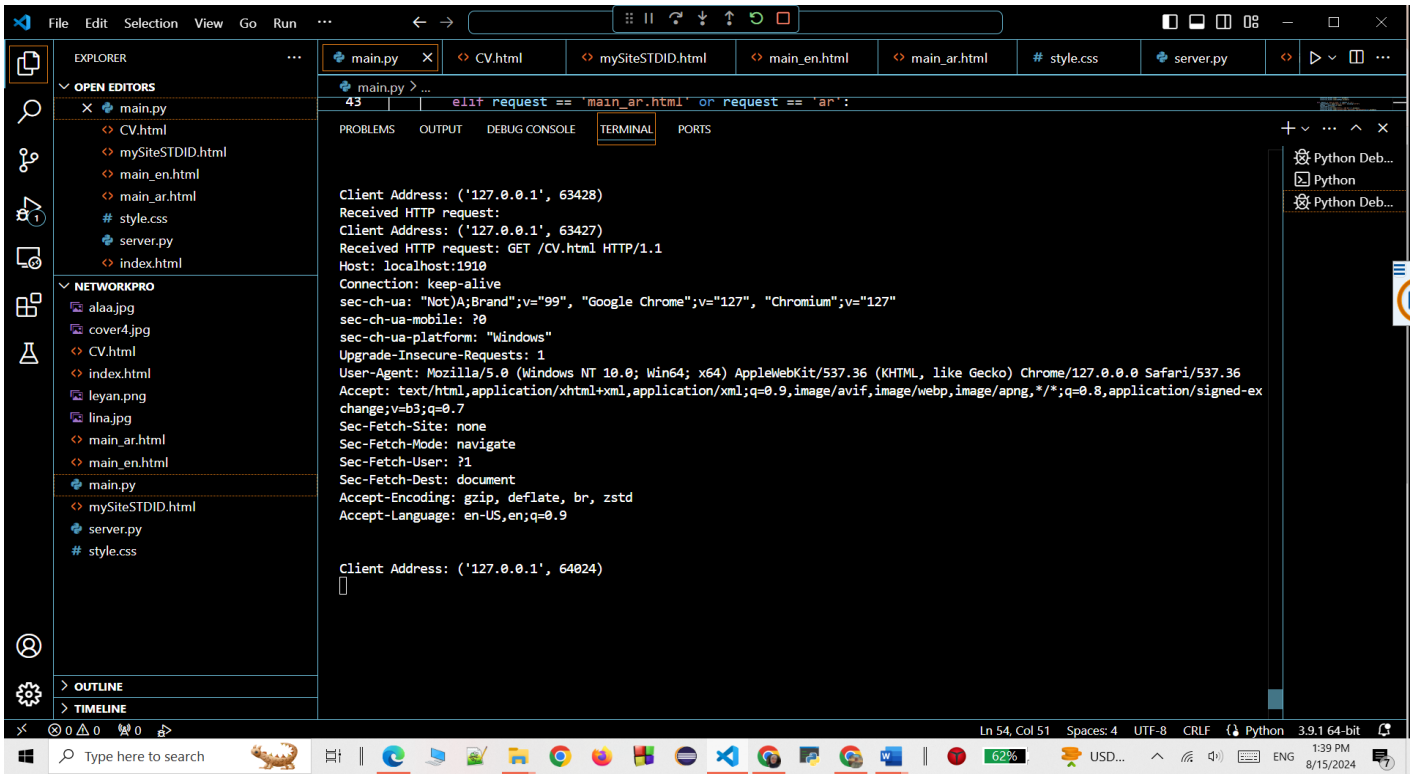


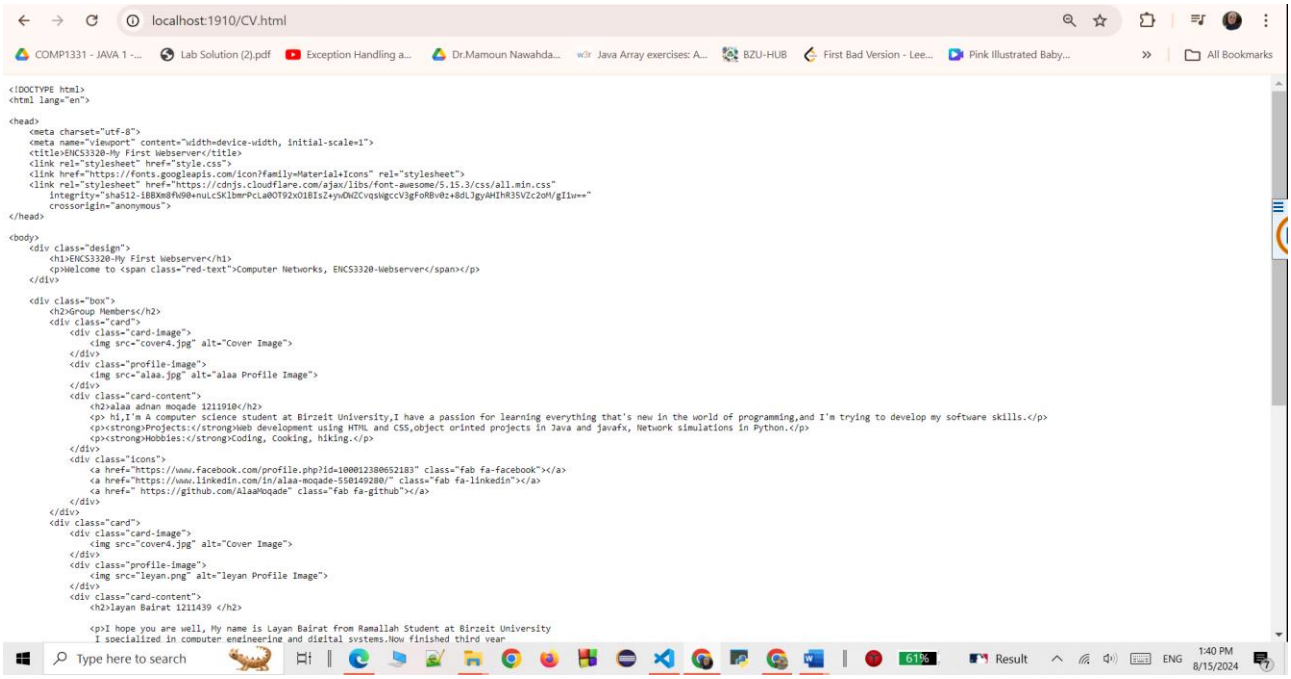
Figure 56Request the html file

➢ The Html Response:



figuer57: create html request

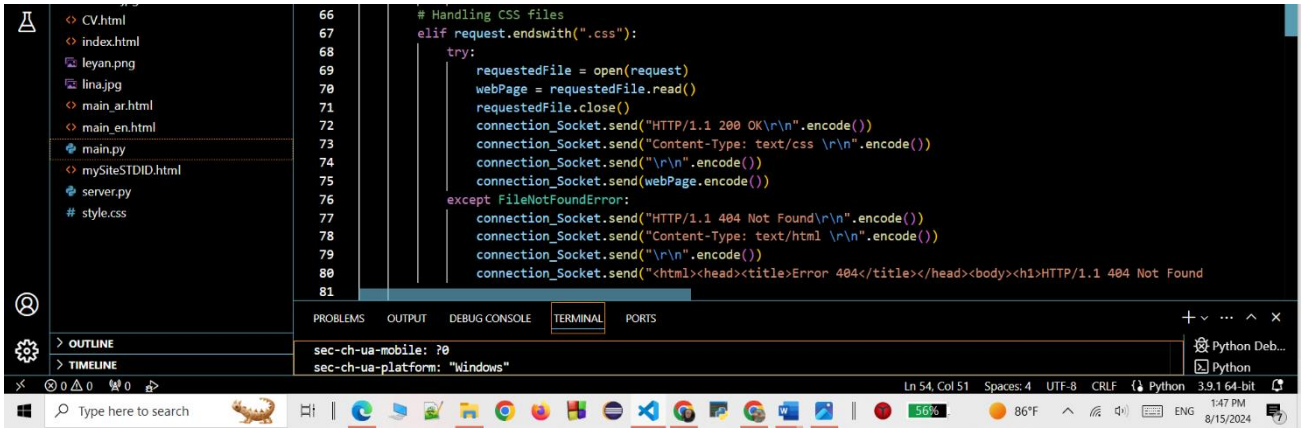
- ***The Html request (CV.html):***



figuer58: html reguest as txt

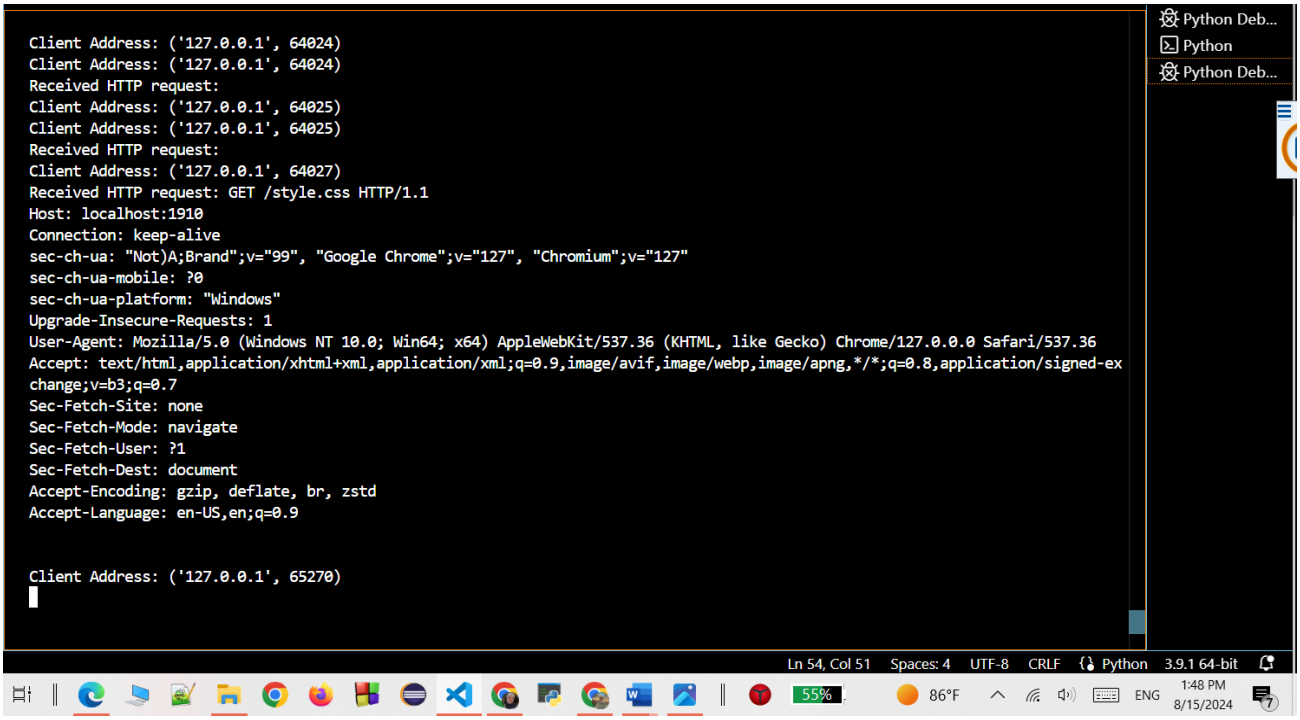
3.4 Request the CSS file:

This code checks if the request is for a CSS file, attempts to serve it, and if not found, it returns a 404 error page



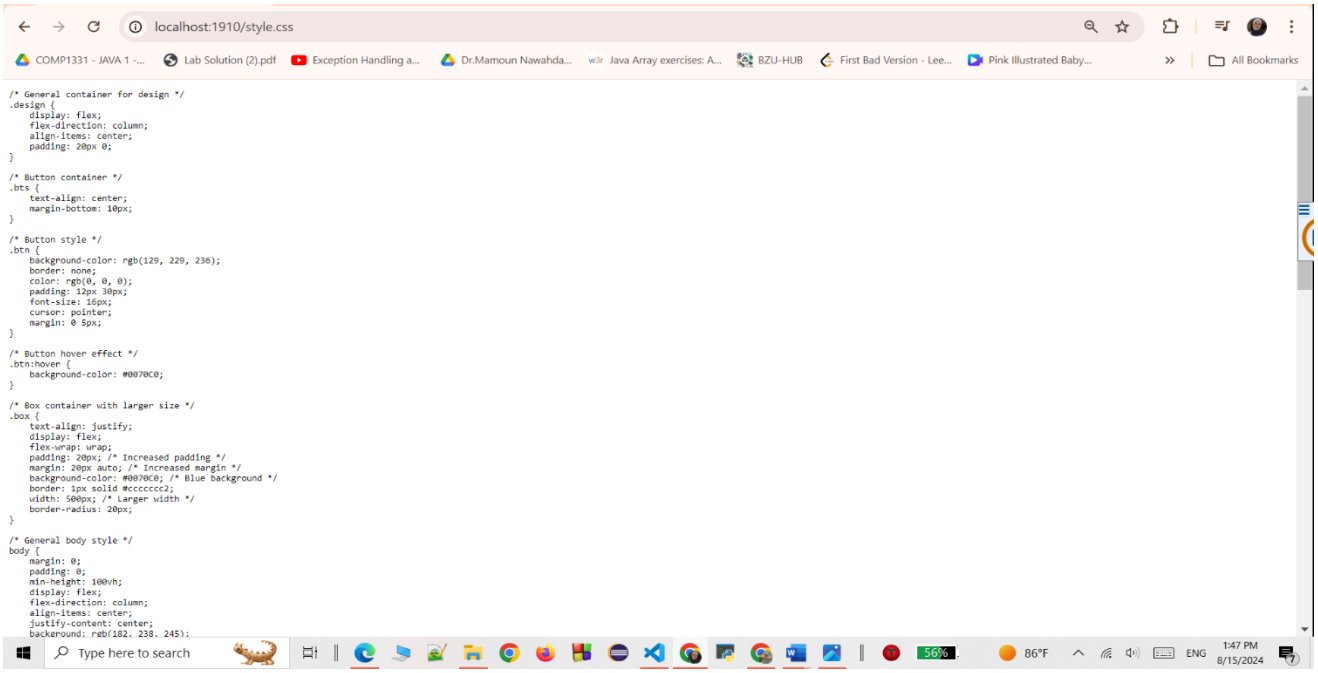
figuer 59: create css request

➢ The CSS Response:



figuer 60 : css response

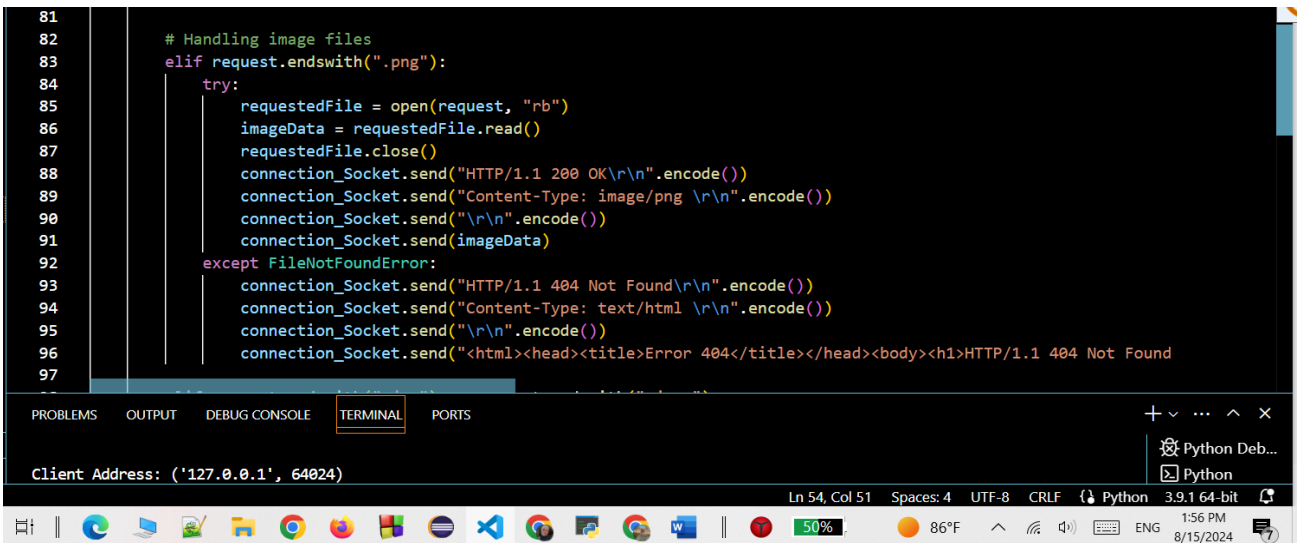
➢ ***The CSS Request:***



figuer 61 : css request as txt

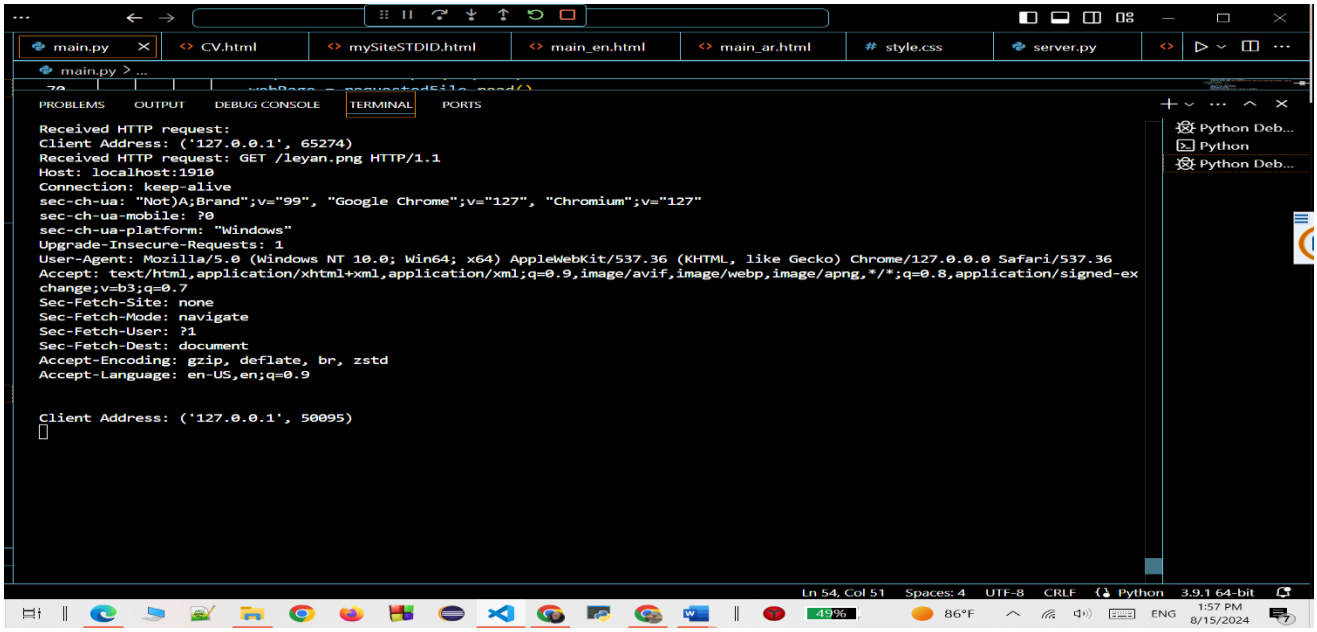
3.5. Request an image with PNG format:

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

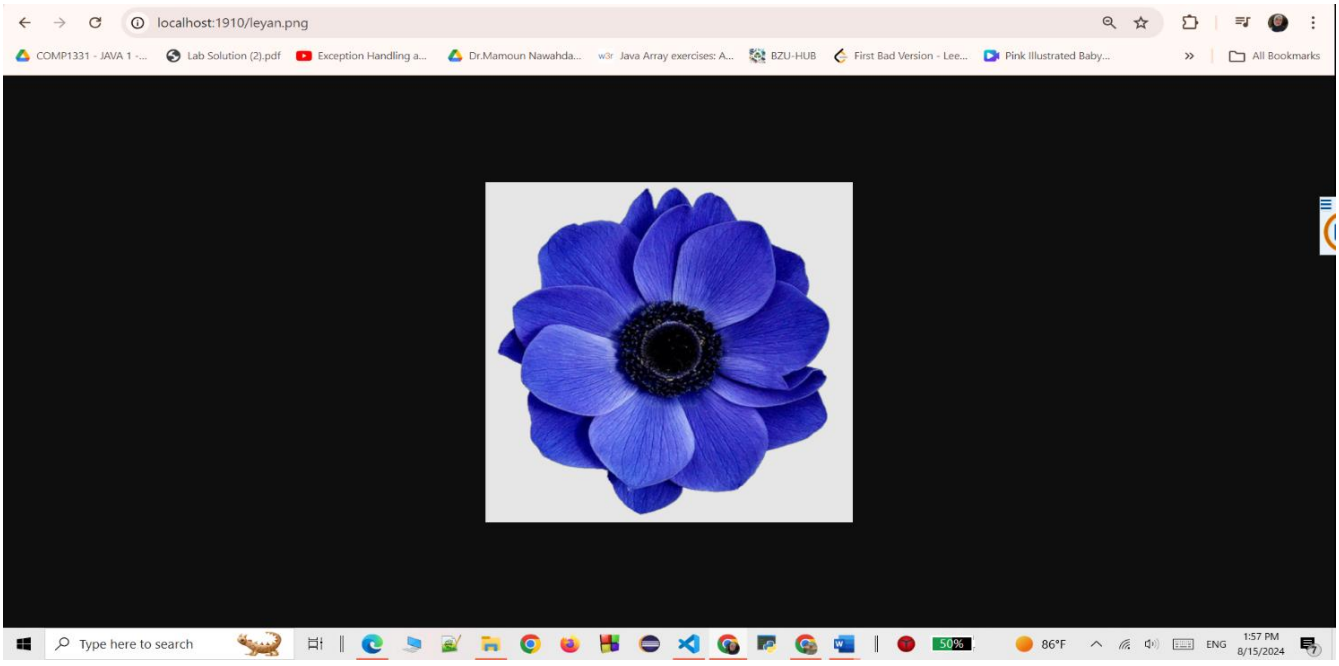


figuer 62: png request image

The response:



figuer 63: png response



figuer 64: png image

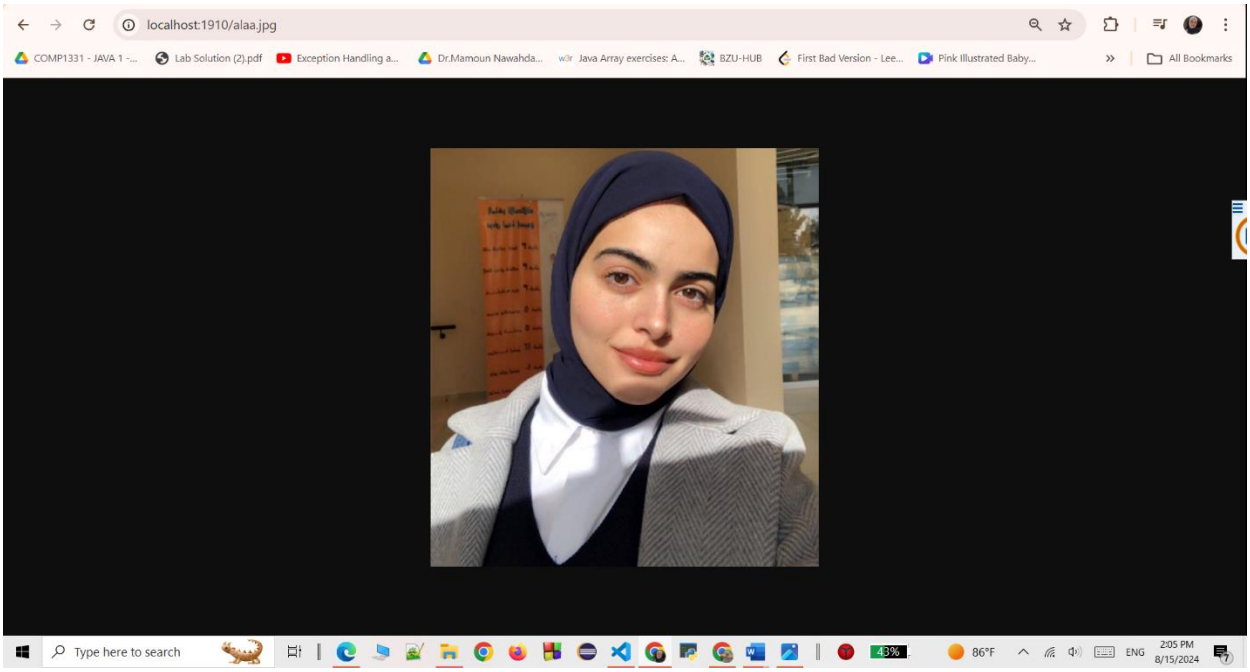
3.6. Request an image with JPG format:

```
97
98      elif request.endswith(".jpg") or request.endswith(".jpeg"):
99          try:
100             requestedFile = open(request, "rb")
101             imageData = requestedFile.read()
102             requestedFile.close()
103             connection_Socket.send("HTTP/1.1 200 OK\r\n".encode())
104             connection_Socket.send("Content-Type: image/jpeg \r\n".encode())
105             connection_Socket.send("\r\n".encode())
106             connection_Socket.send(imageData)
107         except FileNotFoundError:
108             connection_Socket.send("HTTP/1.1 404 Not Found\r\n".encode())
109             connection_Socket.send("Content-Type: text/html \r\n".encode())
110             connection_Socket.send("\r\n".encode())
111             connection_Socket.send("<html><head><title>Error 404</title></head><body><h1>HTTP/1.1 404 Not Found
112
```

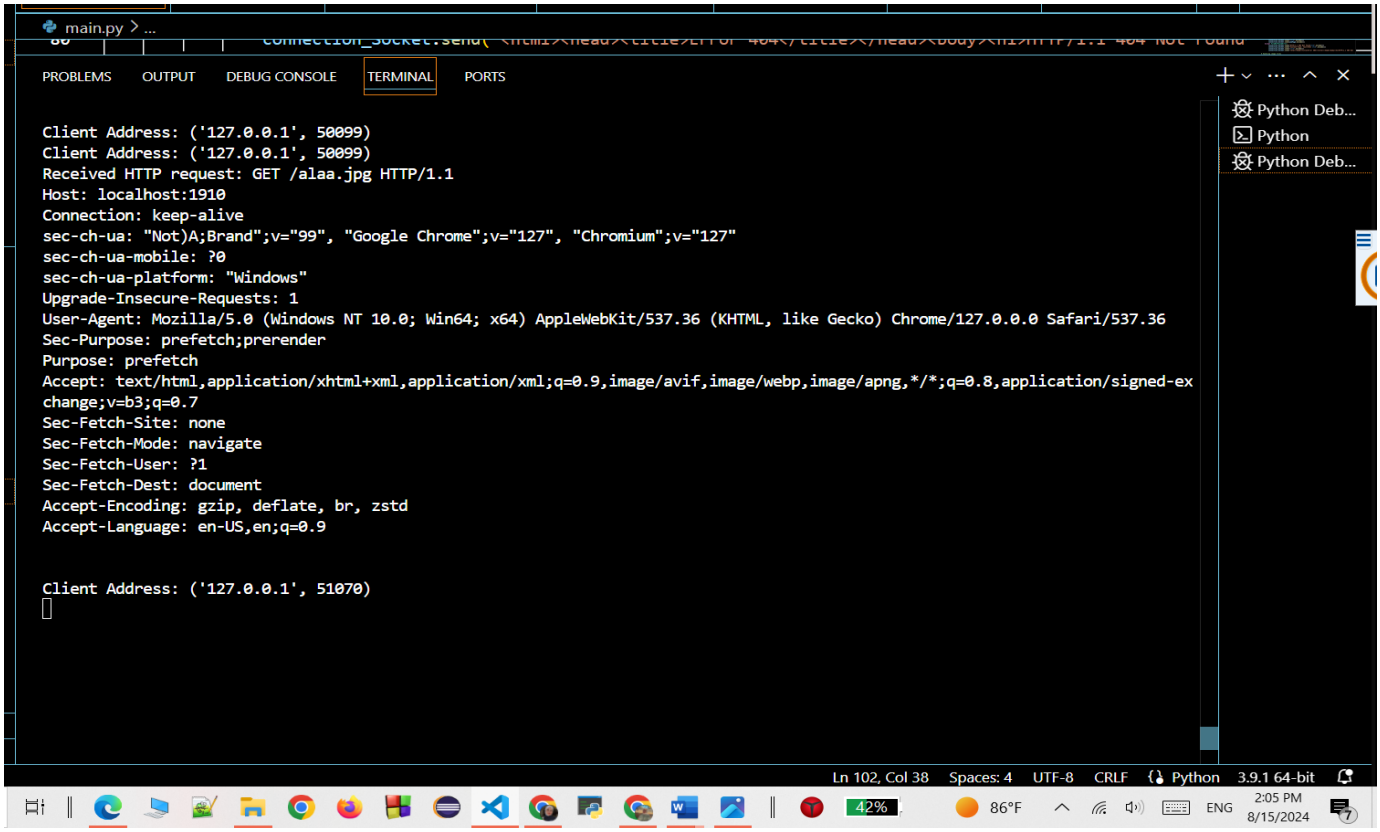
Ln 54, Col 51 Spaces: 4 UTF-8 CRLF Python 3.9.1 64-bit

43% 86°F ENG 2:04 PM 8/15/2024

figuer 65: create jpg request image



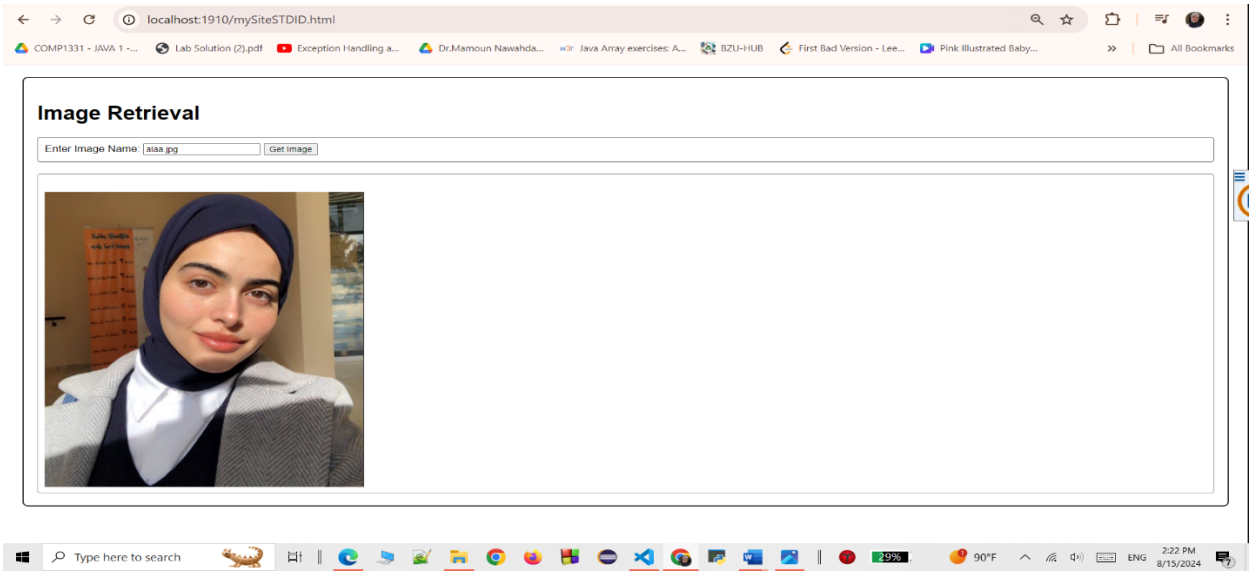
figuer 66: jpg image



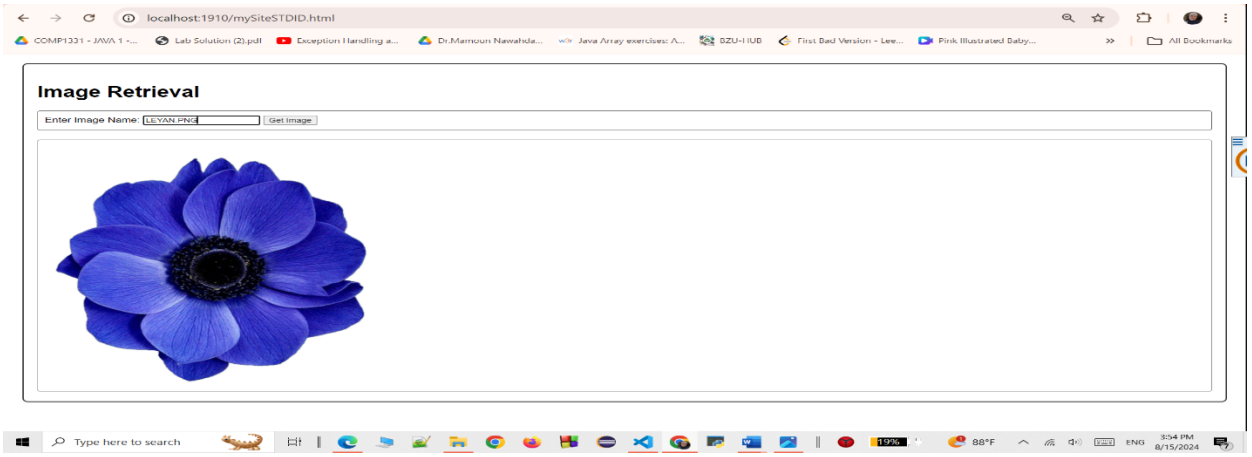
figuer 67: jpg response image

3.7 The HTML file (mySiteSTDID.html) :contains an input box and a submit button. Users can enter the name of the image file they want to retrieve.

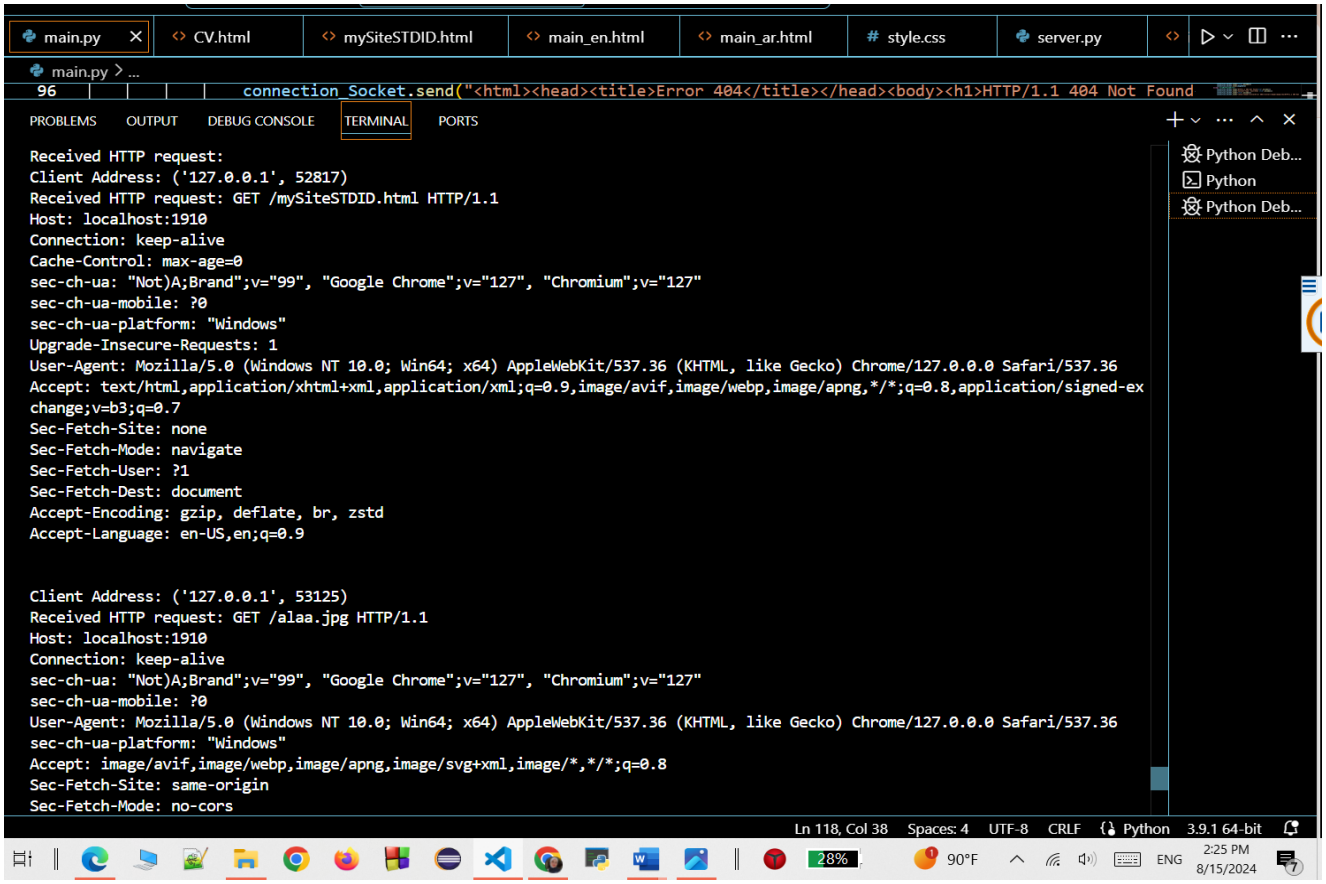
figuer 68: create mySiteSTDID request



figuer 69: jpg image retrieval



figuer 70: png image retrieval

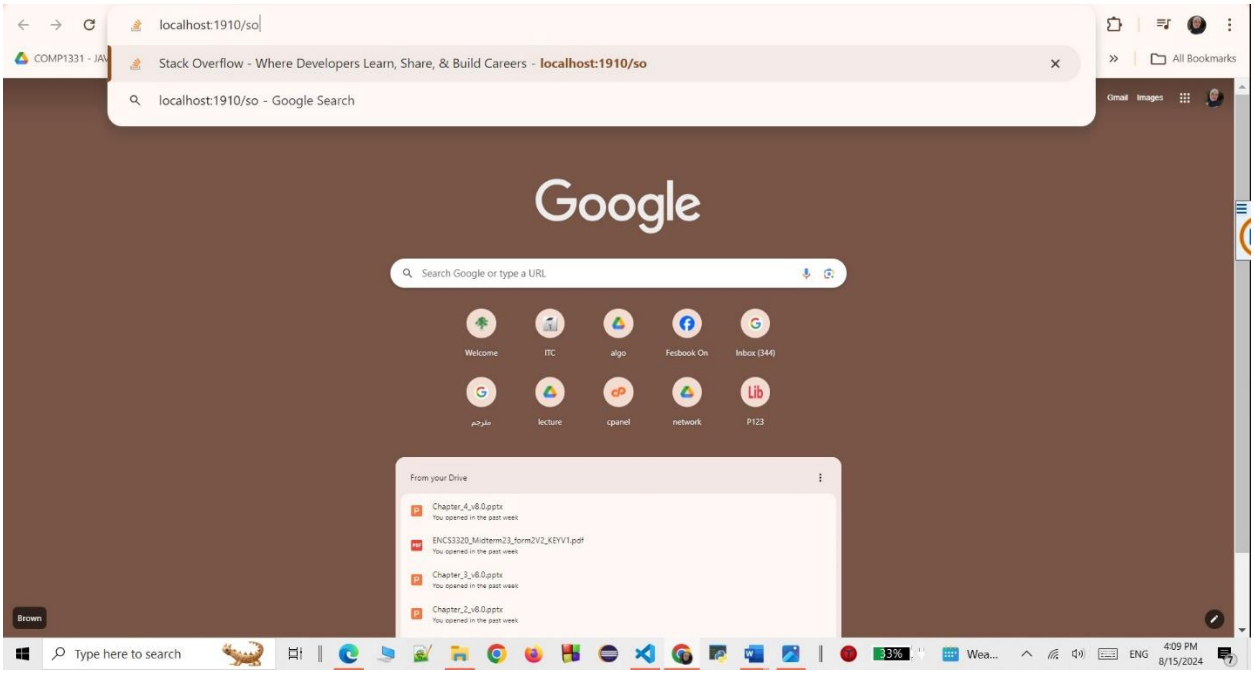


figuer 71: mySitSTDID.html response

3.8. The status code 307 Temporary Redirect: *The server uses a 307 Temporary Redirect to send clients from /so to stackoverflow.com and from /itc to itc.birzeit.edu. The redirect ensures the method and data are preserved during redirection.*

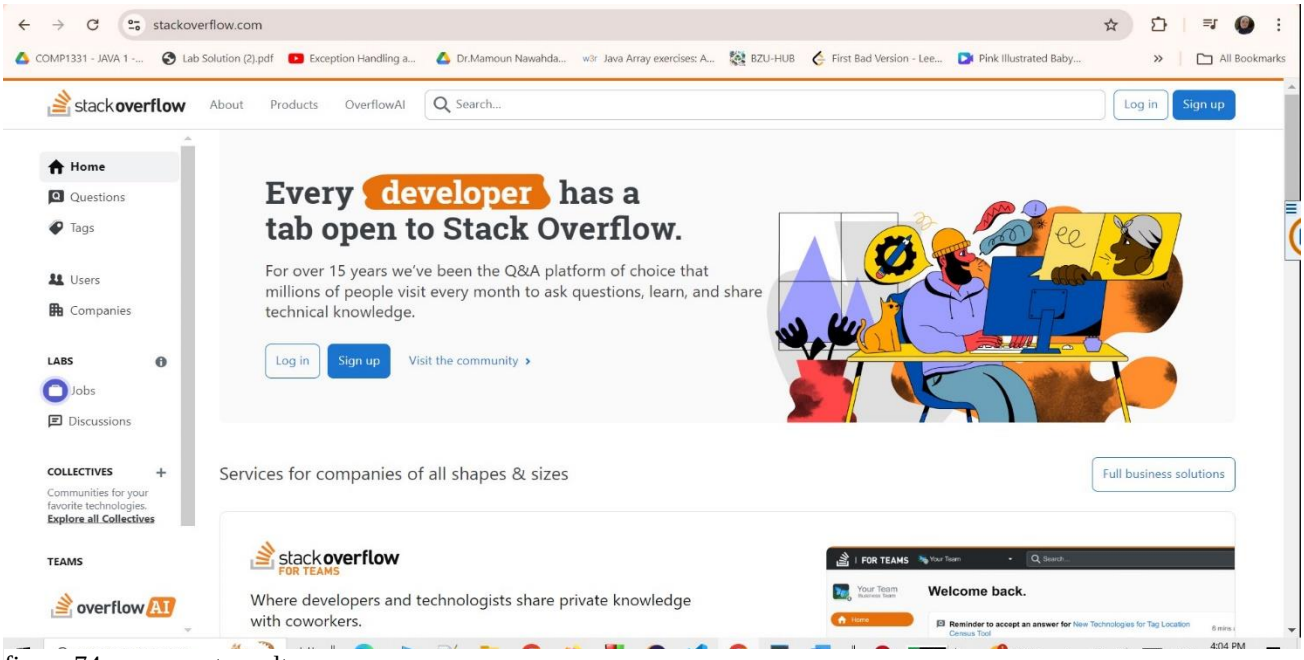
figuer 72: so/itc request

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



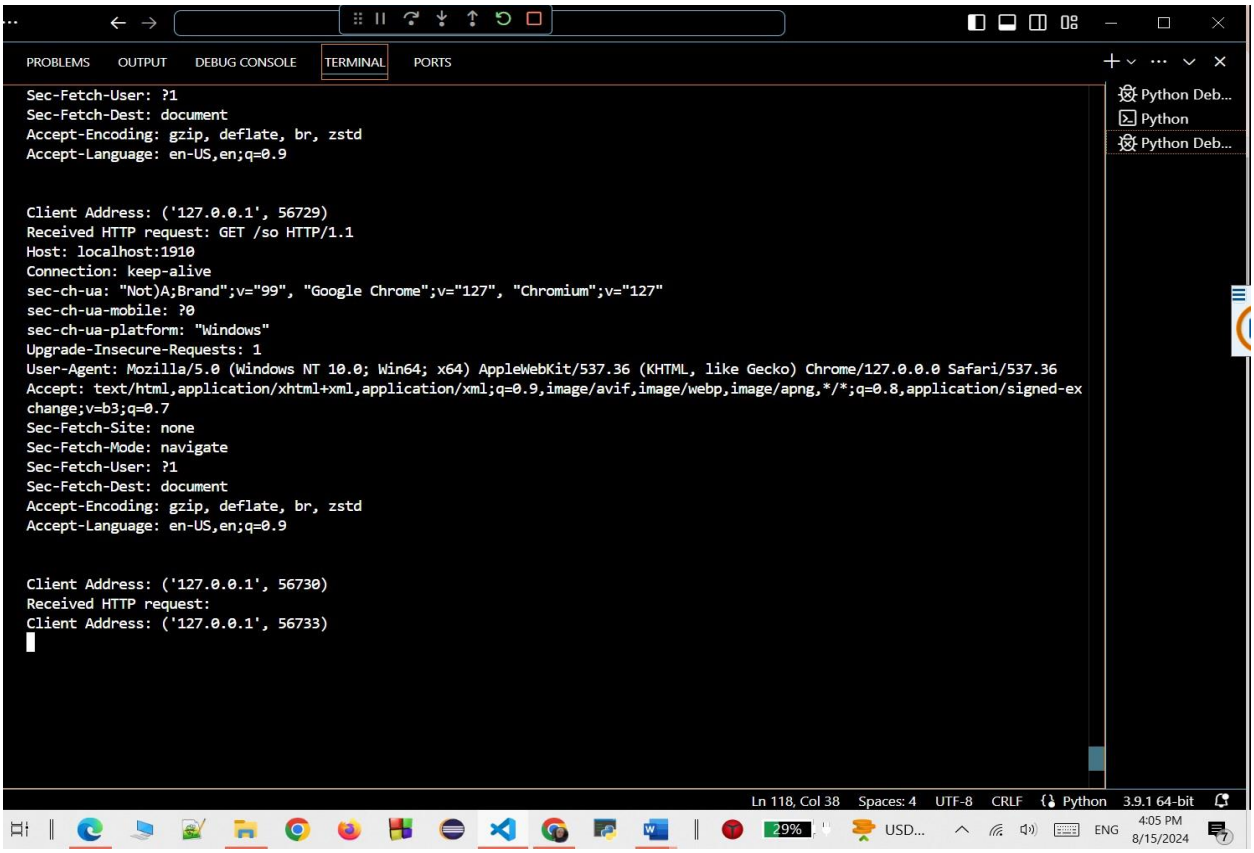
figuer 73: so request

Goes directly to the Stack Overflow website:



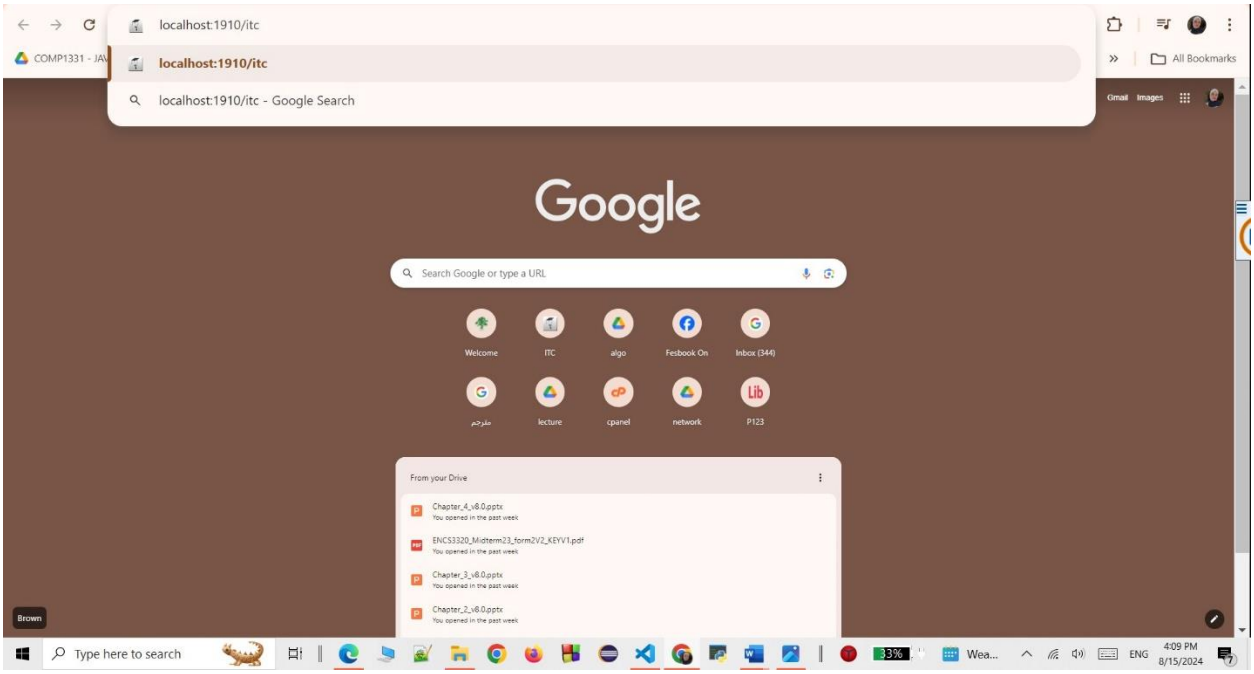
figuer 74: so request result

*The response of **/so:***



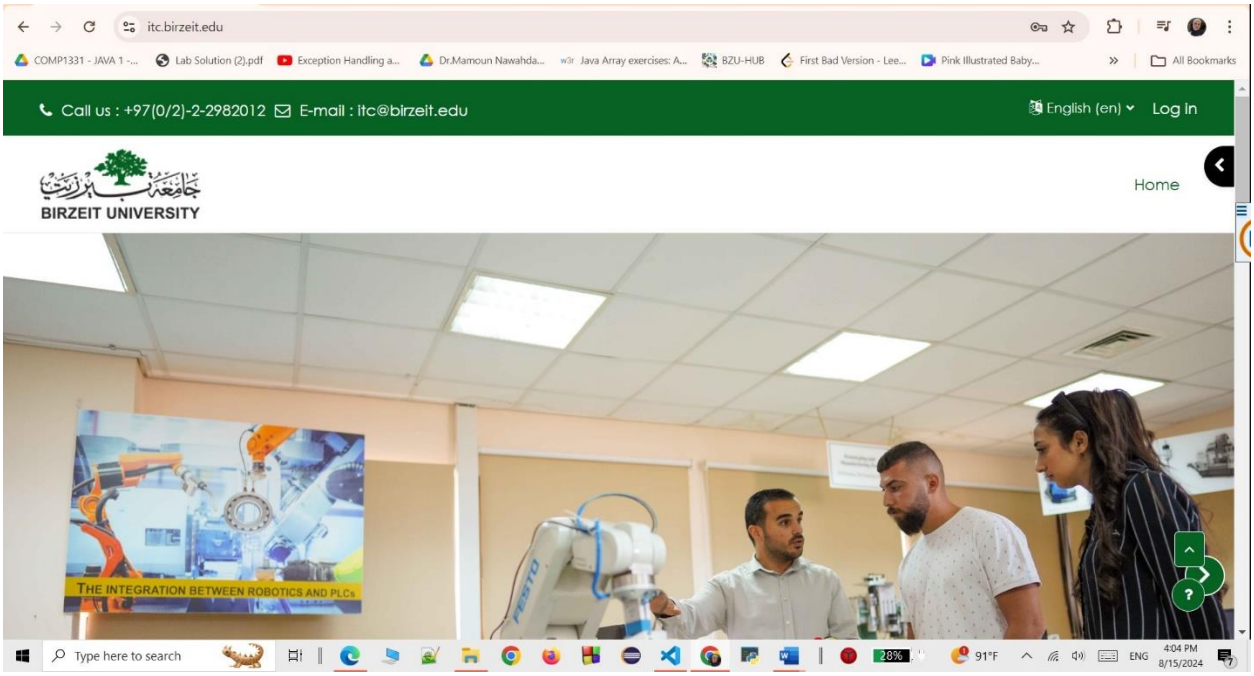
figuer 75: so response

B.*If the request is /itc then redirect to itc website:*



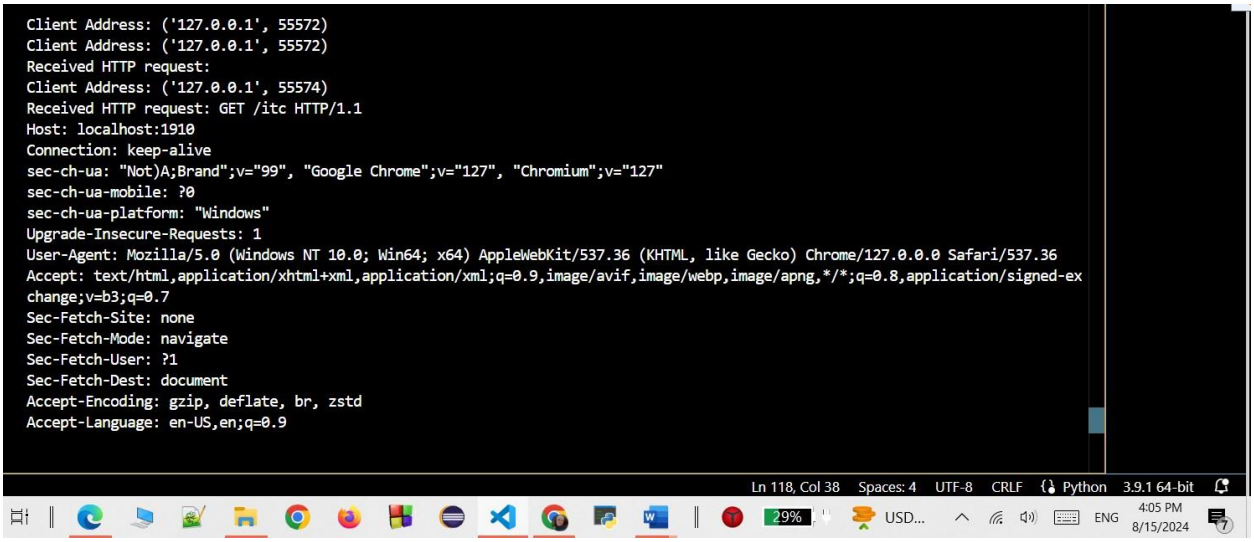
figuer 76: itc request

Goes directly to the itc website:



figuer 77: itc result

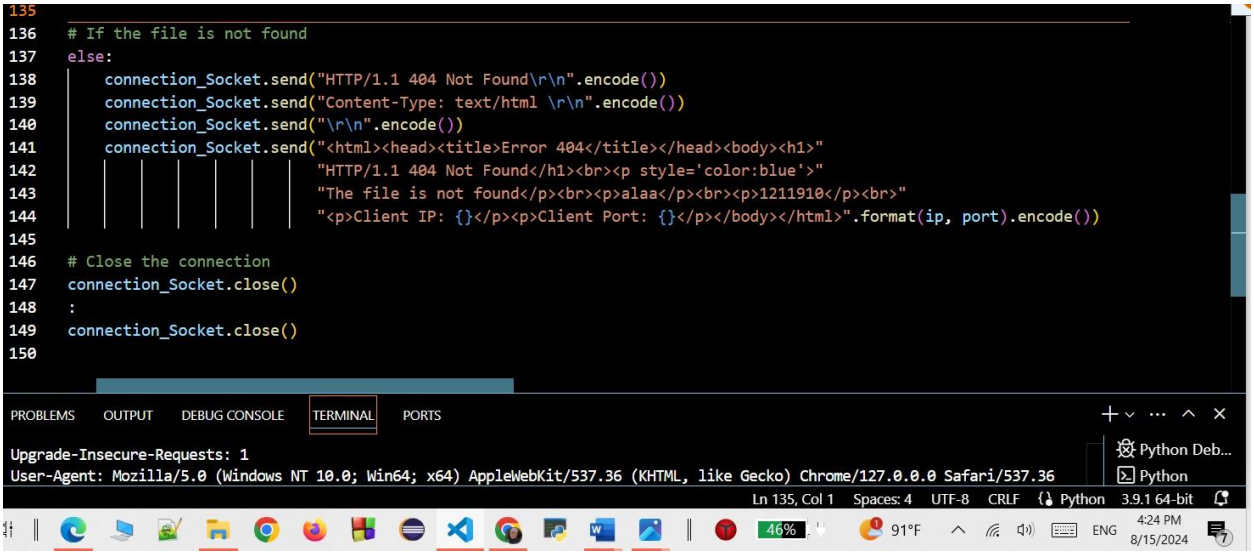
*The response of **/itc:***



figuer 78: itc response

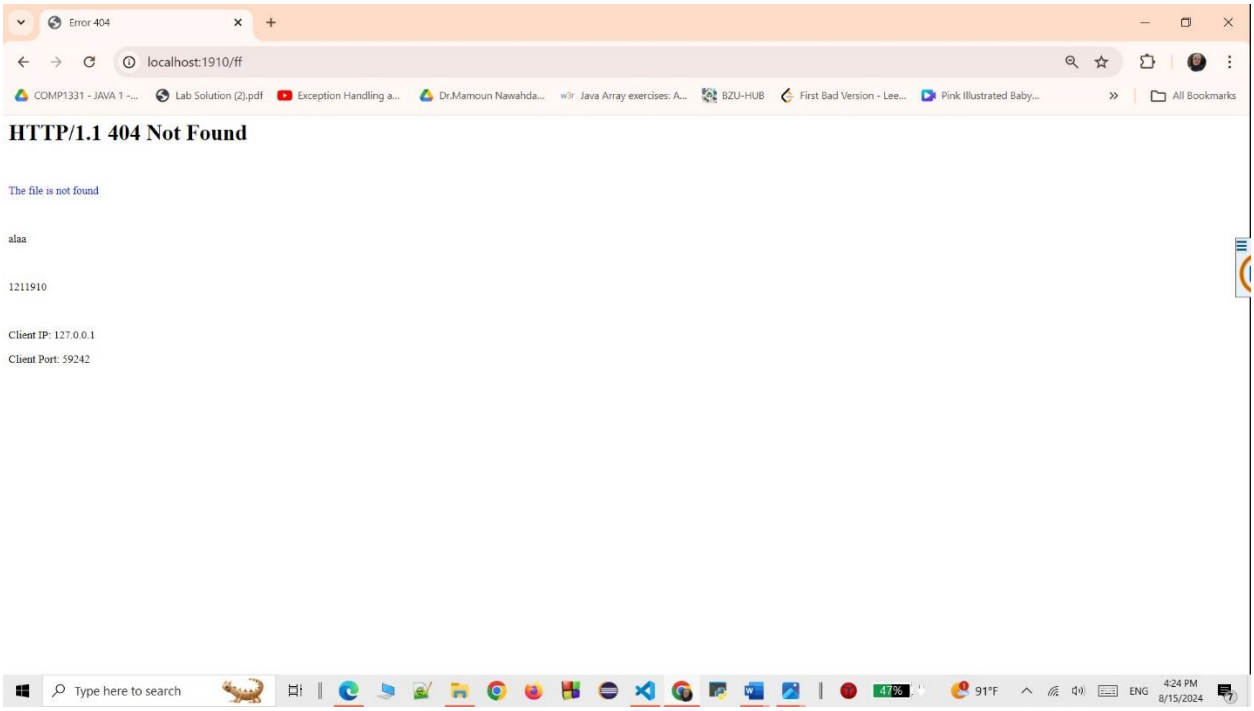
3.10.Wrong request:

If the request is wrong, the server will return an "404 not found" file. And the name and id and IP address for client and port number of client.



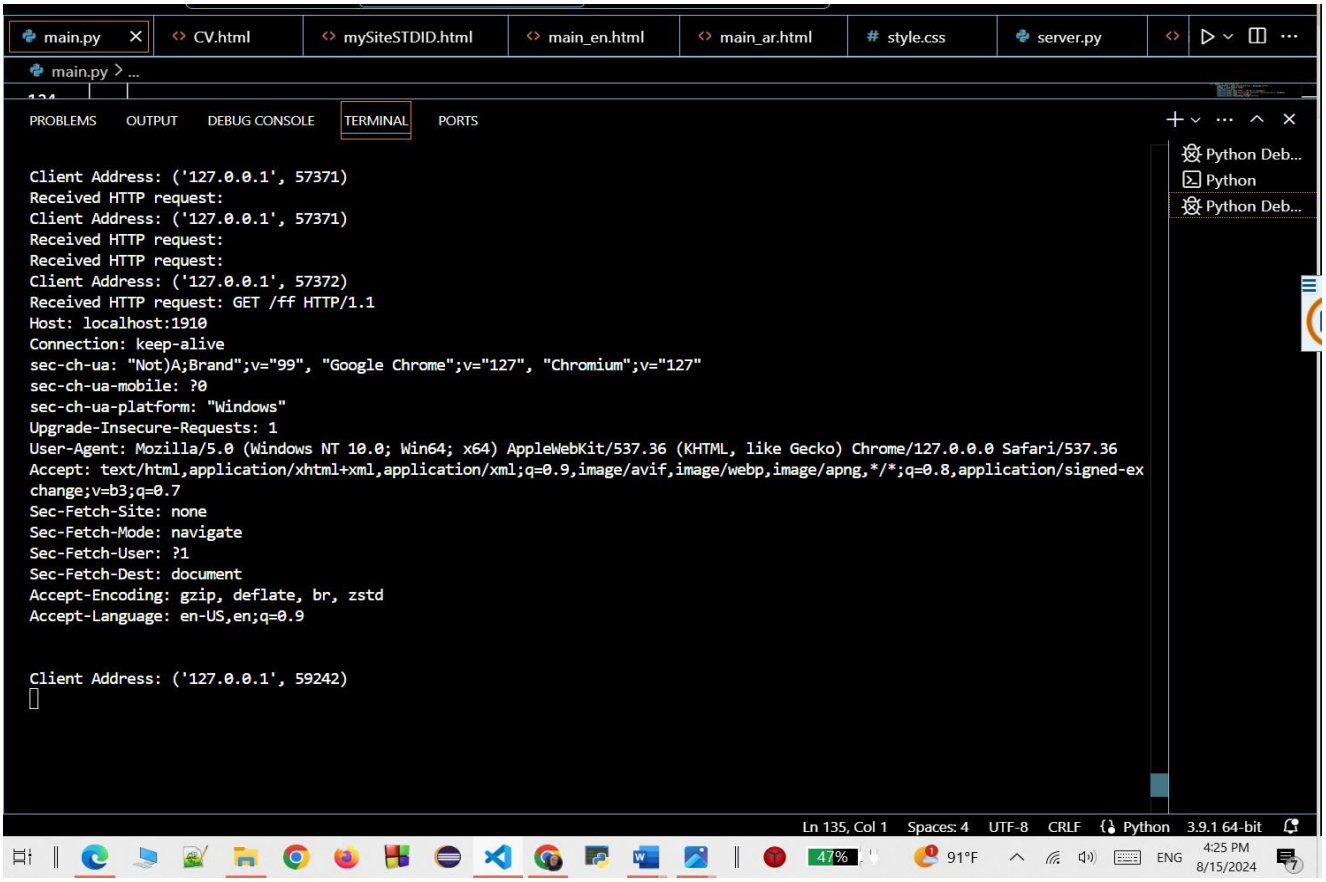
figuer 79: wrong request

The output of wrong Request:



figuer 80: wrong request result

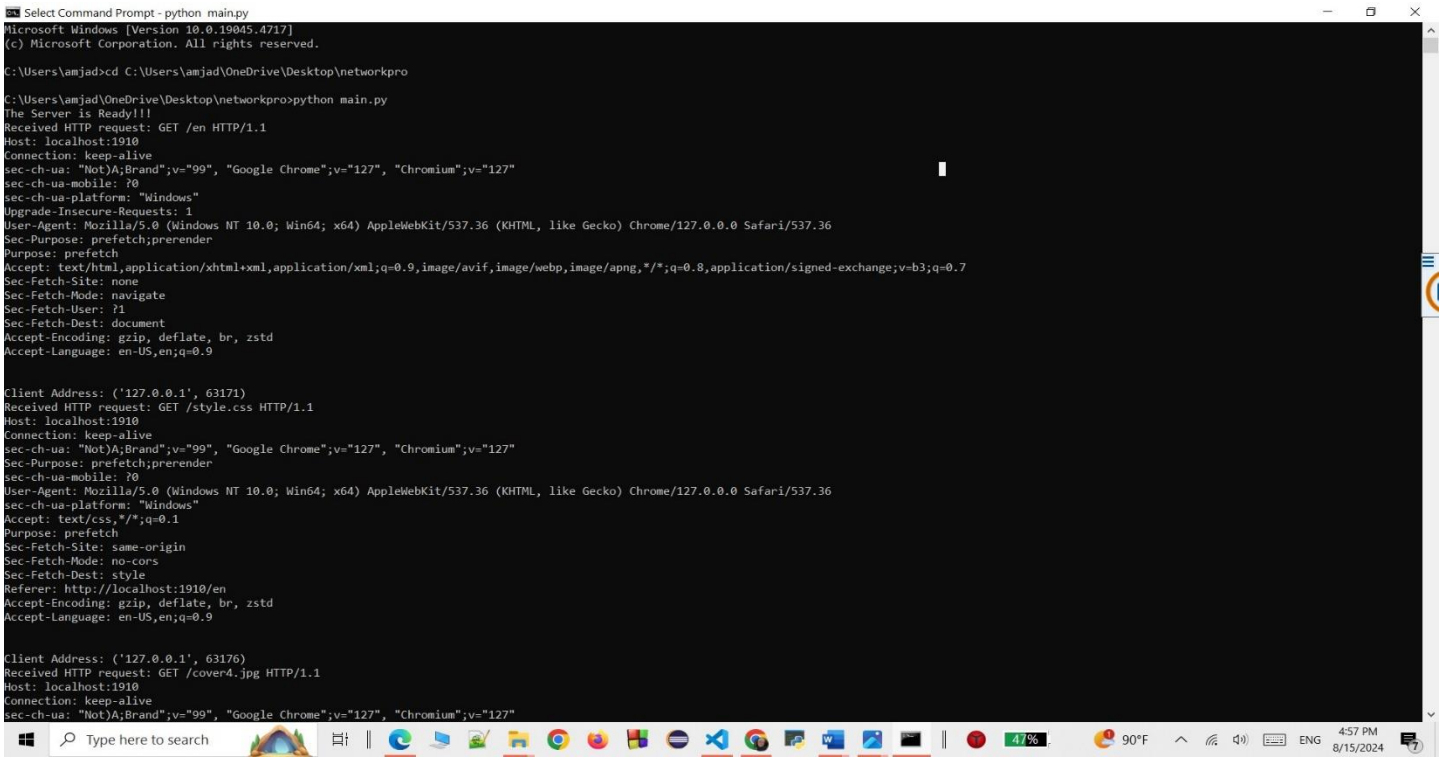
The Response of Wrong Request:



figuer 81: wrong response

3.11 print the HTTP requests on the terminal window:

The program logs incoming HTTP requests by printing the request details (e.g., URL, headers) to the terminal. This allows you to monitor all received requests directly in the command line window.



figuer 82 : http respone on terminal

The code of main_en:

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>ENCS3320-My First Webserver</title>
    <link rel="stylesheet" href="style.css">
    <link href="https://fonts.googleapis.com/icon?family=Material+Icons"
rel="stylesheet">
    <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">
</head>

<body>
    <div class="design">
        <h1>ENCS3320-My First Webserver</h1>
        <p>Welcome to <span class="red-text">Computer Networks, ENCS3320-
Webserver</span></p>
    </div>

    <div class="box">
        <h2>Group Members</h2>
        <div class="card">
            <div class="card-image">
                <img src="cover4.jpg" alt="Cover Image">
            </div>
            <div class="profile-image">
                <img src="alaa.jpg" alt="alaa Profile Image">
            </div>
            <div class="card-content">
                <h2>alaa adnan moqade 1211910</h2>
                <p> hi,I'm A computer science student at Birzeit University,I
have a passion for learning everything that's new in the world of programming,and
I'm trying to develop my software skills.</p>
                <p><strong>Projects:</strong>Web development using HTML and
CSS,object orinted projects in Java and javafx, Network simulations in
Python.</p>
```

```
                <p><strong>Hobbies:</strong>Coding, Cooking, hiking.</p>
            </div>
            <div class="icons">
                <a href="https://www.facebook.com/profile.php?id=100012380652183"
class="fab fa-facebook"></a>
                <a href="https://www.linkedin.com/in/alaa-moqade-550149280/"
class="fab fa-linkedin"></a>
                <a href=" https://github.com/AlaaMoqade" class="fab fa-
github"></a>
            </div>
        </div>
        <div class="card">
            <div class="card-image">
                <img src="cover4.jpg" alt="Cover Image">
            </div>
            <div class="profile-image">
                <img src="leyan.png" alt="leyan Profile Image">
            </div>
            <div class="card-content">
                <h2>layan Bairat 1211439 </h2>

                <p>I hope you are well, My name is Layan Bairat from Ramallah
Student at Birzeit University
                 I specialized in computer engineering and digital systems.Now
finished third year
                I participated in this course because I would like to learn about
its topic</p>
                <p><strong>Projects:</strong> Database management projects.</p>
                <p><strong>Hobbies:</strong> Painting,traveling,and coding.</p>
            </div>
            <div class="icons">
                <a href="https://www.facebook.com/leyan.buirat" class="fab fa-
facebook"></a>
            </div>
        </div>

        <div class="card">
            <div class="card-image">
                <img src="cover4.jpg" alt="Cover Image">
            </div>
            <div class="profile-image">
                <img src="lina.jpg" alt="lina Profile Image">
            </div>
            <div class="card-content">
                <h2>lina Abufarha 121</h2>
```

```
                <p>I'm a Computer Science student at Birzeit University in my
third year.I have experience with various programming languages and am always
looking to learn new skills in programming.</p>
                <p><strong>Projects:</strong>Web scraping using Python</p>
                <p><strong>Hobbies:</strong>Coding, playing chess, and exploring
new technologies.</p>
            </div>
            <div class="icons">
                <a href="https://www.facebook.com/lina.abufarha.1" class="fab fa-
facebook"></a>
            </div>
        </div>
    </div>

    <div class="links">
        <h3><strong>Useful Links</strong></h3>

        <p><a href="mySiteSTDID.html" target="_blank">Open Local HTML
File</a></p>
        <p><a href="https://www.birzeit.edu/" target="_blank">Visit Birzeit
University</a></p>
    </div>
</body>

</html>
```

The code of main_ar

```
<!DOCTYPE html>
<html lang="ar" dir="rtl">

<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>ENCS3320-خادم الويب الصغير</title>
    <link rel="stylesheet" href="style.css">
    <link href="https://fonts.googleapis.com/icon?family=Material+Icons"
rel="stylesheet">
    <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">
</head>

<body>
    <div class="design">
        <h1>ENCS3320-خادم الويب الصغير</h1>
        <p>مرحبًا بك في <span class="red-text">شبكات الحاسوب, ENCS3320-خادم الويب</span></p>
    </div>

    <div class="box">
        <h2>أعضاء المجموعة</h2>
        <div class="card">
            <div class="card-image">
                <img src="cover4.jpg" alt="صورة الغلاف">
            </div>
            <div class="profile-image">
                <img src="alaa.jpg" alt="صورة ملف علاء">
            </div>
            <div class="card-content">
                <h2>ألاء عدنان موقدي 1211910</h2>
                <p> مرحبًا، أنا طالبة في علوم الحاسوب. لدي شغف لتعلم كل ما هو جديد في عالم البرمجة، وأحاول تطوير
مهاراتي في البرمجيات.</p>
                <p><strong>المشاريع:</strong> تطوير ويب باستخدام HTML و CSS ، مشاريع كائنية التوجه
باستخدام Java و JavaFX، محاكاة الشبكات باستخدام Python.</p>
                <p><strong>الهوايات:</strong> البرمجة، الطبخ، المشي في الطبيعة.</p>
            </div>
            <div class="icons">
                <a href="https://www.facebook.com/profile.php?id=100012380652183"
class="fab fa-facebook"></a>
                <a href="https://www.linkedin.com/in/alaa-moqade-550149280/"
class="fab fa-linkedin"></a>
                <a href="https://github.com/AlaaMoqade" class="fab fa-
github"></a>
            </div>
        </div>

        <div class="card">
            <div class="card-image">
                <img src="cover4.jpg" alt="صورة الغلاف">
            </div>
            <div class="profile-image">
                <img src="leyan.png" alt="صورة ملف ليان">
```

```
            </div>
            <div class="card-content">
                <h2>1211439 ليان بيرات</h2>
                <p>أتمنى أن تكون بخير، اسمي ليان بيرات من رام الله، طالبة في جامعة بيرزيت. تخصصت في هندسة الحاسوب والنظم الرقمية. أنهيت سنتي الثالثة وأشارك في هذه الدورة لأنني أرغب في تعلم المزيد عن موضوعها.</p>
                <p><strong>المشاريع:</strong> مشاريع إدارة قواعد البيانات.</p>
                <p><strong>الهوايات:</strong> الرسم، السفر، والبرمجة.</p>
            </div>
            <div class="icons">
                <a href="https://www.facebook.com/leyan.buirat" class="fab fa-facebook"></a>
            </div>
        </div>

        <div class="card">
            <div class="card-image">
                <img src="cover4.jpg" alt="صورة الغلاف">
            </div>
            <div class="profile-image">
                <img src="lina.jpg" alt="صورة ملف لينا">
            </div>
            <div class="card-content">
                <h2>121 لينا أبو فرحة</h2>
                <p>أنا طالبة في علوم الحاسوب في جامعة بيرزيت في سنتي الثالثة. لدي خبرة في العديد من لغات البرمجة وأسعى دائمًا لتعلم مهارات جديدة في البرمجة.</p>
                <p><strong>المشاريع:</strong> سحب البيانات من الويب باستخدام Python.</p>
                <p><strong>الهوايات:</strong> البرمجة، لعب الشطرنج، واستكشاف التقنيات الجديدة.</p>
            </div>
            <div class="icons">
                <a href="https://www.facebook.com/lina.abufarha.1" class="fab fa-facebook"></a>
            </div>
        </div>
    </div>

    <div class="links">
        <h3><strong>روابط مفيدة</strong></h3>

        <p><a href="mySiteSTDID.html" target="_blank">فتح ملف HTML محلي</a></p>
        <p><a href="https://www.birzeit.edu/" target="_blank">زيارة موقع جامعة بيرزيت</a></p>
    </div>
</body>

</html>
```

The code of CSS Style

```css
/* General container for design */
.design {
    display: flex;
    flex-direction: column;
    align-items: center;
    padding: 20px 0;
}

/* Button container */
.bts {
    text-align: center;
    margin-bottom: 10px;
}

/* Button style */
.btn {
    background-color: rgb(129, 229, 236);
    border: none;
    color: rgb(0, 0, 0);
    padding: 12px 30px;
    font-size: 16px;
    cursor: pointer;
    margin: 0 5px;
}

/* Button hover effect */
.btn:hover {
    background-color: #0070C0;
}

/* Box container with larger size */
.box {
    text-align: justify;
    display: flex;
    flex-wrap: wrap;
    padding: 20px; /* Increased padding */
    margin: 20px auto; /* Increased margin */
```

```
    background-color: #0070C0; /* Blue background */
    border: 1px solid #cccccccc2;
    width: 500px; /* Larger width */
    border-radius: 20px;
}

/* General body style */
body {
    margin: 0;
    padding: 0;
    min-height: 100vh;
    display: flex;
    flex-direction: column;
    align-items: center;
    justify-content: center;
    background: rgb(182, 238, 245);
}

/* Card container with larger size */
.card {
    font-family: "Candara", sans-serif;
    width: 480px; /* Larger width */
    overflow: hidden;
    background: #fff;
    border-radius: 15px;
    box-shadow: 0 0 40px rgba(0, 0, 0, 0.5);
    display: flex;
    flex-direction: column;
    margin: 0 auto;
}

/* Card image styling with larger size */
.card-image img {
    width: 100%;
    height: 200px; /* Larger height */
    border-top-left-radius: 10px;
    border-top-right-radius: 10px;
    object-fit: cover;
}

/* Profile image styling with larger size */
.profile-image img {
    z-index: 1;
    height: 150px; /* Larger size */
    width: 150px;
```

```
    position: relative;
    margin-top: -75px; /* Adjusted for larger size */
    display: block;
    margin-left: auto;
    margin-right: auto;
    border-radius: 50%;
    border: 10px solid rgb(243, 157, 157);
    transition-duration: 0.4s;
    transition-property: transform;
}

/* Profile image hover effect */
.profile-image img:hover {
    transform: scale(1.1);
}

/* Card content styling with larger text */
.card-content h2 {
    font-size: 24px; /* Larger font size */
    text-align: center;
    margin: 10px 0; /* Added margin */
}

.card-content p {
    font-size: 18px; /* Larger font size */
    text-align: justify;
    padding: 0 20px 10px 20px; /* Increased padding */
}

/* Icons container */
.icons {
    text-align: center;
    padding-top: 10px; /* Increased padding */
    padding-bottom: 30px; /* Increased padding */
}

/* Icons styling */
.icons a {
    text-decoration: none;
    font-size: 24px; /* Larger font size */
    color: rgb(247, 19, 49);
    padding: 0 18px;
    transition-duration: 0.4s;
    transition-property: transform;
}
```

```css
/* Icons hover effect */
.icons a:hover {
    color: black;
    transform: scale(1.5);
}

/* Blue text styling */
.blue-text {
    color: blue;
    font-weight: bold; /* Optional: Make the text bold */
}

/* Red text styling */
.red-text {
    color: red;
}
```

the Main Code:

```python
from socket import *

# Reserve port 1910 on the computer
serverPort = 1910

# 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))
serverSocket.listen(1)  # Corrected this line
image_extensions = ['.jpg', '.png']

print("The Server is Ready!!!")

# Infinite loop until 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 client and decode it to obtain the string.
    sentence = connection_Socket.recv(1024).decode()
    # Print the received HTTP request
    print("Received HTTP request:", sentence)
    print("Client Address:", address)

    ip = address[0]
    port = address[1]
    words = sentence.split()

    if len(words) >= 2:
        request = words[1].lower().lstrip('/')

        # Handling HTML files
        if request == '' or request == 'index.html' or request == 'main_en.html'
or request == 'en':
            requestedFile = open("main_en.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())

        elif request == 'main_ar.html' or request == 'ar':
            requestedFile = open("main_ar.html", encoding="utf-8")
            webPage = requestedFile.read()
            requestedFile.close()
            connection_Socket.send("HTTP/1.1 200 OK\r\n".encode())
            connection_Socket.send("Content-Type: text/html; charset=utf-
8\r\n".encode())
            connection_Socket.send("\r\n".encode())
            connection_Socket.send(webPage.encode('utf-8'))

        elif request.endswith(".html"):
            try:
                requestedFile = open(request, 'r')
                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())
            except FileNotFoundError:
                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("<html><head><title>Error
404</title></head><body><h1>HTTP/1.1 404 Not Found</h1><br><p
style='color:blue'>The file is not
found</p><br><b>alaa</b><br><b>1211910</b><br><p>Client IP: {}</p><p>Client Port:
{}</p></body></html>".format(ip, port).encode())
        # Handling CSS files
        elif request.endswith(".css"):
            try:
                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())
            except FileNotFoundError:
                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("<html><head><title>Error
404</title></head><body><h1>HTTP/1.1 404 Not Found</h1><br><p
style='color:blue'>The file is not
found</p><br><b>alaa</b><br><b>1211910</b><br><p>Client IP: {}</p><p>Client Port:
{}</p></body></html>".format(ip, port).encode())

        # Handling image files
        elif request.endswith(".png"):
            try:
                requestedFile = open(request, "rb")
                imageData = requestedFile.read()
                requestedFile.close()
                connection_Socket.send("HTTP/1.1 200 OK\r\n".encode())
                connection_Socket.send("Content-Type: image/png \r\n".encode())
                connection_Socket.send("\r\n".encode())
                connection_Socket.send(imageData)
            except FileNotFoundError:
                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("<html><head><title>Error
404</title></head><body><h1>HTTP/1.1 404 Not Found</h1><br><p
style='color:blue'>The file is not
```

```
found</p><br><b>alaa</b><br><b>1211910</b><br><p>Client IP: {}</p><p>Client Port:
{}</p></body></html>".format(ip, port).encode())



        elif request.endswith(".jpg") or request.endswith(".jpeg"):
            try:
                requestedFile = open(request, "rb")
                imageData = 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(imageData)
            except FileNotFoundError:
                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("<html><head><title>Error
404</title></head><body><h1>HTTP/1.1 404 Not Found</h1><br><p
style='color:blue'>The file is not
found</p><br><b>alaa</b><br><b>1211910</b><br><p>Client IP: {}</p><p>Client Port:
{}</p></body></html>".format(ip, port).encode())

        elif request == 'mySiteSTDID.html' :
                requestedFile = open("mySiteSTDID.html", encoding="utf-8")
                webPage = requestedFile.read()
                requestedFile.close()
                connection_Socket.send("HTTP/1.1 200 OK\r\n".encode())
                connection_Socket.send("Content-Type: text/html; charset=utf-
8\r\n".encode())
                connection_Socket.send("\r\n".encode())
                connection_Socket.send(webPage.encode('utf-8'))



        # Redirects
        elif request == "so":
            connection_Socket.send("HTTP/1.1 307 Temporary Redirect
\r\n".encode())
            connection_Socket.send("Location: https://www.stackoverflow.com
\r\n".encode())
            connection_Socket.send("\r\n".encode())

        elif request == "itc":
```

```
            connection_Socket.send("HTTP/1.1 307 Temporary Redirect
\r\n".encode())
            connection_Socket.send("Location: https://itc.birzeit.edu
\r\n".encode())
            connection_Socket.send("\r\n".encode())

        # If the file is not found
        else:
            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("<html><head><title>Error
404</title></head><body><h1>"
                                   "HTTP/1.1 404 Not Found</h1><br><p
style='color:blue'>"
                                   "The file is not
found</p><br><p>alaa</p><br><p>1211910</p><br>"
                                   "<p>Client IP: {}</p><p>Client Port:
{}</p></body></html>".format(ip, port).encode())

        # Close the connection
        connection_Socket.close()
    else:
        connection_Socket.close()
```

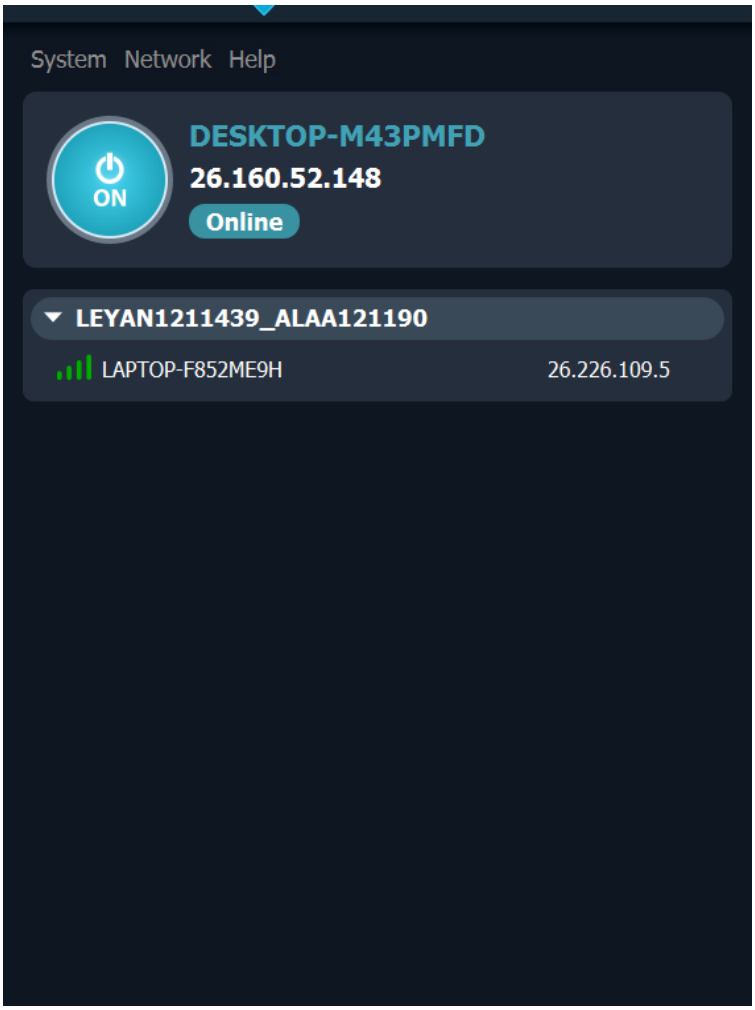
The code of server:

```
from socket import *


def Is_valid_number(number):
    return number == "1211910" or number == "1211439"


serverSocket = socket(AF_INET, SOCK_STREAM)
serverPort = 1910
serverSocket.bind(('', serverPort))
serverSocket.listen(5)

print("The server is listening on port " + str(serverPort))
```

```
while True:
    newSocket, clientAddress = serverSocket.accept()
    studentId = newSocket.recv(1024).decode()

    if Is_valid_number(studentId):
        print("The OS will lock the screen after 10 seconds" + " Server Side ")
        message = "The server will lock the screen after 10 seconds"
        newSocket.send(message.encode())

        import time
        time.sleep(10)  # Wait 10 seconds

        import ctypes

        ctypes.windll.user32.LockWorkStation()

        newSocket.close()
    else:
        print("The number is not valid")

        newSocket.close()
```

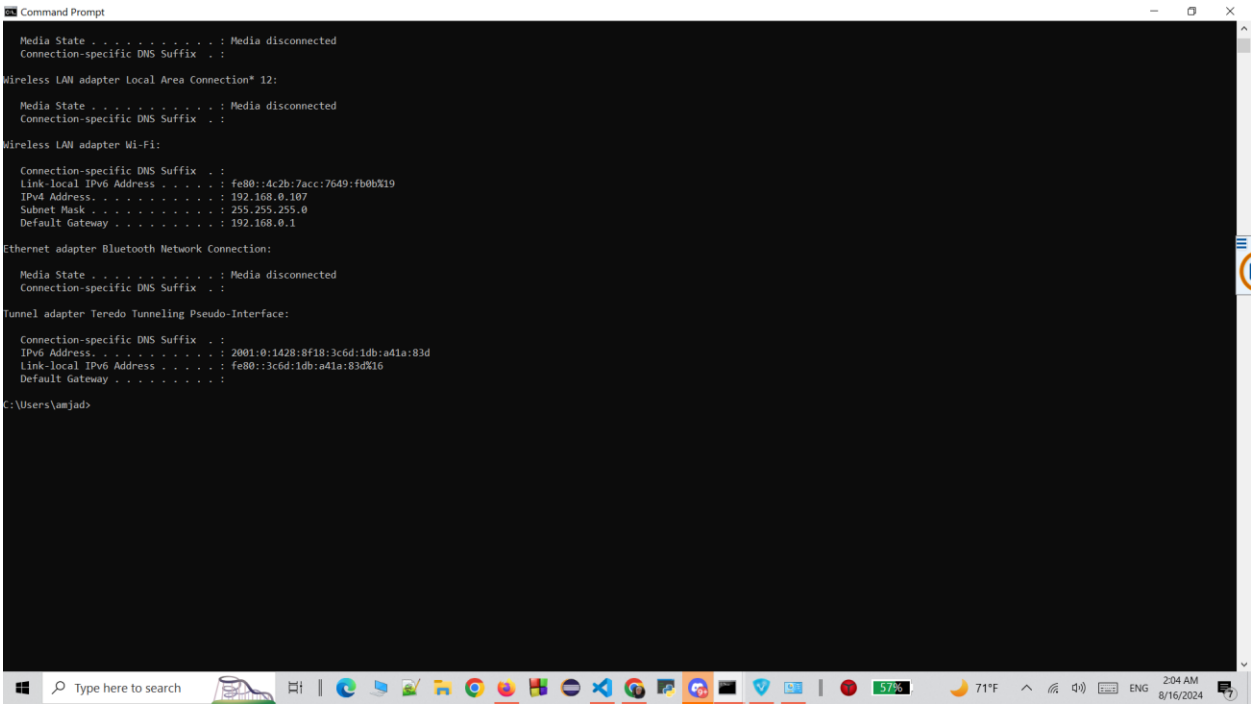
Test part 3 from a many browser:

1-Frome different computer : by work as a same network

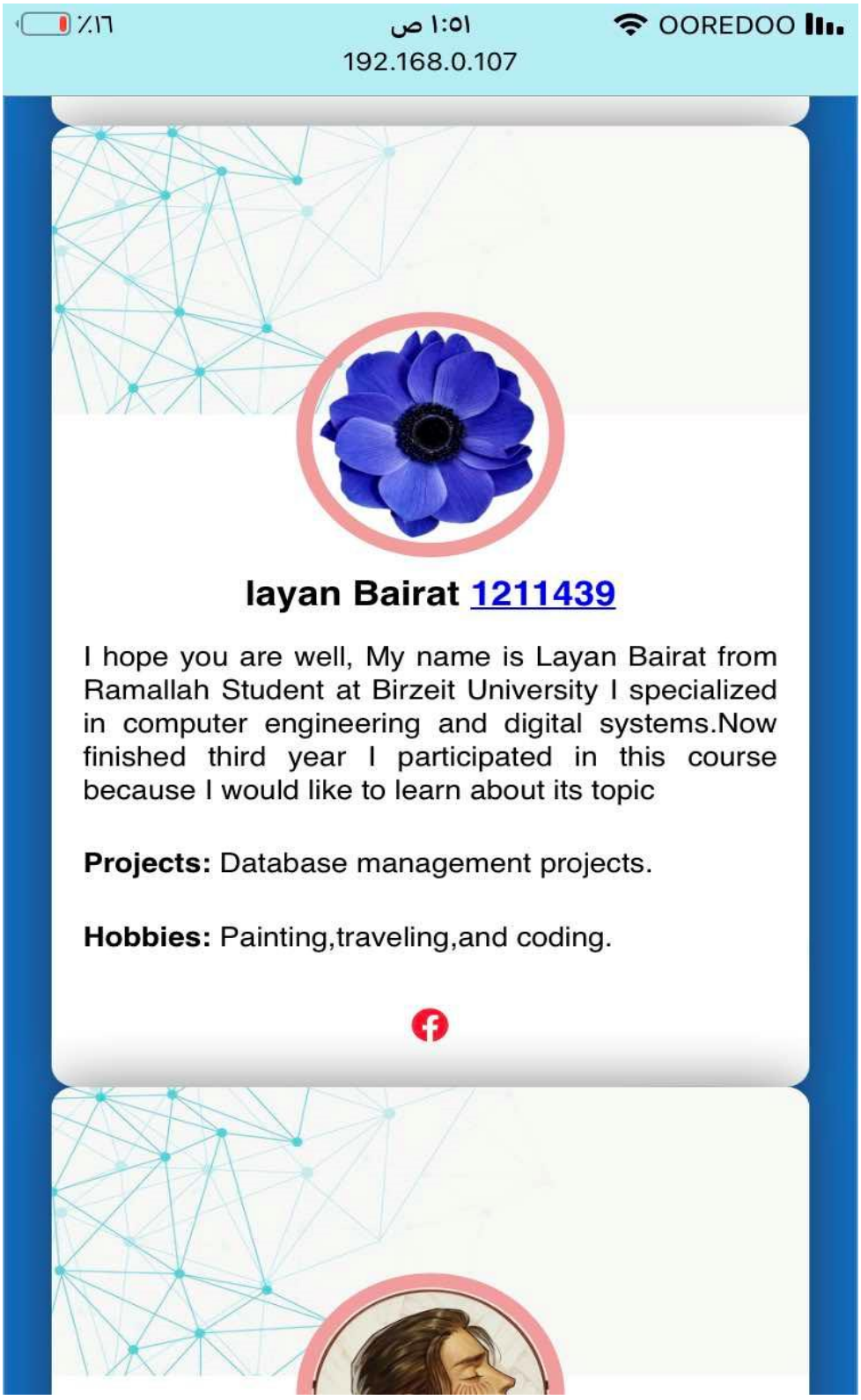


figuer 83: connecting network

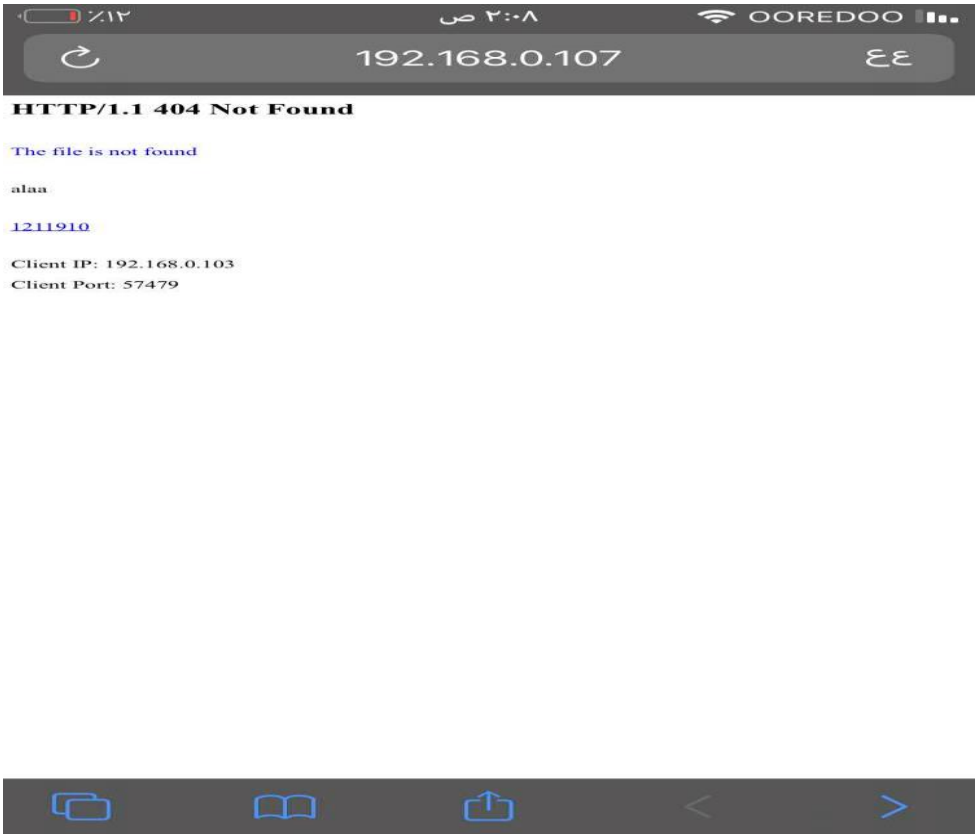
2- different phone : by use the IP address 192.168.0.107, which is your computer's local network IP. Ensure your computer and phone are connected to the same Wi-Fi network. Then, access the server from your phone's browser by typing http://192.168.0.107:1910, where 1910 is the port number.



figuer 84: get ip address



figuer 85: main_en request in phone



figuer 86: wrong request in phone

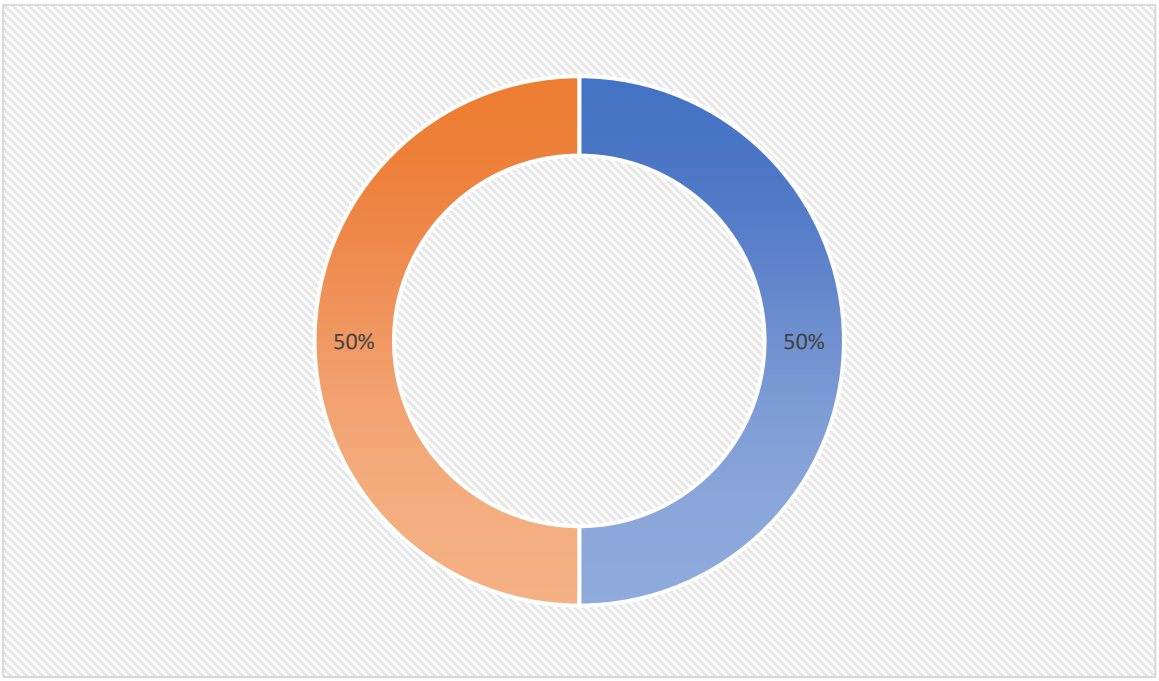
figuer 87: mySitSTDID.html request in phone

figuer 88: style.css request in phone

3.Alternative solutions:

issues and limitations We did not face many problems, but the biggest problem we faced was when we wanted an extension for a file to use it with the code to display the results on the cmd screen, it would show an error because the file extension was wrong. Therefore, the solution to this problem was as follows: We go to the file itself and click on Open the location path, then copy the path and finally return to use it in the code cmd.

4.The work done by each member:



5.Referance :

1_ CMD Ping Command | Test Network Connectivity and Latency (configserverfirewall.com)

At 10:01pm 8\15\2024

2_ How to use the command 'tracert' (with examples) (commandmasters.com)

At 10:00am 8\10\2024

3_ How to Use the nslookup Command {10 Examples} (phoenixnap.com) at 12:50 am 8\10\2024

4_ How to use the command 'telnet' (with examples) (commandmasters.com) at 1:00 am 8\11\2024

5_ What is an autonomous system? | What are ASNs? | Cloudflare at 1:16 am 8\10\2024

6_ DNS in Wireshark - GeeksforGeeks at 4:04 am 8\10\2024

7_https://www.techtarget.com/searchnetworking/definition/ping?fbclid=IwAR2VOdJFaAAQd0DkEahLL8RNPWmvRc8pUh8fCNIz5E3RbflDjmDONQ86deU

8-https://www.lifewire.com/tracert-command-2618101https://www.w#schools.com/python/python_intro.asp