Faculty of Engineering And Technology Electrical And Computer Engineering Department

Computer Networks
ENCS3320

Project No.1_
Socket Programming

Group number 14

Prepared by :
Lana Zaben 1221321
Wala'a Jaradat 1220198
Sadeel Jabareen 1220465

Instructor :
Dr . Ibraheem nemer

Section : 1
Date : 31 - 7 – 2024

Table of Contents :

Table of figures:

Theory :

This project focuses on key aspects of computer networks, including diagnostic commands, socket programming, and building a basic web server.

Ping is used to test the reachability of a device on a network by sending simple echo requests. **Tracert** helps trace the path data takes across a network, showing each hop along the way. **Nslookup** translates domain names into IP addresses, which is essential for resolving DNS issues. **Telnet** allows for remote management of devices through a command-line interface over TCP.

Socket programming plays a central role in network communication. **TCP** ensures reliable, orderly communication between a client and server, with data arriving as expected. **UDP** prioritizes speed, sending data without guaranteeing delivery, making it suitable for applications where speed is more critical than accuracy.

The project also involves building a simple web server using **HTTP**. This server listens on a designated port, serving different types of files like HTML, CSS, and images. It processes client requests, delivering the appropriate response—whether it's a webpage, a redirect, or an error message. This part of the project demonstrates how web content is served and provides practical experience in handling HTTP requests.

Overall, the project offers hands-on experience with fundamental networking concepts, providing a solid understanding of how to work with and manage networked systems.

Procedure:

The project involves several key tasks, each requiring specific components and ideas to ensure proper execution. For the **Commands & Wireshark** task, tools like **Ping**, **Traceroute**, **Nslookup**, **Telnet**, and **Wireshark** are used to test network connectivity, diagnose routing issues, verify DNS information, test remote server connections, and analyze network traffic, respectively. In the **Socket Programming** task, **TCP** and **UDP** client-server applications are implemented to handle reliable data transfer and low-latency communication, with port numbers derived from a student ID to ensure uniqueness. The **Web Server** task involves using socket programming to handle HTTP requests, serving HTML content based on specific URL requests, and selecting a unique port number to avoid conflicts with other services. These components are chosen to address the specific needs of the tasks, ensuring that the network operates effectively and is easy to debug and verify.

Result and disscussion:

Task 1 : Commands & Wireshark

1.In your words, what are ping, tracert, nslookup, and telnet?

ping, tracert, nslookup, and telnet.

Ping: Checks the reachability of a device on a network by sending packets and measuring the response time[1].
Tracert (Traceroute): Maps the path data takes to travel from your computer to a target location, showing each step along the way[1].
Nslookup: Looks up and provides details about domain names and their associated IP addresses from the DNS[1].
Telnet: Allows you to connect and interact with remote devices or servers over a network using the TCP/IP protocol[1].

---

2. Make sure that your computer is connected to the internet and then run the following
commands:
a. ping a device in the same network, e.g. from a laptop to a smartphone.

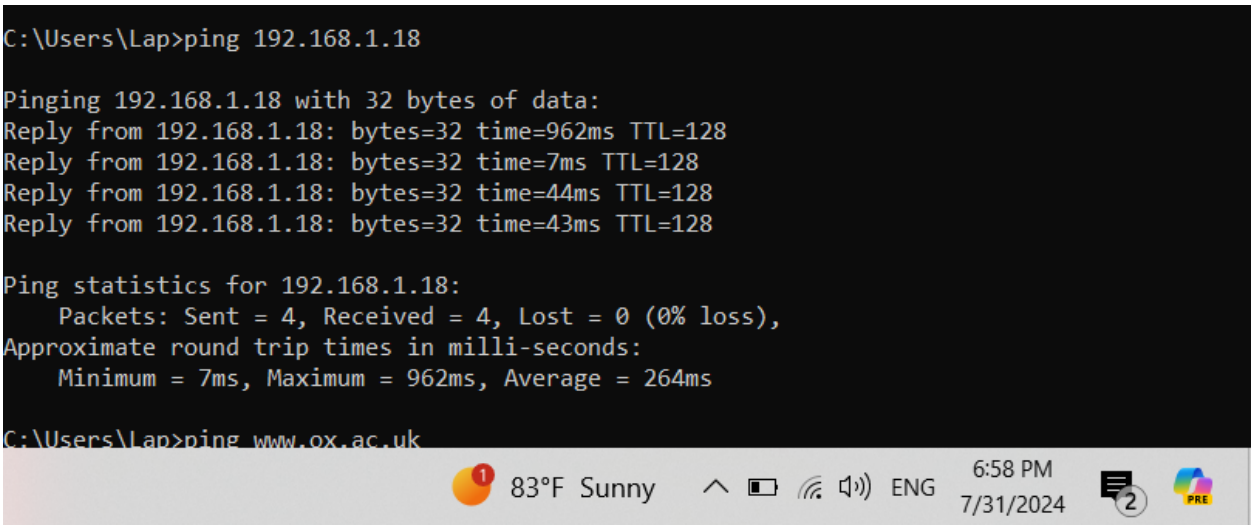


Figure 1:Ping test between two laptops on the same network

The screen above shows the result of a successful ping command executed in the Command Prompt to the IP address 192.168.1.18.

Reply from 192.168.1.18: Confirms the target device responded.
TTL=128: Time To Live value, showing the remaining lifespan of the packet before being discarded.

Packets: Sent = 4: Four packets were sent.
Received = 4: All four packets were received back.
Lost = (0% loss): No packets were lost, indicating a reliable connection.

Device Reachability: The device with IP 192.168.1.18 is reachable, as all packets were received back without loss.
Variable Latency: The round-trip times varied significantly (7ms to 962ms), indicating fluctuations in network performance.
Overall Performance: Despite the high maximum latency, the average round-trip time (264ms) suggests a generally acceptable network performance with some instances of delay.

---

b. ping www.ox.ac.uk.

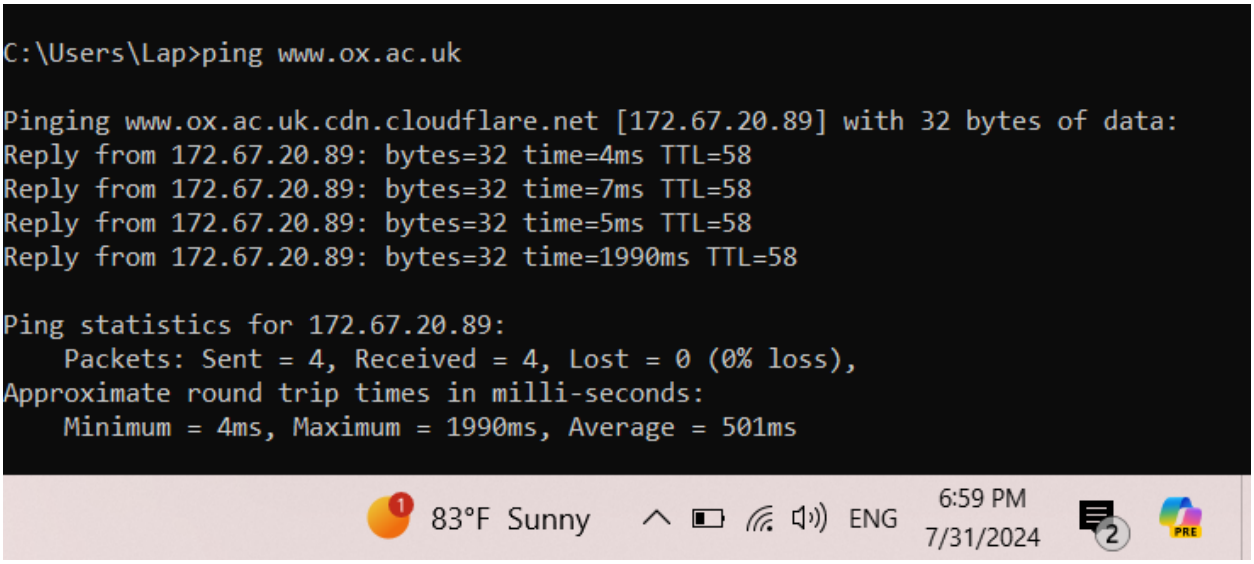


Figure 2:Ping result for www.ox.ac.uk to check internet connectivity.

C:\Users\Lapg>ping www.ox.ac.uk :This command is used to send packets to the domain www.ox.ac.uk.

The domain **www.ox.ac.uk** is resolved to an IP address **172.67.20.89**, indicating that the site is using Cloudflare's CDN services.

bytes=32: Each reply is 32 bytes in size.
time=xms: The time it took for each packet to make a round trip, where x is the time in milliseconds.
TTL=58: Time-to-Live value, which indicates the remaining life of the packet in the network.

Packets: Sent = 4, Received = 4, Lost = (0% loss): All four packets sent were successfully received, indicating no packet loss.

Approximate round trip times in milli-seconds: The time statistics for the round trips:

Minimum time: 4 ms

Maximum time: 1990 ms

Average time: 501 ms

The result indicates that the connection to the server is mostly fast, with one outlier taking significantly longer (1990 ms). This could suggest intermittent network congestion or issues. The overall connectivity is stable as there is no packet loss**.**

---

c. From the ping results, specify the location of the server from where you got the response? Explain your answer briefly.

Ping to 192.168.1.18

- Server Address: 192.168.1.18 (Private IP)
- Location: This server is local, within the same network as the client.
- Explanation: The relatively consistent response times and the private IP address indicate that the server is part of the local network (e.g.home network).

Ping to www.ox.ac.uk

- Server Address: 172.67.20.89
- Location: This server is remote, likely far from the client's location or experiencing network delays.
- Explanation: The wide range of response times (from 4ms to 1990ms) suggests that the server is geographically distant or facing intermittent network issues.

D. tracert www.ox.ac.uk

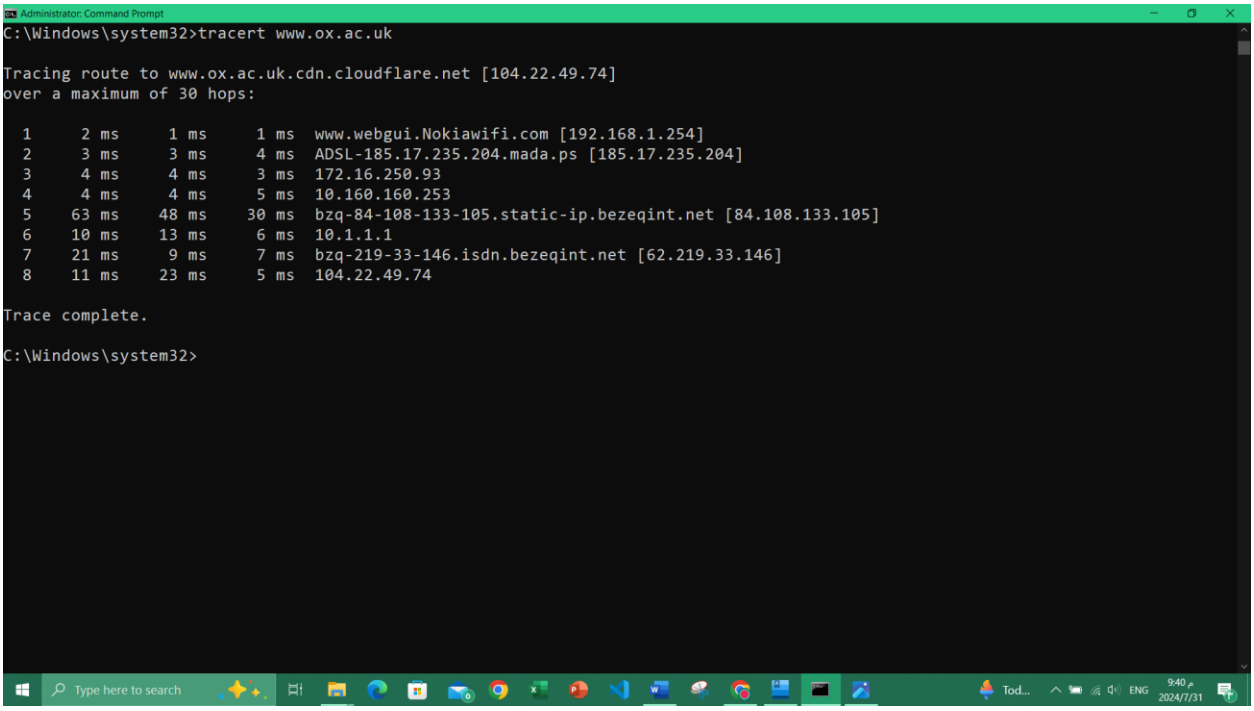


Figure 3 result of command tracert www.ox.ac.uk

The IP address [104.22.49.74] points to a Cloudflare data center, where the server that is answering your request is situated. With a massive global server network, Cloudflare is a content delivery network (CDN) and DDoS mitigation company. This indicates that the server you reached—rather than the one that originally hosted the www.ox.ac.uk website—is probably the nearest Cloudflare edge server to your location.

• Cloudflare's Function: The last hop, a Cloudflare IP, shows that www.ox.ac.uk traffic is being handled by Cloudflare. This indicates that one of Cloudflare's edge servers is serving the real website content.

• Efficiency and Security: By ensuring that the content is delivered swiftly and securely from a nearby location, Cloudflare lowers latency and speeds up load times.

Explanation of the Trace Route Result:

1. Hop 1: www.webgui.Nokiawifi.com [192.168.1.254]
    - This is your local network router, typically the first hop in any tracert command.
2. Hop 2: ADSL-185.17.235.204.mada.ps [185.17.235.204]
    - This is the first external hop, usually your Internet Service Provider's (ISP) gateway.
3. Hop 3: 172.16.250.93
4. Hop 4: 10.160.160.253
5. Hop 5: bzq-84-108-133-105.cablep.bezeqint.net [84.108.133.105]
    - This is a public IP associated with Bezeq International, indicating a transit point within the ISP's broader network.
6. Hop 6: 10.1.1.1
7. Hop 7: bzq-219-33-146.isdn.bezeqint.net [62.219.33.146]
    - Another transit point within the ISP, but with a public IP address.
8. Hop 8: 104.22.49.74
    - This is the IP address of the Cloudflare server that is providing the response. This hop shows that the Cloudflare server is acting as an intermediary, which is typical for websites using Cloudflare for content delivery and security services.

E. nslookup www.ox.ac.uk.

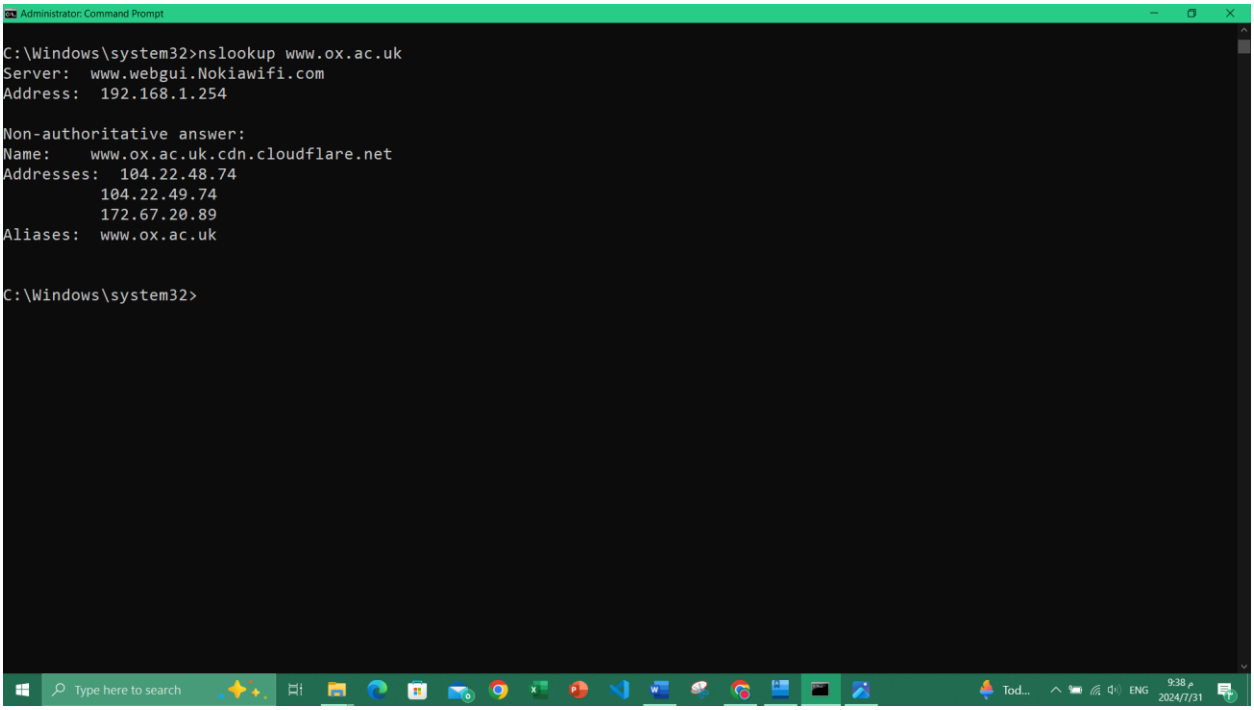


Figure 4 result of command nslookup www.ox.ac.uk

• www.webgui.Nokiawifi.com is the server.

• This shows that your local network router—a Nokia WiFi router—is the DNS server being utilized for the lookup.

192.168.1.254 is the address.

• This is the IP address of the local router in your network, which serves as the DNS server.

• Non-authoritative response: This indicates that the DNS response is not directly coming from the domain's authoritative DNS server, but rather from a cache or another non-primary source.

Name: www.cloudflare.net.cdn.ox.ac.uk

• This demonstrates that www.ox.ac.uk is a CNAME (alias) for www.ox.ac.uk.cdn.cloudflare.net, suggesting that Cloudflare is in charge of this domain's DNS management and potentially its content delivery.

• Addresses: 172.67.20.889; 104.22.48.74; 172.67.20.89

The IP addresses linked to www.ox.ac.uk.cdn.cloudflare.net are listed here. Several IP addresses are used by Cloudflare for load balancing and redundancy.

• Aliases: www.ox.ac.uk

•This confirms that www.ox.ac.uk is an alias for www.ox.ac.uk.cdn.cloudflare.net.

---

f. telnet www.ox.ac.uk

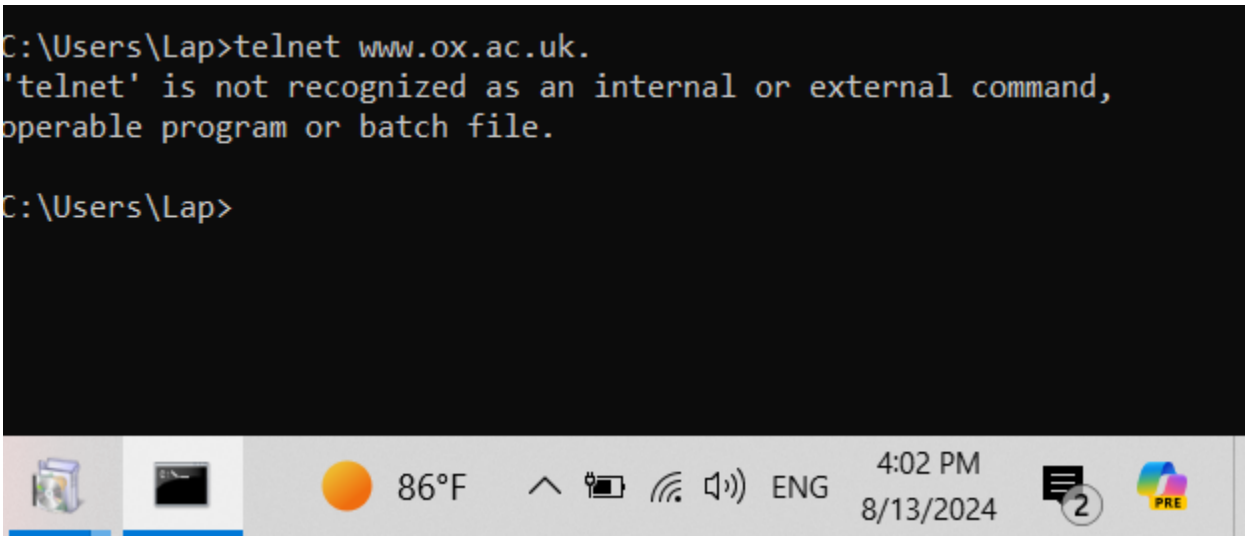


Figure 5:Result of telnet www.ox.ac.uk command before enable the telnet client

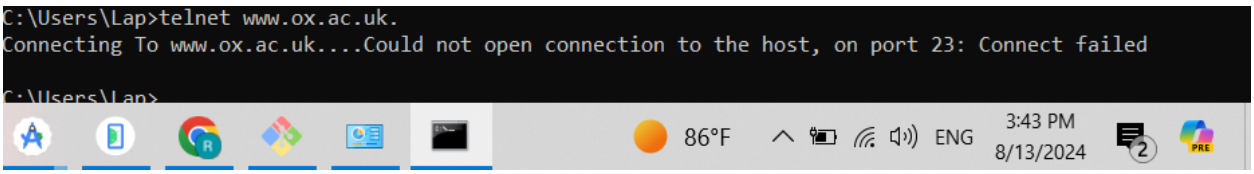


Figure 6:Result of telnet www.ox.ac.uk command before change the port

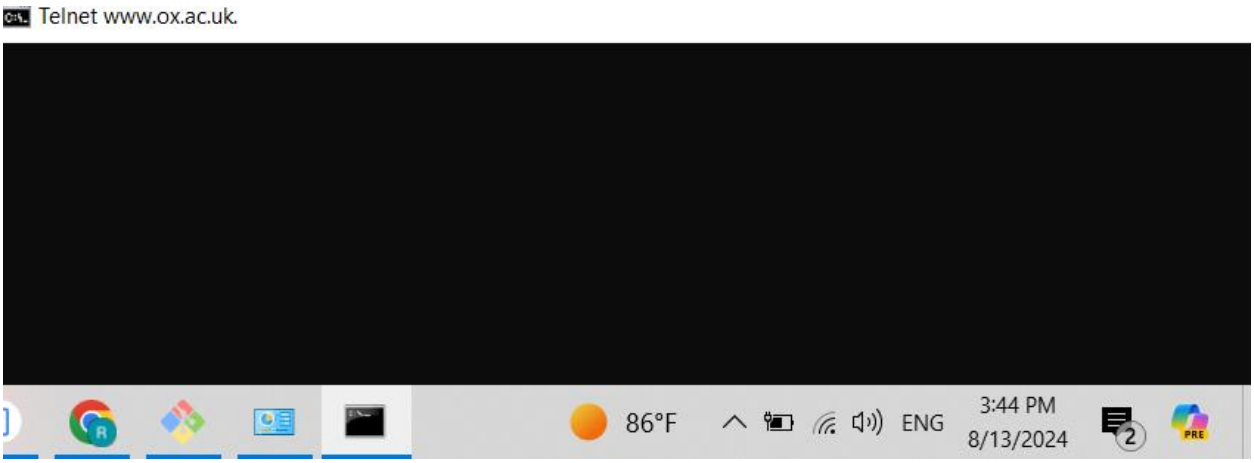


Figure 7:Result of telnet www.ox.ac.uk command after changing the port to 80

What the Command Does:

1. telnet: This command-line utility connects to distant PCs via the Telnet protocol over a network. It can be applied to verify that a remote server's port is reachable.

2. The website of the University of Oxford is located at www.ox.ac.uk.
You are attempting to get a Telnet connection to the domain when you type {telnet www.ox.ac.uk}. The normal Telnet port, 23, is the port that Telnet attempts to connect to by default.

When we write the telnet www.ox.ac.uk command for the first time it does not work ,then we enable the telnet client from the settings but it still does not work until we change the port from port 23 to port 80 .

---

3. Give some details about autonomous system (AS) number, number of IPs, prefixes, peers, name of Tier1-ISP of www.ox.ac.uk.

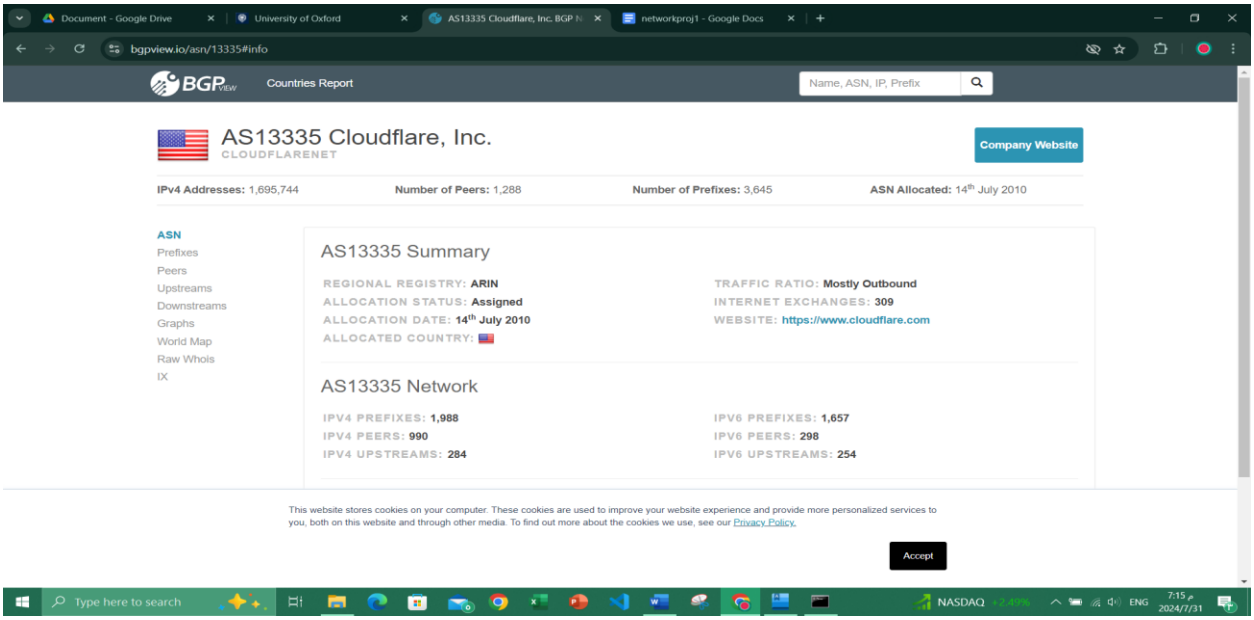


Figure 8 entire network figures for the oxford website on BGP

• AS Number: AS13335 is the University of Oxford's Autonomous System (AS) number. This AS number is owned by Cloudflare, which offers www.ox.ac.uk CDN and other internet services.

•Number of IPs: Because of Cloudflare's vast global network, AS13335, which is part of the company, manages a lot of IP addresses. Although a summary of the precise number of IP addresses is not easily accessible, Cloudflare maintains a large range of IPs in order to support its CDN services.

•Prefixes: Because Cloudflare's AS13335 caters to a worldwide audience, it has multiple IP prefixes. These prefixes, which offer a strong and dispersed network, include ranges like 104.22.49.0/24, 172.67.0.0/16, and others.

• Peers: To guarantee effective and dependable content delivery, Cloudflare peers with many networks throughout the world. Major global network operators and ISPs are among these peers.

• Tier 1 ISP: Cloudflare provides its services by peering with multiple Tier 1 ISPs; it is not a Tier 1 ISP in and of itself. Companies like AT&T, CenturyLink, and NTT Communications are a few well-known Tier 1 ISPs.

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

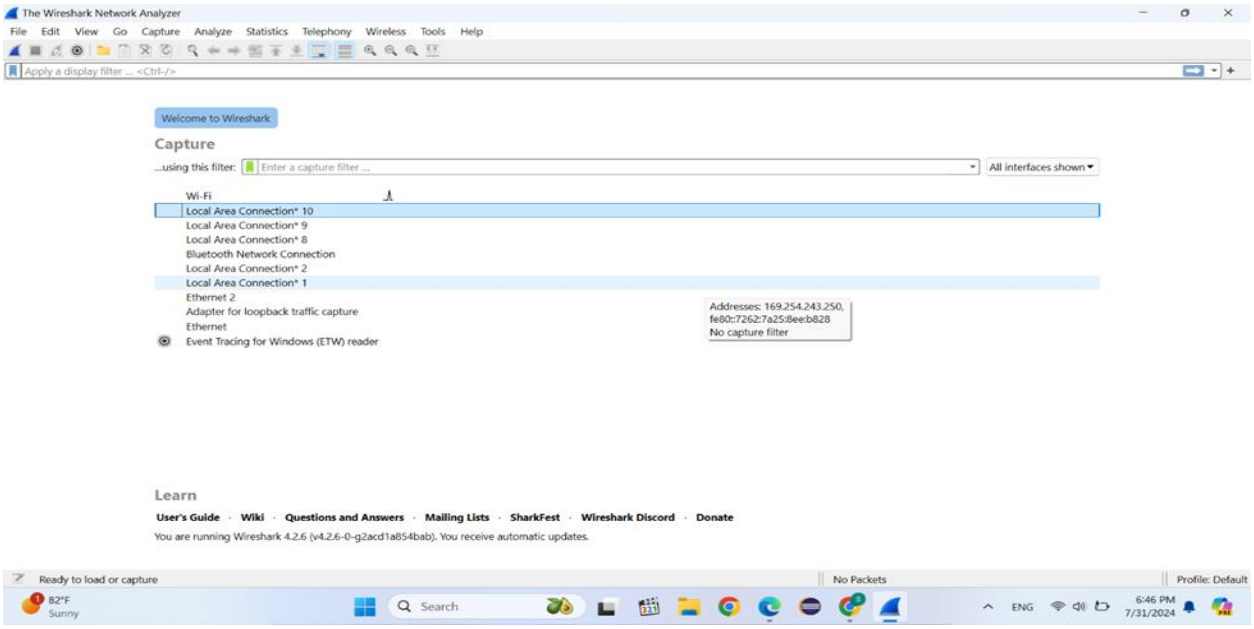


Figure 9 first interface in wireshark

We downloaded the wireshark application and the first time I opened it, this interface appeared

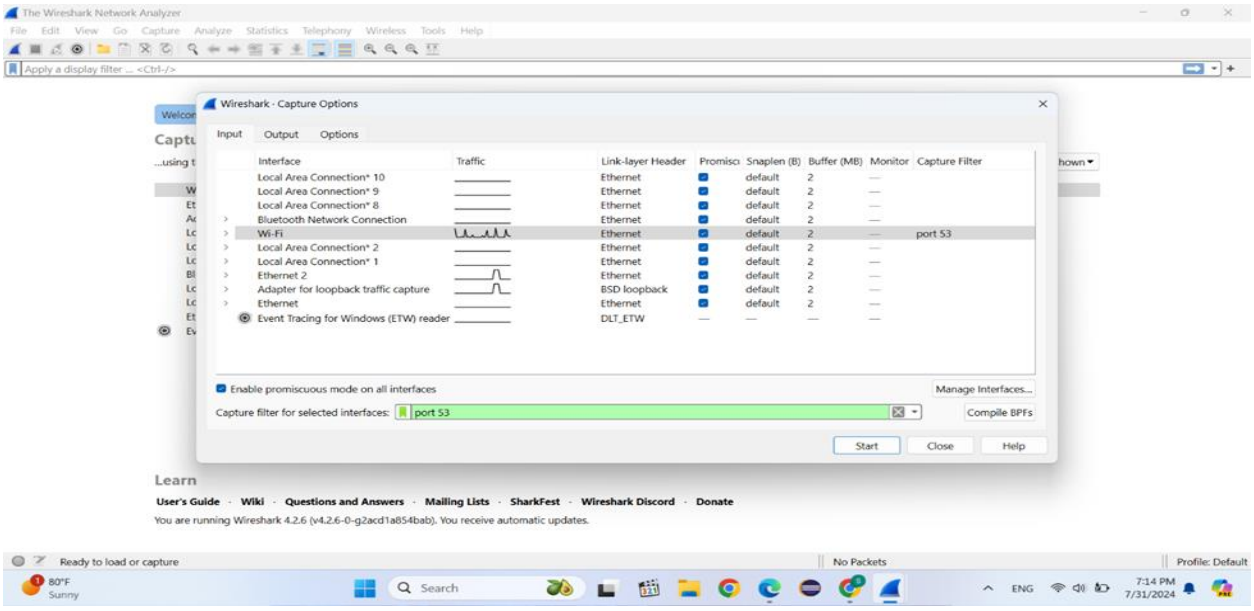


Figure 10 interface to choose port

Then we went to the bar at the top and chose capture, then we chose option and we specified wifi and port 53.

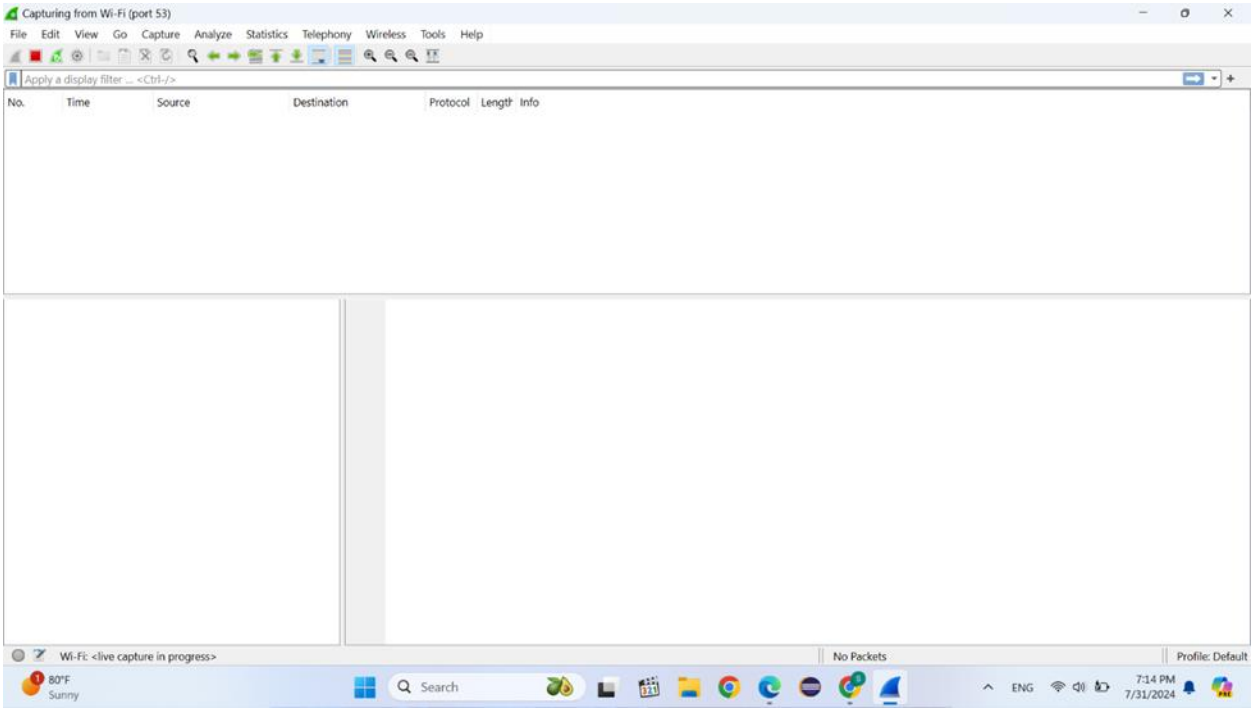


Figure 11 before show capture some DNS messages

After confirming what was required in the previous process, this interface appears.

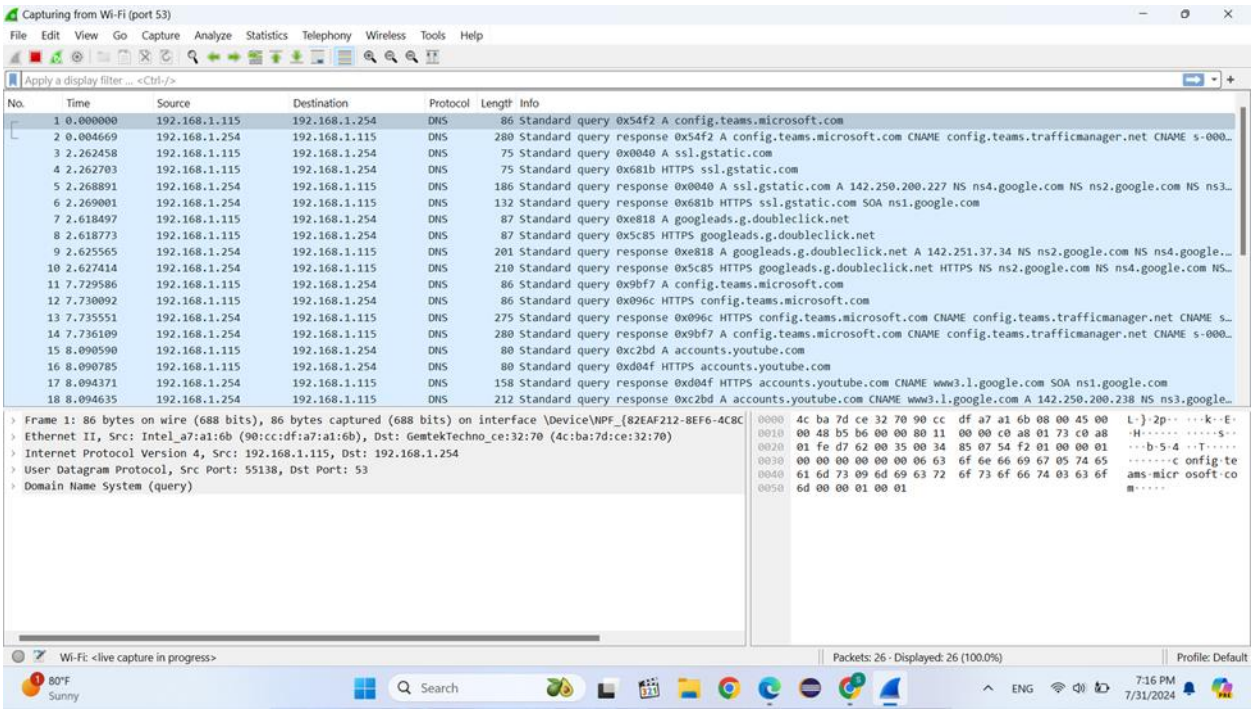


Figure 12 capture some DNS messages

This interface log is a packet capture of network traffic, specifically focusing on DNS (Domain Name System) queries. Let's break down the details:

Packet Capture Details:

- Interface: Captured from Wi-Fi (port 53)

- Protocol: DNS (Domain Name System)

- Total Packets Displayed: 26

*Frame 1:

Size: 86 bytes on the wire, 86 bytes captured

Interface: \Device\NPF_{82EAF212-8EF6-4C8C-8DAD-123456789}

Ethernet II:

- Source: Intel_a7:a1:6b (MAC address 90:cc:df:a7:a1:6b)

-Destination: GemtekTechno_ce:32:70 (MAC address 4c:ba:7d:ce:32:70)

IPv4:

-Source: 192.168.1.115

- Destination: 192.168.1.254

UDP:

- Source Port: 55138

- Destination Port: 53 (DNS)

DNS Queries:

- Standard query 0x54f2: A config.teams.microsoft.com

Subsequent Frames:

* Frame 2 to Frame 18: Similar DNS queries and responses as Frame 1

  - Protocols: DNS

  - Length: Varies (e.g., 86 bytes to 280 bytes)

  - Source: 192.168.1.115

  - Destination: 192.168.1.254

  - DNS Queries and Responses:

        config.teams.microsoft.com

        ssl.gstatic.com

        googleads.g.doubleclick.net

        accounts.youtube.com

Example Detailed Frame:

* Frame 8:

  Size: 210 bytes

  Source: 192.168.1.115

  Destination: 192.168.1.254

  DNS Query: Standard query response 0xe818 A googleads.g.doubleclick.net

Packet Capture Details:

- Captured by: Ethernet II protocol

- Addressing: Utilizes IPv4 addressing

- DNS Queries and Responses: Transmitted using the User Datagram Protocol (UDP) on port 53

This log essentially traces DNS communication, capturing queries for different types of records (e.g., A records for IPv4 addresses, HTTPS records) and their corresponding responses.

Task2: Socket Programming (TCP and UDP):

TCP client server python applications .

1. 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.
When using the same computer in TCP it will be easier to received all packet after send them because they are in same computer and same program.

A screenshot for the server

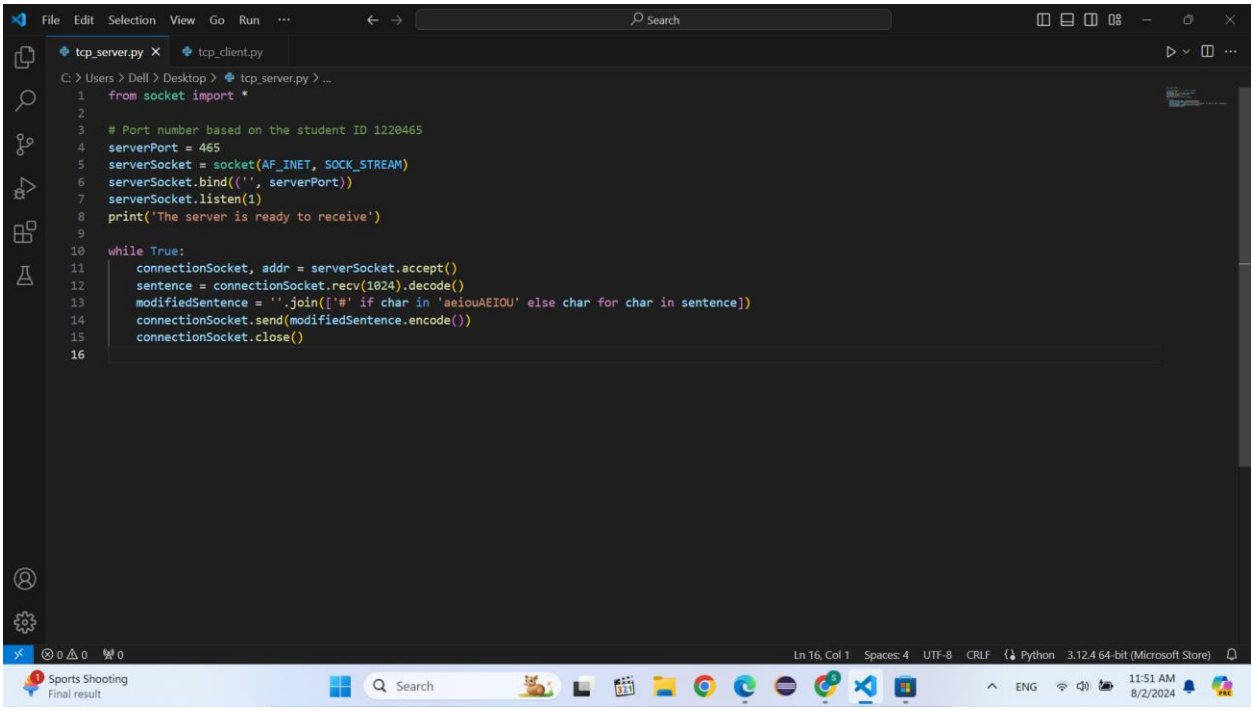


Figure 13 server code in TCP

1. Based on the student's ID, 1220465, the server is listening on port 465.
2. It receives a sentence from the client and accepts incoming connections.
3. The sentence has been changed by adding # in place of the vowels.
4. The customer receives the amended sentence back, and the communication is ended.

A screenshot for the client

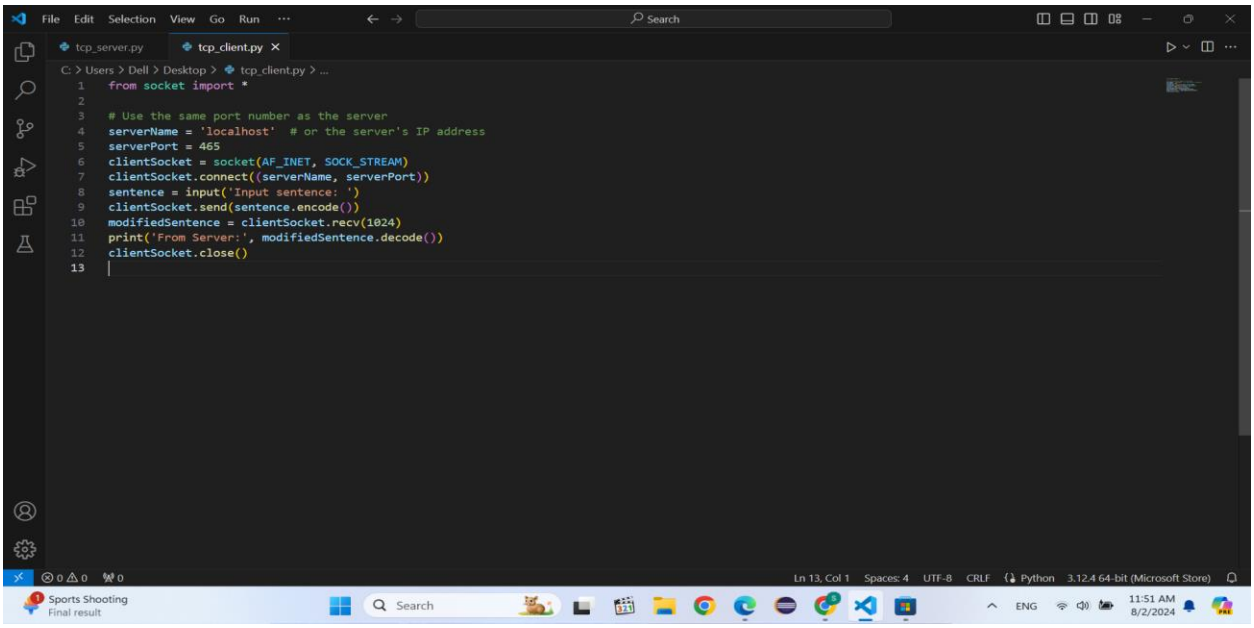


Figure 14 client code in TCP

1. The server and client establish a connection across port 465 and localhost.
2. The sentence entered by the user is transferred to the server.
3. After receiving the amended text (with the # symbol in place of the vowels), it prints it.
4. After that, the connection is closed.

The input and output screenshot

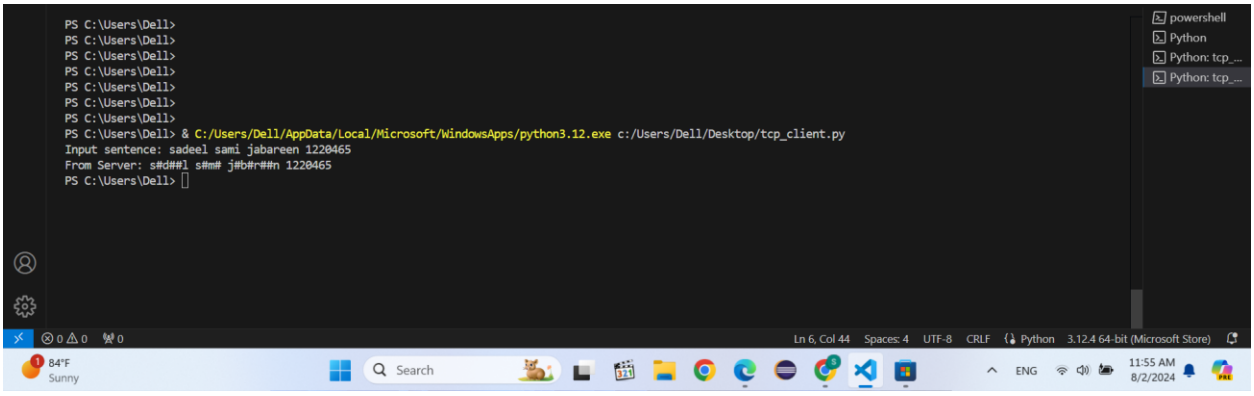


Figure 15 server result in TCP

Input sentence: sadeel sami jabareen 1220465
From Server: s#d##l s#m# j#b#r##n 1220465

How They Work Together:

1. Server Side: The server operates and waits for clients to connect.
When a client connects, it gets a sentence from the client, processes it by replacing vowels with #, and then returns the changed phrase to the client.
2. Client Side: The client connects to the server, submits the sentence, and waits for the updated sentence.When the amended sentence is received, it publishes the result and terminates the connection.

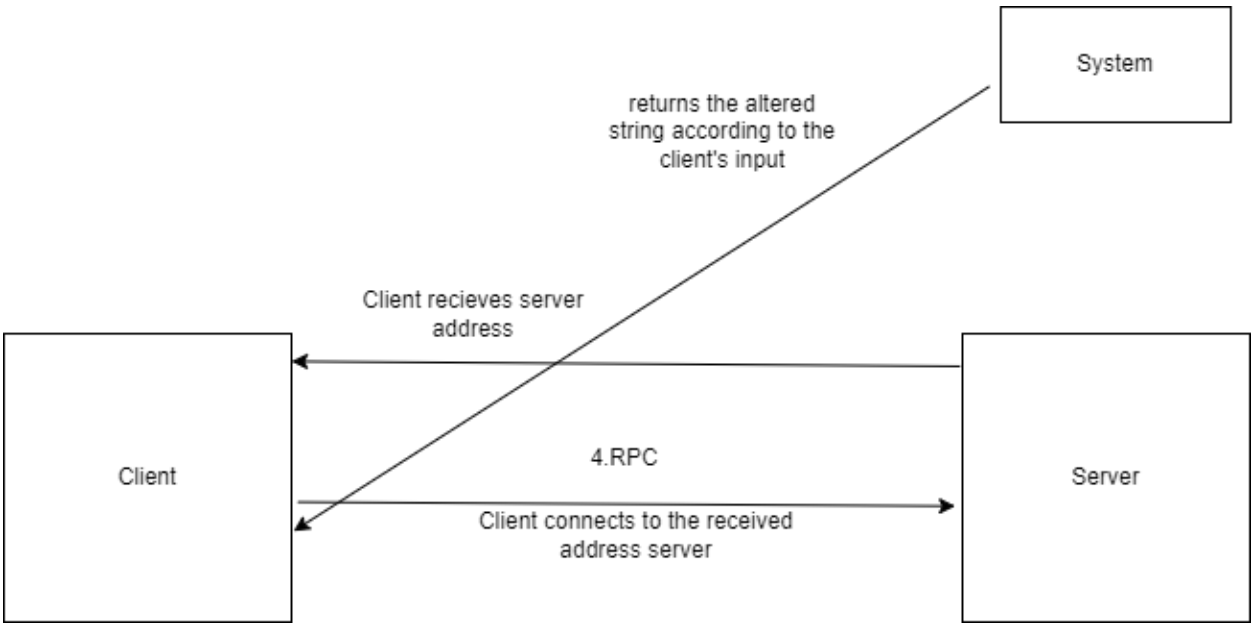


Figure 16 Flowchart for TCP client-server

UDP client and server applications

2.Using socket programming, implement UDP client and server applications (in go, python, java or C). The server (peer) should listen on a port number; this number should be selected based on the std.ID of one of the students in the team (e.g. 1201515 → port is 1515). All peers (clients and server) can send and receive messages. In other words, a message sent by a peer will be received by another peer called server at this moment, others can also send to the same peer (server) after each other. The message should include peer (client) and its number as well as any message (e.g. “Hello”). The peer (server) lists the last

received message from a peer (client) based on its communication with all peers.

First, I built a server class, and a client manager class using java.

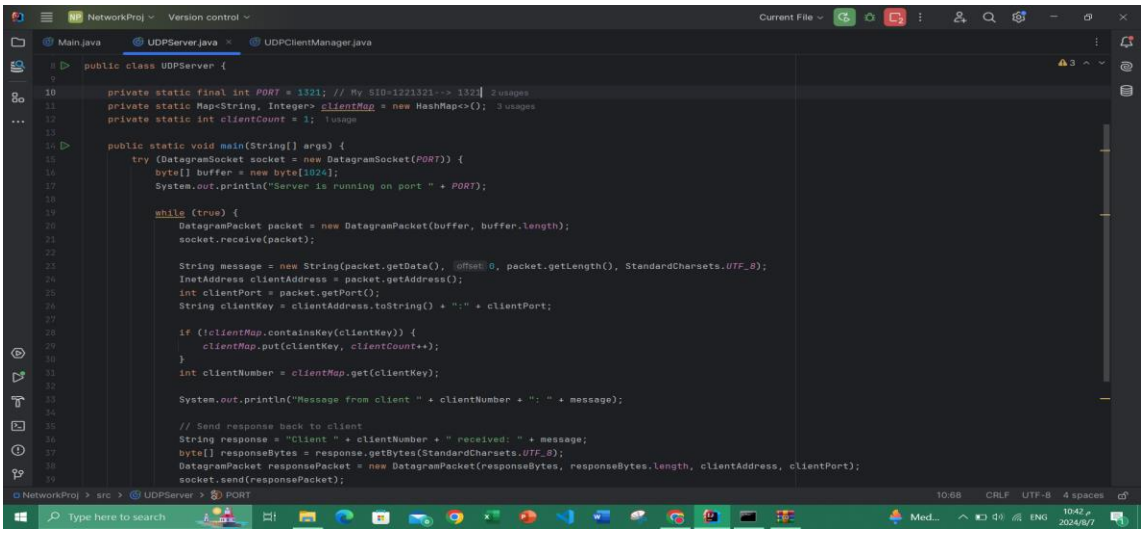


Figure 17 Server class

The server's port number is extracted from the student's D 1221321→ 1321
Client class:

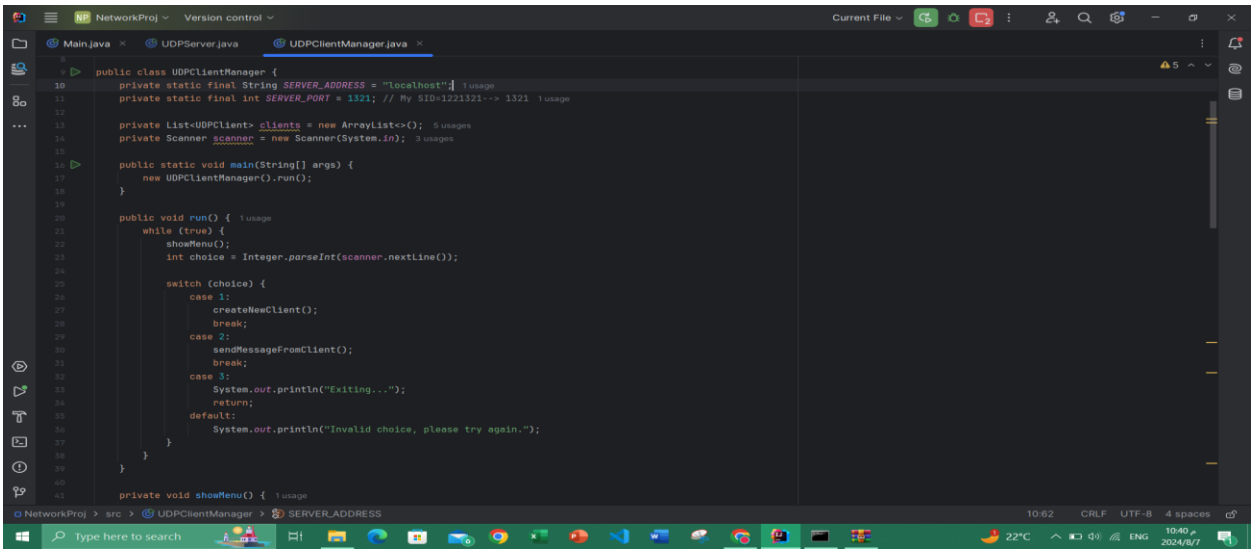


Figure 18 client manager class, part1

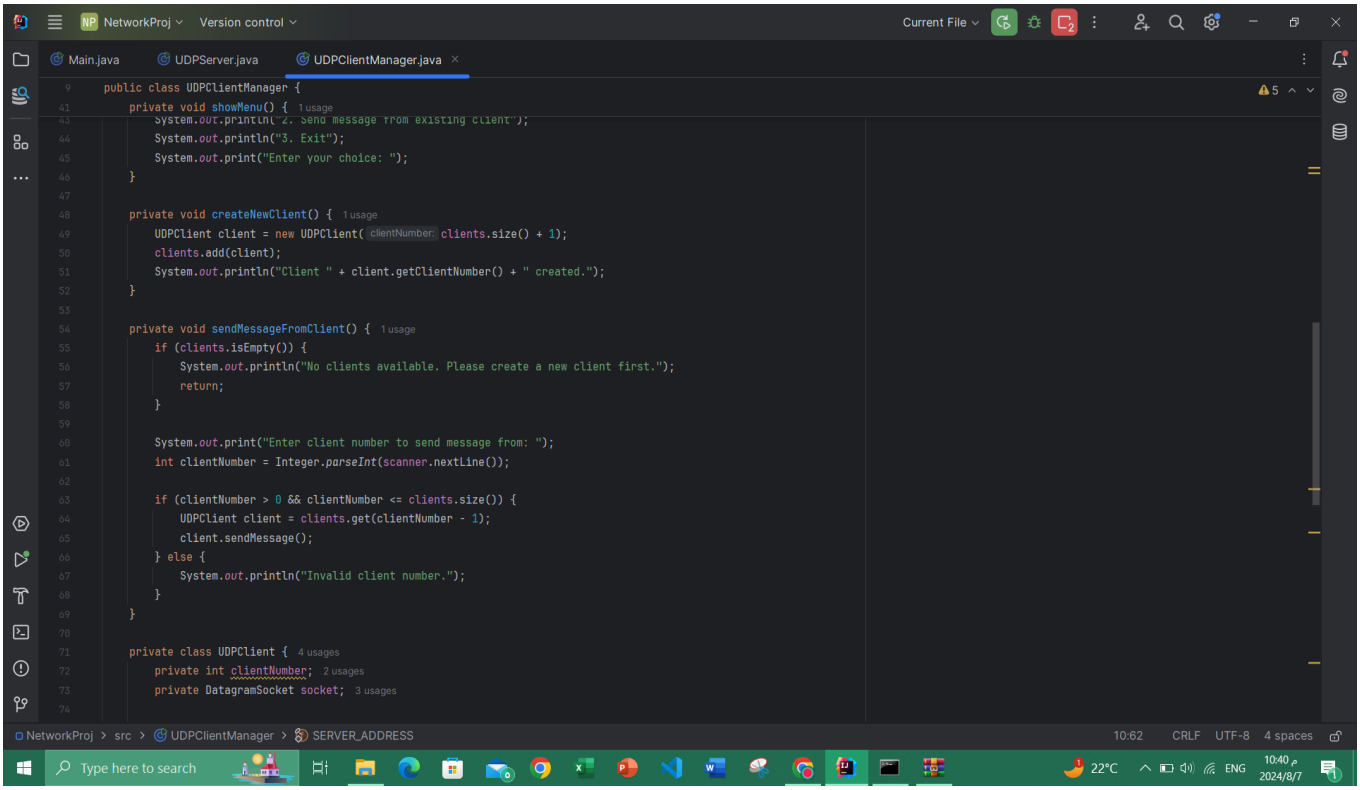


Figure 19 client manager class, part2

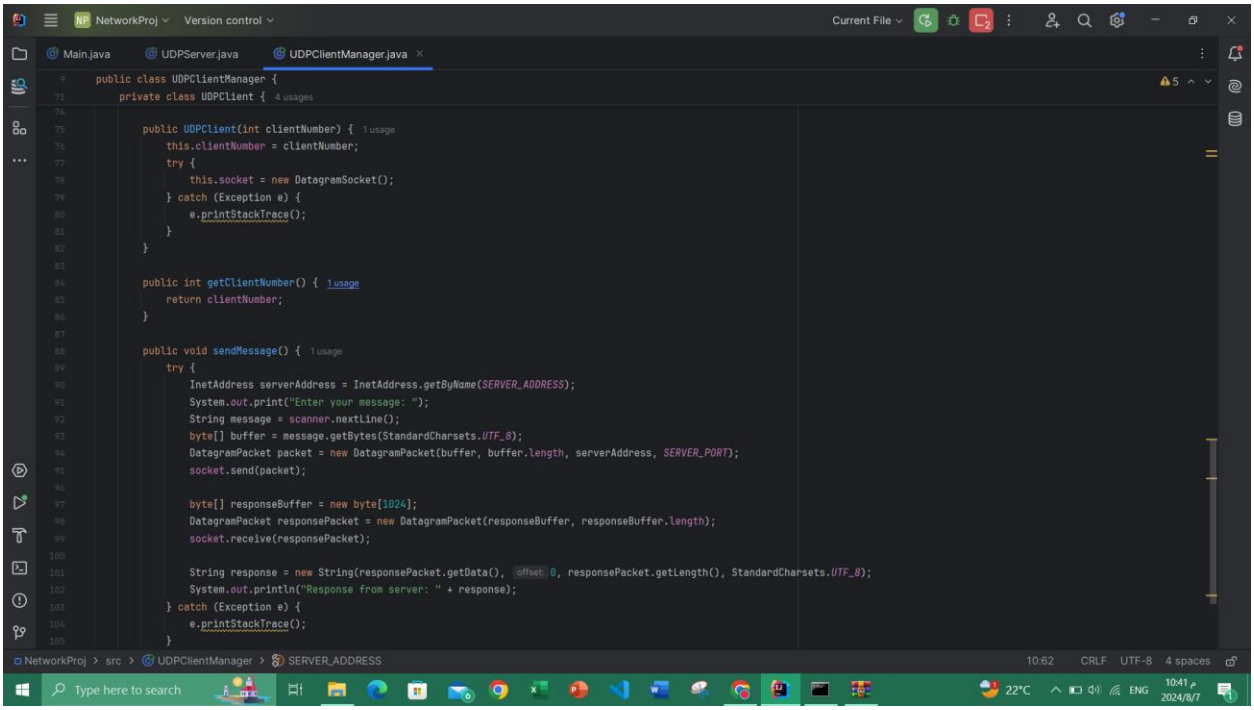


Figure 20 client manager class,part3

- To establish the idea of several clients that are able to send messages, even when a new client has been created, I made a menu that consists of the

choices to either create a new client, or send a message from an already created client and enter its number, or exit the program.

- **UDPClientManager (Client-side)**
  - **DatagramSocket**: Each client creates its own DatagramSocket to send and receive UDP packets.
  - **DatagramPacket**: Used to wrap the data (in bytes) being sent or received.
  - **InetAddress**: Represents the IP address of the server to which the client sends messages.
  - The client sends a message to the server and waits for a response. After receiving a message, it prints the server's response.
- **UDPServer (Server-side)**
  - **DatagramSocket**: The server listens on a specific port (1321 in this case) for incoming UDP packets.
  - **DatagramPacket**: The server receives packets from clients and sends back a response.
  - **Client Tracking**: The server tracks clients using a Map that binds each client's unique IP address and port combination to a client number.

Workflow

1. **Client Creation**: The UDPClientManager allows creating multiple clients. Each client operates independently, with its own socket.
2. **Message Sending**: A client sends a message to the server by creating a DatagramPacket with the server's address, port, and the message content.
3. **Server Reception**: The server receives the packet, extracts the message, and identifies the client by its IP address and port. It assigns a unique client number if the client is new.
4. **Server Response**: The server sends a response back to the client, echoing the received message along with the client's assigned number.
5. **Client Reception**: The client receives the server's response and displays it.

This setup is ideal for simple, low-latency communication where the overhead of establishing a connection (like in TCP) is unnecessary.

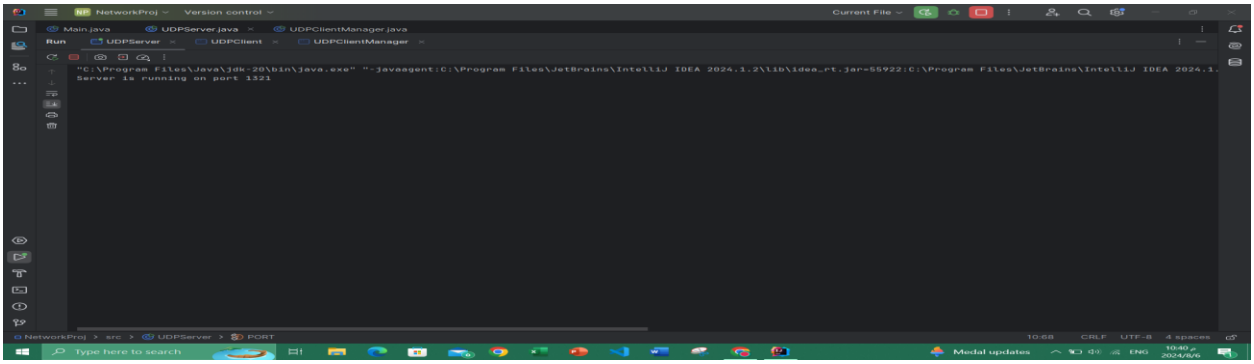


Figure 21 The server run

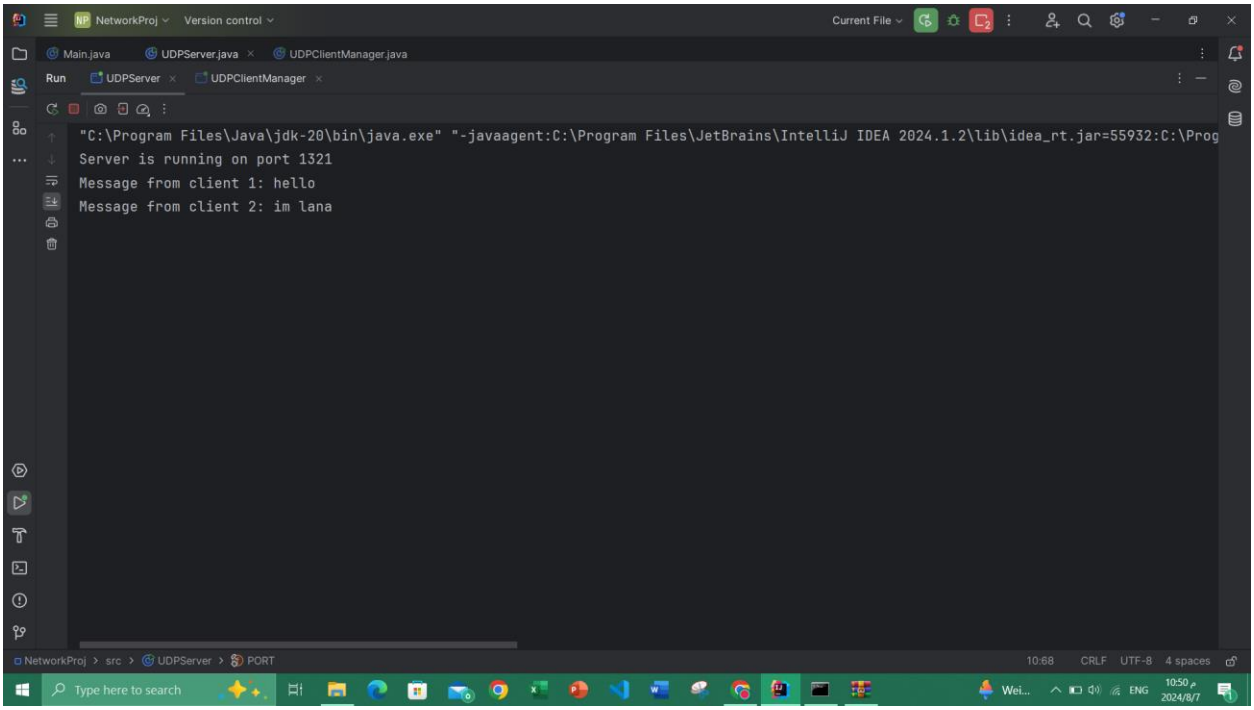


Figure 22 after the process is complete

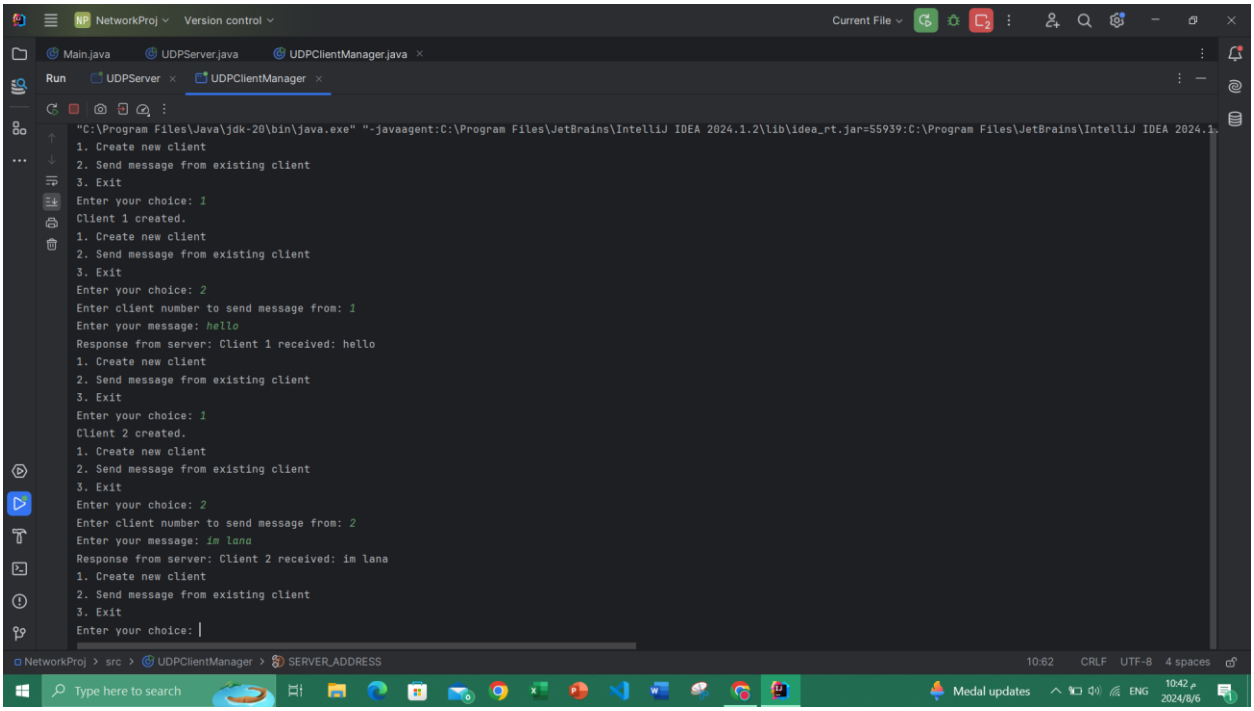


Figure 23 client class run

As shown in the picture above, the user can easily transmission between clients to send messages to the server, and the server receives the fully without any losses or delay.

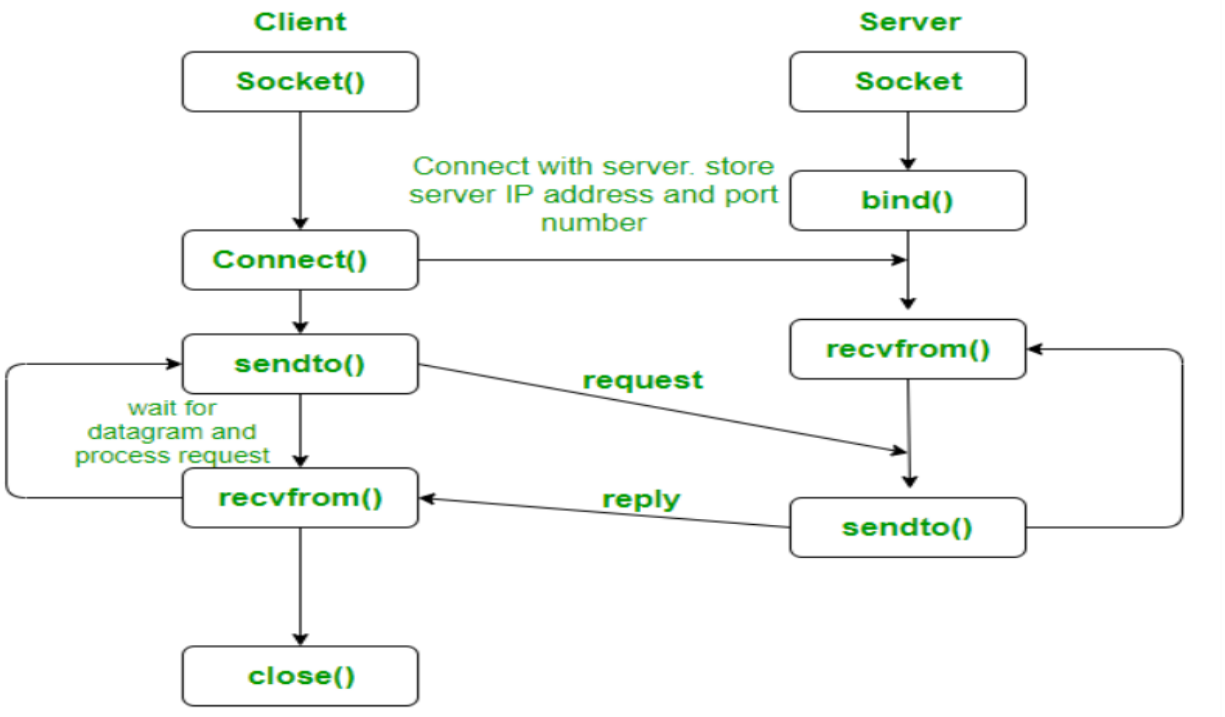


Figure 24 Flowchart for UDP client-server

Task3: Web Server :

Using socket programming, implement a simple but a complete web server (in go, python, java or C) that is listening on a port number; this number should be selected based on the std.ID of one of the students in the team (e.g. 1201515 → port is 1515). Make the code as general as possible. The user types in the browser something like http://localhost:1515/ar or http://localhost:1515/en.

- Note : The ID is 1221321 → Port is 1321

1- If the request is / or /index.html or /main_en.html or /en (for example localhost:1515/ or localhost:1515/en) then the server should send main_en.html file with Content-Type: text/html.
The main_en.html file should contain HTML webpage that contains:
a. "ENCS3320-My First Webserver" in the title.
b. "Welcome to Computer Networks, ENCS3320-Webserver" (part of the phrase is in RED).
c. Group members' names and IDs.
d. Some information about the group members. For example, projects you have done during different course (programming, electrical, math, etc), skills, hobbies, etc.
e. Use CSS to make the page looks nice.
f. Divide the page in different boxes and put student's information in the different boxes.
g. Include CSS as a separate file.
h. The page should contain at least an image with extension .jpg and an image with extension .png.
i. A link to a local html file (mySiteSTDID.html).
j. A link to https://www.birzeit.edu/.

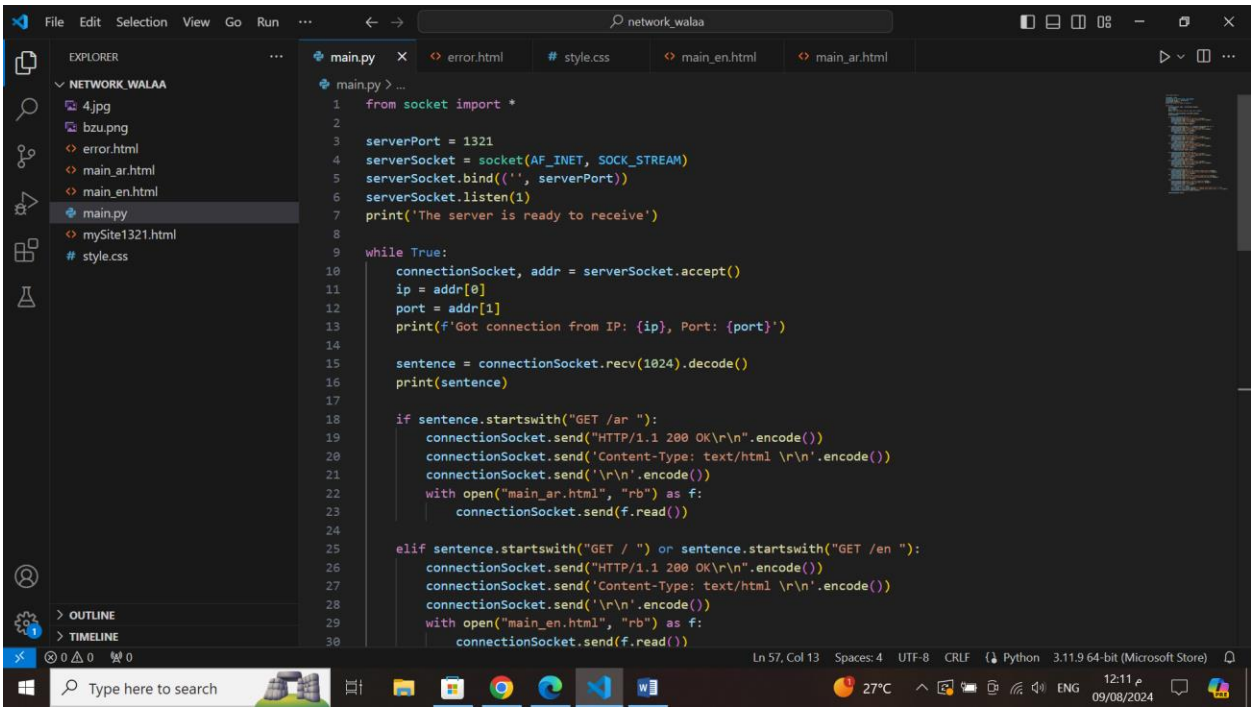


Figure 25: The code of main class of web server

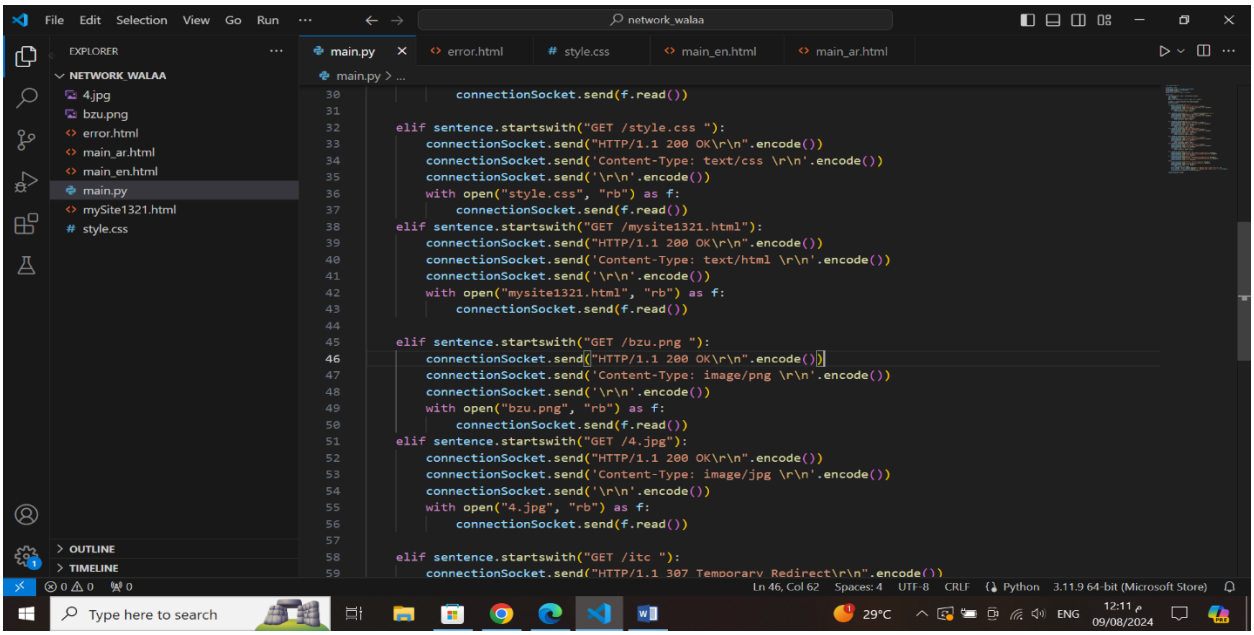


Figure 26: The code of main class of web server

figure (23,24) is a Python code implements a simple HTTP server using the socket module. The server listens for incoming connections and responds with different types of content based on the HTTP request received. It handles various types of requests including HTML files, CSS files, images, and redirects.

When a client makes a request to the server with the path / or /en (e.g., http://localhost:1321/ or http://localhost:1321/en), the server responds with the main_en.html

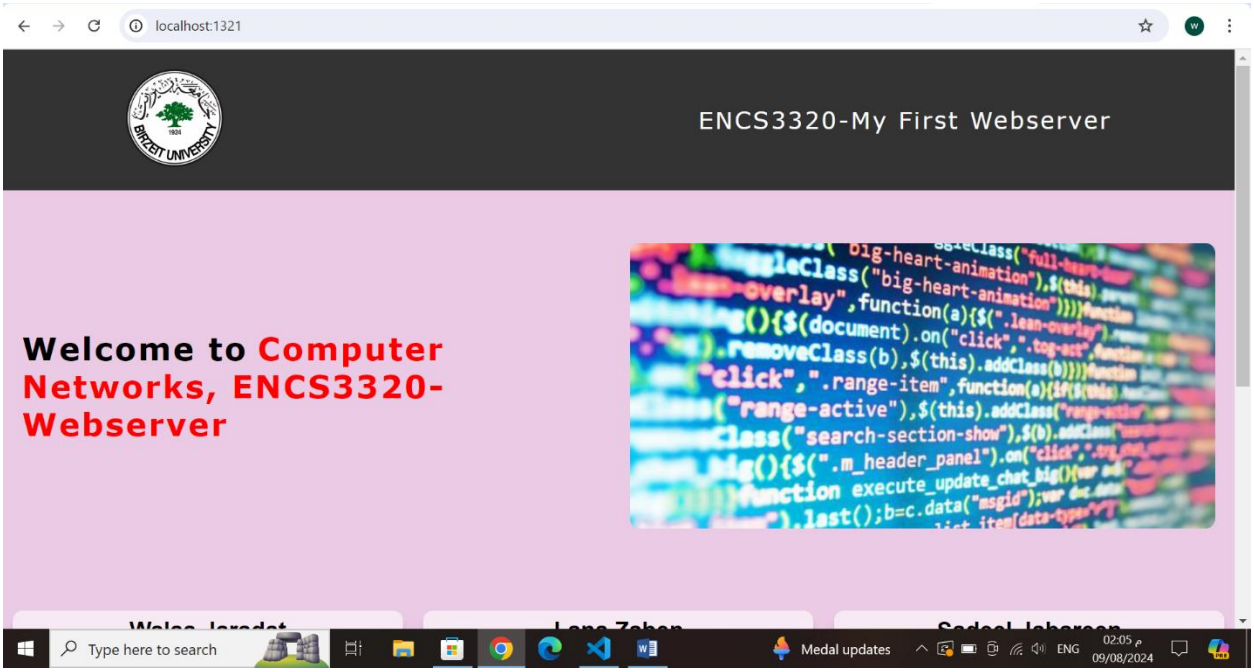


Figure 27: The web site of localhost:1321 /en

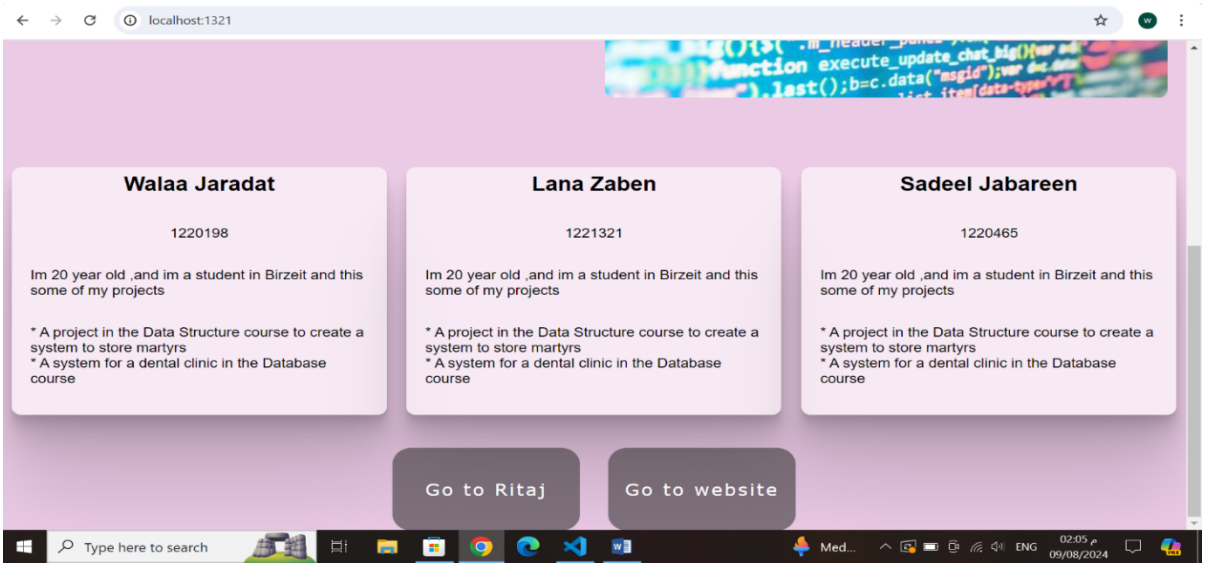


Figure 28: The web site of localhost:1321/en

When a client requests the path / or /en, the server responds with the same HTML page, main_en.html . providing a well-structured HTML page with a title, welcome message, group

information, CSS styling, images, and relevant links (when we click on **Go to Ritaj** ritaj website will open and when we click on **Go to Website** local html file (mySite1321.html will open .

2- If the request is /ar then the server response with main_ar.html which is an Arabic version of main_en.html.

When a client makes a request to the server with the path /ar (e.g., http://localhost:1321/ar), the server responds with the main_ar.html

Figure 29 : The web site of localhost:1321/ar

Figure 30:The web site of localhost:1321/ar

3- If the request is .html file, then the server should send the requested html file with Content-Type: text/html. You can use any html file. Make it general (not only for specific filename).

When a client makes a request to the server .html,

- (e.g.,http://localhost:1321/mysite1321.html), the server responds with the mysite1321.html .

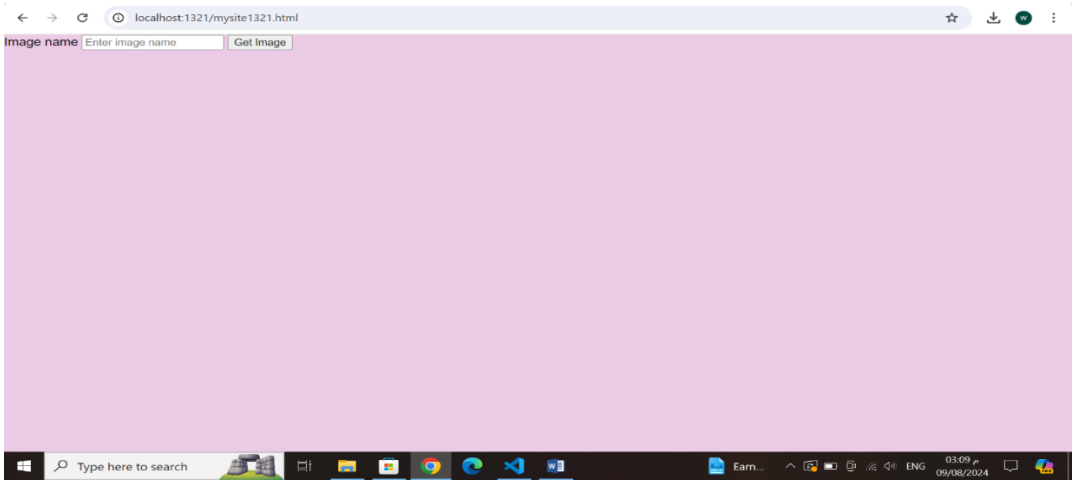


Figure 31:The web site of localhost:1321/mysite1321.html

4- If the request is .css file, then the server should send the requested css file with Content-Type: text/css. You can use any CSS file. Make it general (not only for specific filename).

When a client makes a request to the server .css, (e.g.,http://localhost:1321/style.css), the server responds with the style.css .

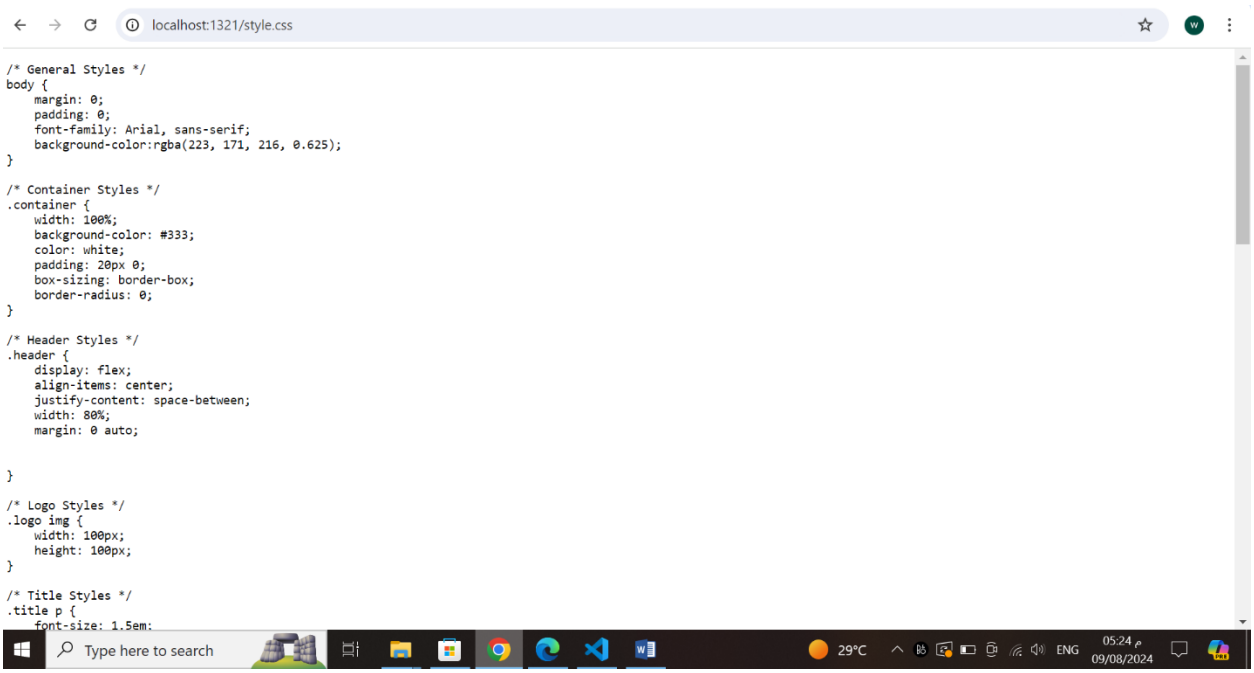


Figure 32:Content of Style.css

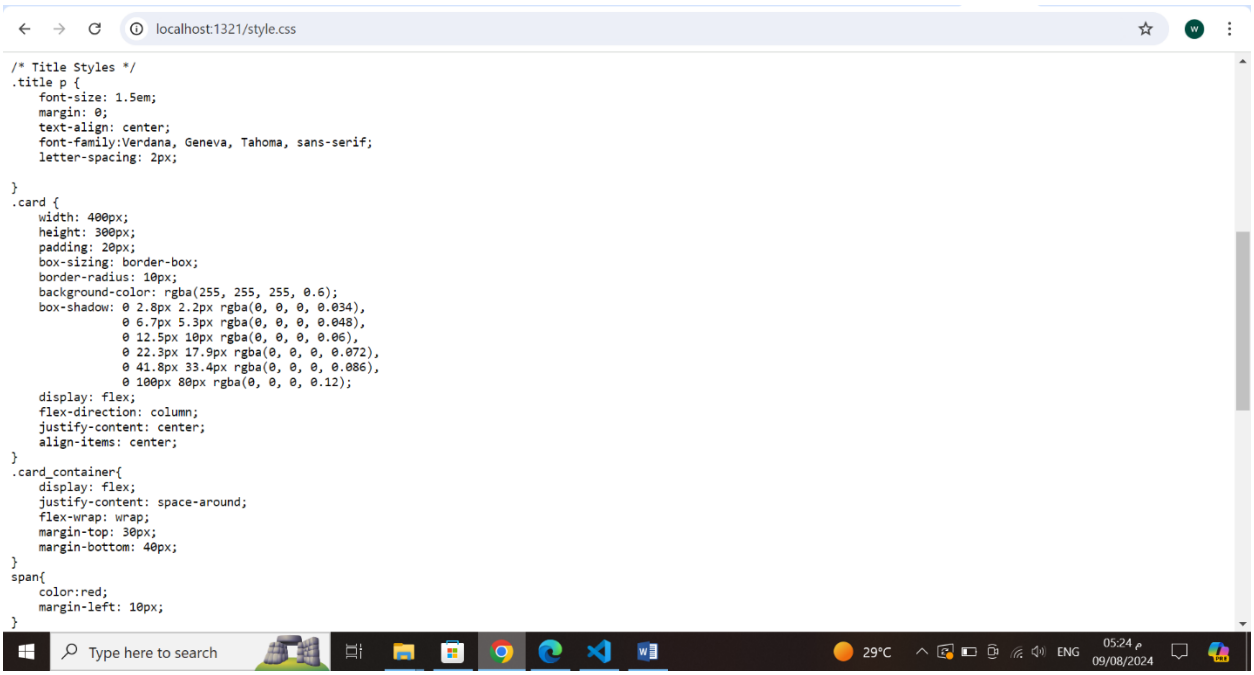


Figure 33:Content of Style.css

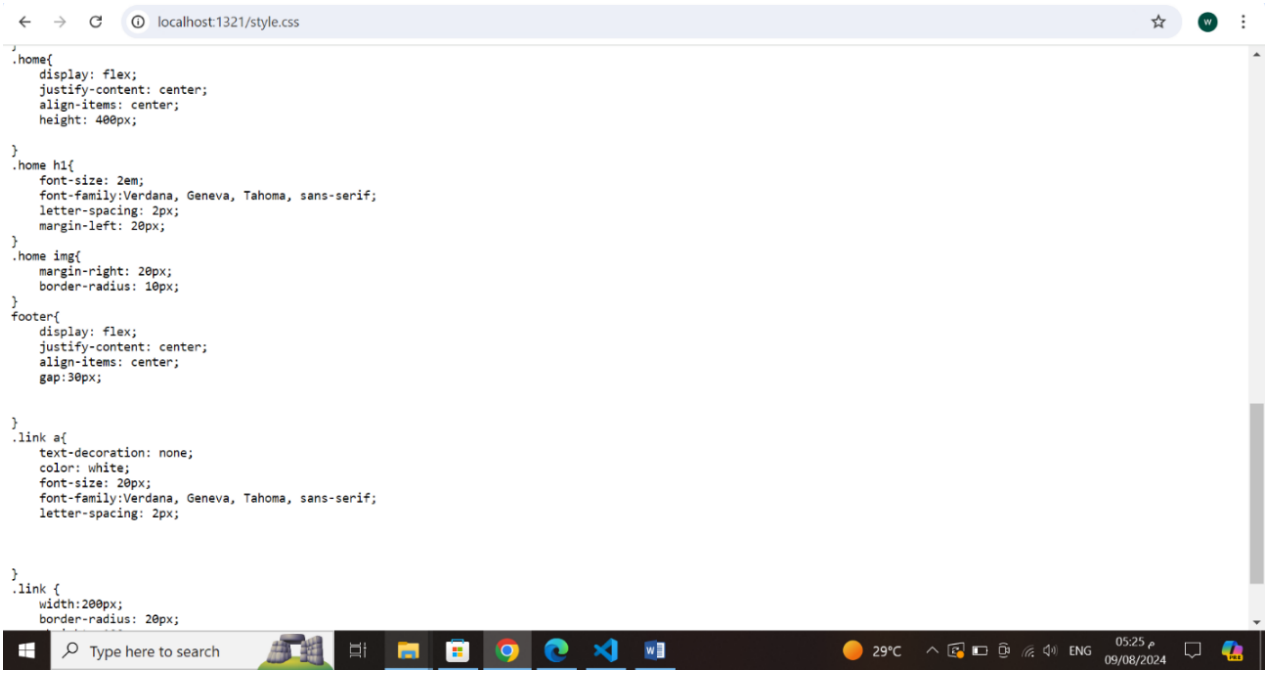


Figure 34:Content of Style.css

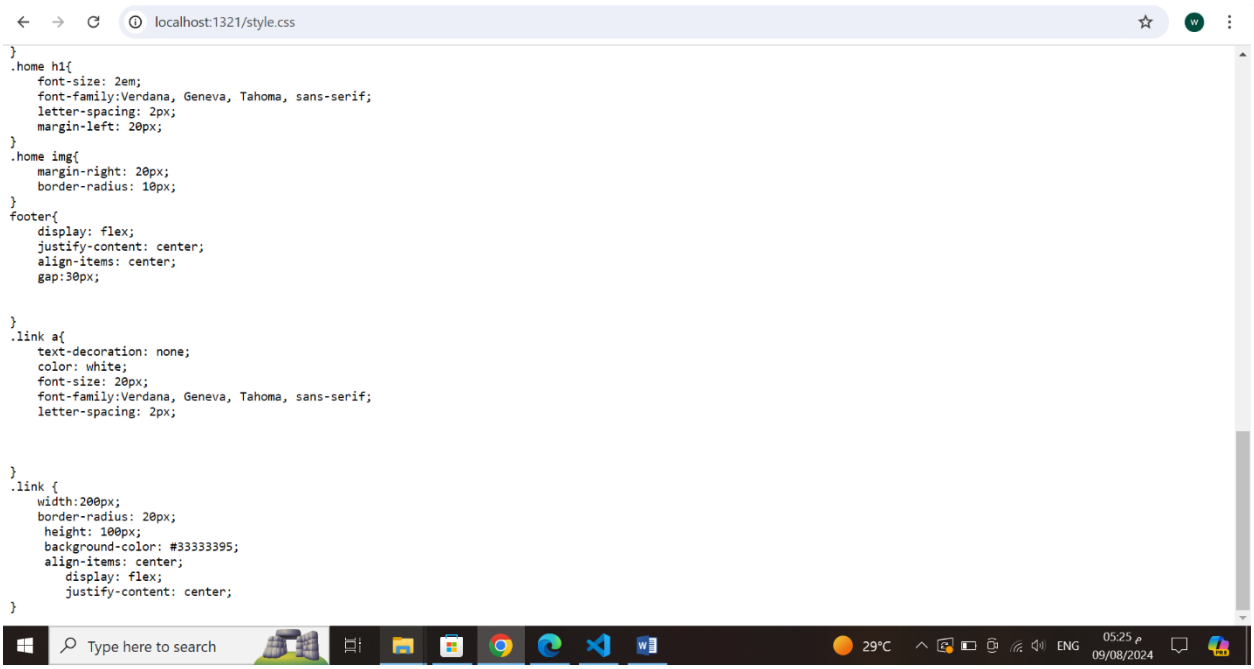


Figure 35:Content of Style.css

5- If the request is .png, then the server should send the png image with Content-Type: image/png. You can use any image. Make it general (not only for specific filename).

When we run the code and opened localhost:1321 in the browser, the terminal will display information similar to the image we provided. It will log incoming connections, the request details, and how the server handles each request.

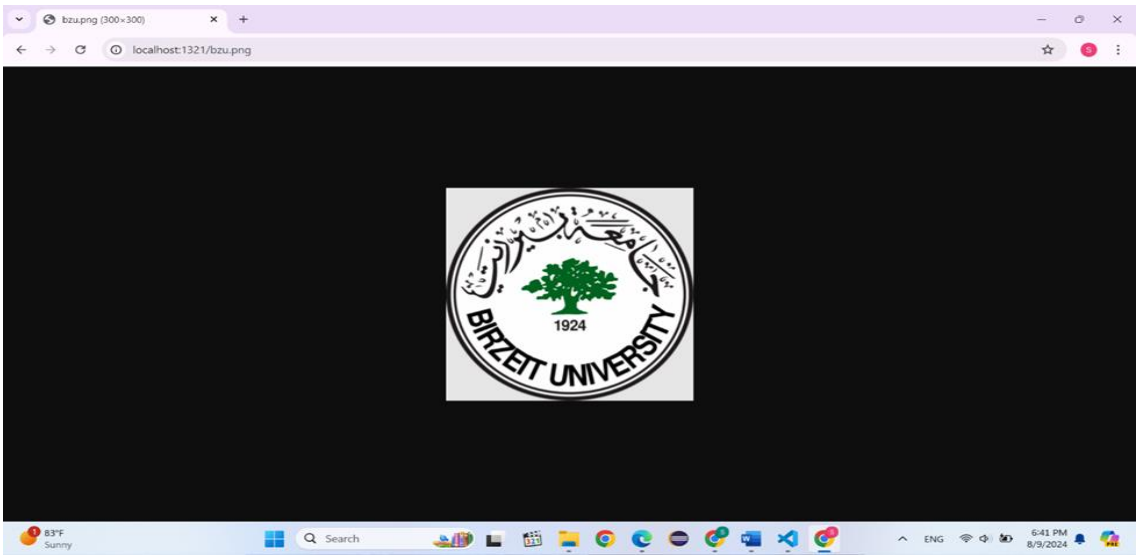


Figure 36:The result output of png

```
The server is ready to receive
Got connection from IP: 127.0.0.1, Port: 64893
GET /bzu.png HTTP/1.1
Host: localhost:1321
Connection: keep-alive
sec-ch-ua: "Not)A;Brand";v="99", "Google Chrome";v="127", "Chromium";v="127"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Windows"
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/127.0.0.0 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exc
hange;v=b3;q=0.7
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate, br, zstd
Accept-Language: en-US,en;q=0.9


Got connection from IP: 127.0.0.1, Port: 64894
```

Figure 37:Response an image with png format

6- If the request is .jpg, then the server should send the jpg image with Content-Type: image/jpeg. You can use any image. Make it general (not only for specific filename).

When we run the code and opened localhost:1321 in the browser, the terminal will display information similar to the image we provided. It will log incoming connections, the request details, and how the server handles each request.

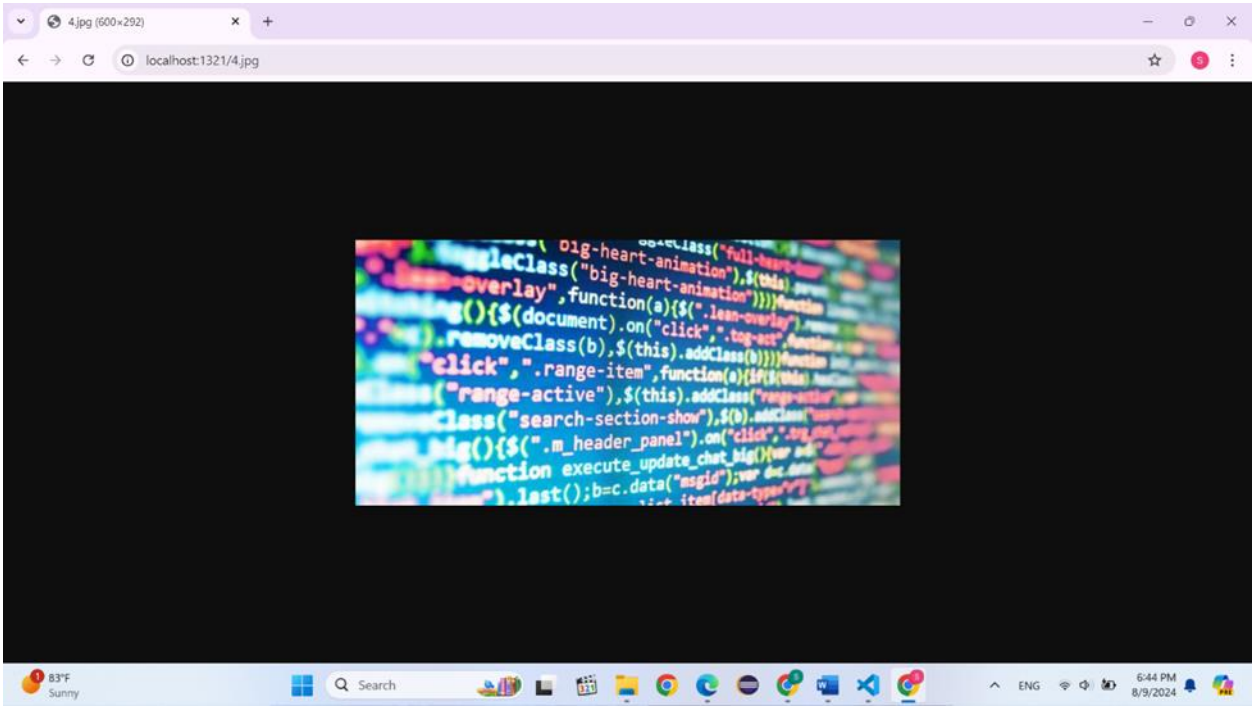


Figure 38:The result output of jpg

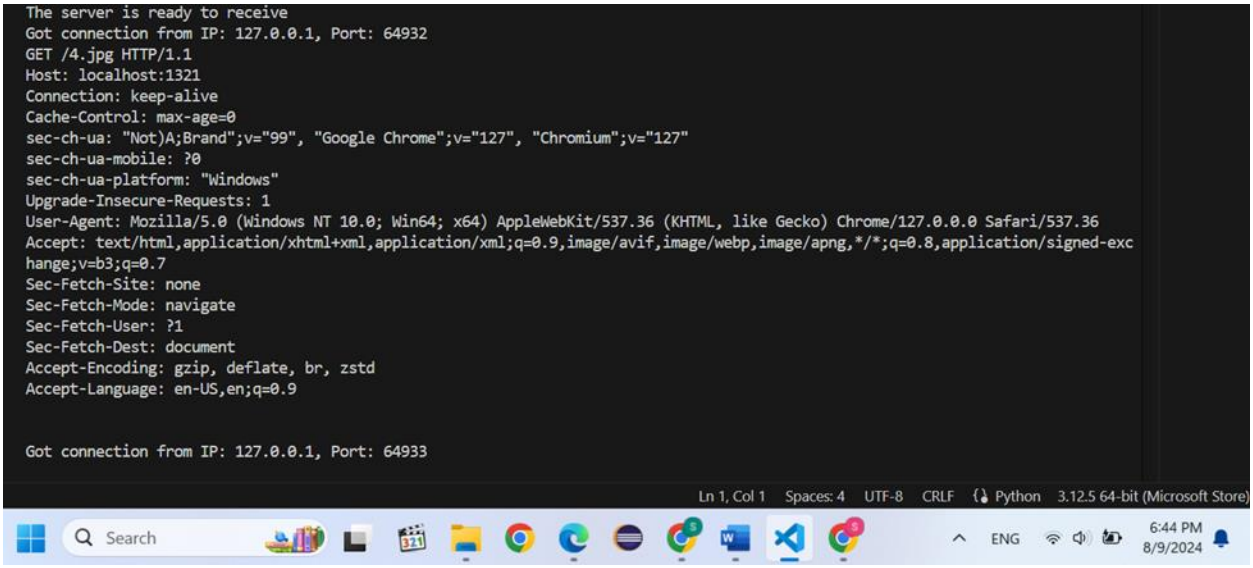


Figure 39:Response an image with jpg format

8- Use mySiteSTDID.html to get image by typing the name of the image in a box For instance:

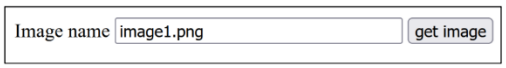


Figure 40:style of box in 8

When a client makes a request to server.html (e.g., http://localhost:1321/mySite1321.html), the server responds with mySite1321.html.
After selecting an image and entering her name in the box, press the "get image" button to display the image.

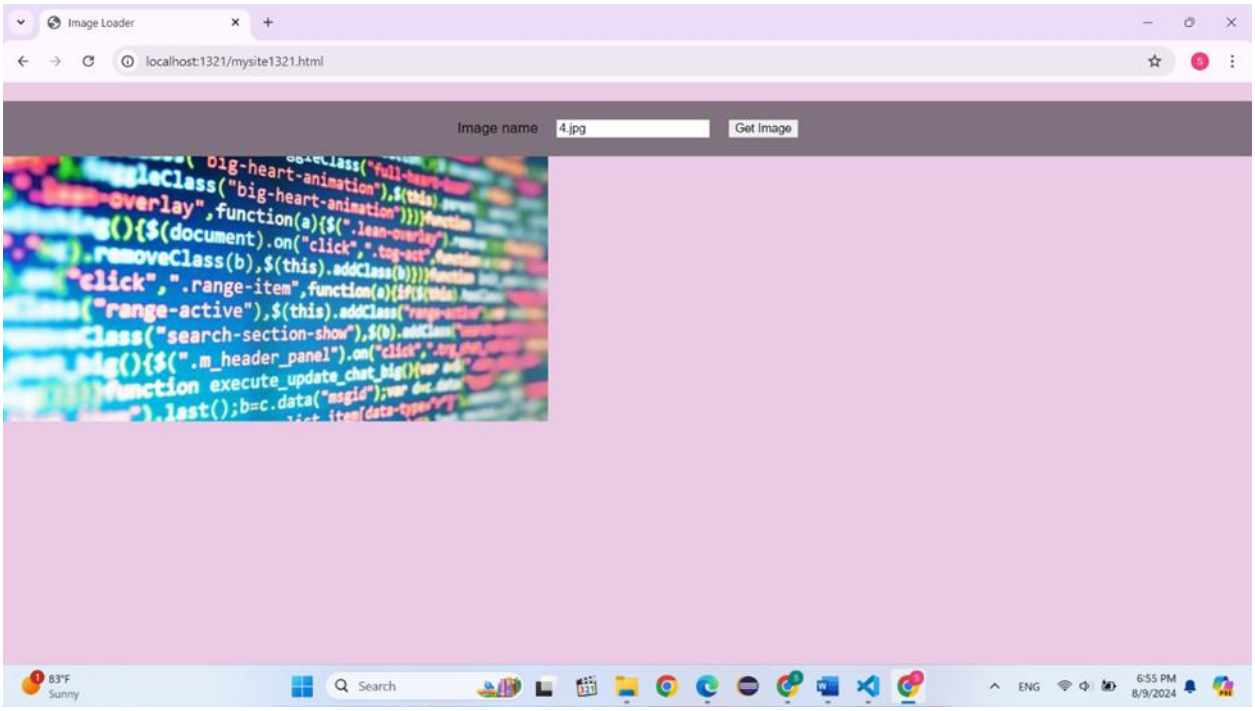


Figure 41:result of Operation

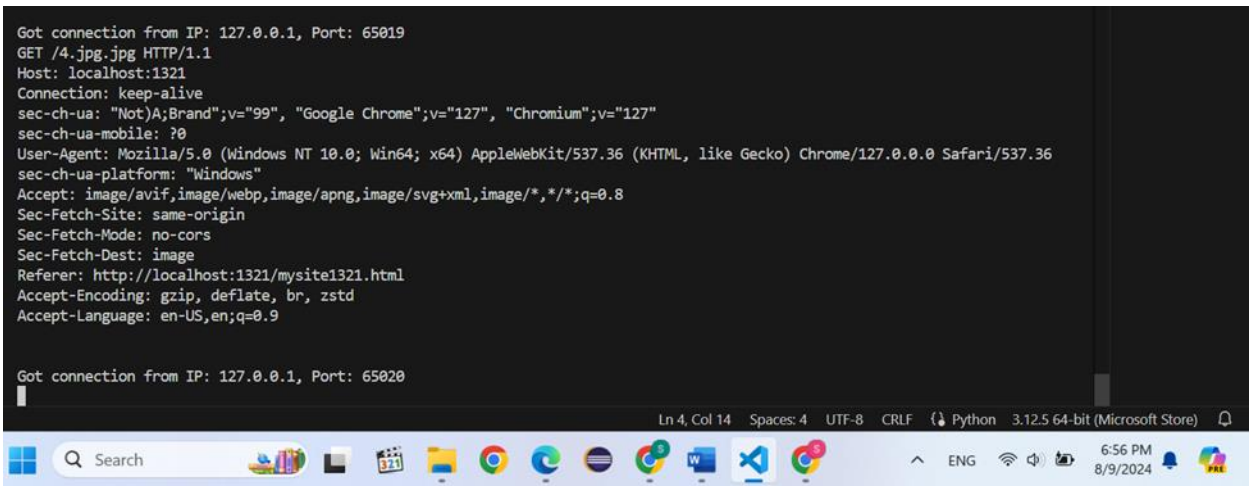


Figure 42:Response server with mySite1321.html

9- Use the status code 307 Temporary Redirect to redirect the following;

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

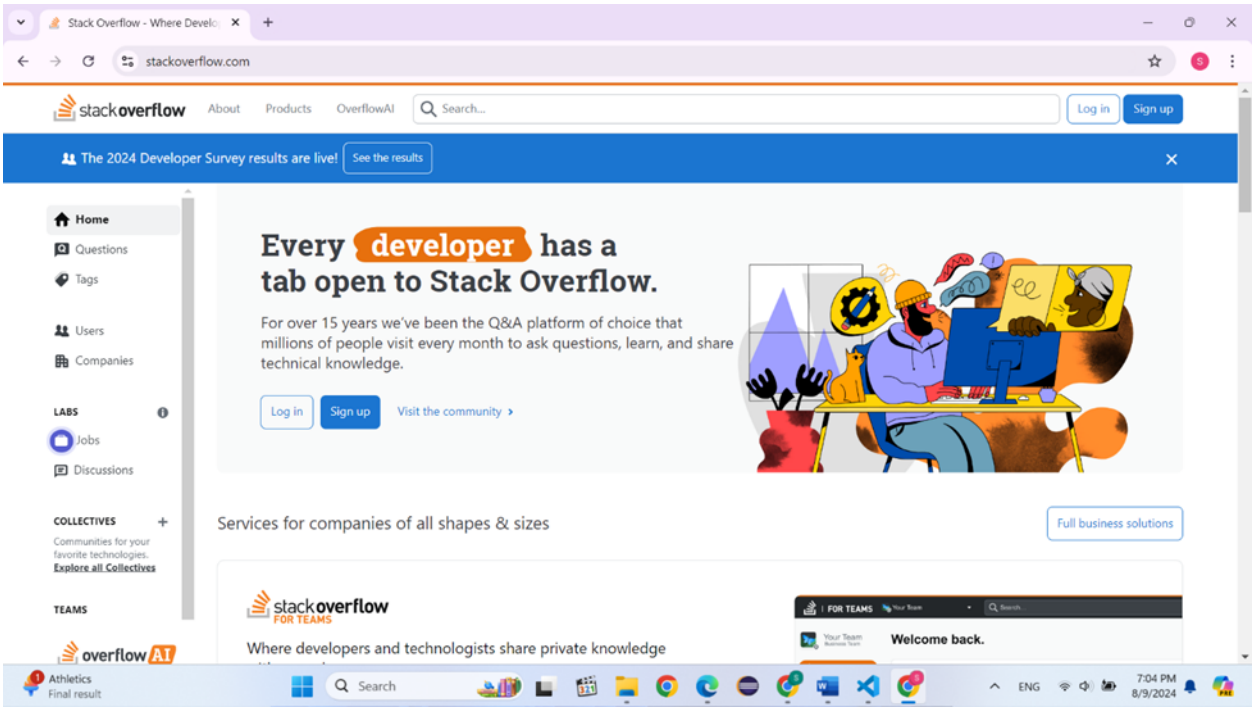


Figure 43;Redirect to stackoverflow.com

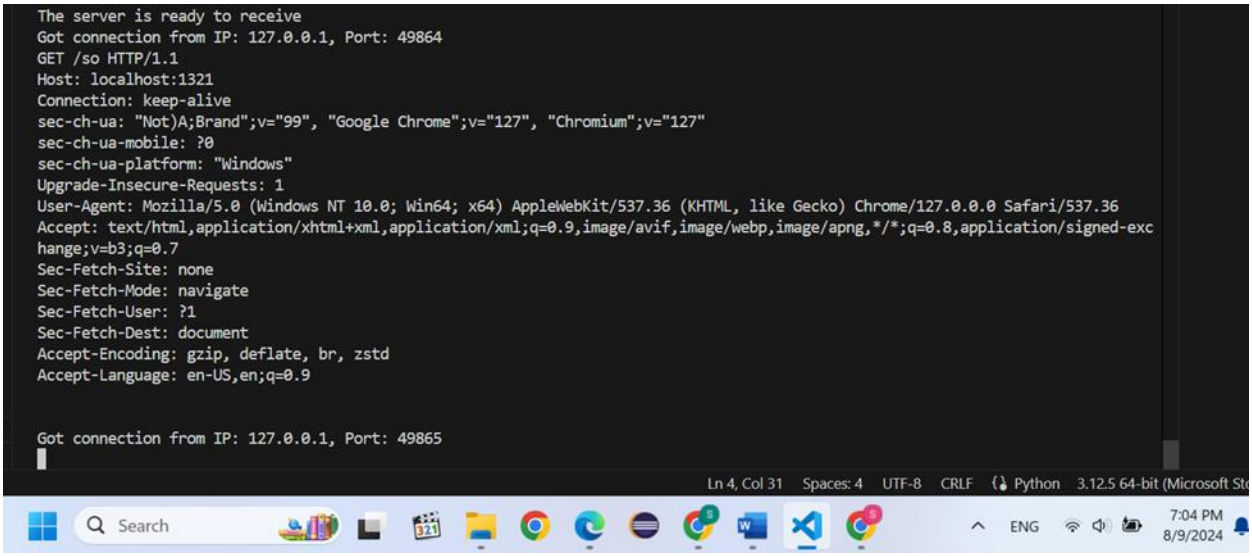


Figure 44:Response that redirects the client to the respective stackoverflow

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

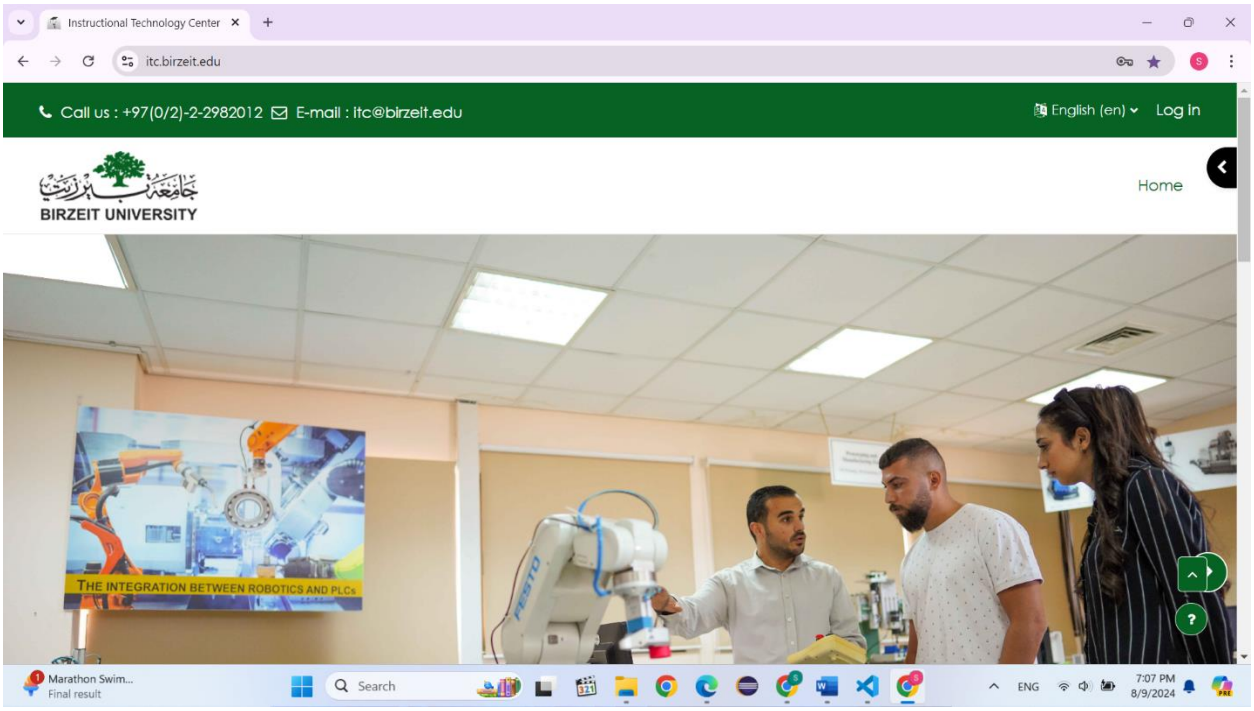


Figure 45:Redirect to itc.birzeit.edu

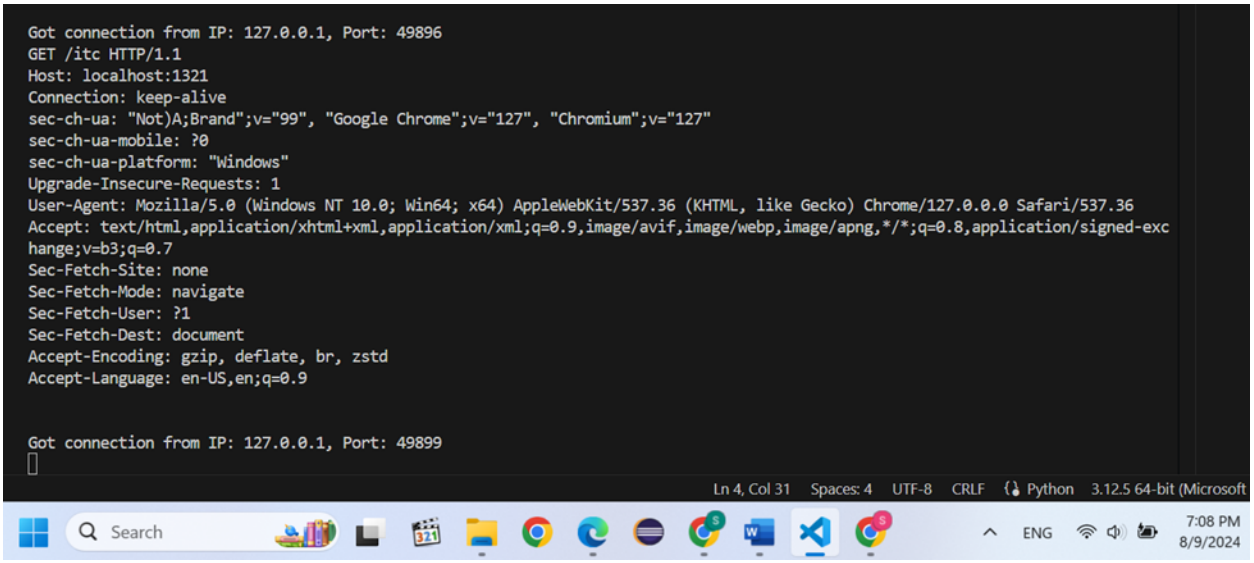


Figure 46:Response that redirects the client to the respective itc

307 Temporary Redirect: This status code tells the client that the requested resource has been temporarily moved to another URL. [1]
The client should continue using the original request method (GET, POST, etc.) for subsequent requests to the new URL .

The server listens for requests and, based on the requested path (e.g., /so or /itc), it sends a response that redirects the client to the respective website (Stack Overflow or Birzeit University ITC).

10- If the request is wrong or the file doesn't exist, the server should return a simple HTML webpage that contains (Content-Type: text/html)
- "HTTP/1.1 404 Not Found" in the response status
- "Error 404" in the title
- "The file is not found" in the body in BLUE
- Your names and IDs in Bold
- The IP and port number of the client

For example the client entered this link using 1321 port(which is the port for the server were using) but asked for permisson for an unknown file in the web for this server it will return a page that contain the error 404, which means source not found, in this example the user entered http://localhost:1321/lll, the file lll is not an object in this server:

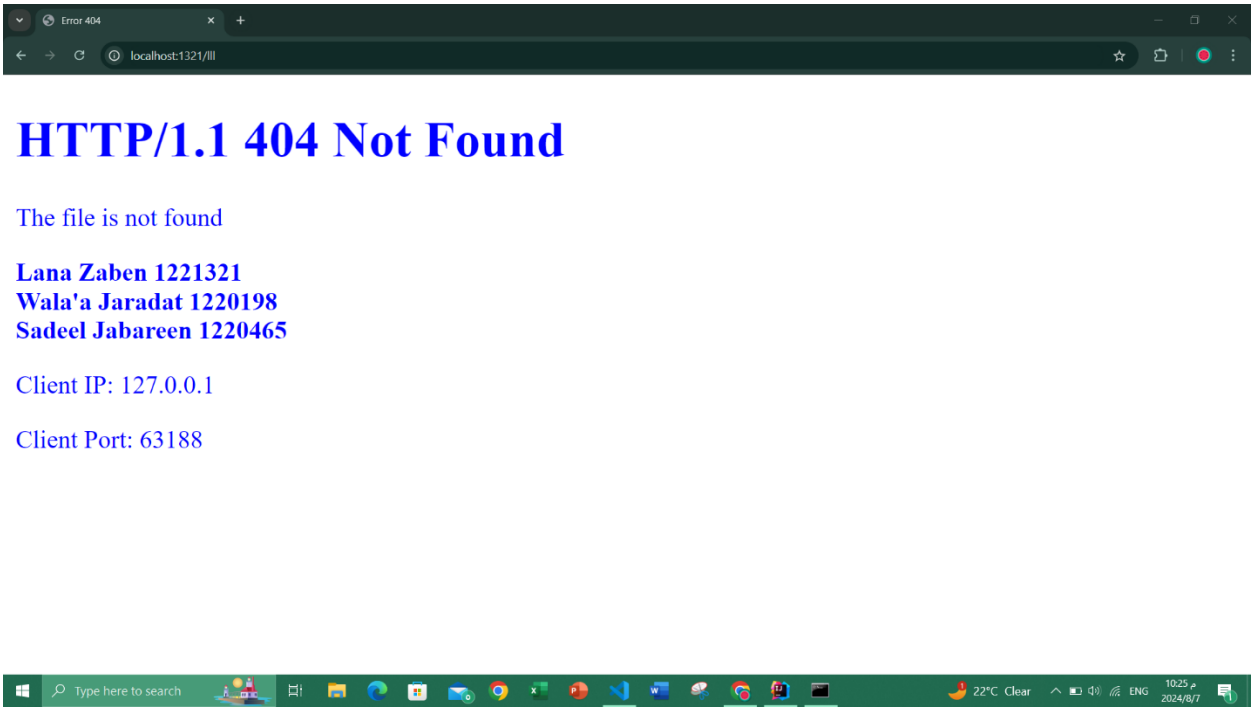


Figure 47:error page after searching 'lll' nonexistent file

Error class code snippet:

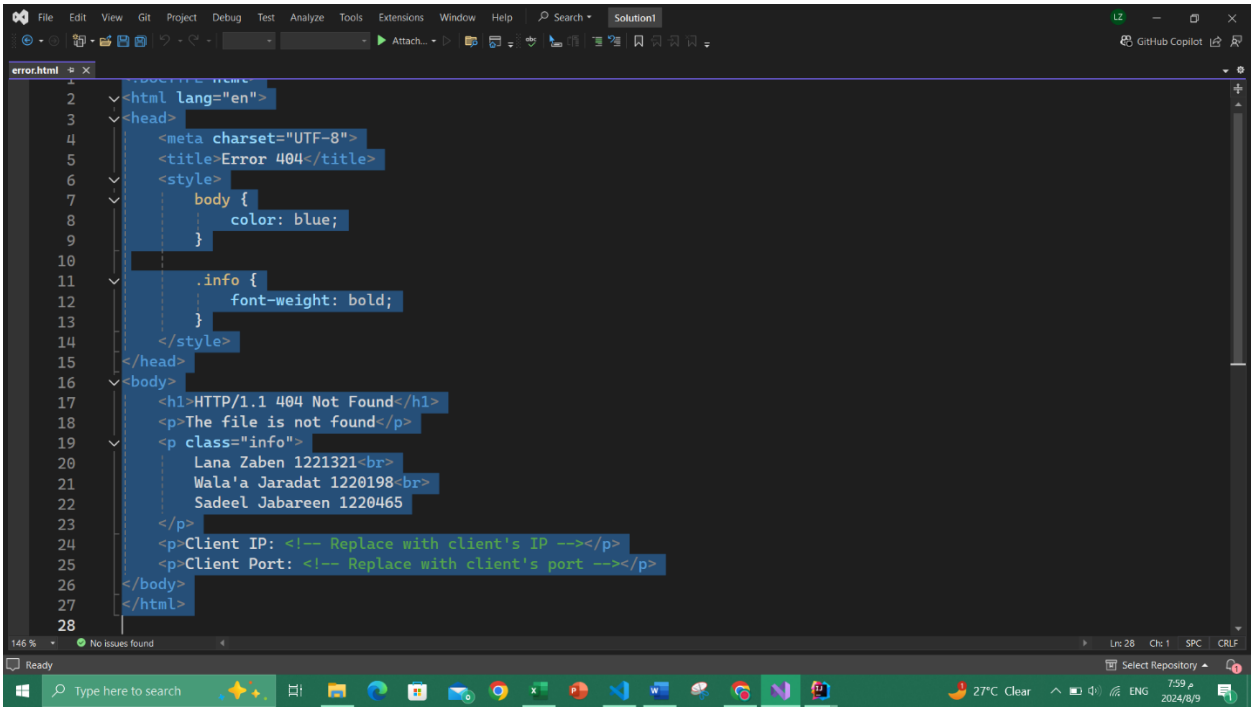


Figure 48:Error code snippet

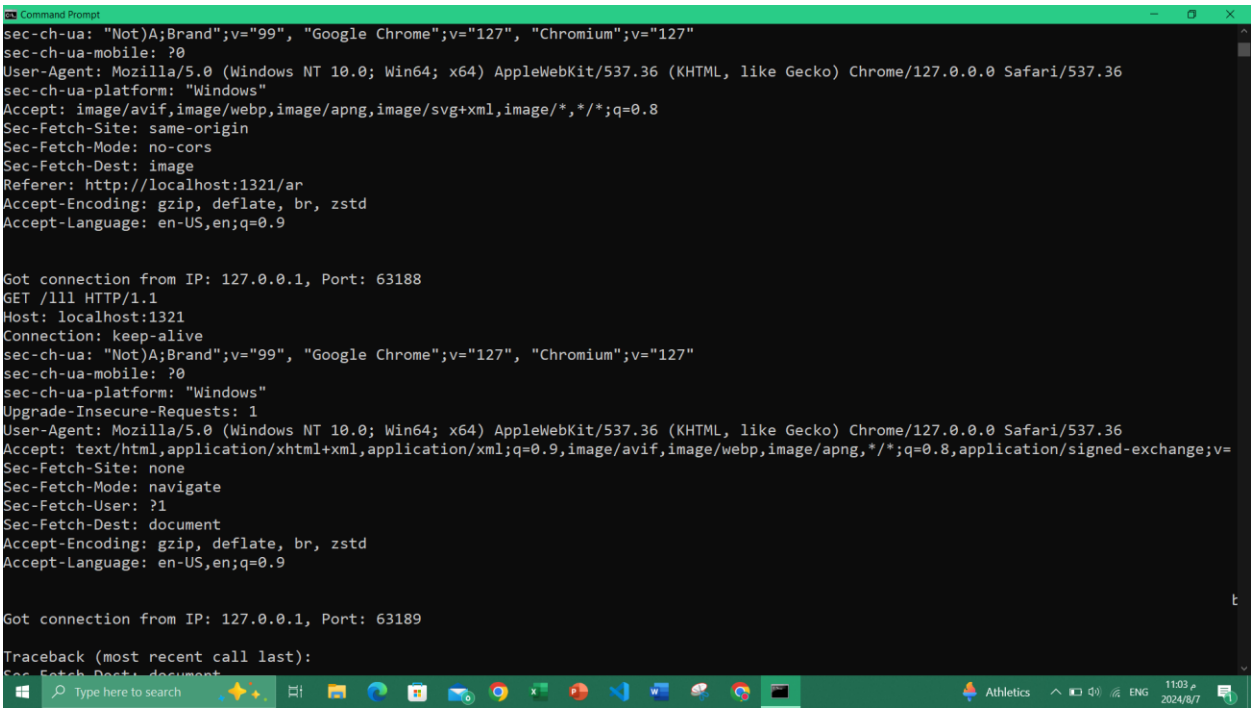


Figure 49:Command prompt interface

I accessed the files for the webpage through the command prompt interface, and run the main python class, which made it ready to service. When opening the webpage on google, and then trying to access an unknown file this is the result in the command prompt interface. It is proof that the error page was evoked.

11- The program should print the HTTP requests on the terminal window (command line window).

Here is some example for terminal window after doing HTTP requests :

- **The HTTP request : Localhost:1321**

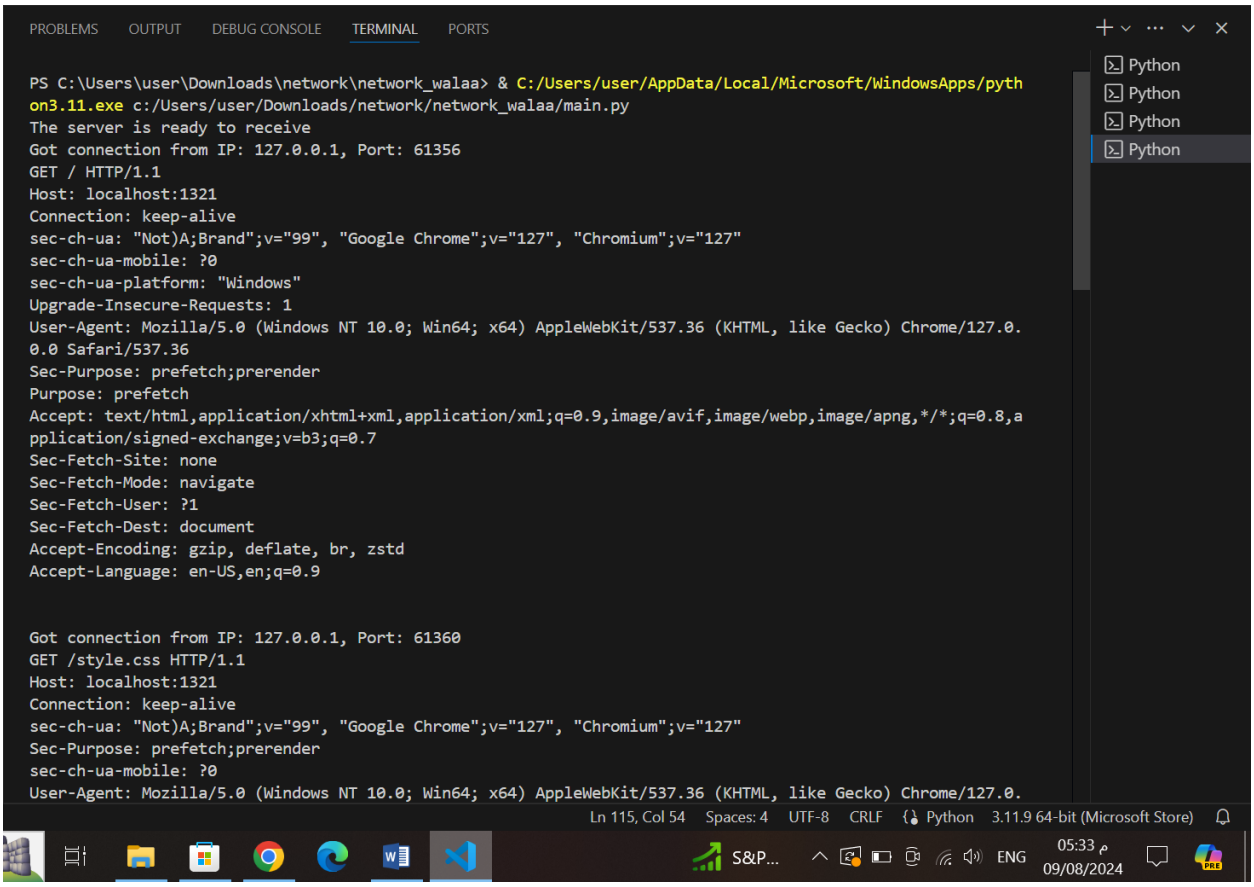


Figure 50:Terminal windo after doing HTTP requests

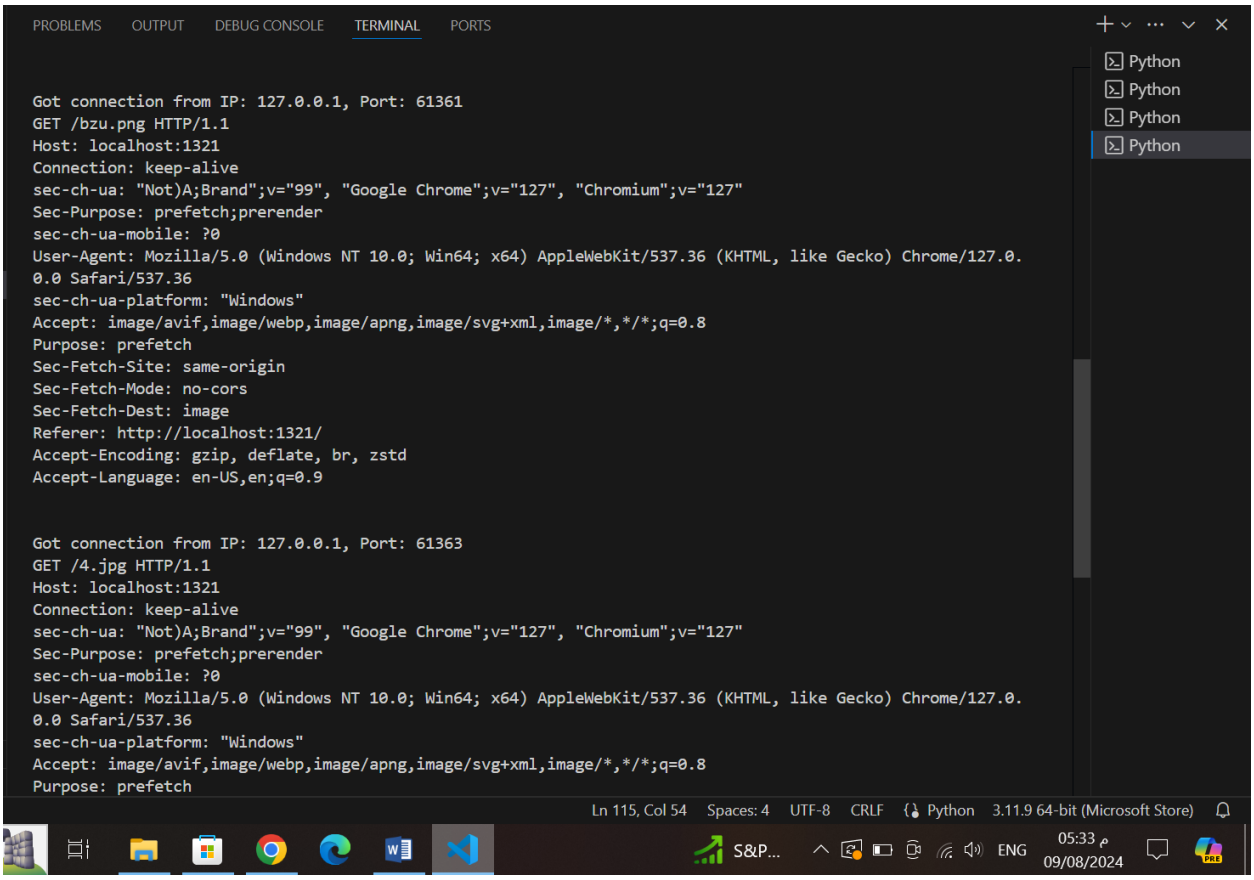


Figure 51:Terminal windo after doing HTTP requests

- **The HTTP request : localhost:1321/mysite1321.html**

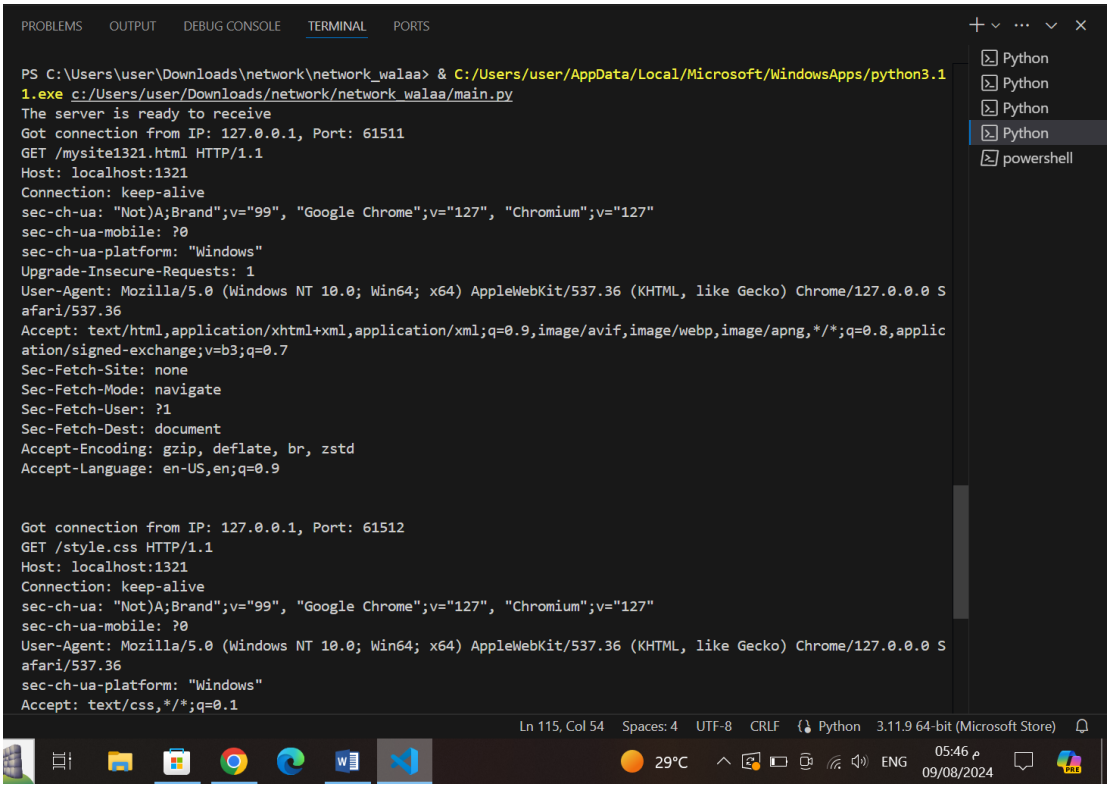


Figure 52:Terminal windo after doing HTTP requests

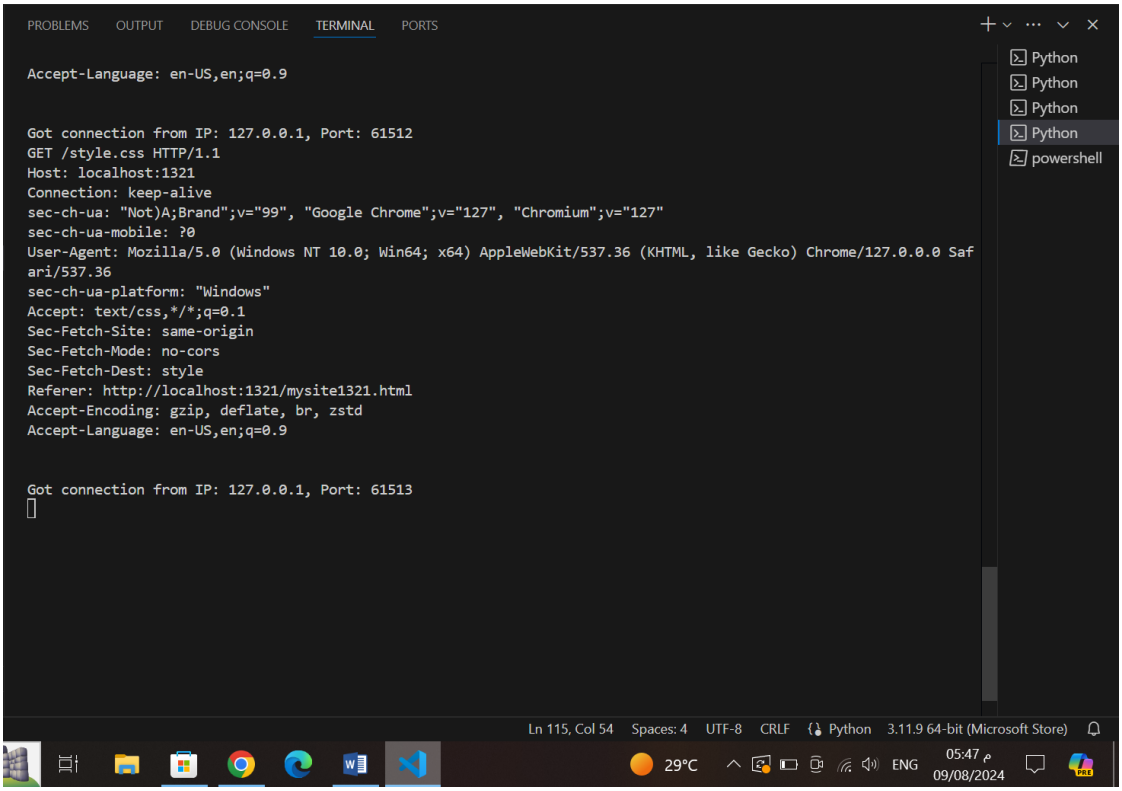


Figure 53:Terminal windo after doing HTTP requests

The program prints incoming HTTP requests on the terminal to display key information such as the client's IP address and port, the requested resource, and various HTTP headers (e.g., Host, User-Agent, Accept). This output helps in understanding and debugging how the server processes client requests.

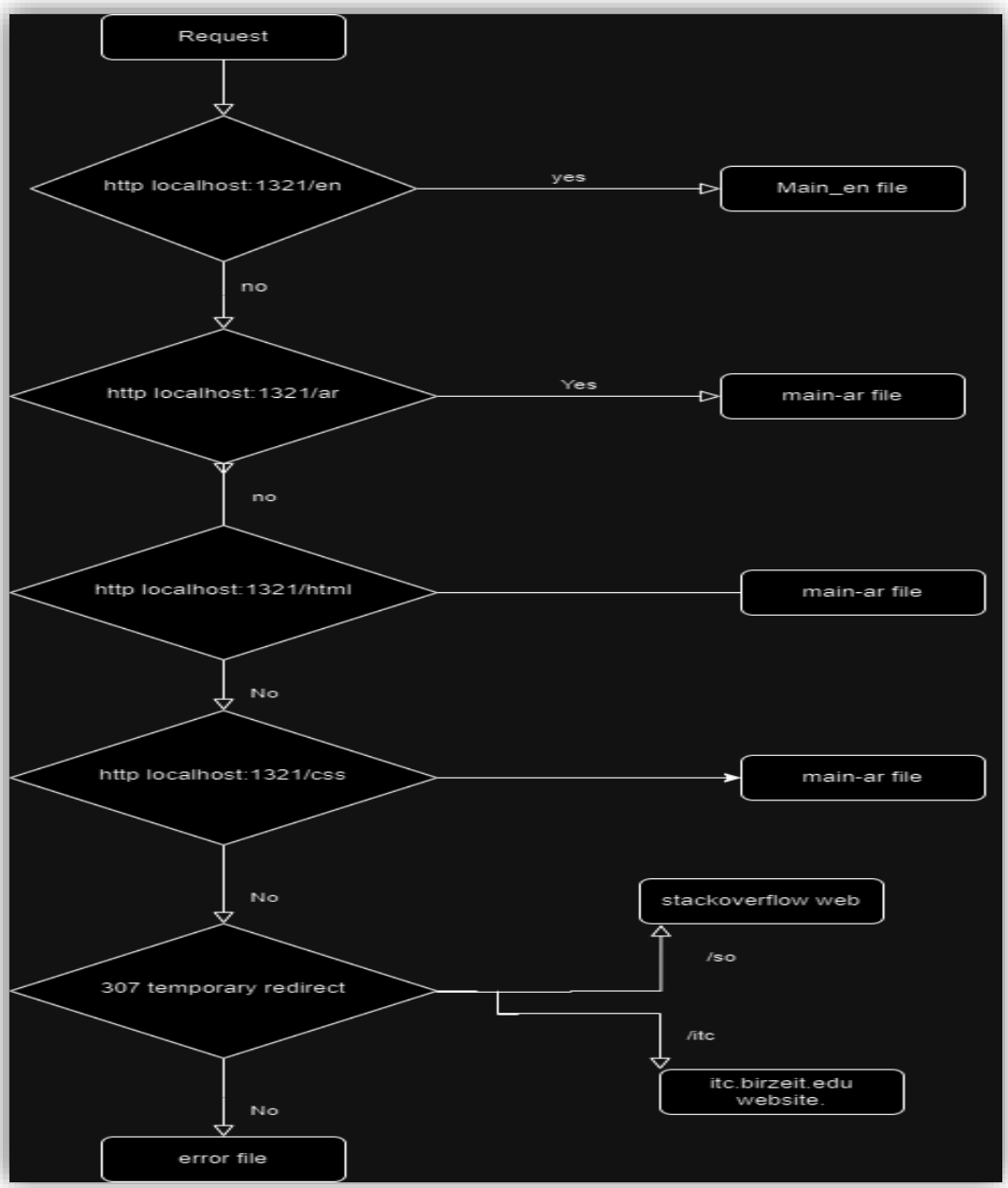
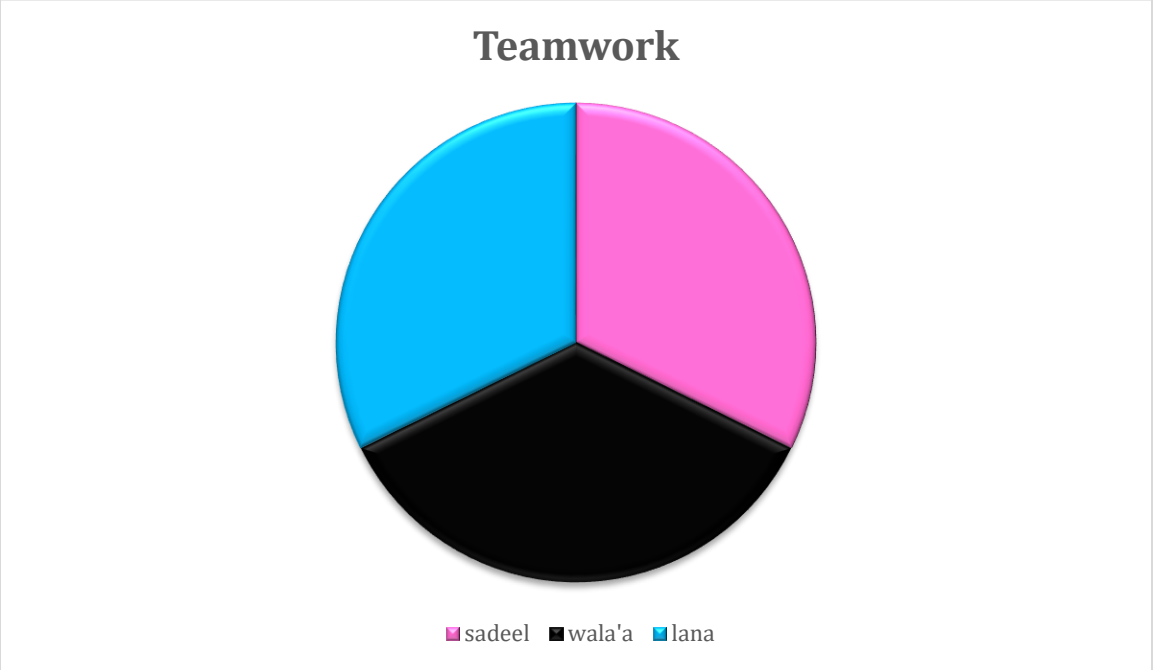


Figure 54:Flowchart diagram for the process

Teamwork:

All of us worked together on design, implementation, simulation, testing, and reporting, but we divided the task into the following:

Wala’a : 1.1 , 1.2.a ,1.2.b , 1.2.c , 2.1, 3.1 , 3.2 , 3.4 , 3.7
Sadeel : 1.2.f , 1.4 , 2.1 , 3.5 , 3.6 , 3.8 , 3.9 , 3.11
Lana : 1.2.d , 1.2.e , 1.3 , 2.2 , 3.10 , all flowcharts



References :

1) https://www.namecheap.com/support/knowledgebase/article.aspx/9667/2194/what-are-traceroute-ping-telnet-and-nslookup-commands/
2) https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/307
3) https://bgpview.io
4) https://app.diagrams.net