ENCS3320-COMPUTER NETWORKS

PROJECT#1

AHMAD TOFFAHA 1220476
ANAS AL SAYED 1221020
ABD AL-RHEEM YASEEN 1220783
Group #15 Sec :1

Instructor: Ibrahim Nemer

15/8/2024

BIRZEIT UNIVERSITY
لن يستطيعوا كسر إرادتنا، كلنا غزة
OUR WILL IS UNBREAKABLE... WE ARE ALL GAZA

Table of Contents

***Procedure &Theory*: 3**

Task 1: Commands & Wireshark 3

Task 2: Socket Programming (TCP and UDP) 3

Task 3: Web Server 4

Task 1: Commands & Wireshark 5

1) In your words, what are ping, tracert, nslookup, and telnet? 5

1-Ping:- 5

2-Tracert:- 6

3- Nslookup :- 7

4- Telnet :- 8

2) Make sure that your computer is connected to the internet and then run the following: 9

b. ping www.ox.ac.uk. 10

c. From the ping results, specify the location of the server from where you got the response? 11

d. tracert www.ox.ac.uk. 12

3) Give some details about autonomous system (AS) number 16

4) Use wireshark (free tool and available online) to capture some DNS messages. 18

Task 2: Socket Programming (TCP and UDP) 23

1)Using socket programming, write simple TCP client server. 23

2)Using socket programming, implement UDP client and server? 26

Task 3: Web Server 32

Project Screen Record: 32

Browser Screenshots: 33

From phone ScreenShote: 44

HTTP request: 48

Code Screenshots: 53

The Html file: 55

Table of Figure 62

References: 63

Teamwork bar: 64

PROCEDURE &THEORY :

This project is divided into three major tasks that help you understand networking concepts and socket programming by using TCP and UDP protocols. Below is an outline to approach each task:

Task 1: Commands & Wireshark

Theory:

The task involves analyzing the network associated with www.ox.ac.uk, including details about its Autonomous System (AS), IP ranges, and Tier 1 ISPs, using tools like BGPView.

Procedure:

- We ping a device on the same network (e.g., laptop to smartphone).
- We use ping www.ox.ac.uk and analyze the response to determine the server location.
- We take the previous ping result we specify the location of server using Ip look up tool.
- We use the tracert command on www.ox.ac.uk to trace the route to the server.
- We use nslookup to get the DNS information about www.ox.ac.uk.
- We attempt to connect using telnet and observe the response.
- We use an online tool like BGPView.io to find details about the autonomous system number, IP ranges, prefixes, and Tier 1 ISP associated with www.ox.ac.uk.
- We use Wireshark to capture DNS messages. Then we analyze the captured packets.

Task 2: Socket Programming (TCP and UDP)

Theory:

TCP (Transmission Control Protocol): is a connection-oriented protocol that ensures reliable communication by establishing a connection before data is sent and ensuring all data arrives in order and without error.
UDP (User Datagram Protocol): is a connectionless protocol that sends data without establishing a connection, making it faster but less reliable than TCP.

Procedure:

TCP Client-Server Application:

- We write a Python code to create a TCP server and client. The client sends a string to the server, which replaces all vowels with '#' and sends it back. The client then prints the modified string.
- We choose the port number based on a student ID (e.g., ID 1221020 → port 1020).

UDP Client-Server Application:

- We implement a UDP-based application where multiple peers (clients) can send and receive messages to/from a server. Each client message should include the client's ID and the message. The server should display communication history with each client.
- We choose the port number based on a student ID (e.g., ID 1220476 → port 476).

Task 3: Web Server

Theory:

This task talks about Web servers and HTTP: A web server handles HTTP requests from clients (browsers), serving HTML pages, images, CSS files, etc.., The status codes indicate the success or failure of these requests (e.g., 200 OK, 404 Not Found).

Procedure:

Simple Web Server Implementation:

- We write a Python script that creates a web server using sockets, listening on a specific port (e.g., ID 1220783 → 783).
- We handle various HTTP requests (/, /en, /ar, etc.) and respond with suitable HTML, CSS, and image files.
- We implement redirection using the HTTP 307 status code to redirect to StackOverFlow.
- Also, we handle errors and serve a custom 404 error page.
- Print HTTP requests to the terminal.

HTML and CSS Design:

- We create an index.html or main_en.html with the specified content and design using CSS.
- We create an Arabic version of the page (main_ar.html).
- We include links to other HTML files, external websites, and images like our pictures and a link to visit Birzeit website.

TASK 1: COMMANDS & WIRESHARK

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

Answer:

1-PING:-

Ping command, you can check the connectivity to devices within your local network and to remote servers on the internet. The IP address in the ping response can help you determine the physical location of the server, using online IP lookup tools.

Example use:

Checking if a website is online.
determine the physical location of the server.

Simple command:
ping www.google.com

Screenshots:

```
Microsoft Windows [Version 10.0.22621.3880]
(c) Microsoft Corporation. All rights reserved.

C:\Users\HP>ping www.google.com

Pinging www.google.com [172.217.22.100] with 32 bytes of data:
Reply from 172.217.22.100: bytes=32 time=21ms TTL=117
Reply from 172.217.22.100: bytes=32 time=22ms TTL=117
Reply from 172.217.22.100: bytes=32 time=9ms TTL=117
Reply from 172.217.22.100: bytes=32 time=11ms TTL=117

Ping statistics for 172.217.22.100:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 9ms, Maximum = 22ms, Average = 15ms
```

Figure 1 use ping www.google.com

2-TRACERT:-

Tracert (trace route): A command that can be used to see the exact path that the data packet is taking on its way to the destination.

Example use:
Finding where delays are happening in a network.

Simple command:
tracert www.google.com

Screenshots:

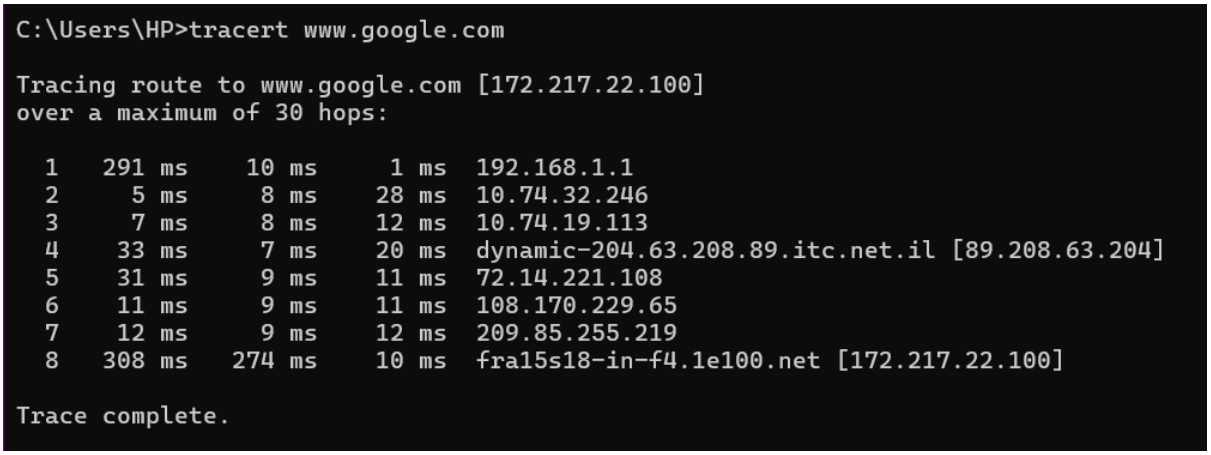


Figure 2 use tracert www.google.com

3- NSLOOKUP :-

Nslookup (name server lookup): A command that can be used to fetch the DNS records for a given domain name or IP address.

Example use:
This command will return the IP address associated with "www.google.com".

Simple command:
nslookup www.example.com

- **Screenshots:**

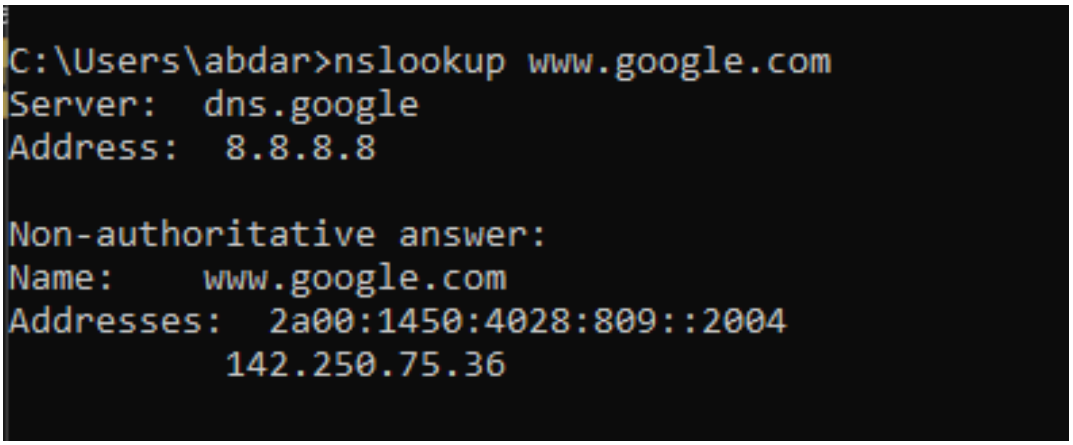


Figure 3 use nslookup www.example.com

4- TELNET :-

Telnet is a network protocol that lets you access and control a remote computer over the internet using a text-based interface. Unlike HTTP or FTP, which are for transferring files, Telnet allows you to log in and use the remote computer as if you were there.

Example use:

This command will return the IP address associated with "www.google.com".

Simple command:

telnet the.libraryat.whatis.edu

- **Screenshots:**

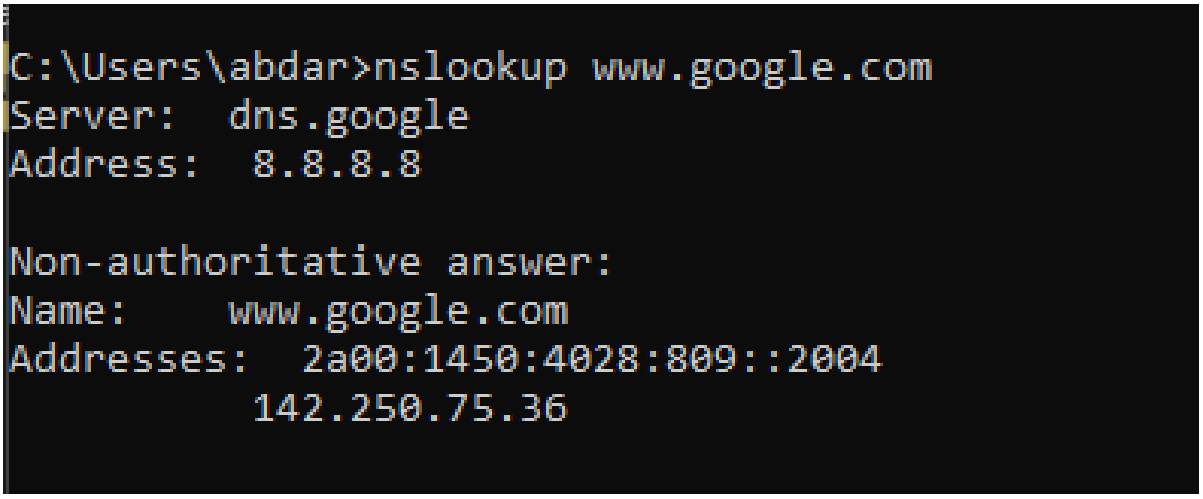


Figure 4 telnet the.libraryat.whatis.edu

2) Make sure that your computer is connected to the internet and then run the following:

a. ping a device in the same network, e.g. from a laptop to a smartphone.

Answer:

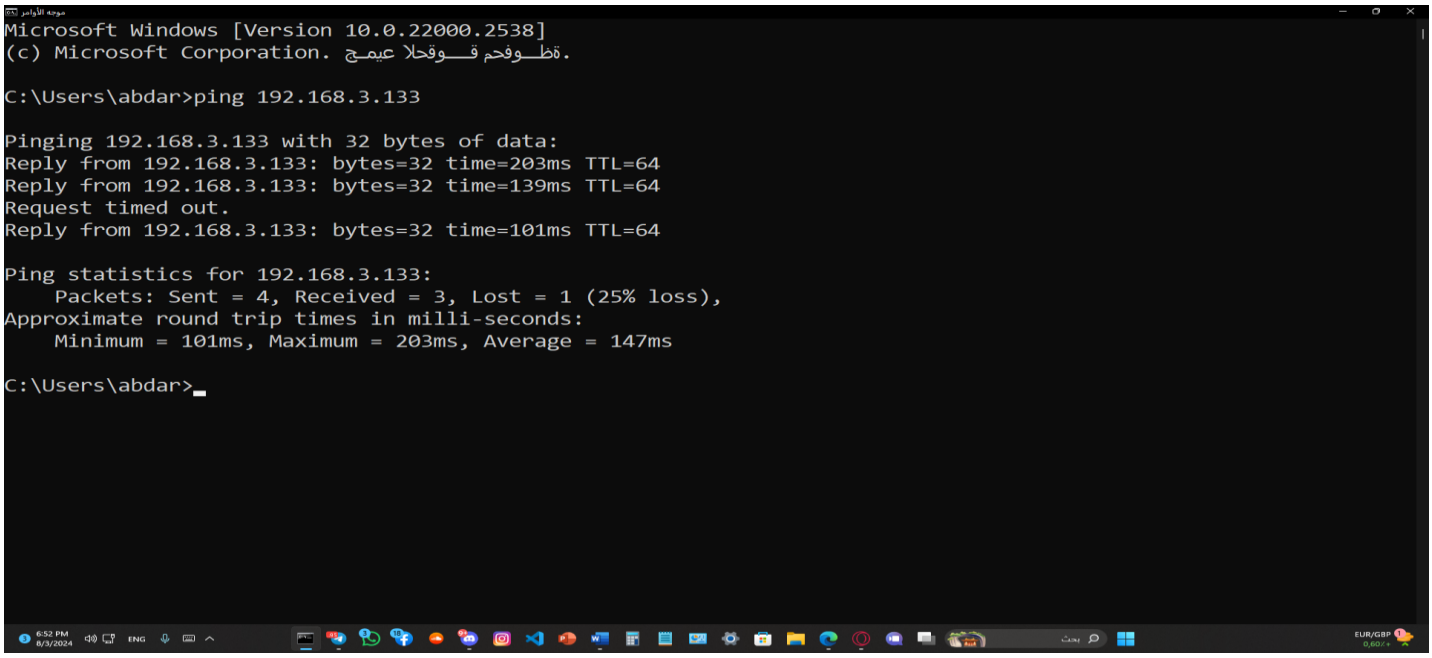


Figure 5 shows ping IP address 192.168.3.133.

Consider figure 5 that shows ping IP address 192.168.3.133.

By default 4 ping packets are sent to the destination IP address we chose, then the destination will send the data packets back to us as a reply If we received a reply then that means there is a network connectivity between us and the destination, but if we didn't get a reply, then that means that there is no network connectivity between us, and here the ping is successful because you can see a reply from the destination.

Note: if we pinged a host and we got a message that says "request timed out", then that could mean that the host is down or it's blocking all the ping requests.

b. ping www.ox.ac.uk.

Answer:

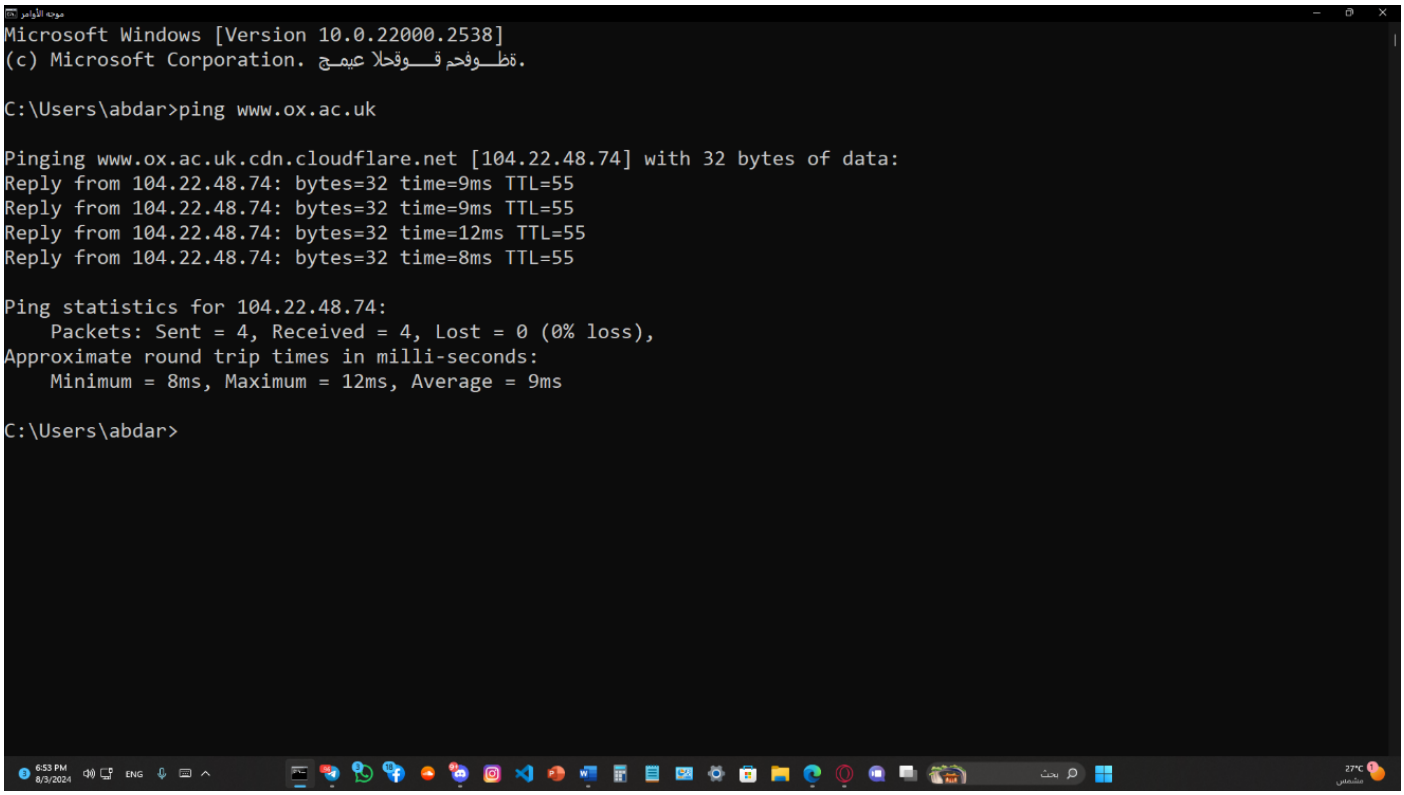


Figure 6 the domain name " www.ox.ac.uk.".

Consider figure 6 that shows a ping the domain name " www.ox.ac.uk.".
"www.ox.ac.uk" is just an alias; as we can see when we run this command "ping www.ox.ac.uk" a message has been appeared "Pinging www.ox.ac.uk.cdn.cloudflare.net " and not "www.ox.ac.uk", that is because "www.ox.ac.uk" is just an alias of "www.ox.ac.uk.cdn.cloudflare.net ".
104.22.48.74 is the IP address of " www.ox.ac.uk.cdn.cloudflare.net ".
You can see a reply from the destination and that indicates that the name resolution by DNS is working fine.

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

Answer:

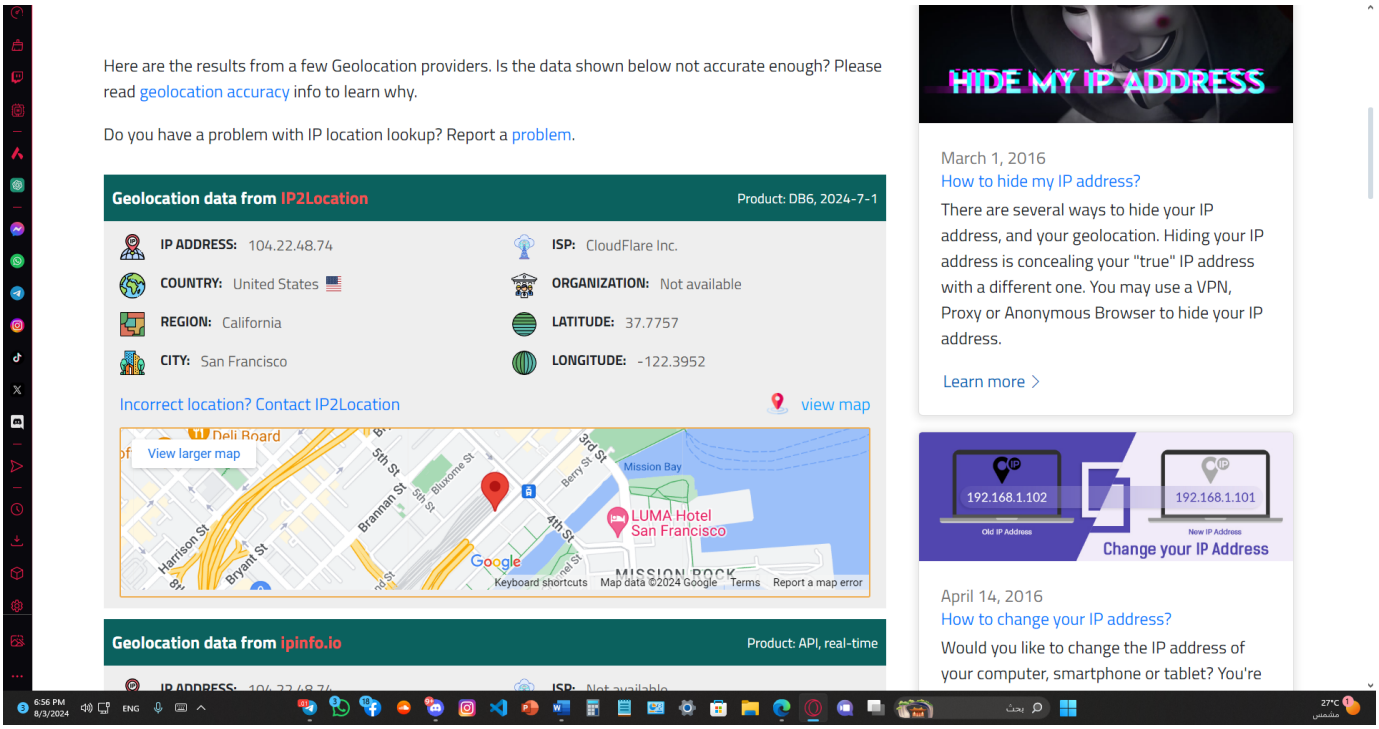


Figure 7 location of the Ip address using Ip geolocation

According to previous result, The IP address 104.22.48.74 is registered to the University of Oxford we get an Ip address 104.22.48.74 , Using any IP geolocation website such as Ip location Lockup, we get Country of United States , California, San Fransisco.

d. tracert www.ox.ac.uk.

Answer:

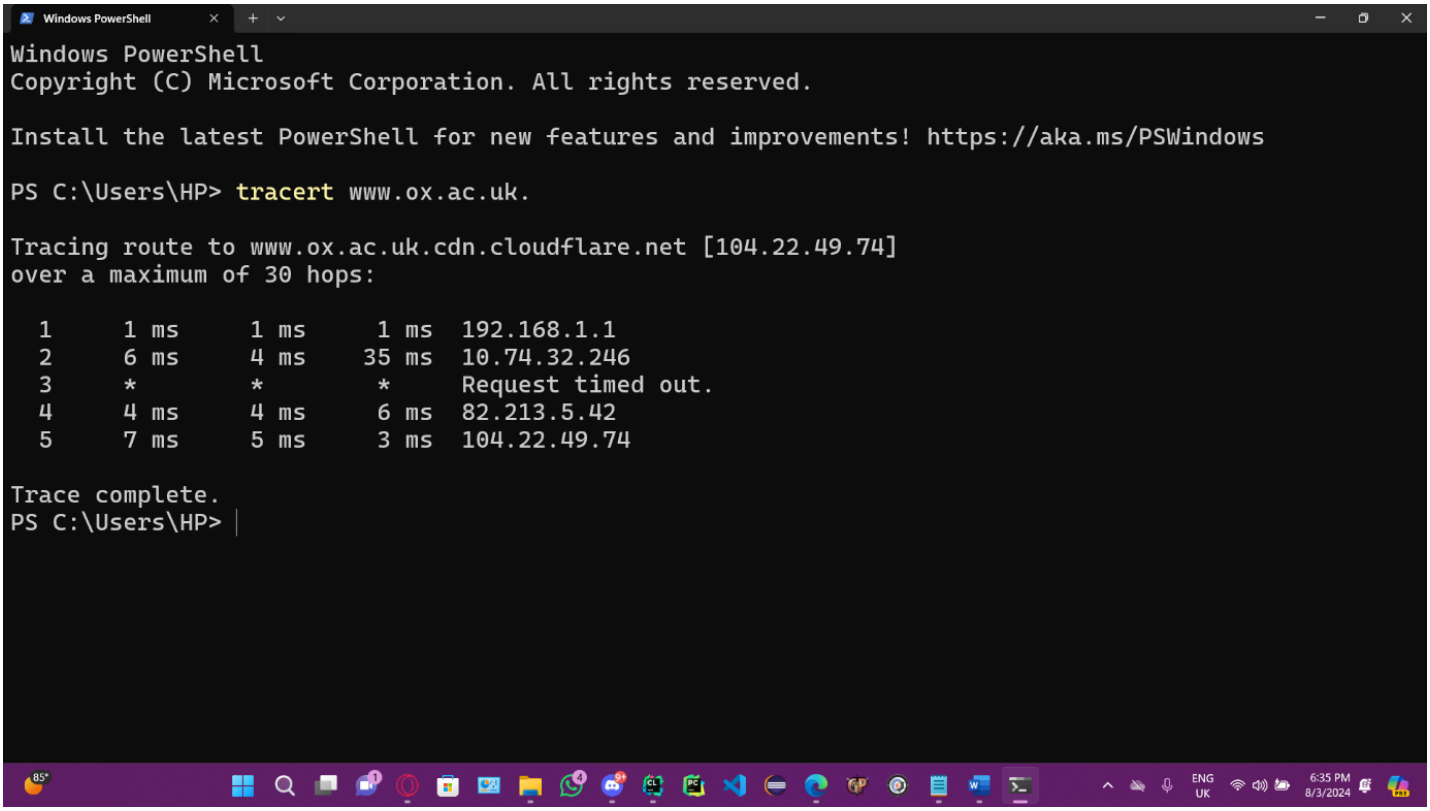


Figure 8 shows a tracing route to "www.ox.ac.uk".

Consider figure 8 that shows a tracing route to "www.ox.ac.uk".
The data packets will find its way to the destination www.ox.ac.uk. and each time it reach a router on its path it will report back information about that router (The IP address and the time it took between each hop).

The message "Over a maximum of 30 hops" : if you hit more than 30 routers to reach the destination the tracert command will stops at the 30 hop, so the maximum amount of hops you can trace by default is set to be 30 .

The first column from the left represents the number of the hop along the route the final router (or hop) number is 5 so it took 5 routers (5 hops) for my device to reach the destination "www.ox.ac.uk".

The last column represents the IP address of the hop.

e. nslookup www.ox.ac.uk.
Answer:

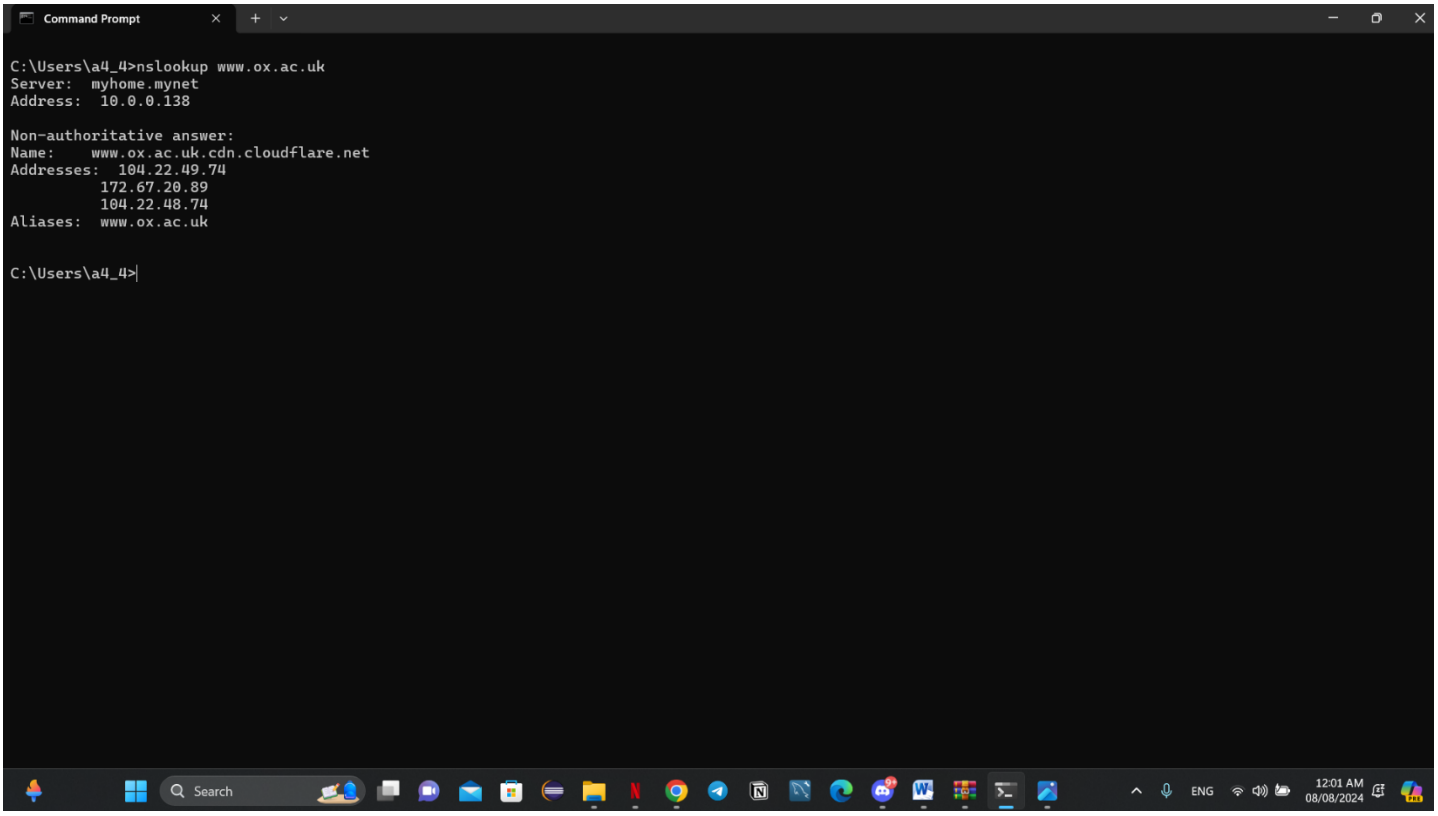


Figure 9 talking to is " myhome.mynet".

Consider Figure 9

The server that I'm talking to is " myhome.mynet".

Note: DNS resolves domain names to IP addresses

.

Nslookup command fires a DNS query which is here "myhome.mynet" to resolve " www.ox.ac.uk.", then sends ICMP (Internet Control Message Protocol) requests to the IP address obtained from the DNS query.

On successful response, it then queries DNS for the domain of the source of the ICMP reply.

Here, the name would differ from the name we are looking for because the name we are looking for "www.ox.ac.uk" is just an alias as we can see when running nslookup "www.ox.ac.uk".

f. telnet www.ox.ac.uk.

Answer:

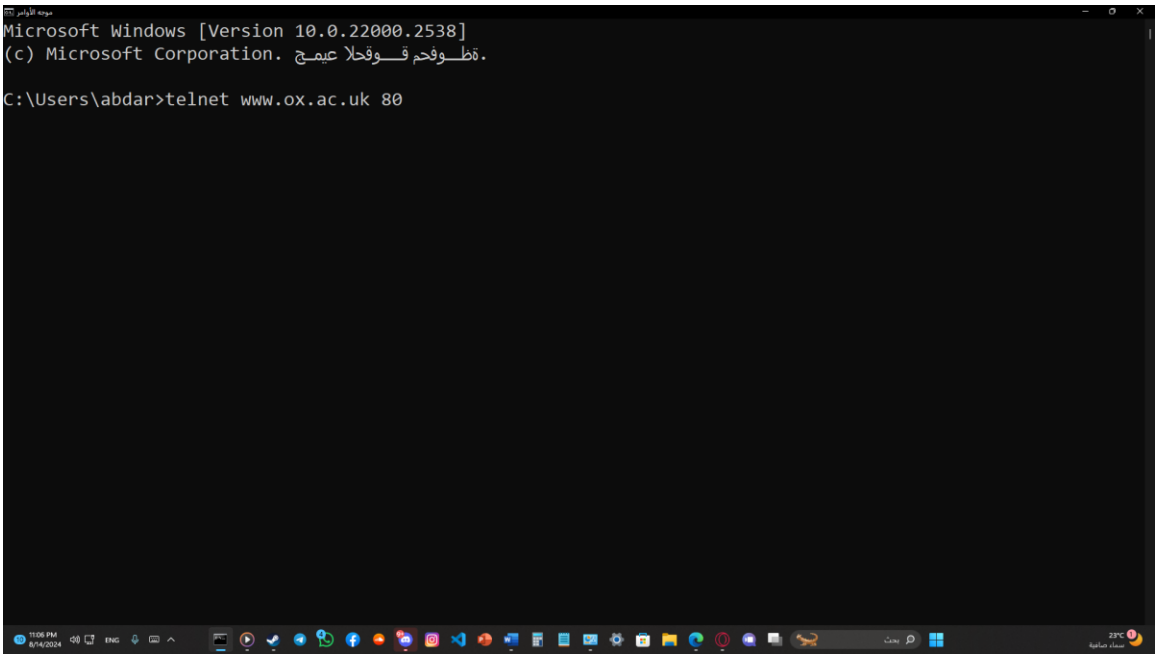


Figure 10: Execute the Telnet Command in CMD with port#

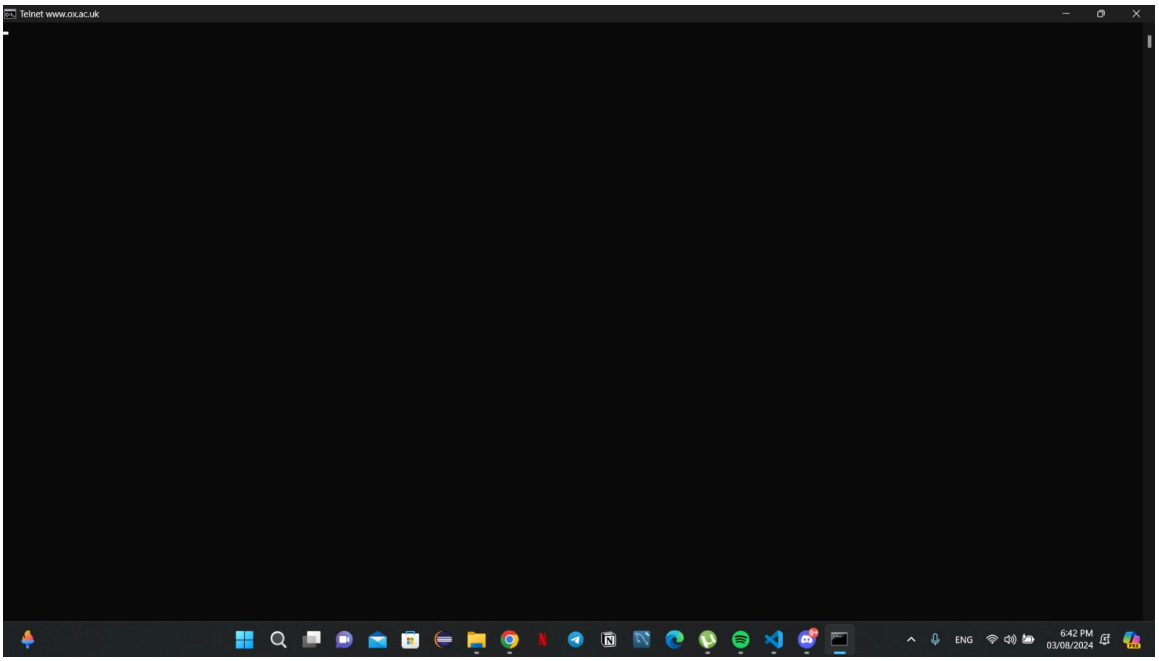


Figure 11 Waiting for response of connection

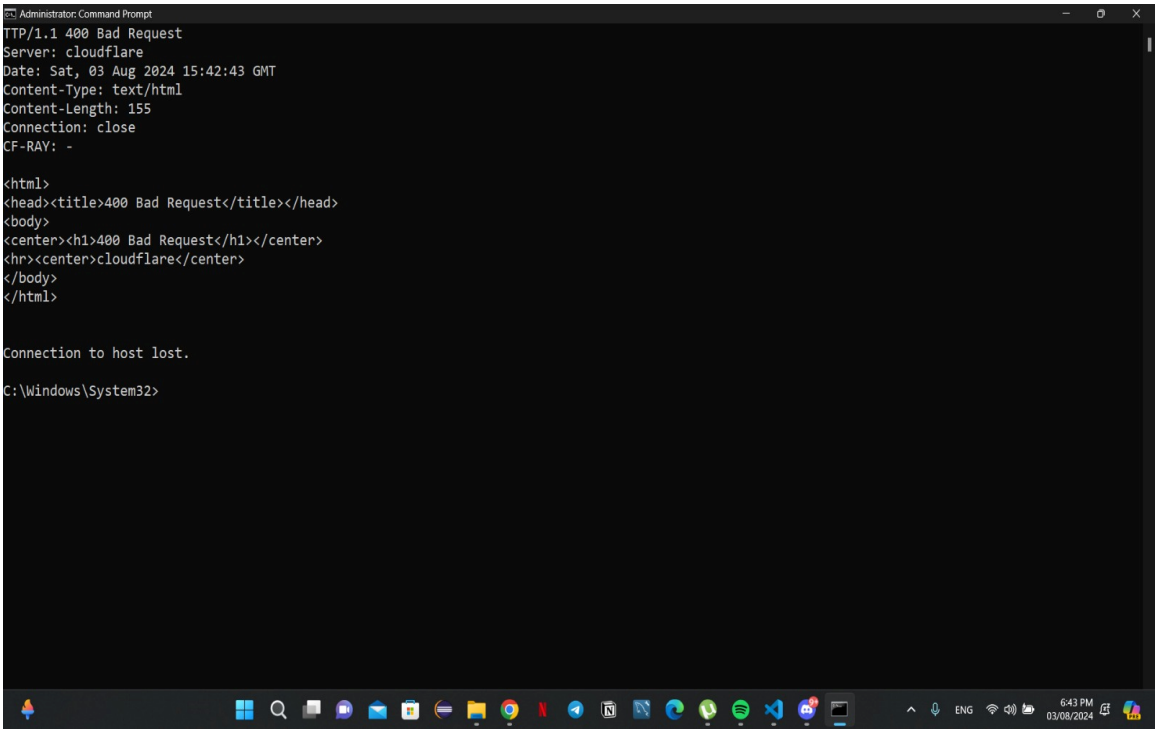


Figure 12 Bad request from the server

We open the CMD then execute the command of telnet using port number of 80 to call HTTP request, after that the CMD window will clear, and if the connection is successful, we'll see a blank screen or a welcome message from the remote server, but in our case, we got a bad request message from the server, this typically means that the server did not understand the request made by the Telnet client. Telnet is more suited for interacting with services like mail servers, routers, or other devices using plain text commands, not for web servers that require HTTP requests.

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

Answer:

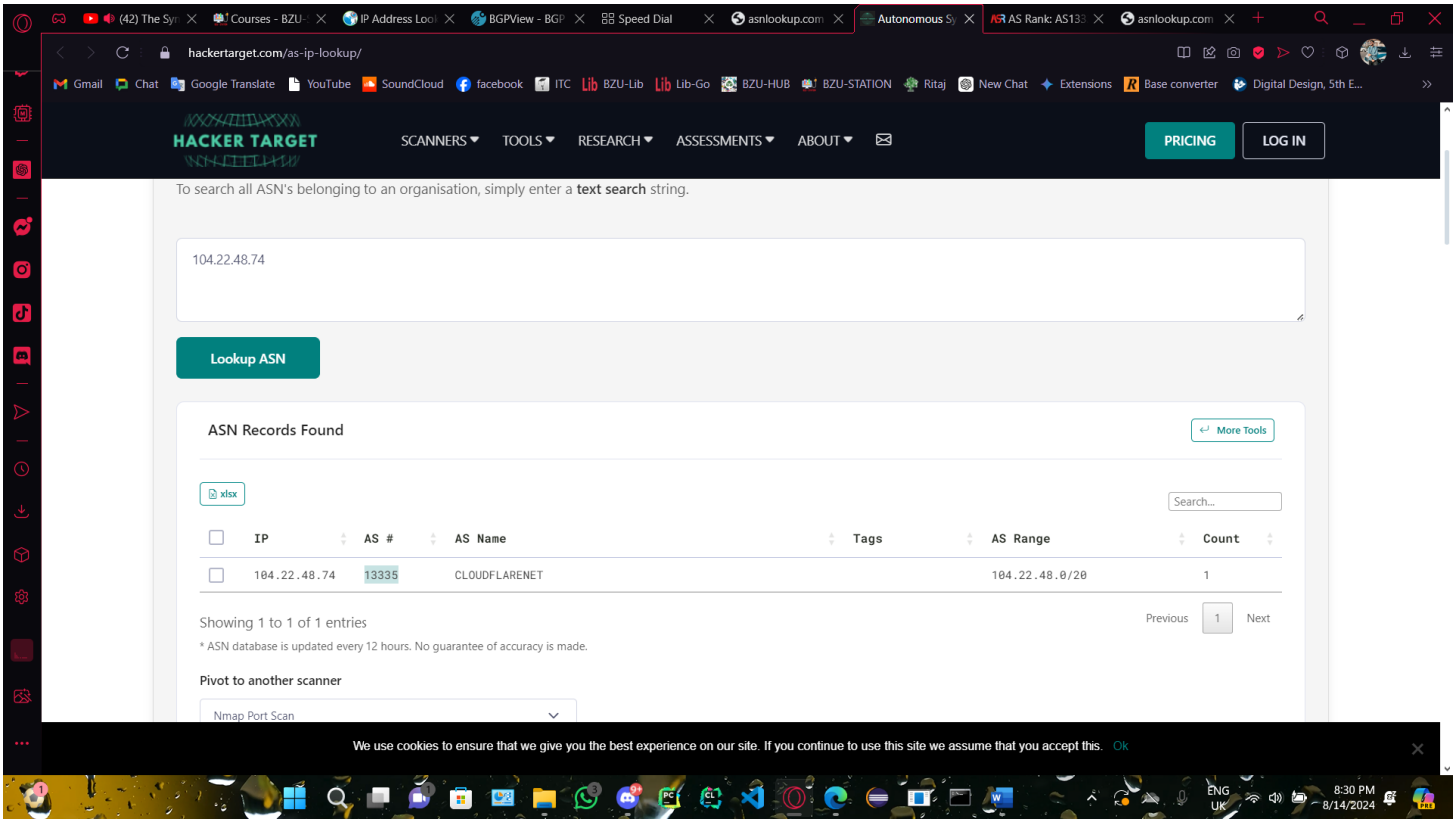


Figure 13: Lookup ASN to get the AS number

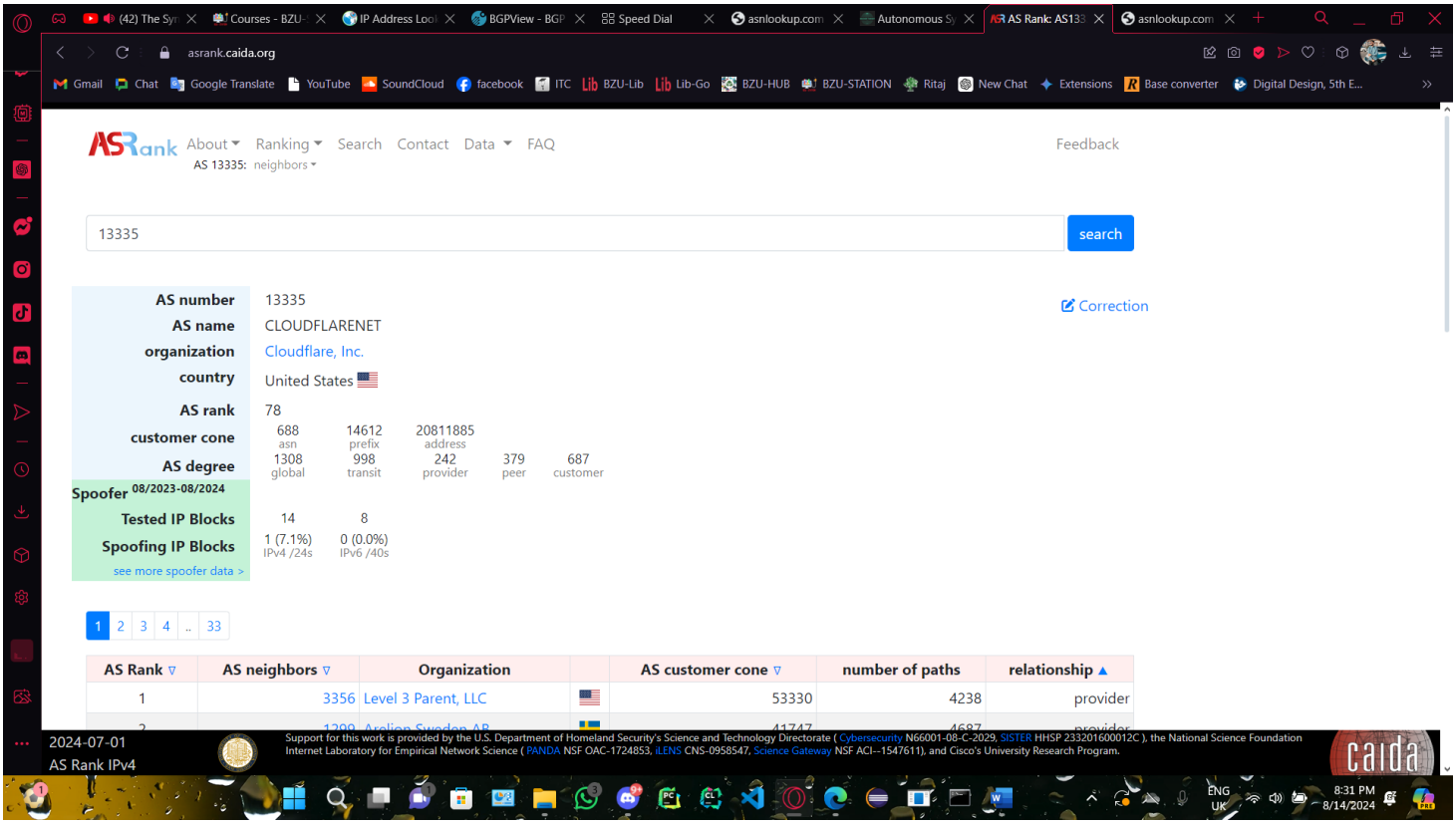


Figure 14: use AS Rank to get info

We bring the Ip of **www.ox.ac.uk** using telnet in the previous task an use it on the **(www.hackertarget.com)** to get the **AS Number**, and after that we use **www.asrank.com** to get the other information.

The University of Oxford's domain, **www.ox.ac.uk**, is associated with the Autonomous System (AS) number **13335**. This AS is responsible for the university's network, managing a variety of IP prefixes.

Details:

- **AS Number**: 13335
- **Number of IPs**: The AS covers a range of IP addresses, though the exact count may vary based on subnet allocations.
- **Prefixes**: Cloudflare's AS 13335 includes numerous prefixes, with some of the notable ones being:
  198.41.240.0/23
  172.68.192.0/18
  141.101.82.0/23
  8.17.205.0/24
  8.34.202.0/24
  103.22.200.0/22
- **Peers**: Cloudflare peers with numerous networks worldwide, including major internet exchange points (IXPs) such as BBIX, BCIX, BNIX, and many others across different regions like the United States, Europe, and Asia.

  Cloudflare's extensive peering strategy helps them maintain a strong and resilient network, improving the performance and reliability of their services. They operate with multiple connectivity options, ranging from 10G to 200G, depending on the specific peer location.

- **Tier 1 ISP:** CloudFlare Inc.

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

Answer:

1. Start Wireshark and Select Interface:

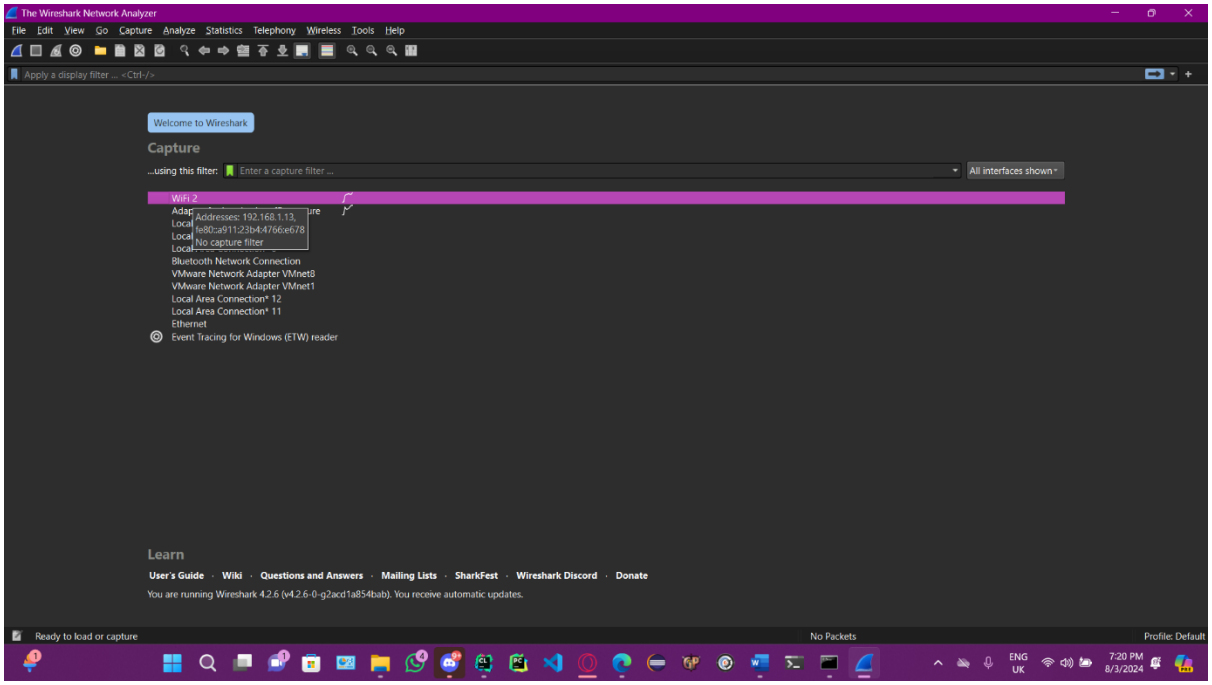


Figure 15 Home Page of wireshark

Choose the network interface you want to capture traffic on. This is usually the one connected to the internet (e.g., Ethernet, Wi-Fi).

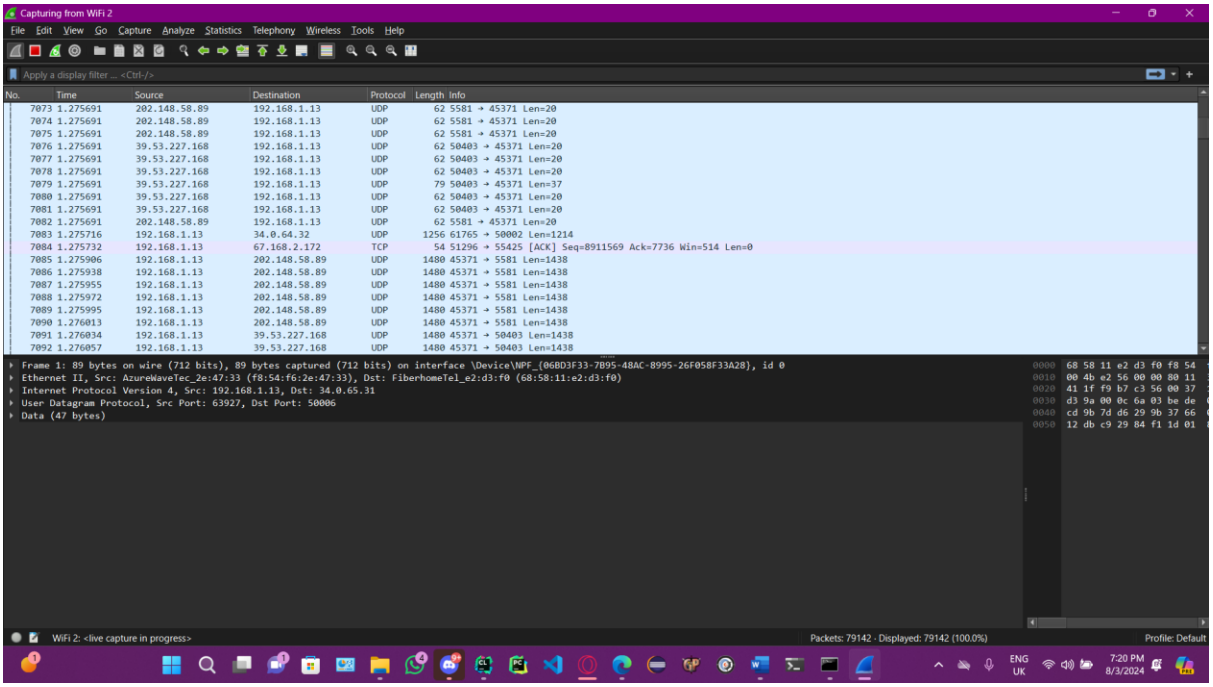


Figure 16. Start Capturing Traffic

Start capturing by clicking the blue shark fin icon on the toolbar to start capturing packet

2.Apply DNS Filter:

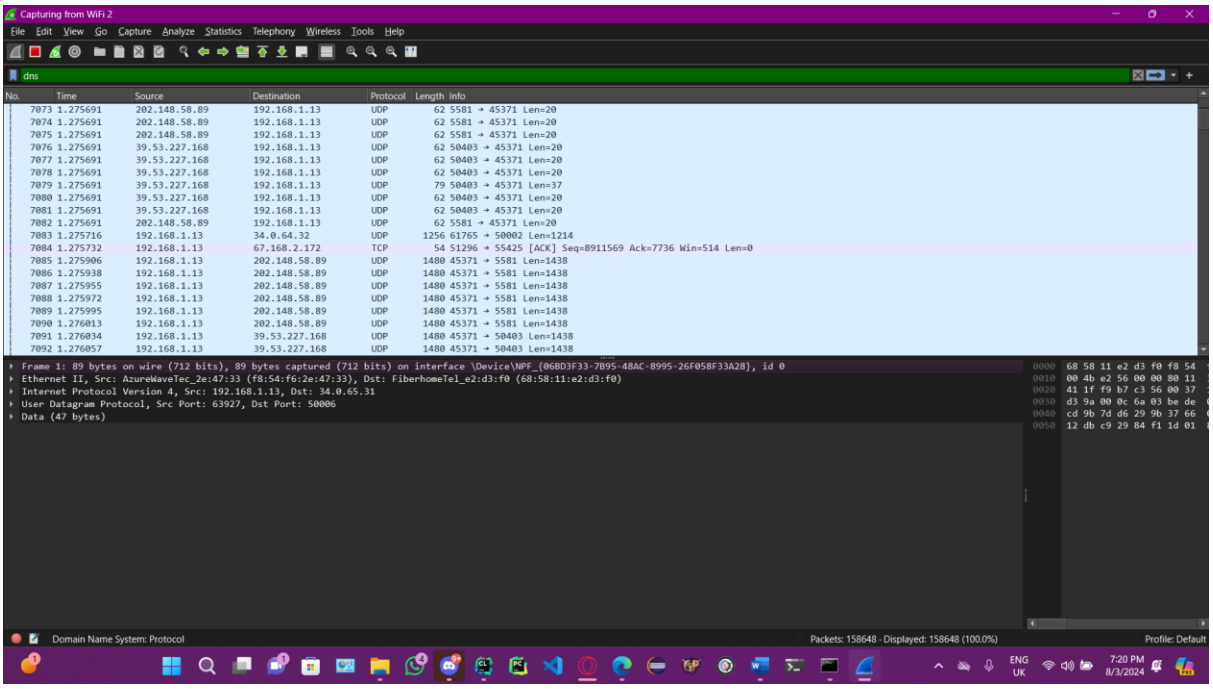


Figure 17 Before applying a Filter

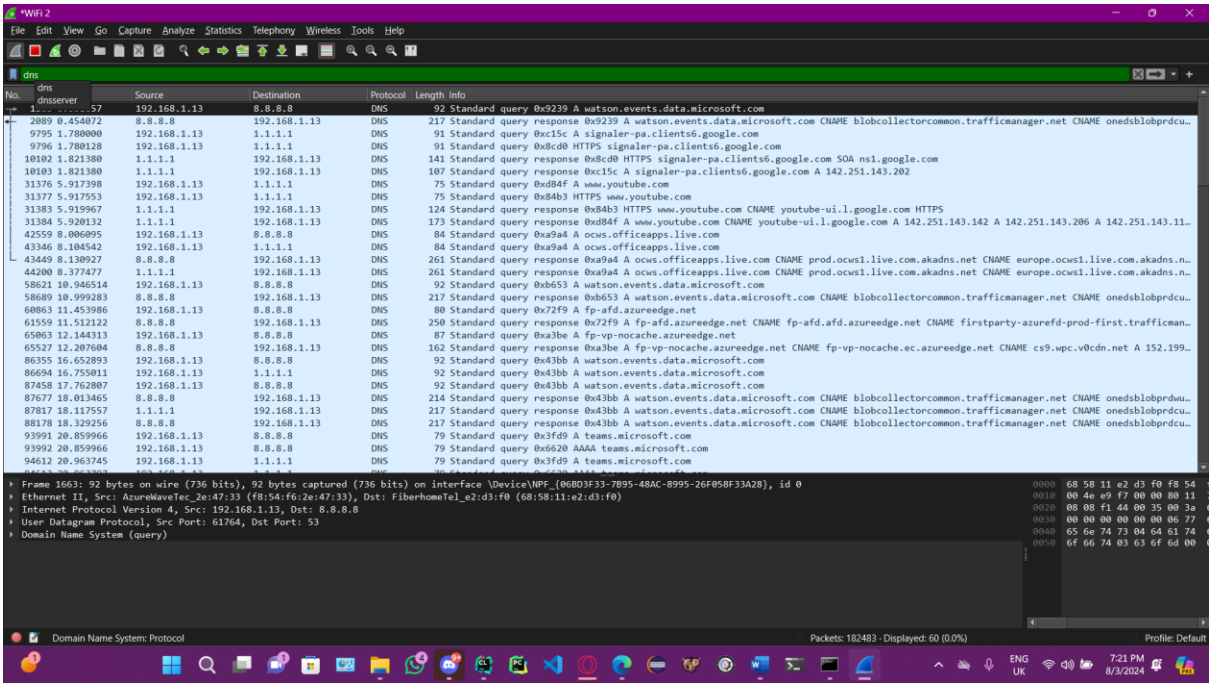


Figure 18 After applying a Filter

In the filter bar at the top, type “dns” and press Enter. This will filter out all traffic except DNS messages.

3. *Generate DNS Traffic:*

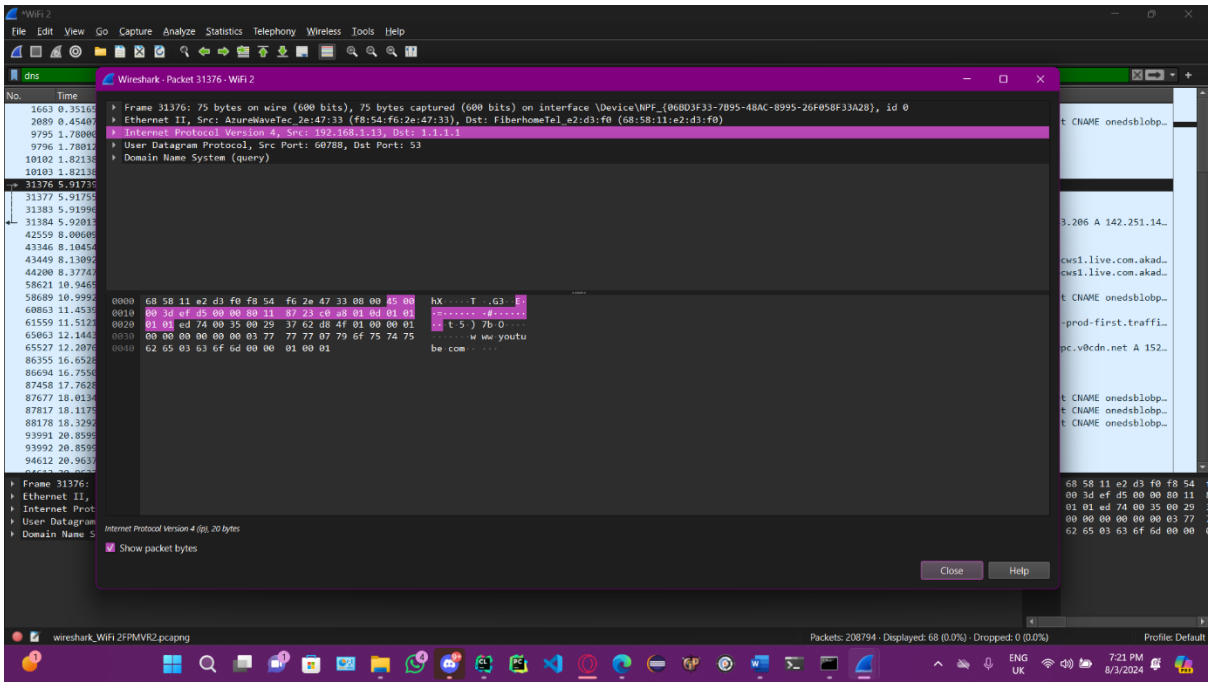


Figure 19 Stop Capturing

Click the red square icon on the toolbar to stop capturing packets once you have enough data

4.Analyze DNS Packets:

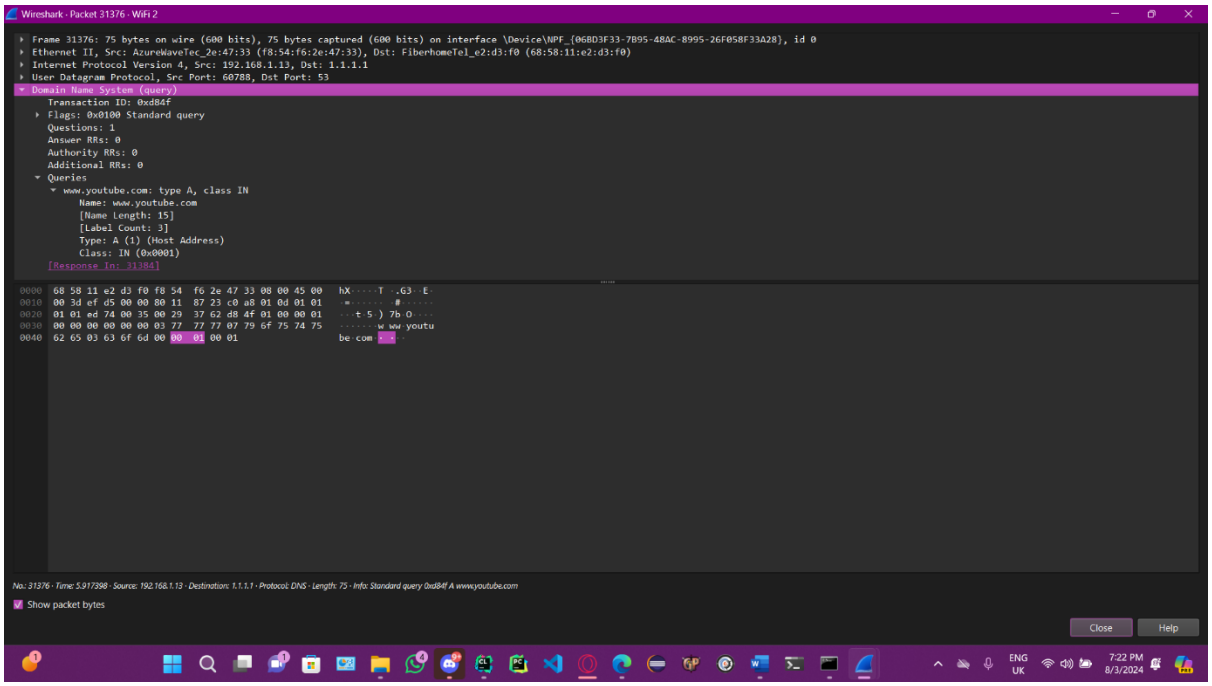


Figure 20 Inspect DNS Packets

Look through the captured packets in the main Wireshark window, DNS packets will be labeled with the protocol "DNS", Click on a DNS packet to see detailed information in the lower pane, including the query and response sections.

5.DNS Message Analysis:

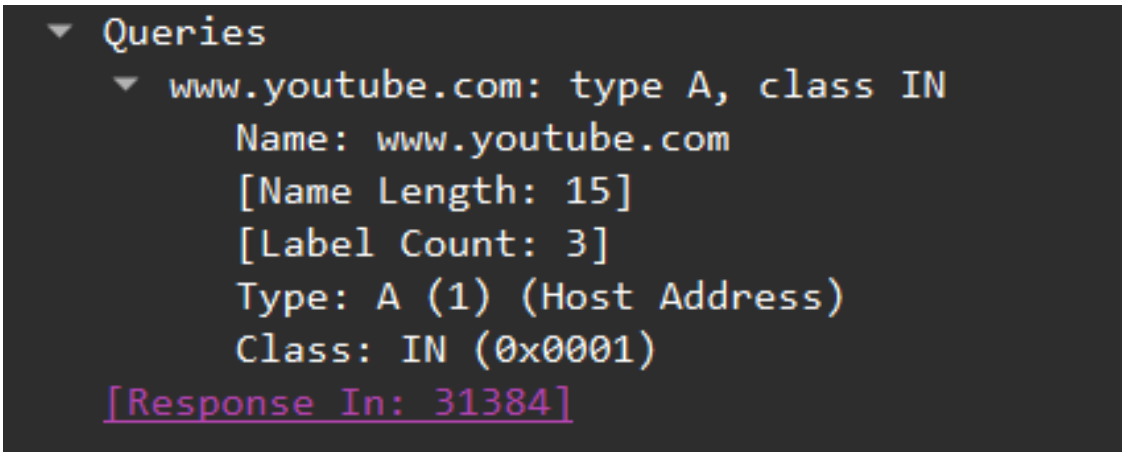


Figure 21 DNS Message Analysis

DNS Message contains:

1- Query: Expand the details in the lower pane to see the query name (in our example we have www.youtube.com).
2- Response: Expand the details to see the answer section, including the IP address resolved from the DNS query.

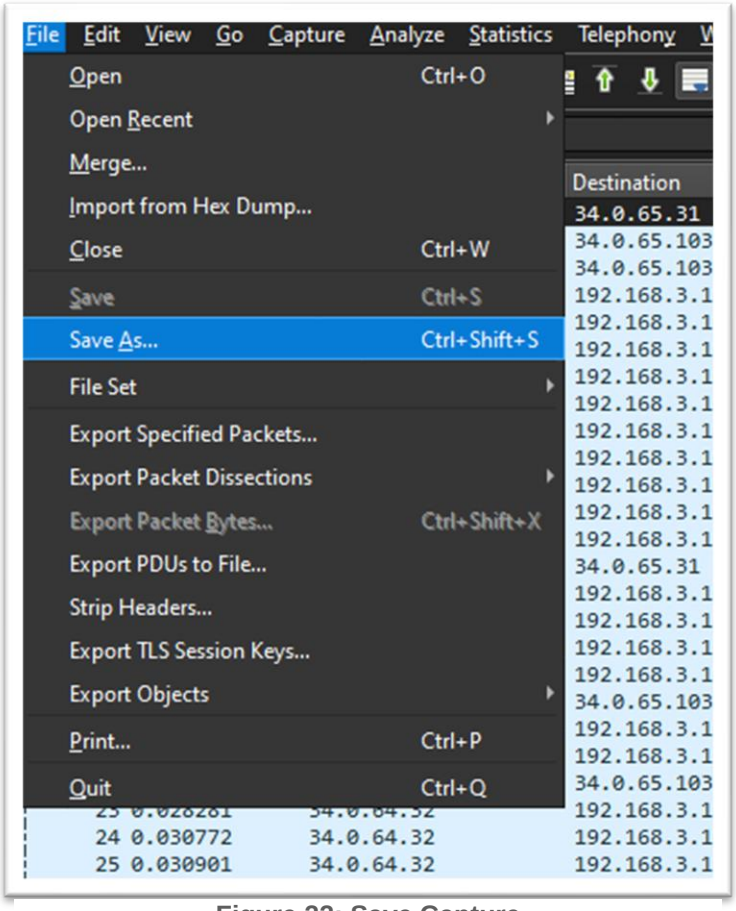


Figure 22: Save Capture

6.Saving the Capture:

Choose a location and save the capture file for later analysis.

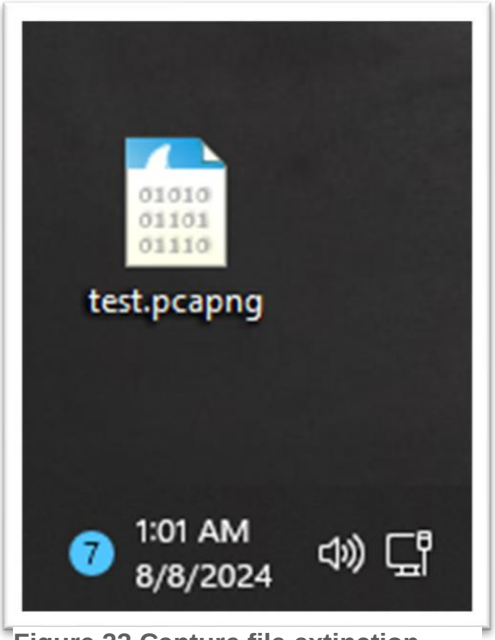


Figure 23 Capture file extinction

TASK 2: SOCKET PROGRAMMING (TCP AND UDP)

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.

Answer:

To run the programs on the same computer, start the server first and then start the client. The server will output **"The server is ready to receive"** and the client will output a message **"Input sentence"**.
For TCP we use a "AF_INET" and "SOCK_STREAM" for the socket creation. The main difference between TCP and UDP is that with UDP there is no connection between the client and server, and messages can be lost or delivered out of order. Therefore, you will need to add code to handle these cases if you want to use UDP.

Code Screenshots: ➔

The TCP server:

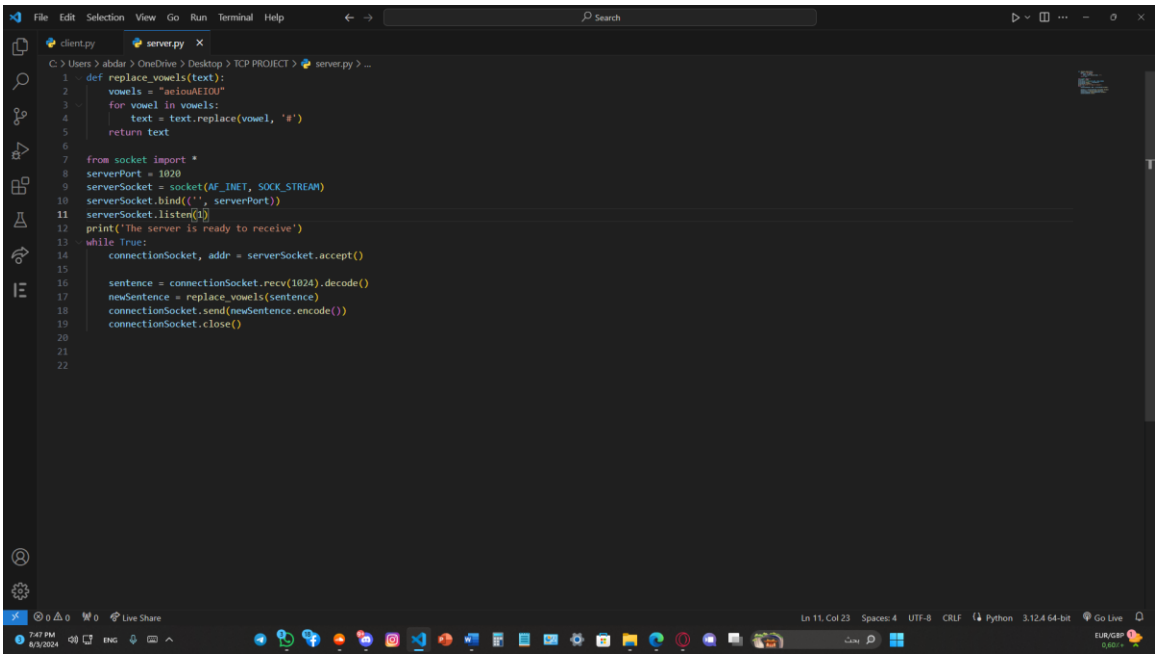


Figure 24 Code of TCP Server

The server code sets up a TCP server that listens for connections on a specified port (1020). When a client connects, the server receives a string from the client. It then processes this string by replacing all vowels with the # character using the replace_vowels function. After processing, the server sends the modified string back to the client. Finally, the server closes the connection, waiting for another client if needed.

Replace_vowels Function:

The replace_vowels function in the server code takes a string as input and replaces all vowels (both uppercase and lowercase) with the # character. It defines a string containing all vowels ("aeiouAEIOU") and iterates over each vowel in this string. For each vowel, it uses the replace() method to substitute the vowel with # throughout the input string. The function returns the modified string with all vowels replaced.

➔ *The TCP client:*

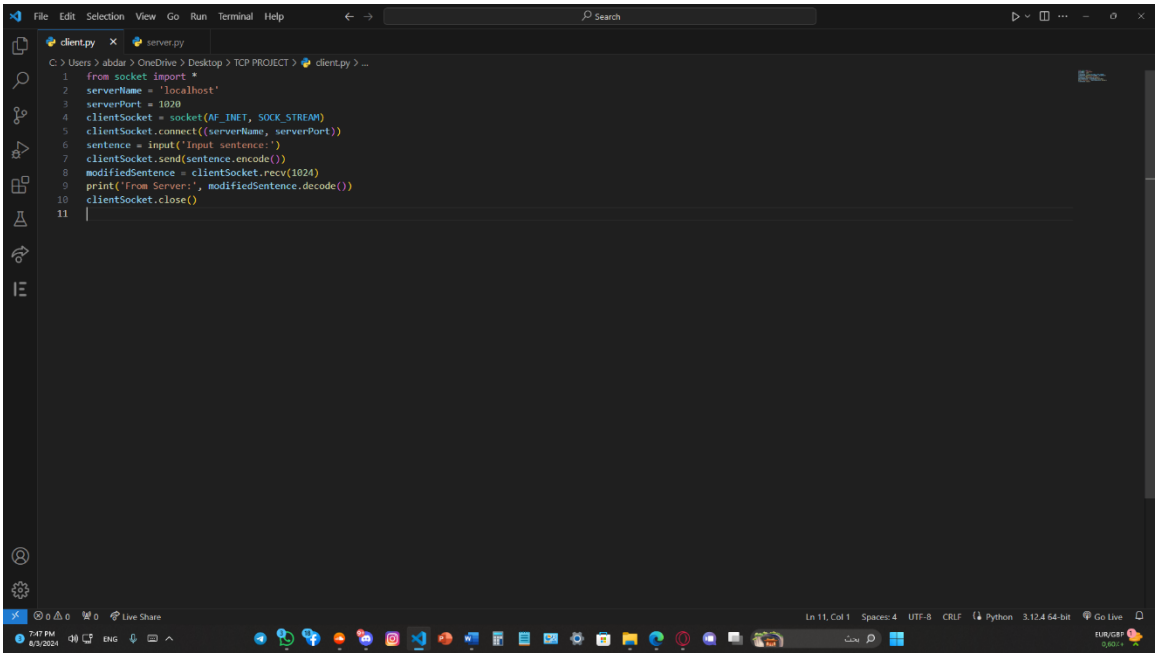


Figure 25 Code of TCP Client

The client code creates a TCP socket to connect to a server on localhost using port 1020. It prompts the user to enter a string, which is then sent to the server. The client waits for the server to process the string (replacing vowels with #) and then receives the modified string back. Finally, the client prints the modified string and closes the connection.

Runs output Screenshots:

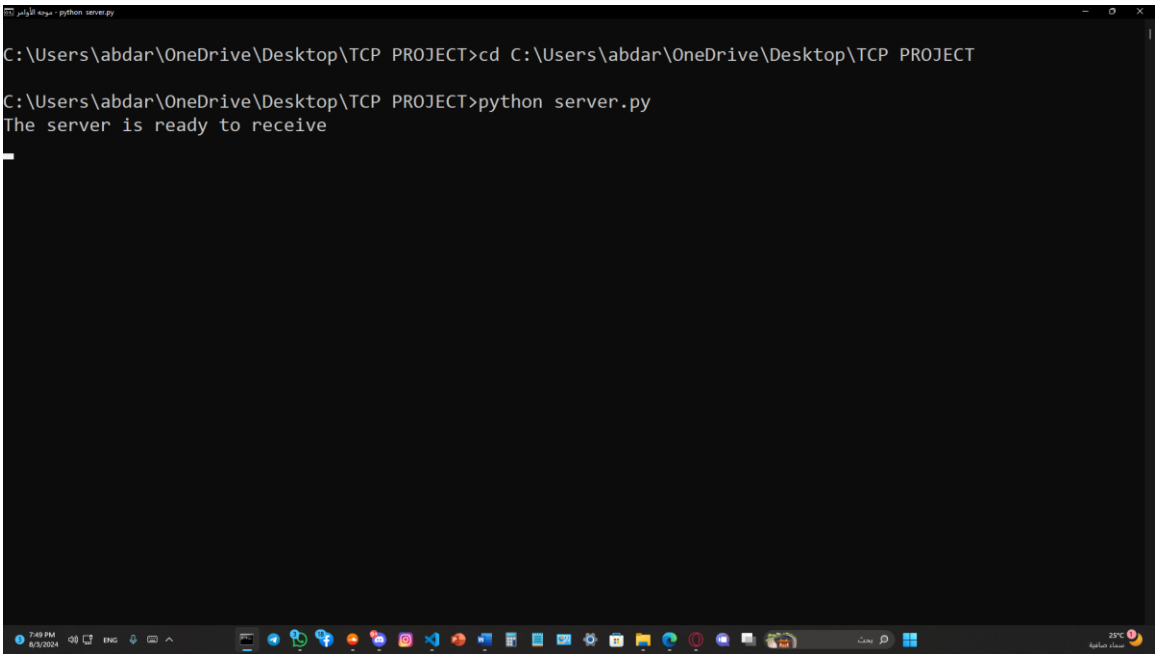


Figure 26 Run the server python code form CMD

To run the Server, you must run the CMD for your device, then you must enter the file containing the python code of server by using cd command, after that we use python server.py to run the server.
As you can see after running the server we got a message from server that it ready to receive.

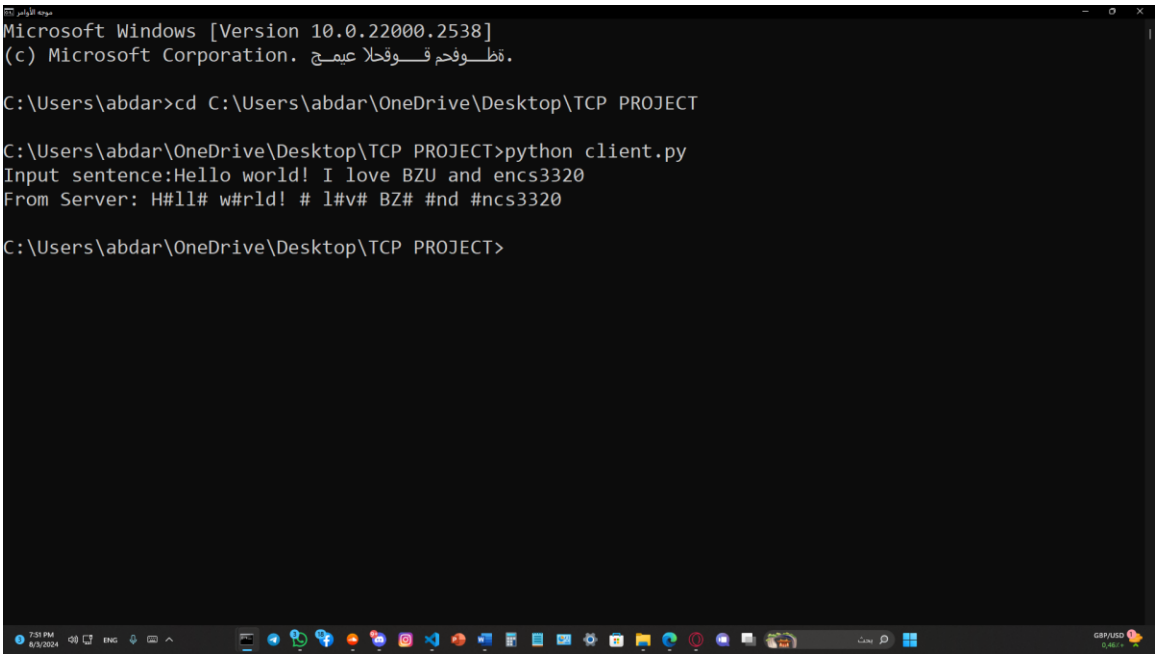


Figure 27 Run client python code form CMD

To run the Client, you must run the CMD for your device, then you must enter the file containing the python code of Client by using cd command, after that we use python client.py to run the client code.
As you can see after running it we got a message to insert to a sentence to send it to the server.

Explanation of input and output examples:

We insert "Hello world! I love BZU and encs3320" sentence and should be sent to the server
Then all vowels (both lowercase and uppercase) should be replaced with #. Vowels are: a, e, I , o, u and their uppercase counterparts A, E, I, O, U.
After replacing vowels, the sentence will be: "H#ll# w#rld! # l#v# B# #nd #ncs3320" .

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. You can have this style of printing for server in the communication between peer (server) and peer (client 1)?

Answer:

In this question we show how to use UDP for communication between multiple clients and a server. Each peer **(client and server)** can send and receive messages, and the server maintains a history of the last messages received from each client.

To run the programs on the same computer, start the server first and then start the client.

1. **Server**: The server listens on a port number (746). It can receive messages from multiple clients and respond to each client. It also keeps track of the last message received from each client and prints them in order.
2. **Client**: Each client sends a message to the server and can receive a response from the server. Each client only displays its communication with the server.

Code Screenshots:

➔ *The UDP client:*

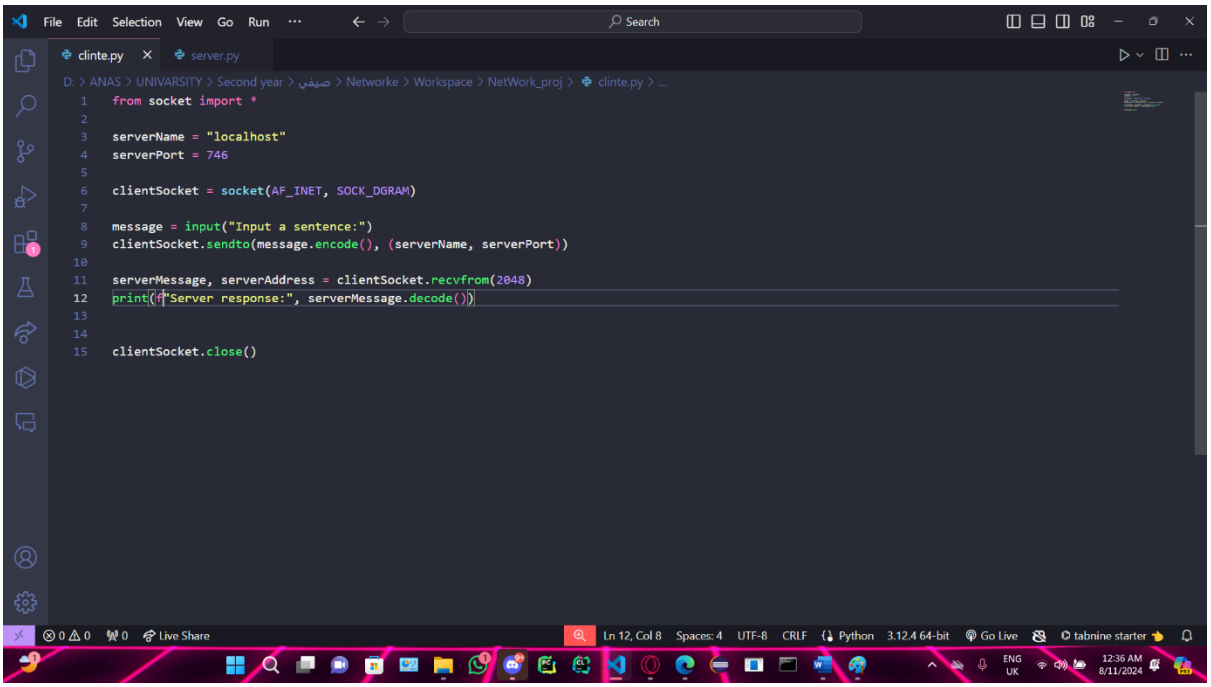


Figure 28 UDP client code

➔ *The UDP server:*

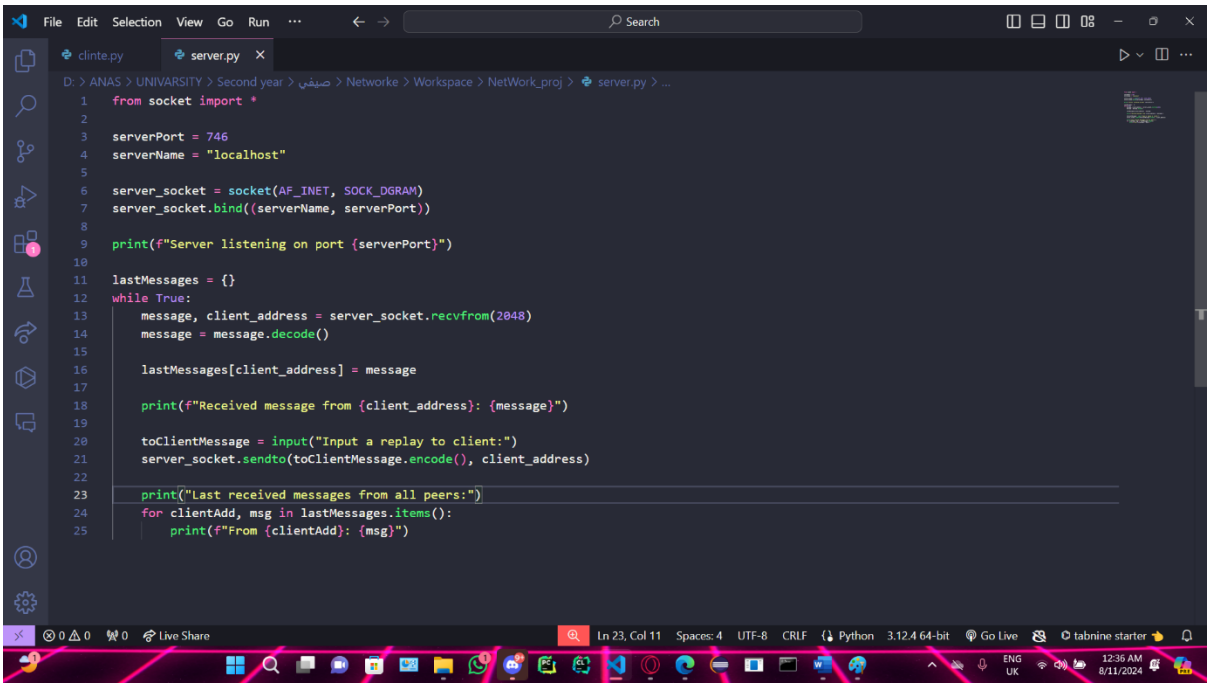


Figure 29 UDP server code

Runs output Screenshots:

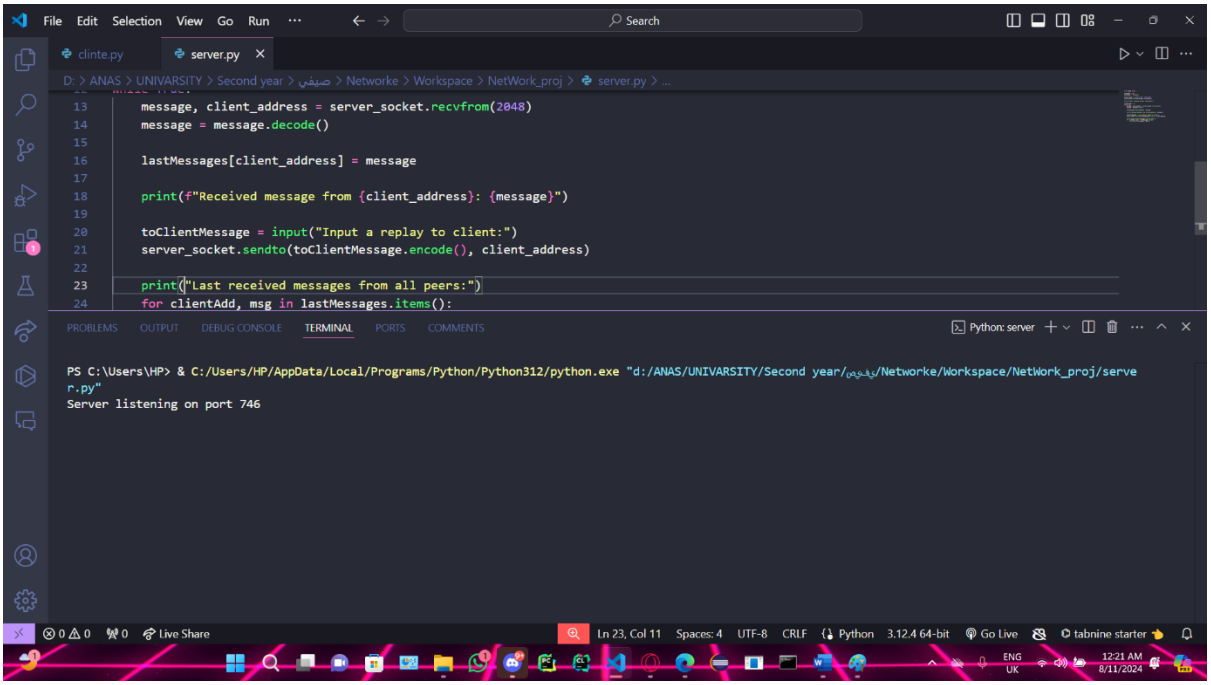


Figure 30 The server listens on port

Now we run the server and it listens on port 476.

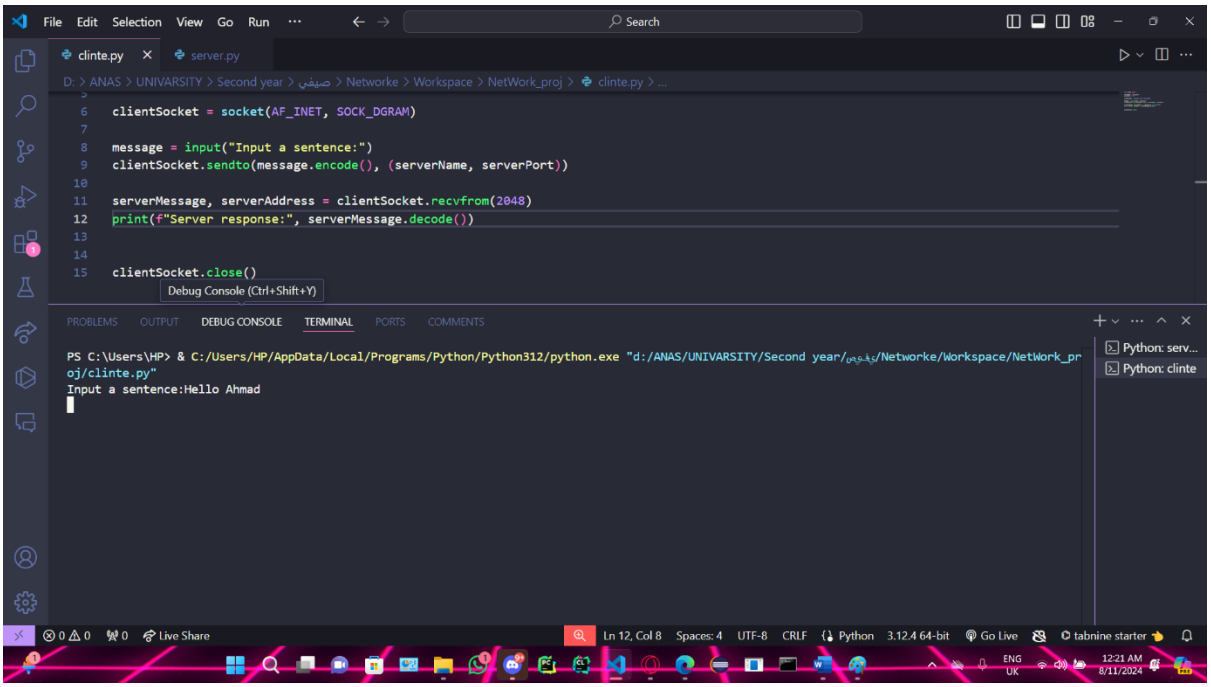


Figure 31 The client sends a message to the server on port 476

Client 1 sends a message ”Hello Ahmad” to the server on port 476,
And after sending the message, it waits for a response from the server and displays it.

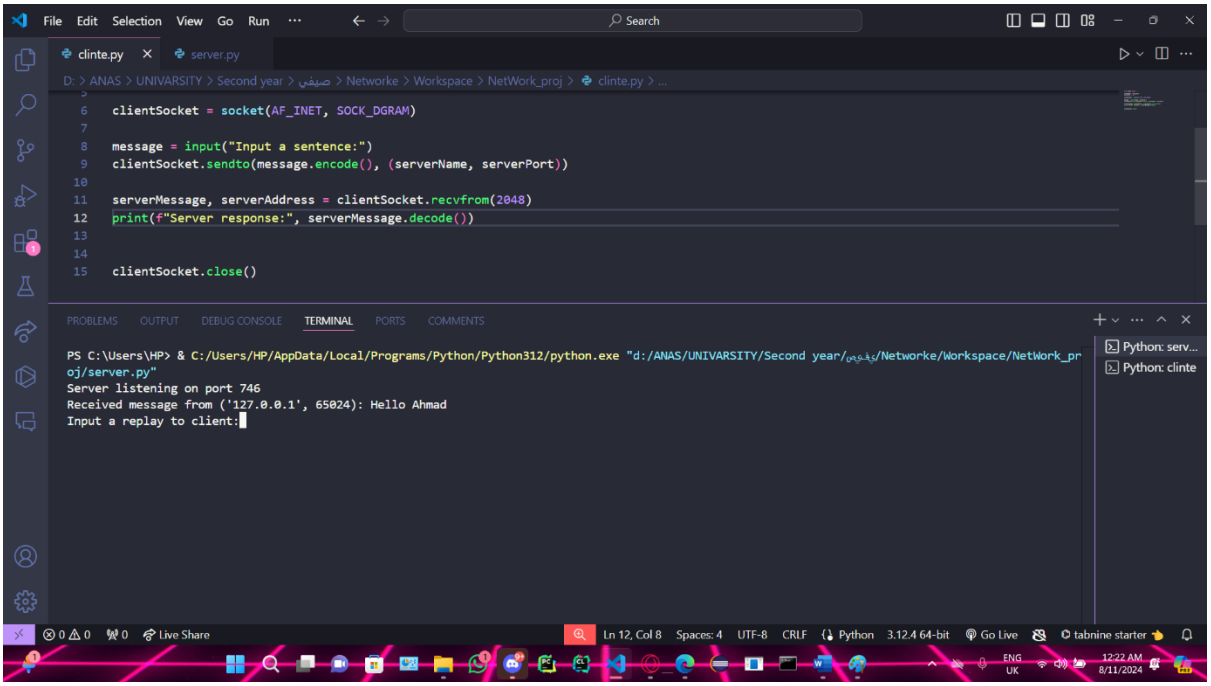


Figure 32 The server receives messages from clint 1

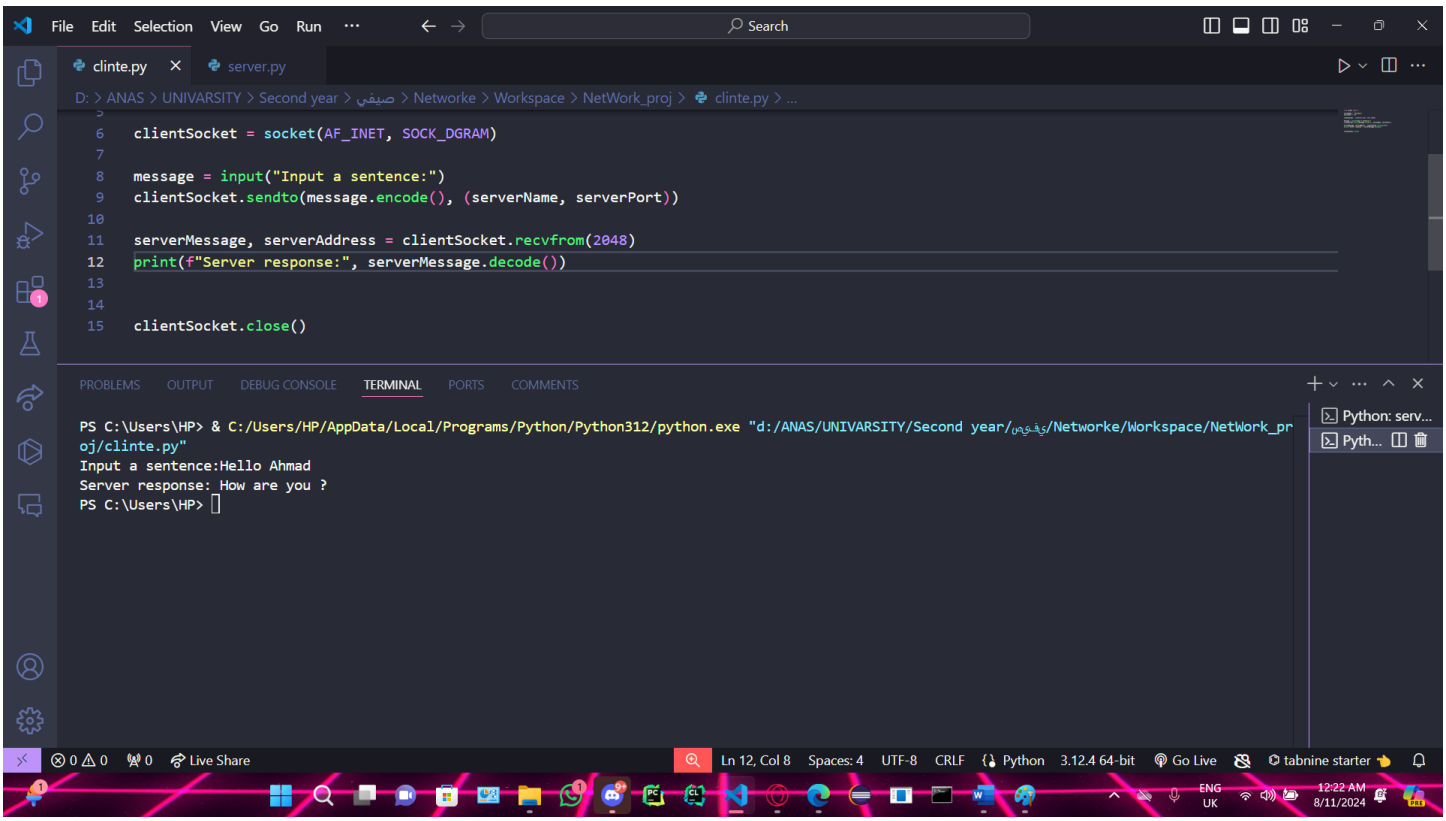


Figure 33 Clint 1 messages from The server

The server receives a message from client 1(show his client address) and, stores it received from each client, the server enter a response, which is sent back to the client .

Figure 34 Client 2 sends a message to the server on port 746

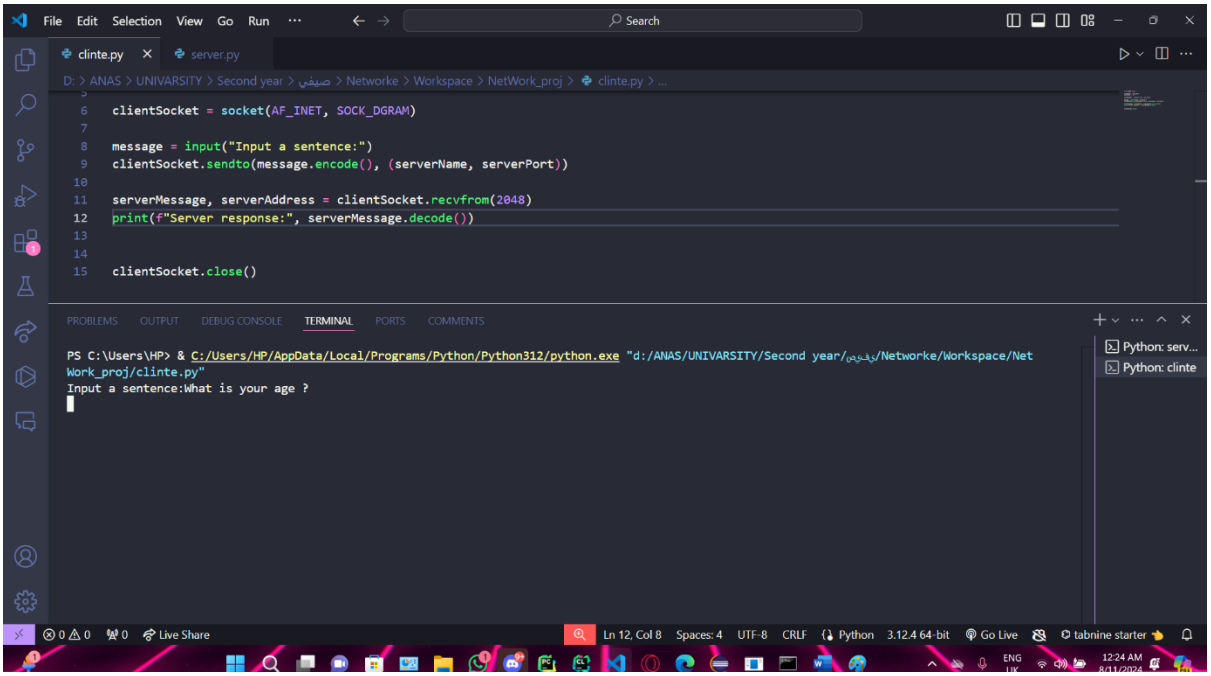


Figure 32: Client 2 sends a message to the server on port 746

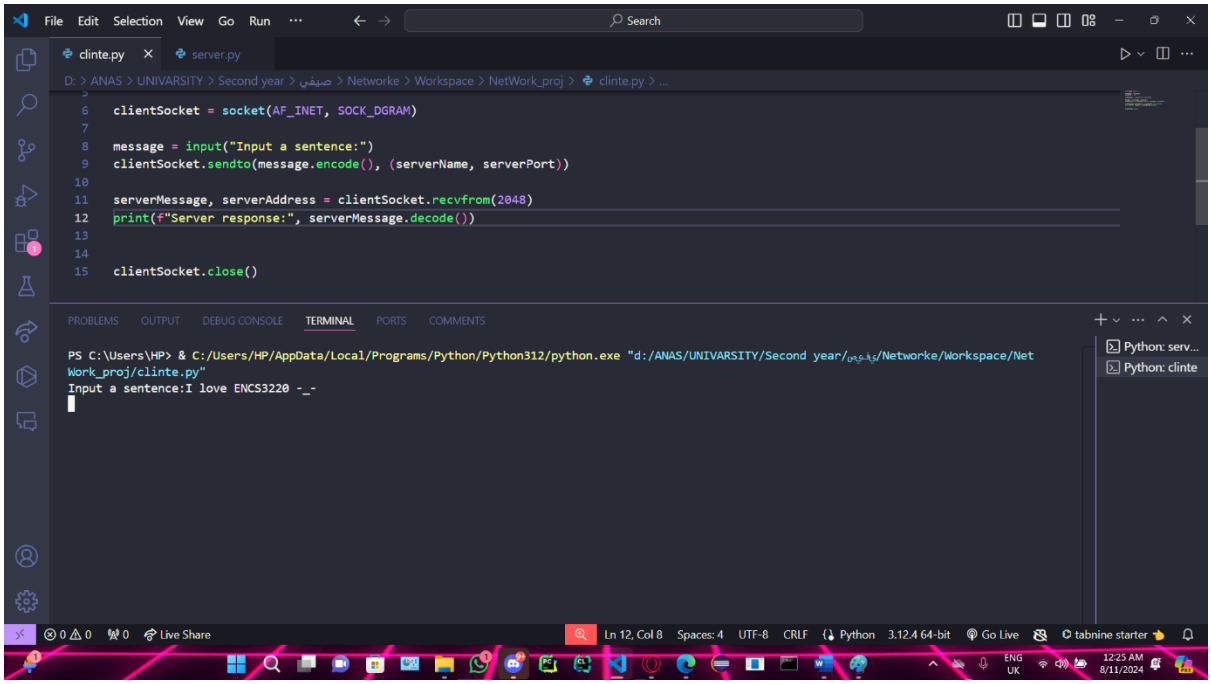


Figure 33:Client 3 sends a message to the server on port 746

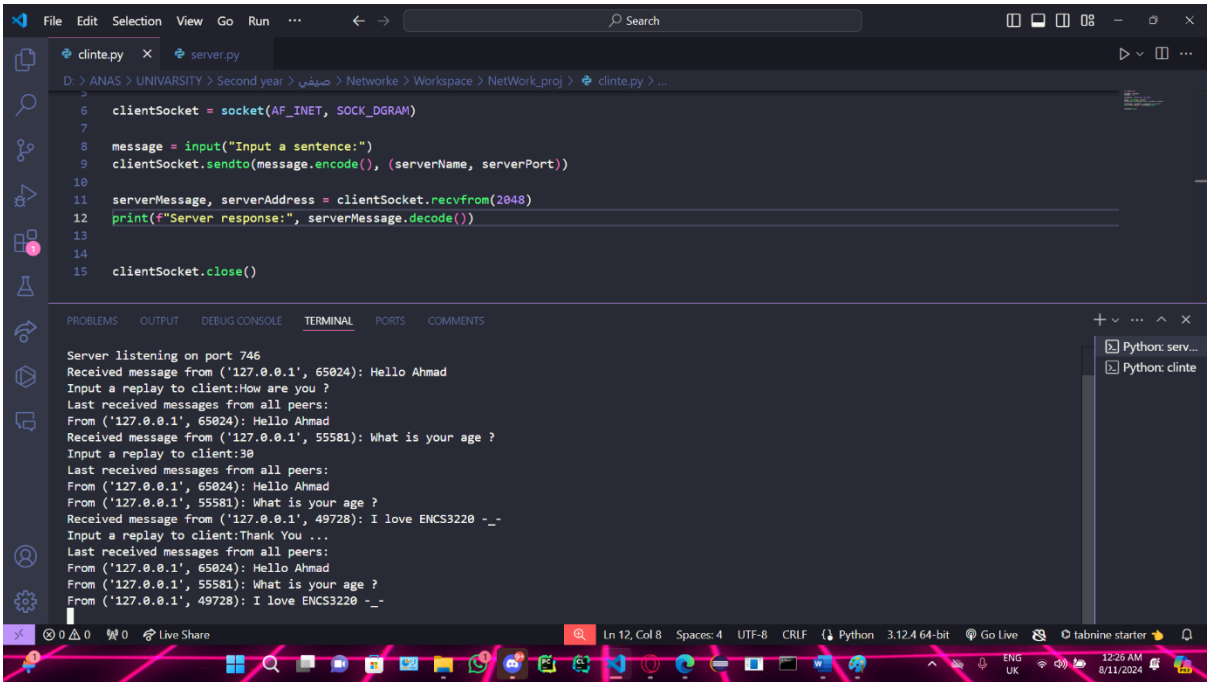


Figure 35:The server shows a list for all received massage.

After receiving a message, the server enter a response, which is sent back to the client, it also prints the entire communication history with all clients after each message.

TASK 3: 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.

Answer:

Starting by creating Server Socket with a specific Port number, and waiting for a TCP connection from one client.
If there is a client asking for a connection with the server, then it's creates a connection socket, and address of the client, then receive a Client Request.
By splitting the client request, we got the object that the client asked for, depends on that the server response vary.

Project Screen Record:

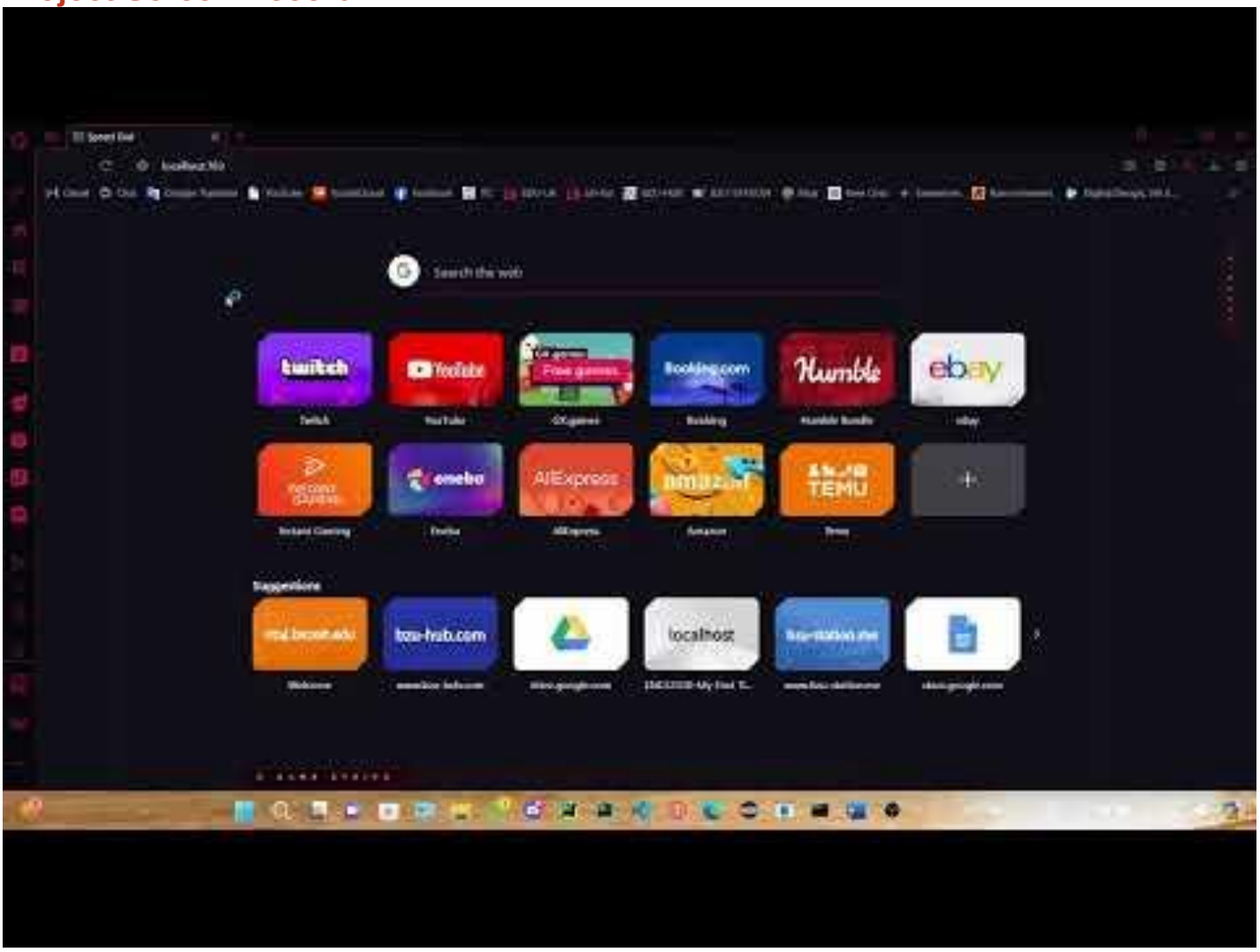


Figure 36: Project Screenrecord:

Browser Screenshots:

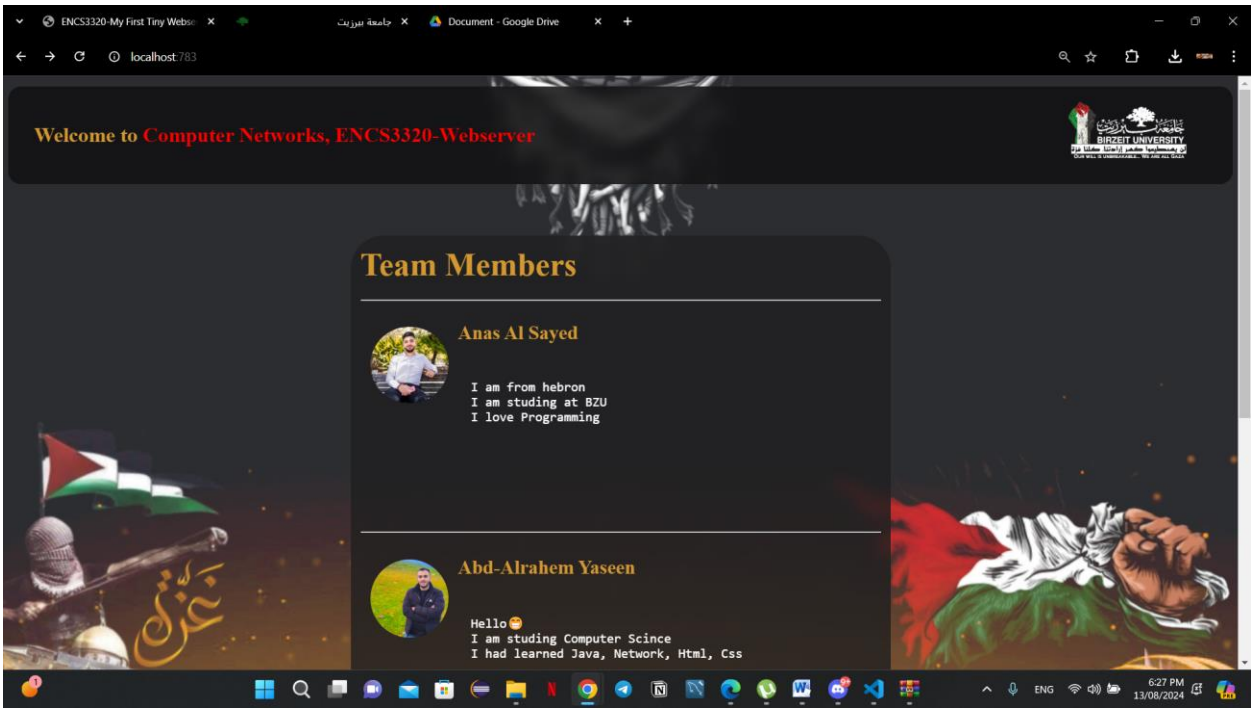


Figure 37 regular request localhost:783

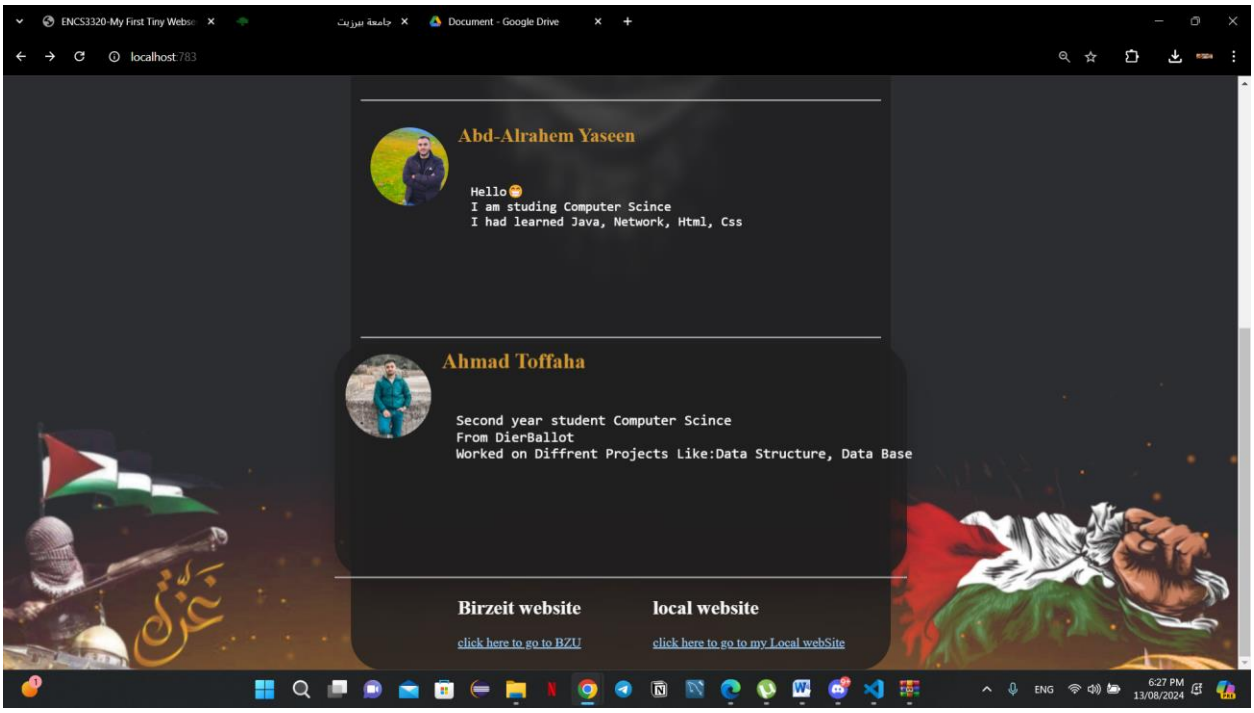


Figure 38 Default website

As shown in figure 37,38 we used the default URL "localhost:783"

It shows the default web page which shows information about us and pictures throw a design we made , also show a links to Birzeit website and link to our website in the last part of the page .

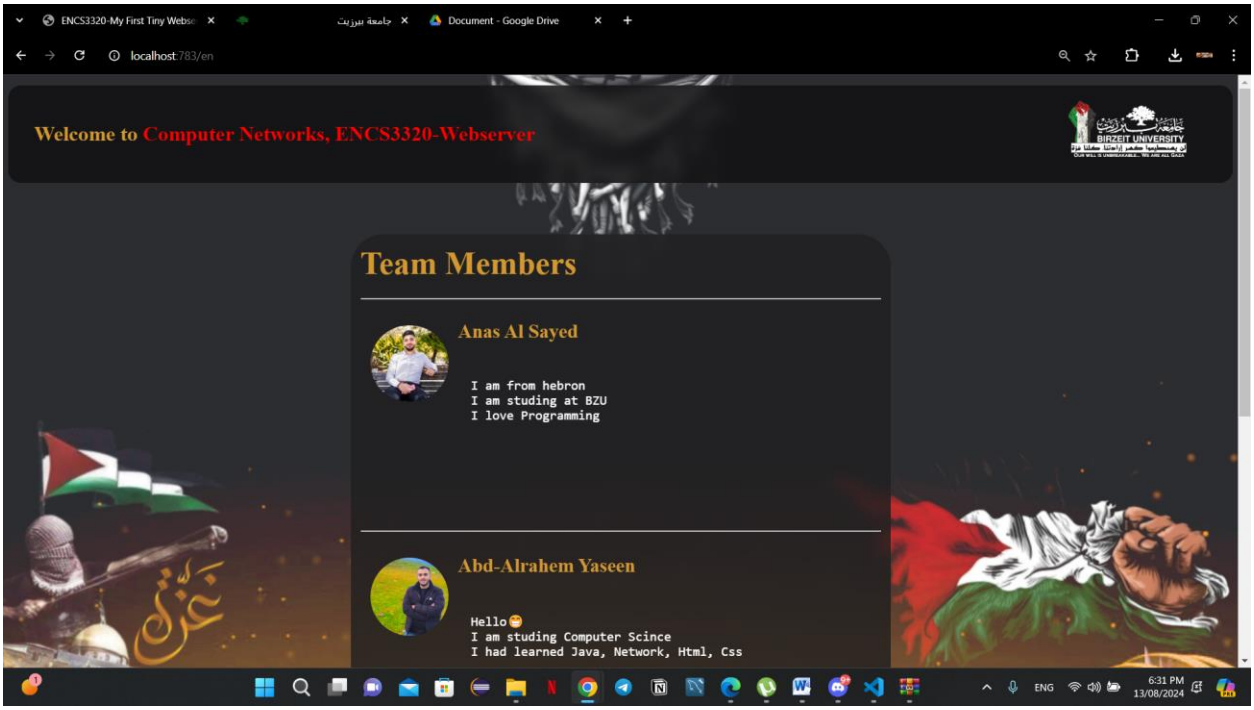


Figure 39 request for /en

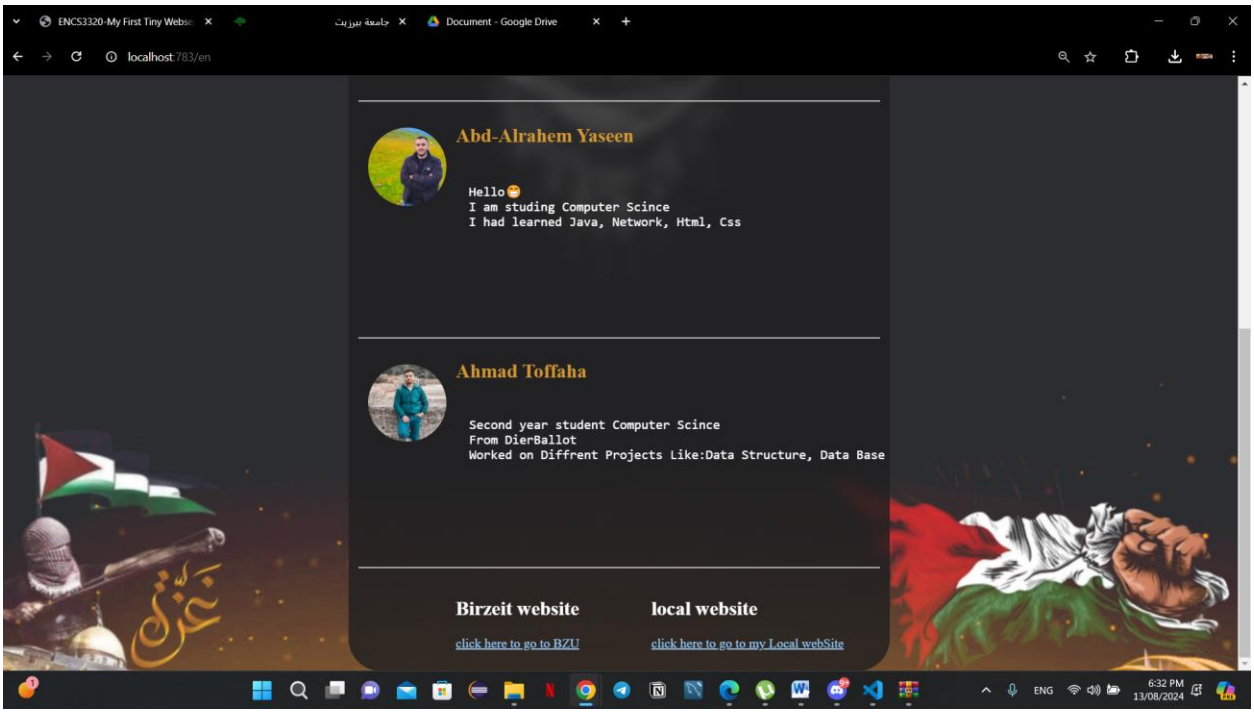


Figure 40 the English version /en

As for pictures 39,40 we used the URL "localhost:783/en"
It shows the default web page in English language which shows information about us and pictures throw a design we made , also show a links to Birzeit website and link to our website in the last part of the page .

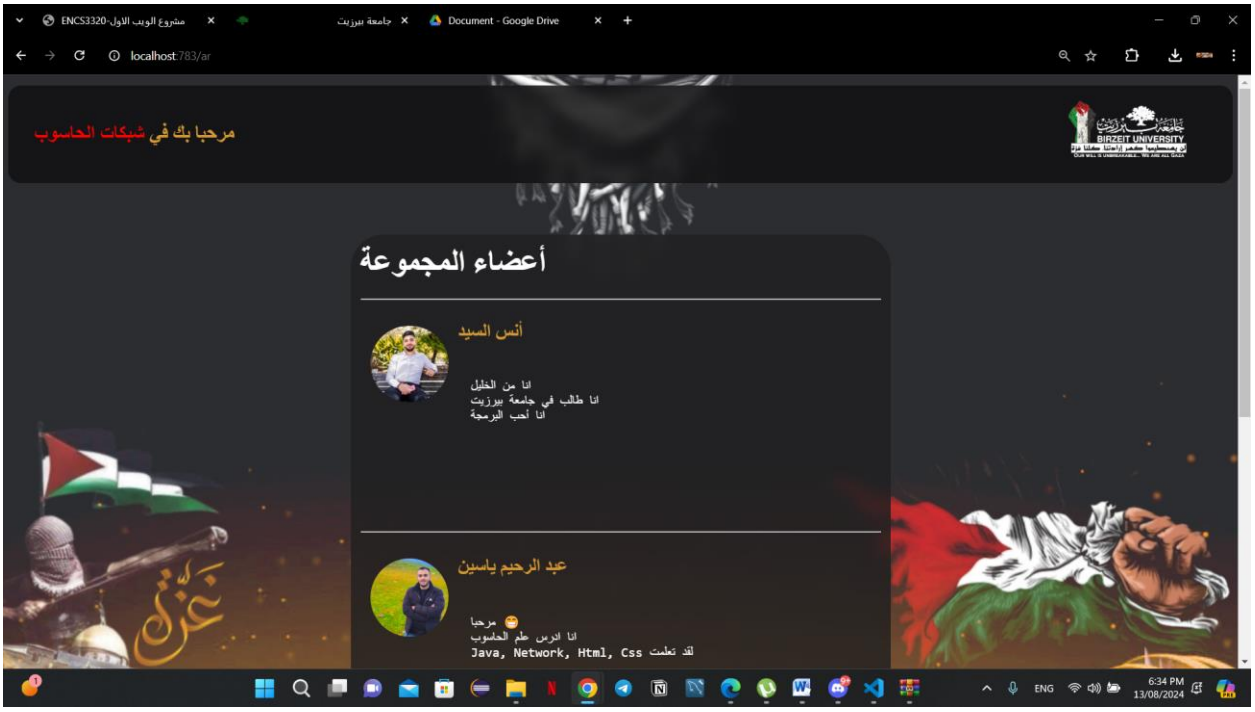


Figure 41 request for /ar

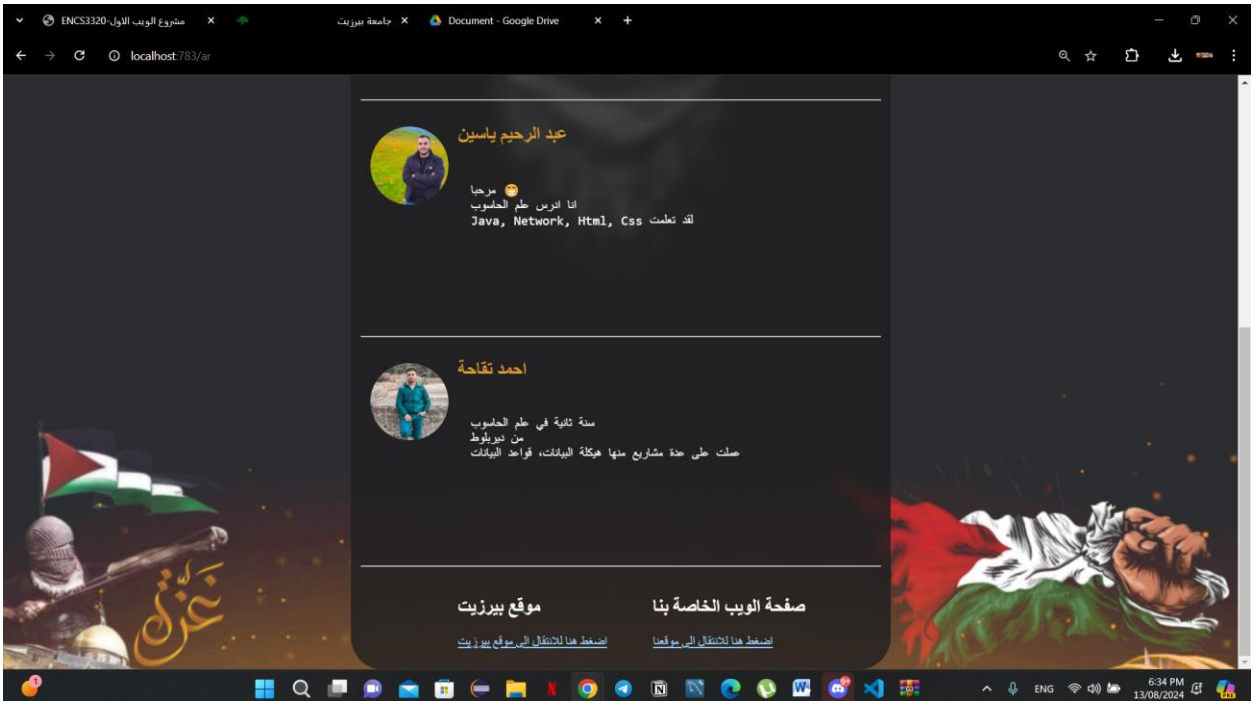


Figure 42 the Arabic version /er

In pictures 41,42 we used the URL "localhost:783/ar"
It shows the default web page in Arabic language which shows information about us and pictures throw a design we made , also show a links to Birzeit website and link to our website in the last part of the page All in Arabic language .

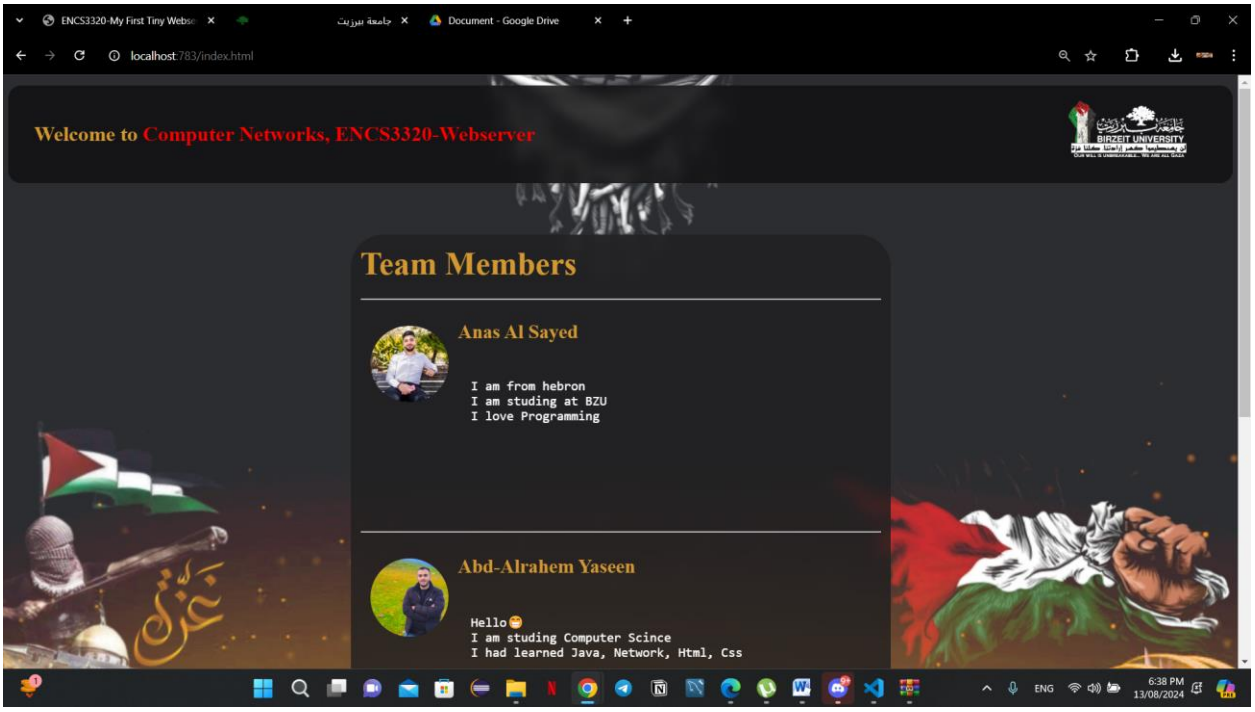


Figure 43 request for /index.html "default"

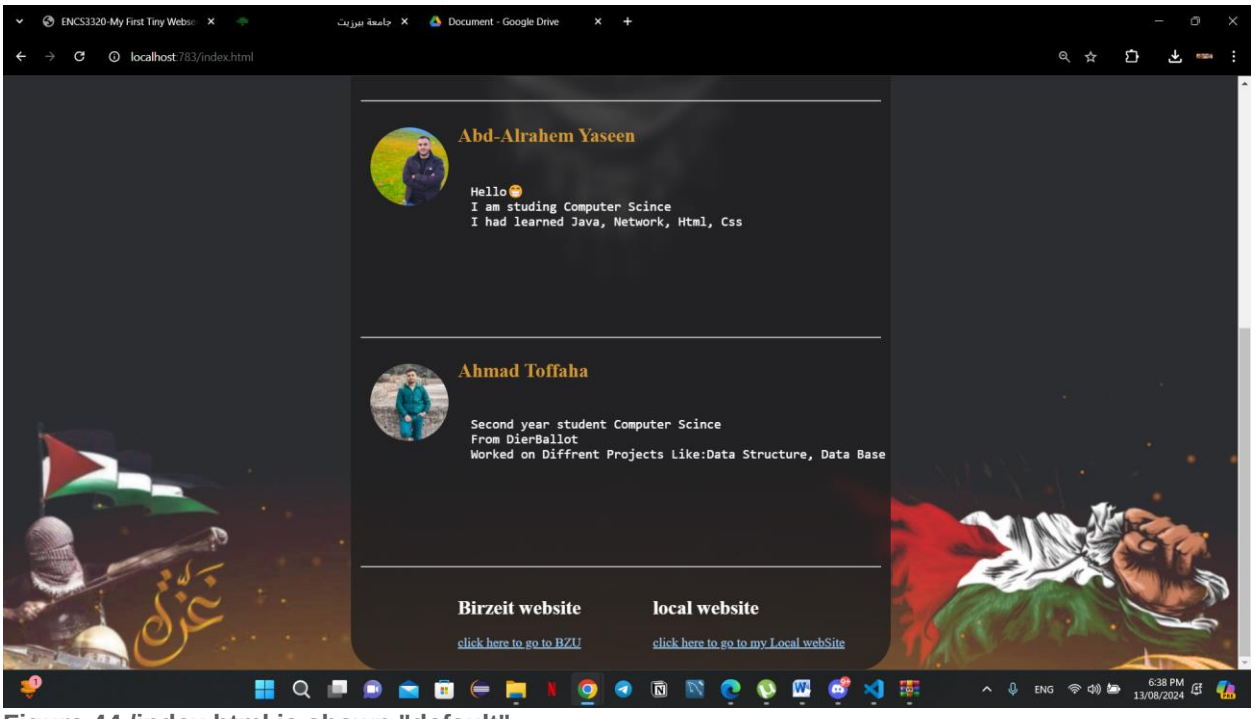


Figure 44 /index.html is shown "default"

In pictures 43,44 we used the URL "localhost:783/index.html"
Which is same as the default web page which shows information about us and pictures throw a design we made , also show a links to Birzeit website and link to our website in the last part of the page.

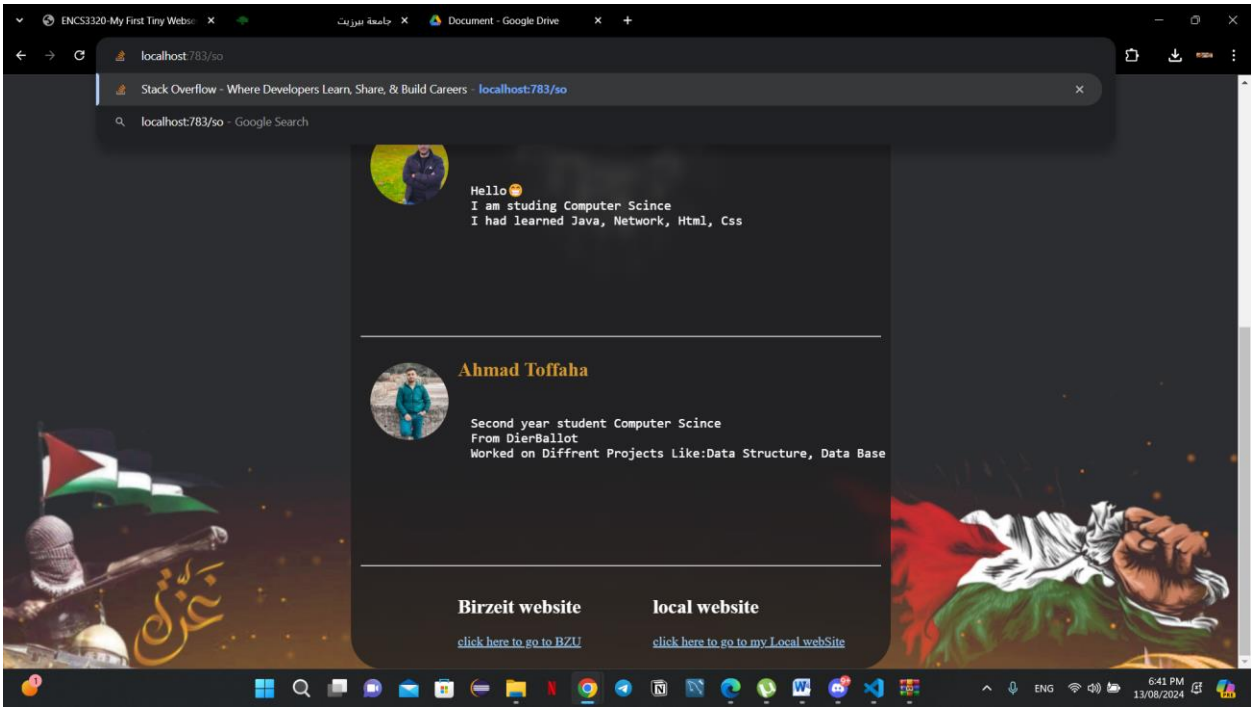


Figure 45 request for /so

When the client asks for a /so, then the connection socket sends a response with "307 Temporary Redirect" to tell the client that the object requested has been temporarily moved to the URL given which is https://www.stackoverflow.com, then the socket opens a link to stackoverflow web page.

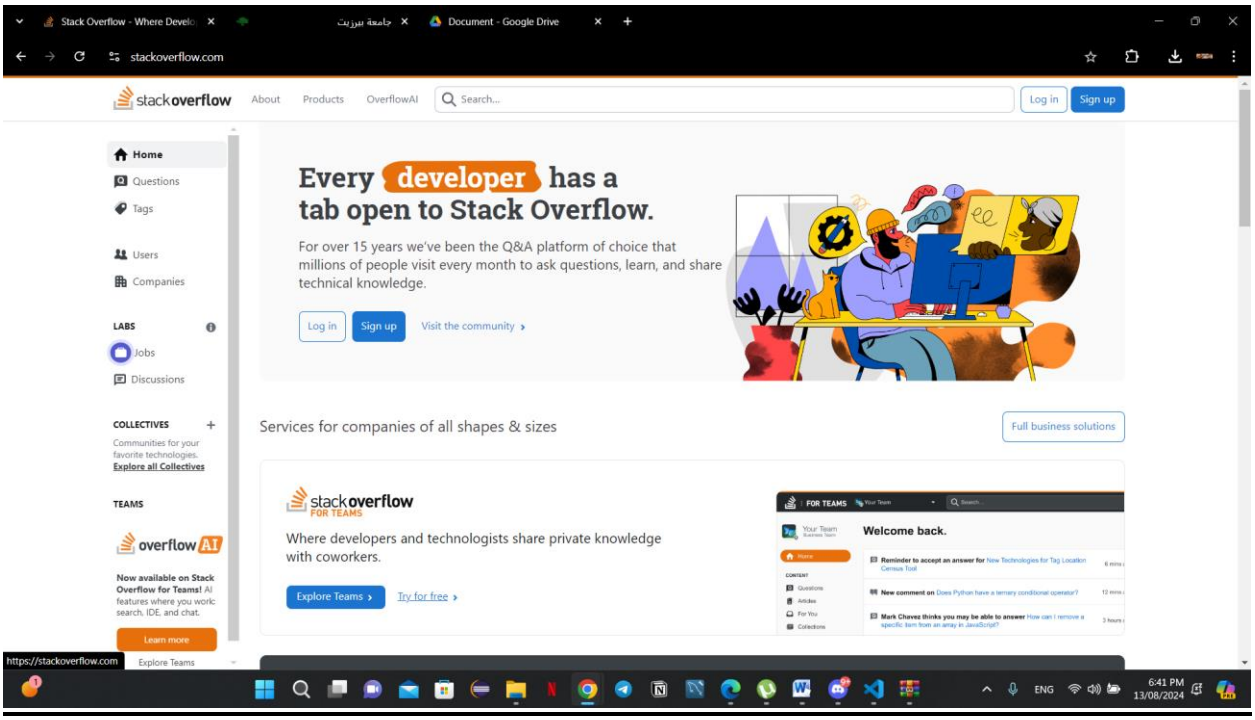


Figure 46 stack over flow website shown

In pictures 45,46 we used the URL "localhost:783/so"
Which takes us to the stack over flow website though the URL we made .

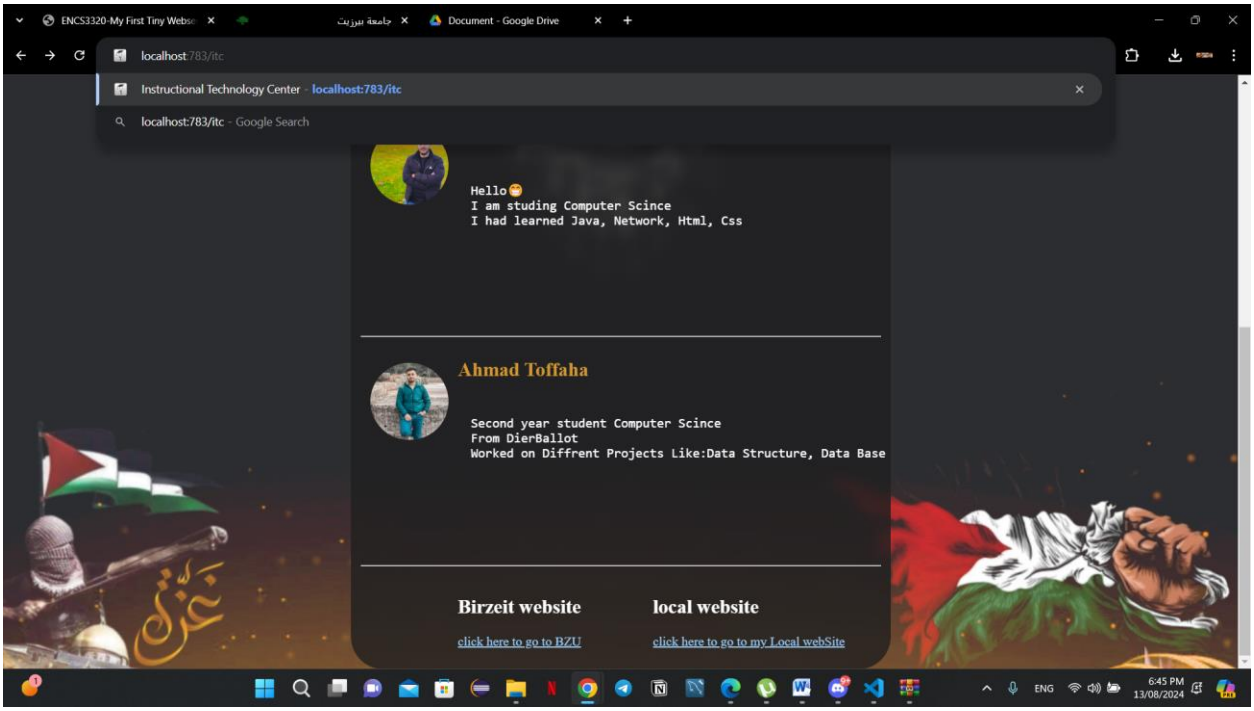


Figure 47 request for /itc

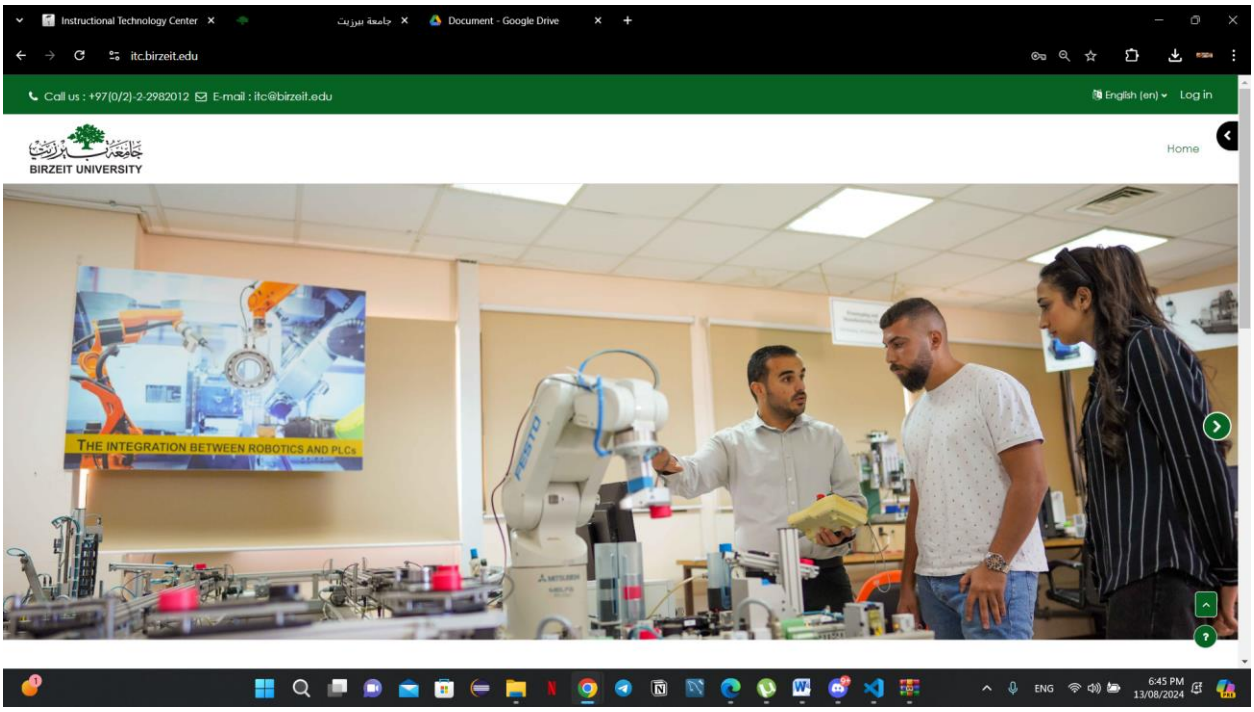


Figure 48 itc website

In pictures 47,48 we used the URL "localhost:783/itc"
Which takes us to the itc website through the URL we made .

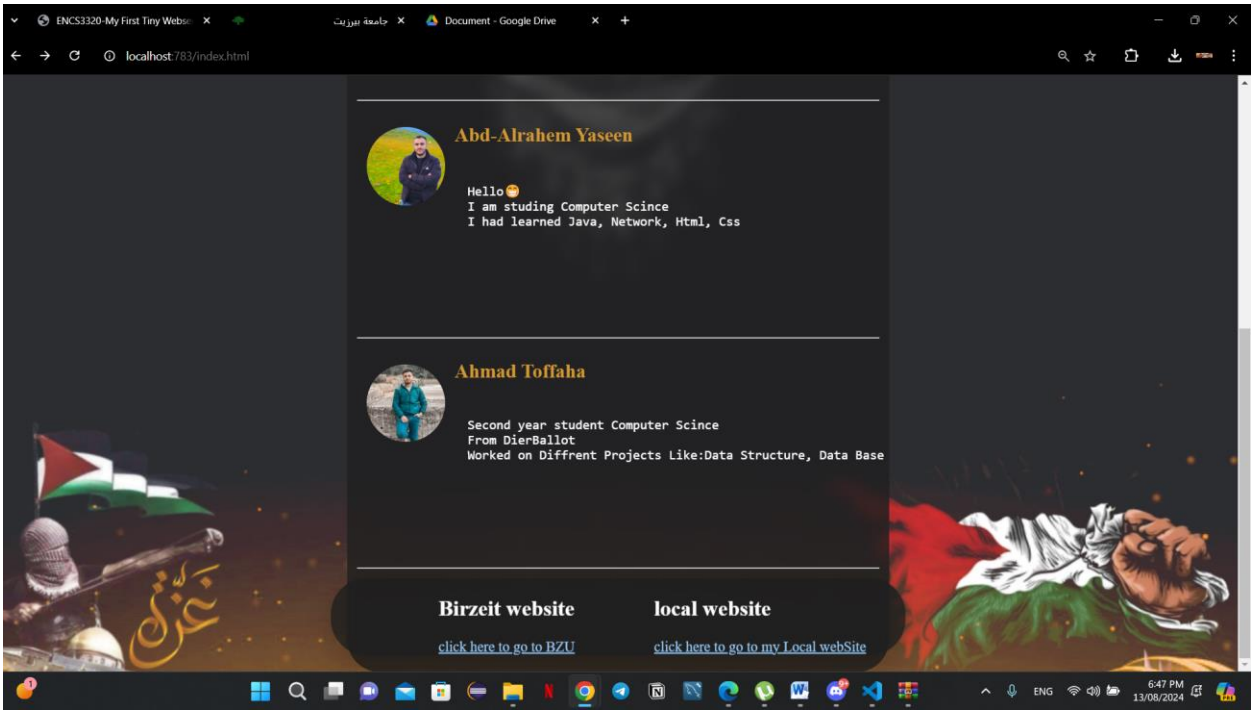


Figure 49 Birzeit link at the end of the website

When the client asks for a /bzu, then the connection socket sends a response with "307 Temporary Redirect" to tell the client that the object requested has been temporarily moved to the URL given which is https://www.birzeit.edu/en, then the socket opens a link to birzeit web page.

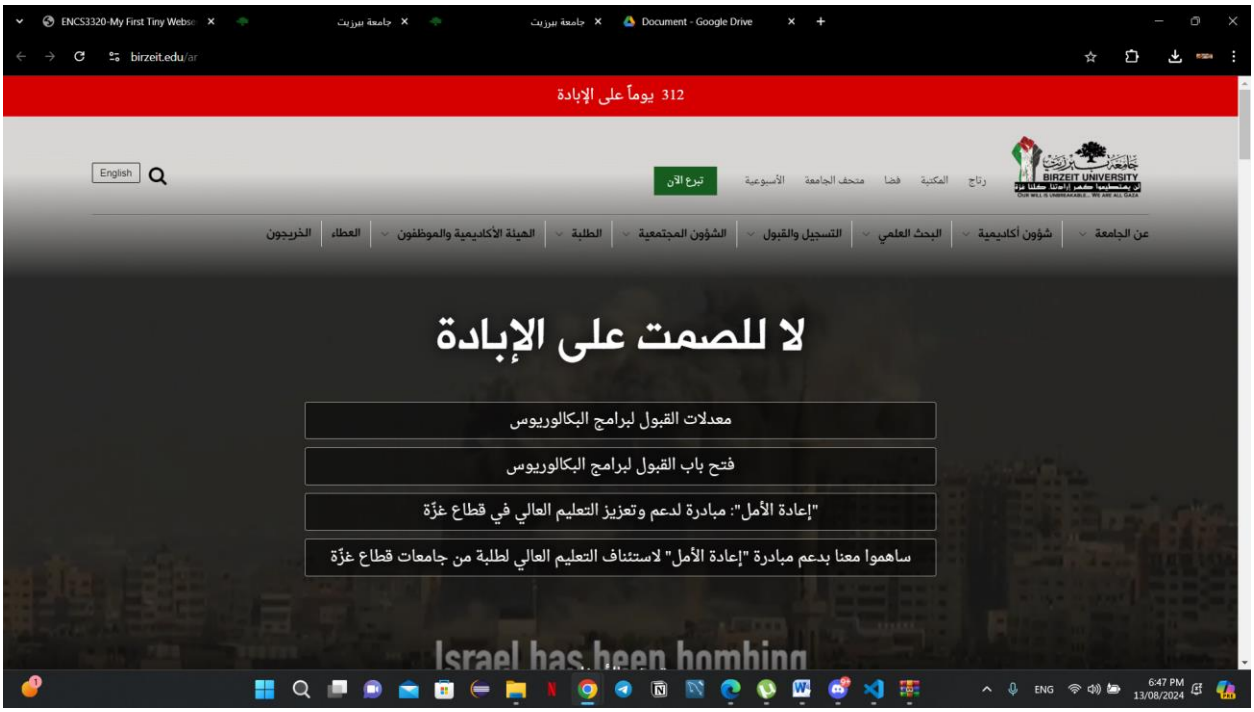


Figure 50 Birzeit website shown.

In pictures 49,50 we clicked on the link at the last part of the page ,

Which takes us to the Birzeit website as shown in picture.

---*---*---*---*---*---*---*---

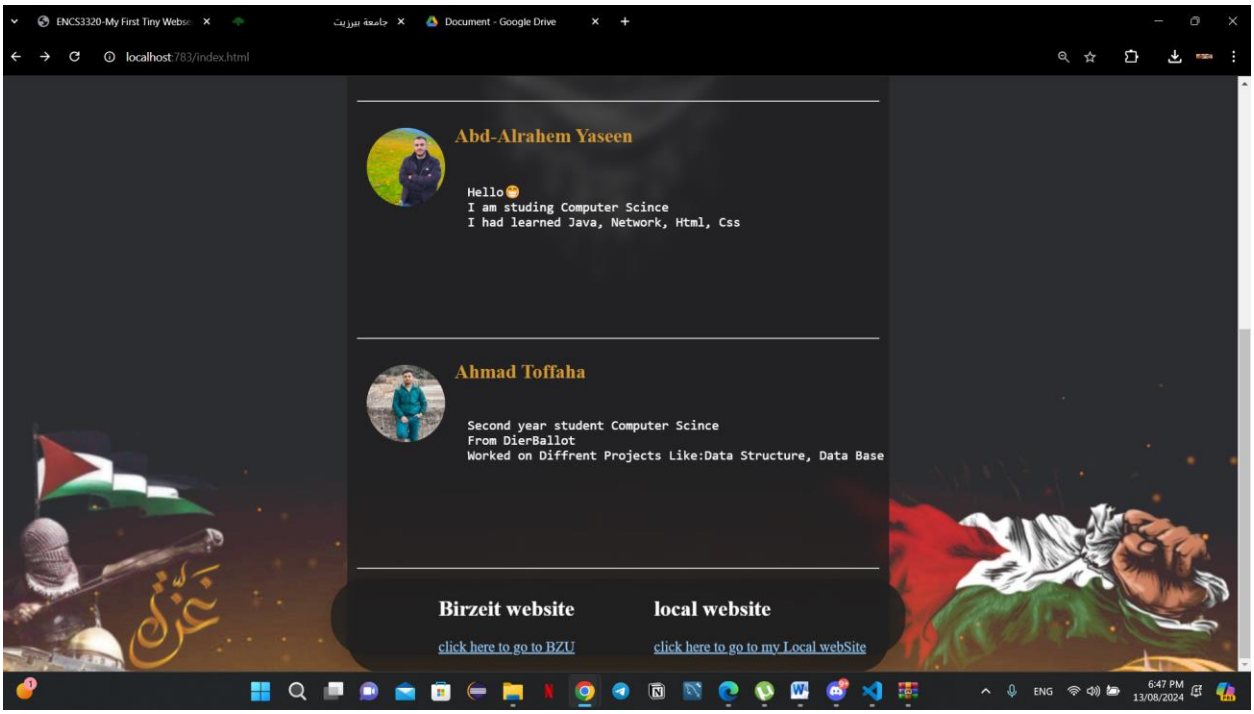


Figure 51 our website link at the end of the page

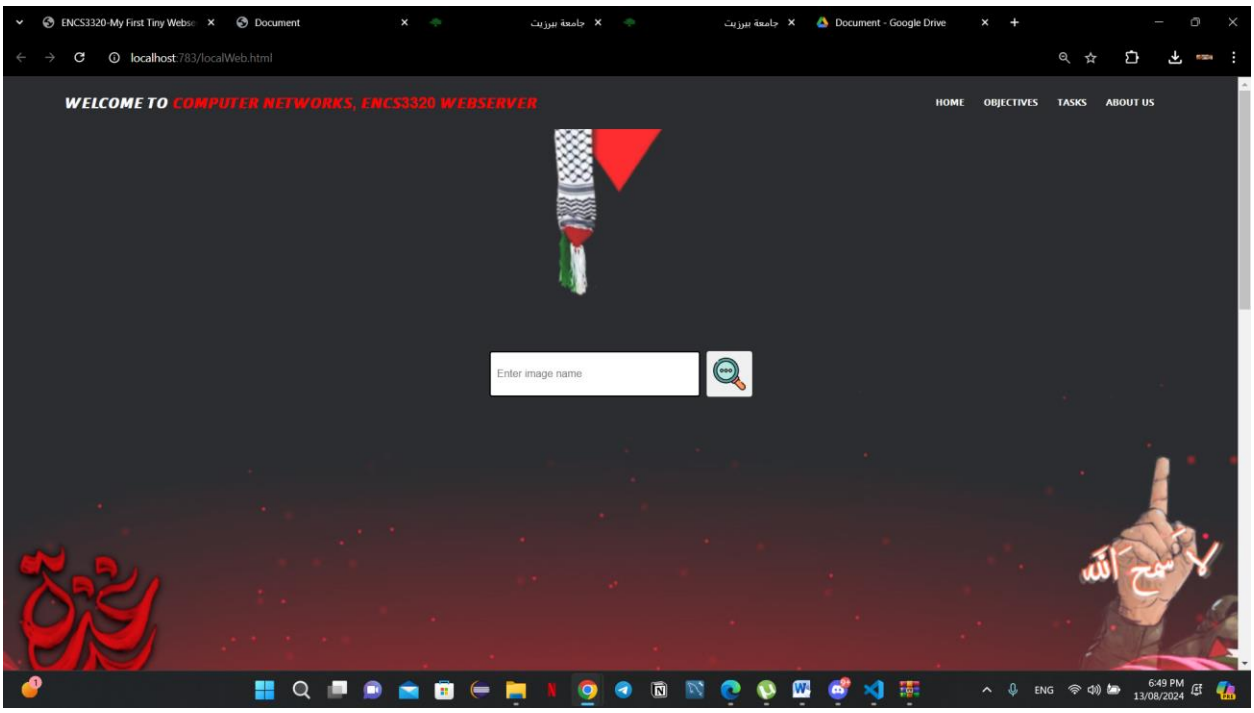


Figure 52 our website shown.

In pictures 51,52 we clicked on the link at the last part of the page ,
Which takes us to our website that we made we use to search for some pictures.

----------*---*---*---*---*---*---*----------

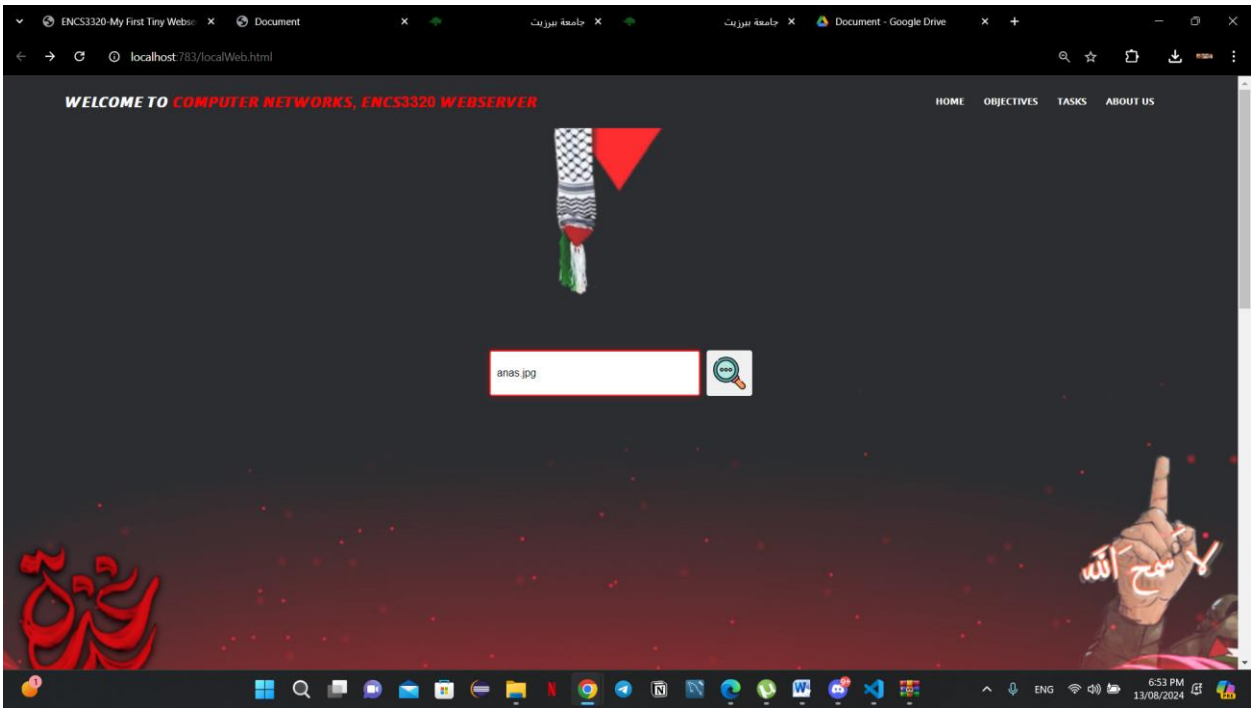


Figure 53 Search for anas.jpg.

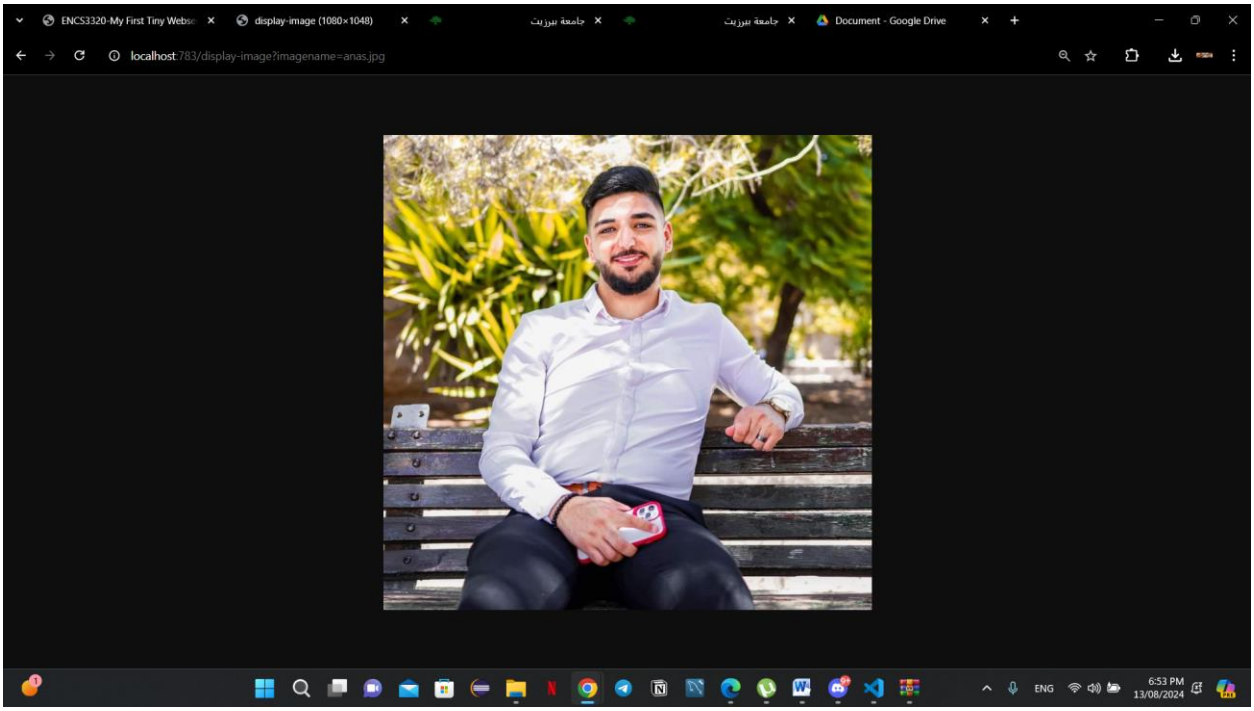


Figure 54 jpg photo shown.

In pictures 53,54 we searched for anas.jpg which is stored in pictures file ,
And it shows the picture in the website.

----------*---*---*---*---*---*---*----------

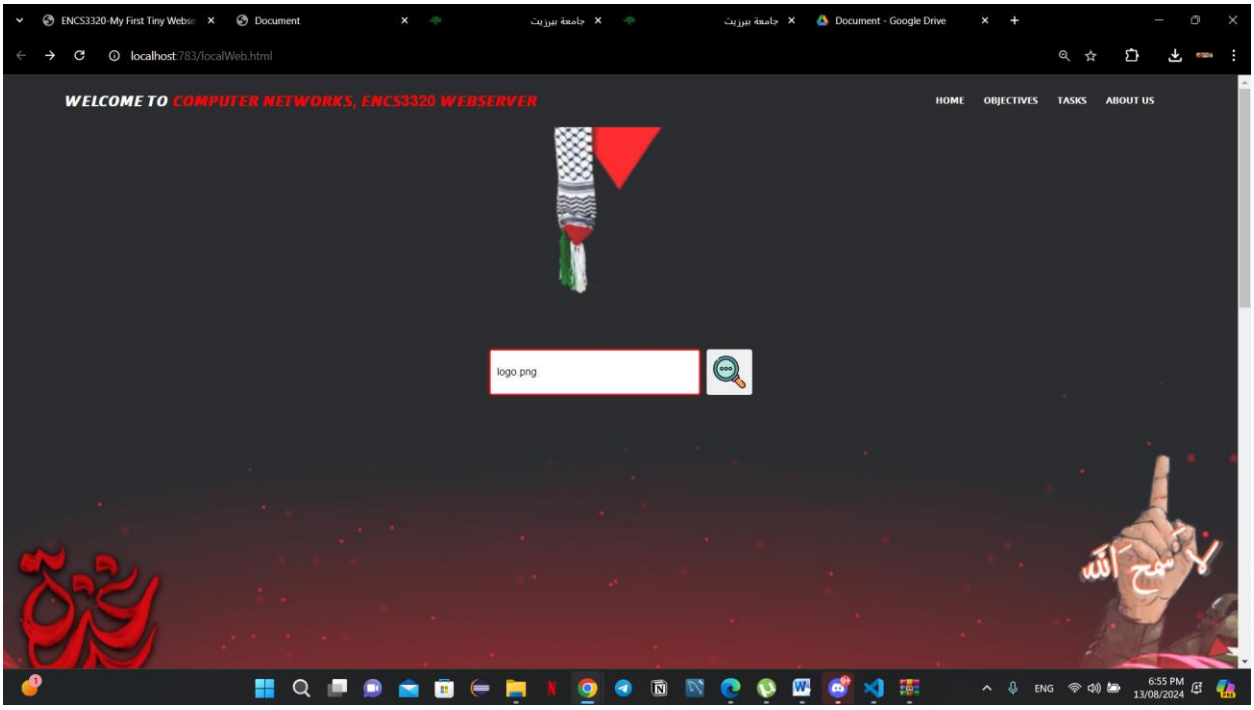


Figure 55 Search for logo.png.

When the client asks for a .png input, then the connection socket sends a response with "HTTP/1.1 200 OK" to tell the client that the object is found and everything is going great, then the socket sends a png picture.

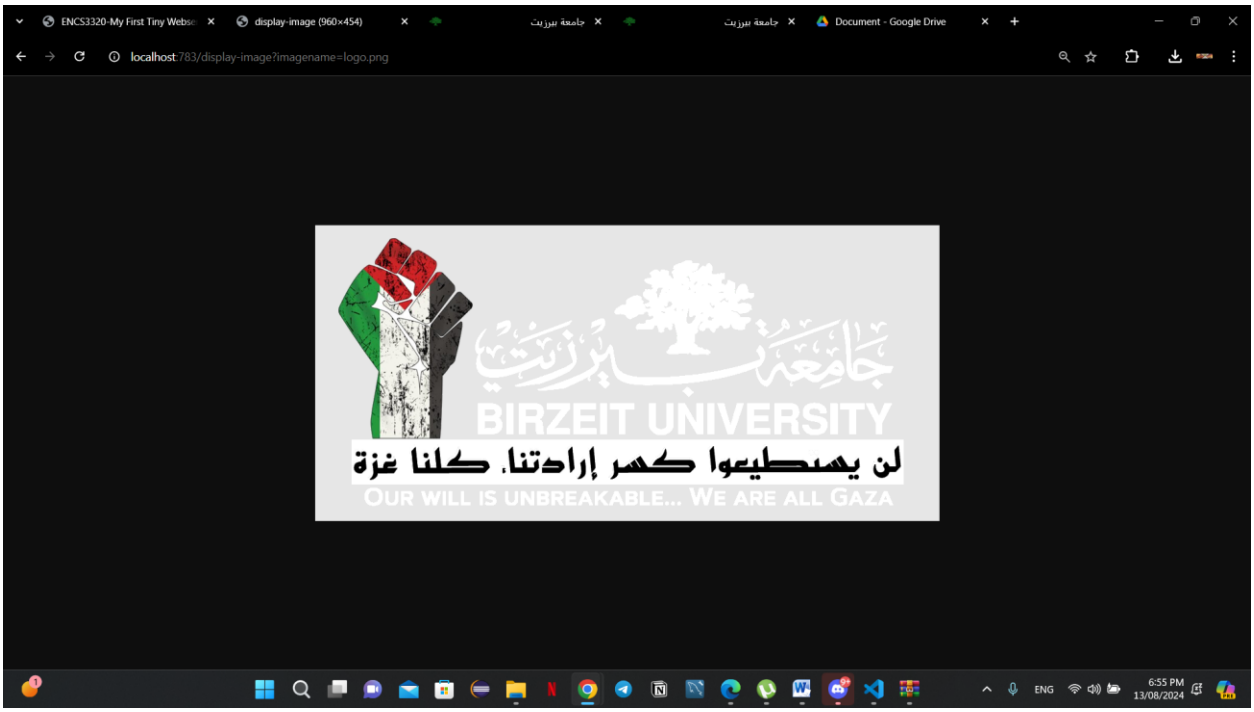


Figure 56 Logo photo is shown.

In pictures 55,56 we searched for logo.png which is stored in pictures file ,
And it shows the picture in the website.

----------*---*---*---*---*---*---*----------

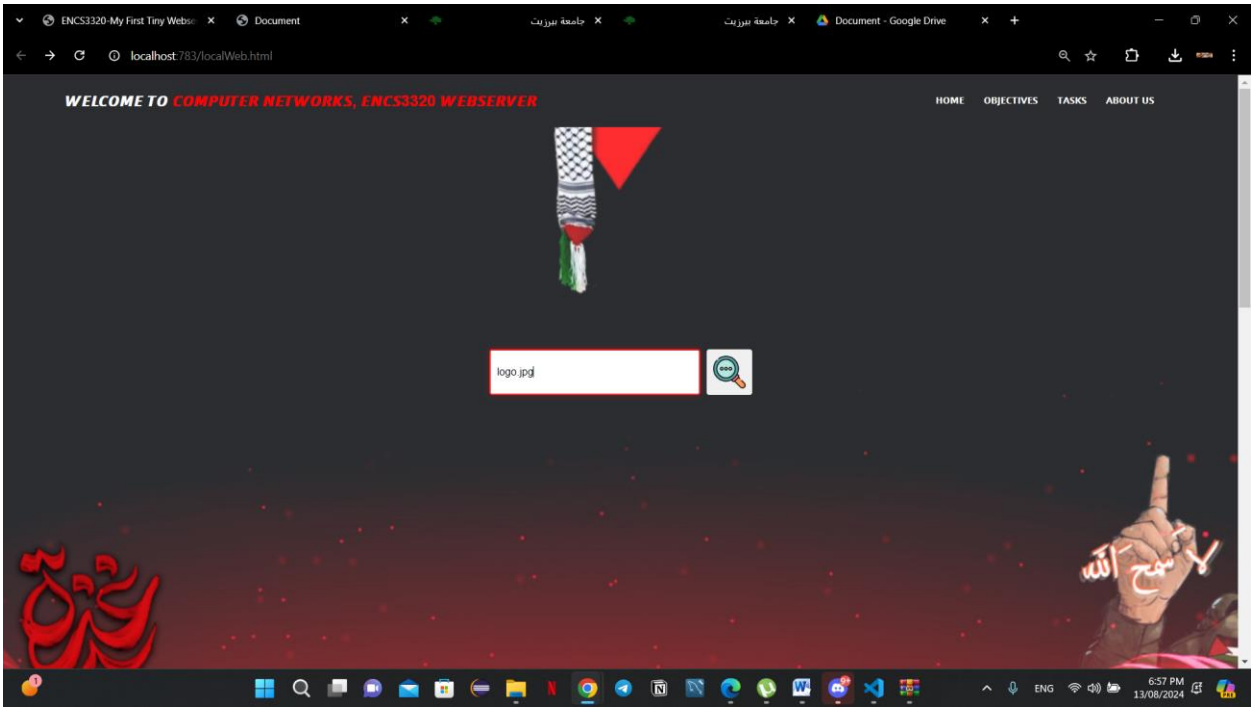


Figure 57 search for logo.jpg which is not found

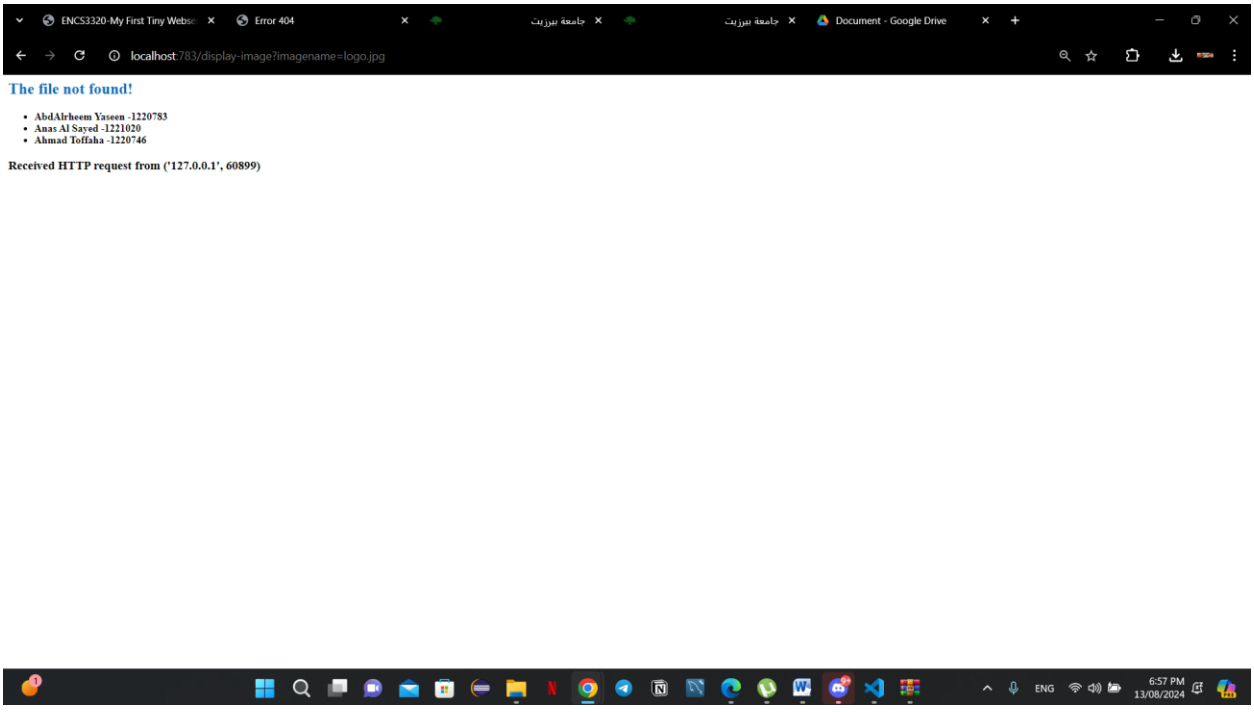


Figure 58 Error massage is shown.

In pictures 57,58 we searched for logo.jpg which is not stored in pictures file , And it shows the error massage in the website.

Screenshots of the browser to show that our project works as expected. (/main_en.html /imagename.png, /go, etc.) . Test the project from a browser on the same computer and from a different computer or phone.

From phone ScreenShote:

for jpg ,png , file not founded, main_en.html file and main_ar.html

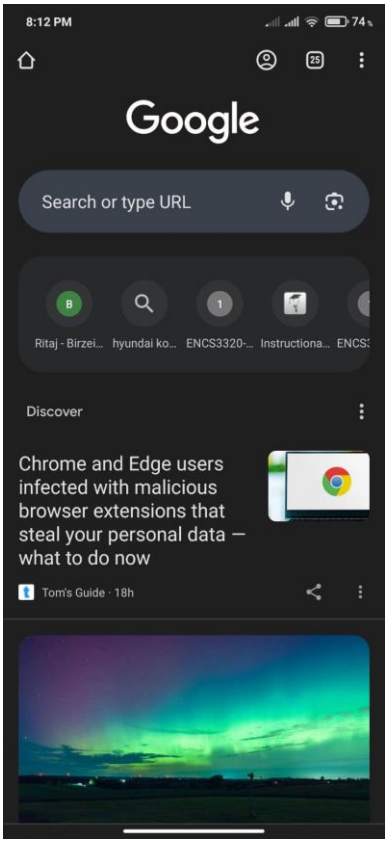


Figure 57 Testing project form Mobile

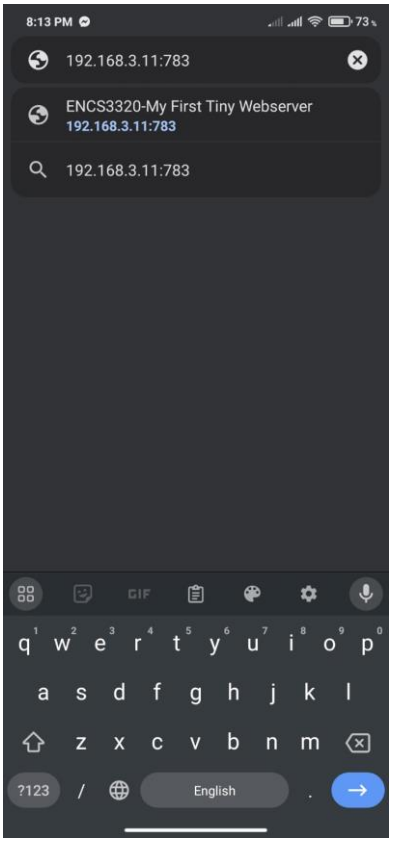


Figure 58 Entering Ip address of server with port#

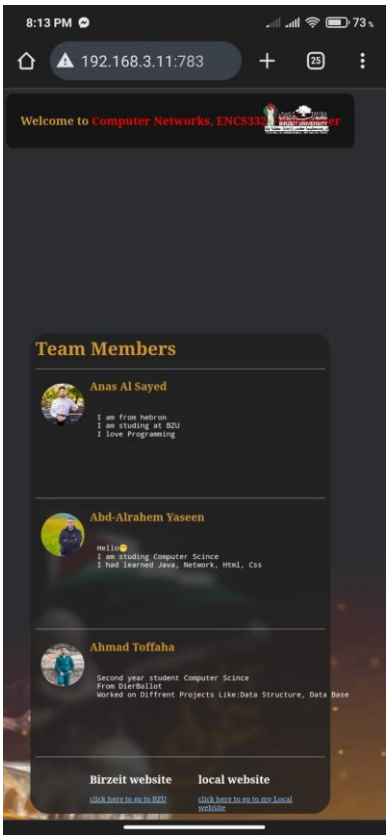


Figure 60 Accessing our website

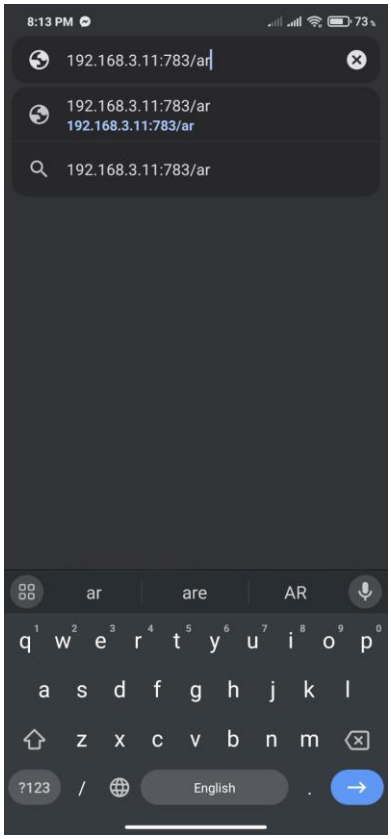


Figure 59 Entering Ip address of server with port# for arabic version

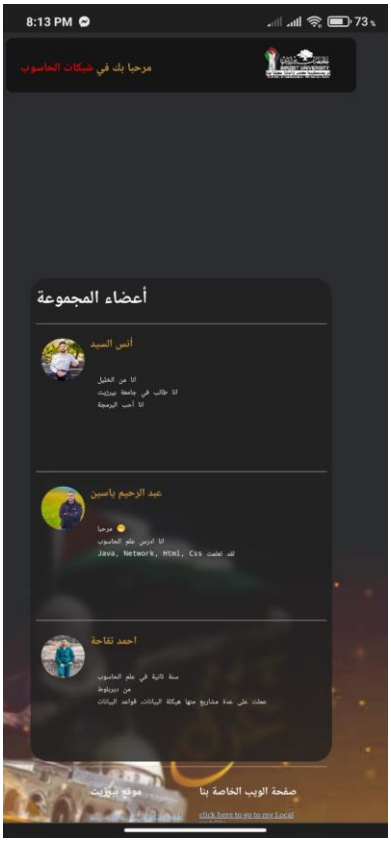


Figure 62 Accessing our arabic version of website

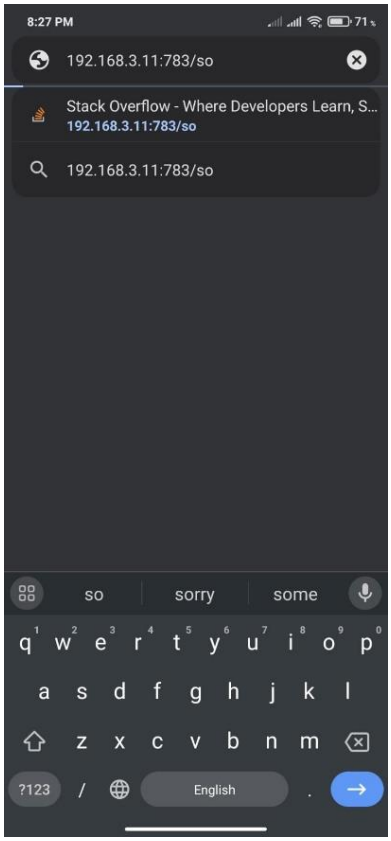


Figure 61 Entering Ip address of server with port# for StackOverFlow

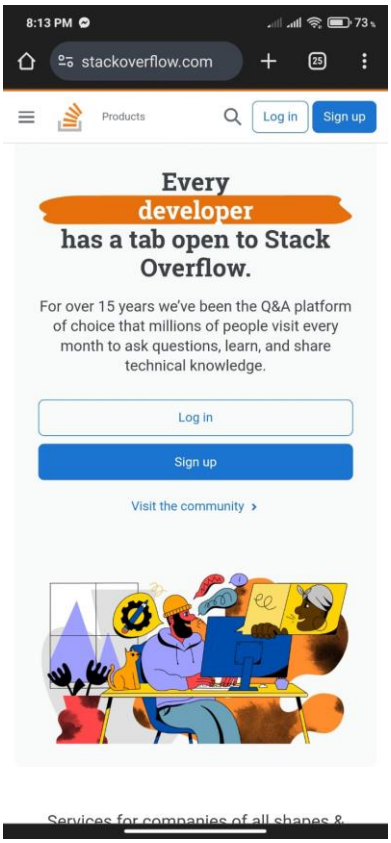


Figure 63 Accessing StackOverFlow

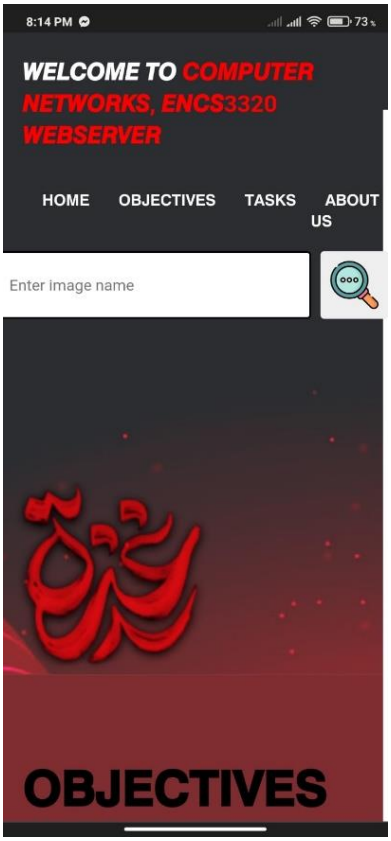


Figure 64 Accessing our website for searching images

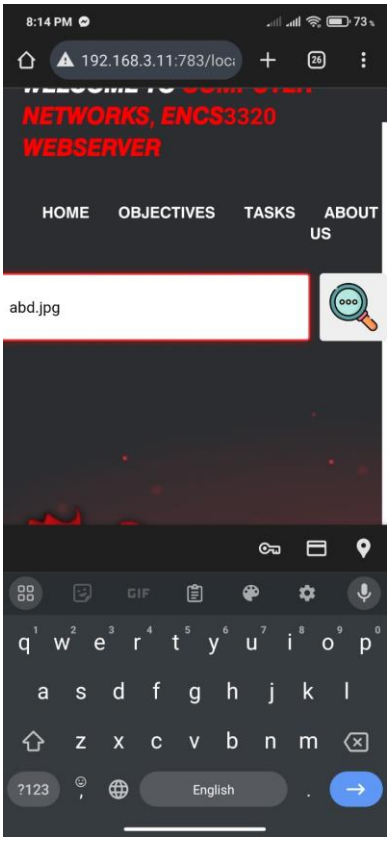


Figure 64 Searching for abd.jpg

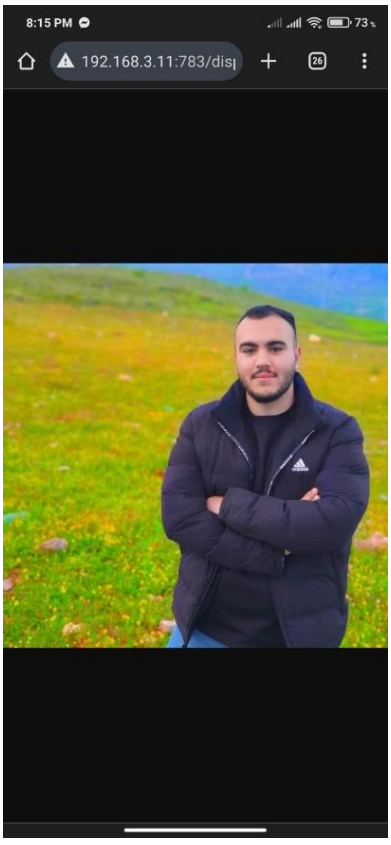


Figure 63 Accessing the image

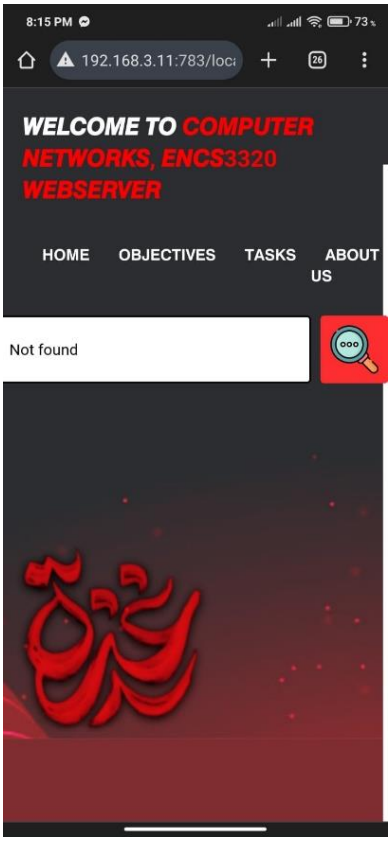


Figure 67 searching for not exist image

Figure 68 simple HTML for file not found

HTTP request:

```
Server is ready!
Received HTTP request from ('127.0.0.1', 51795):
GET / HTTP/1.1

Host: localhost:783

Connection: keep-alive

sec-ch-ua: "Not/A)Brand";v="8", "Chromium";v="126", "Op

sec-ch-ua-mobile: ?0
```

Figure 65 http response

Starting Server and the terminal is empty:

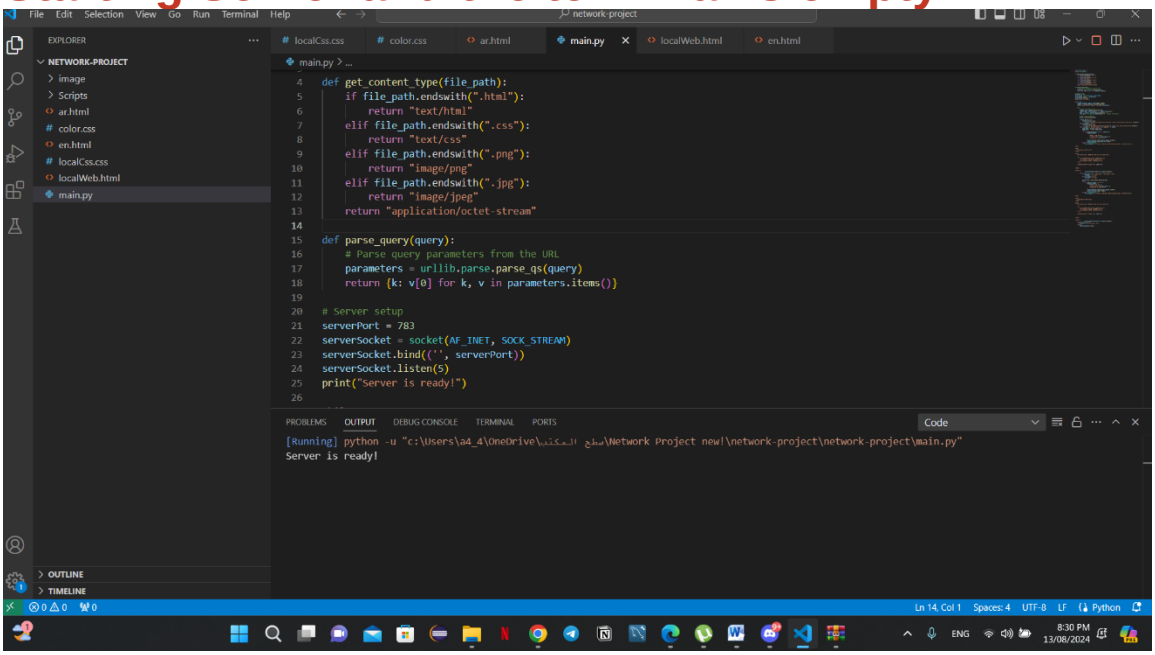


Figure 66 Empty terminal

Requesting the english version of our website:

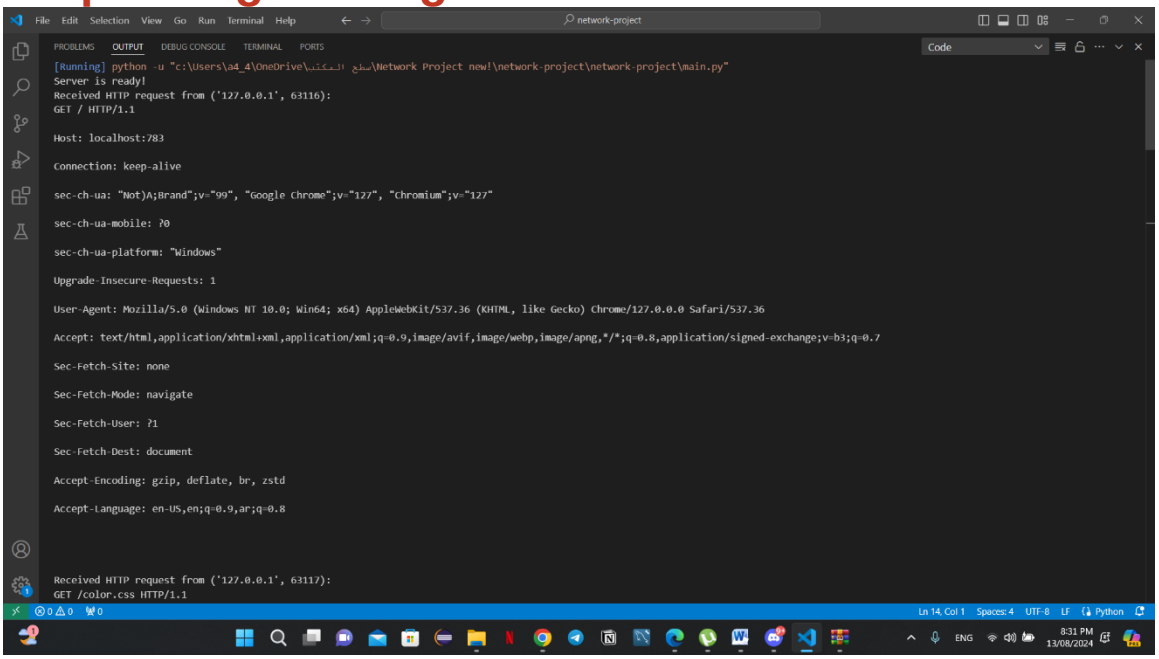


Figure 67 request for server to get Html file

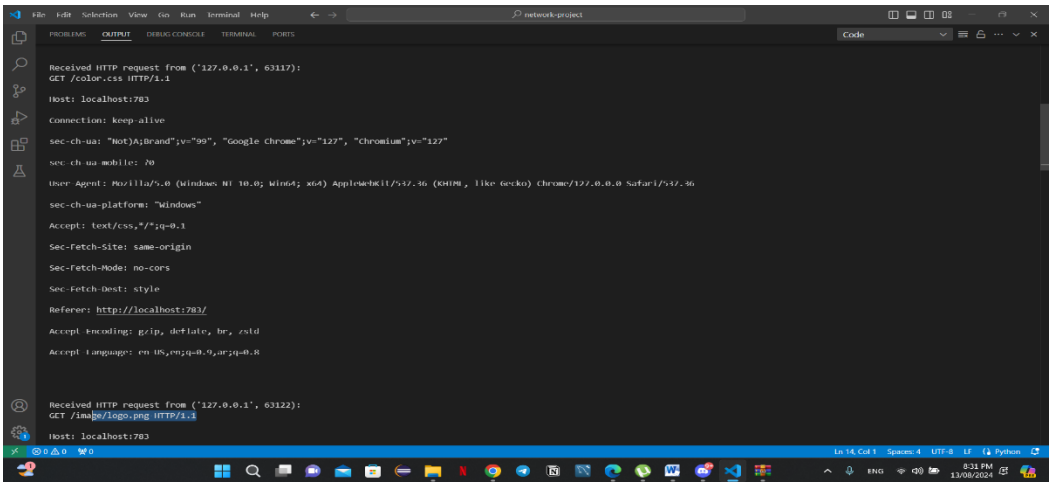


Figure 68 request for server to get CSS file

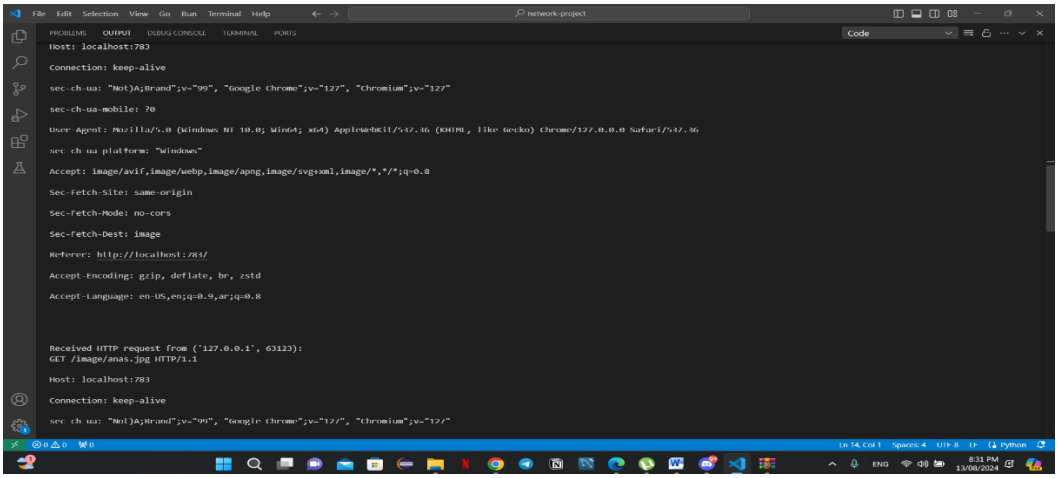


Figure 69 request for server to get logo image

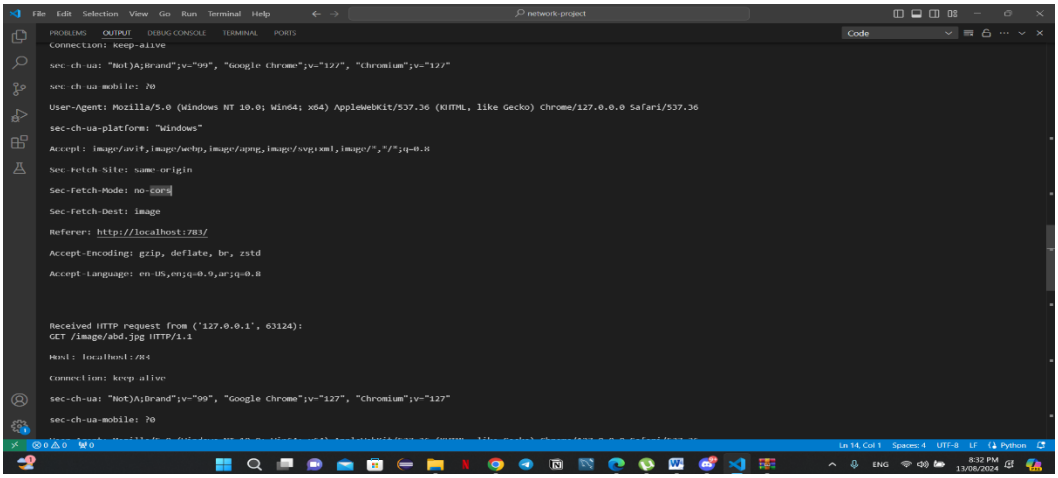


Figure 70 request for server to get images

Requesting the arabic version of our website:

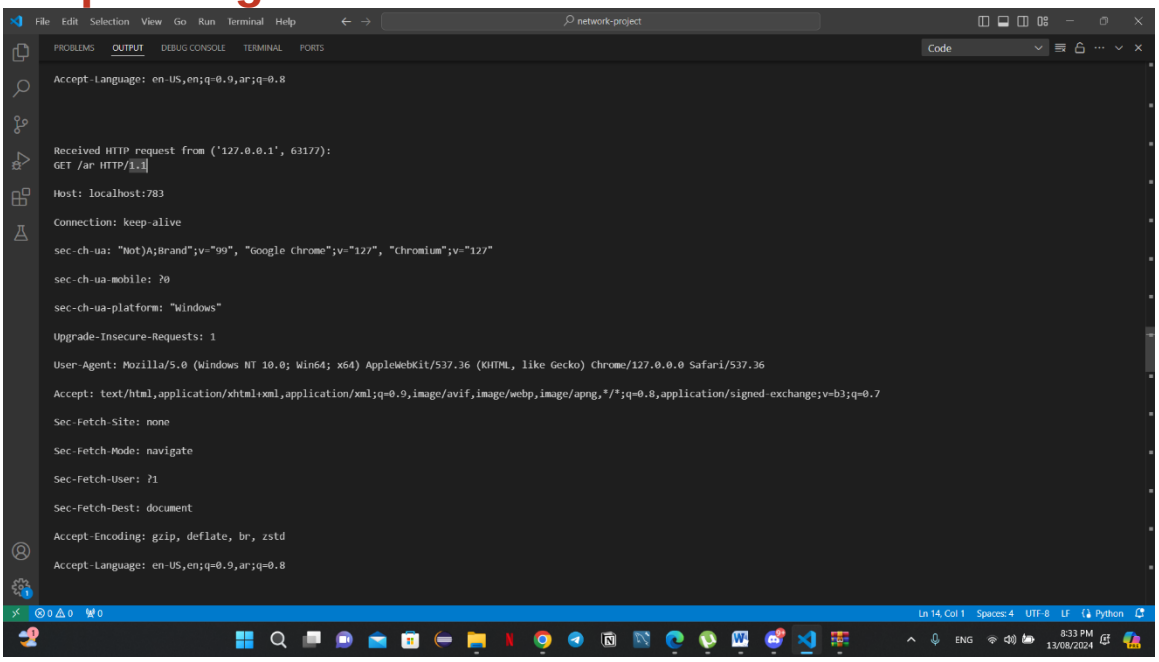


Figure 71 request for server to get Arabic HTML file

Requesting Stack OverFolw:

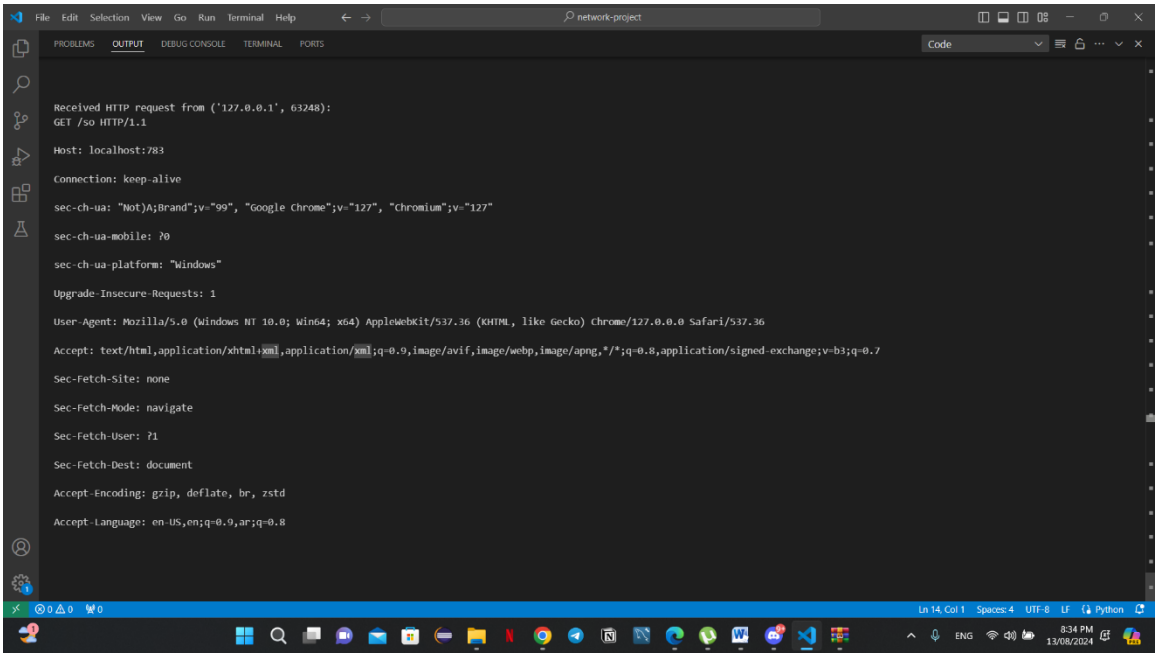


Figure 72 request for server to redirect to Stack Overflow

Requesting our local website:

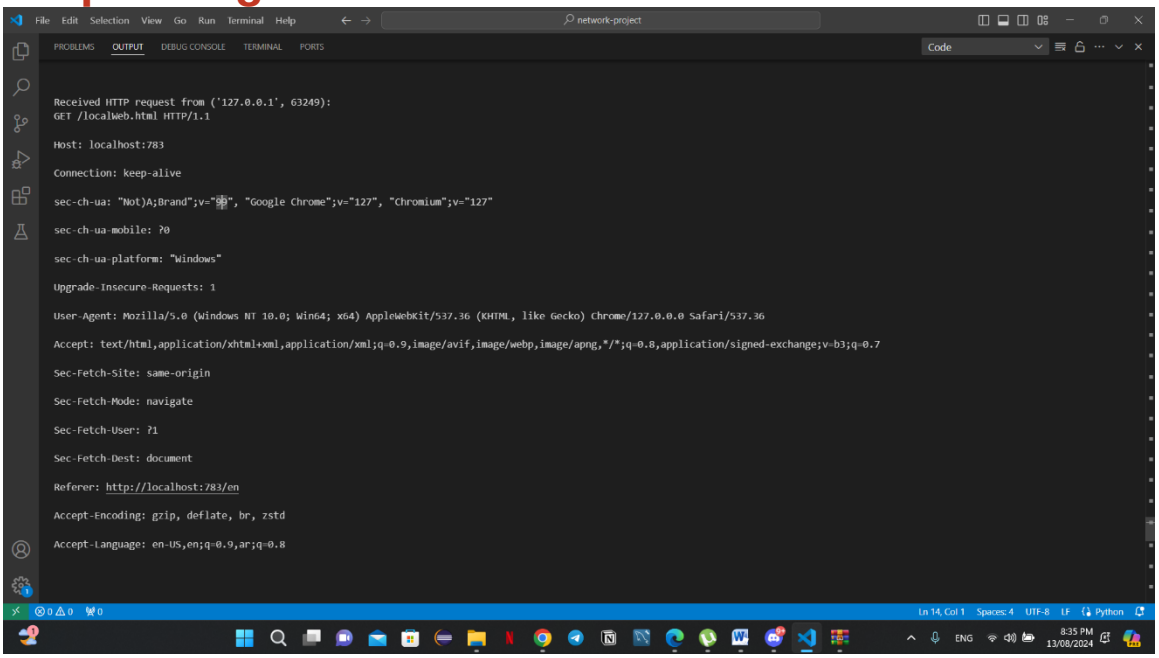


Figure 73 request for server to get our local website

Requesting an Image form our local website:

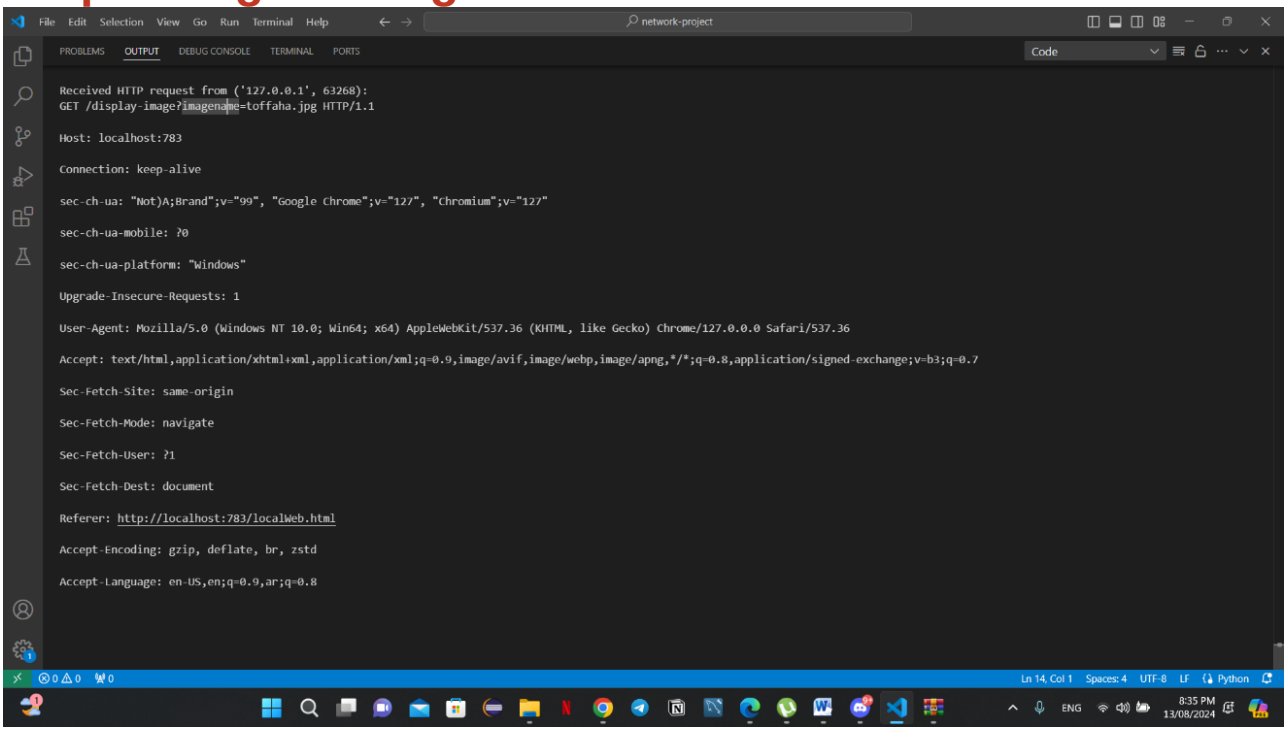


Figure 74 request for a server from our website to get an existing image from the server

Requesting a not exist Image form our local website:

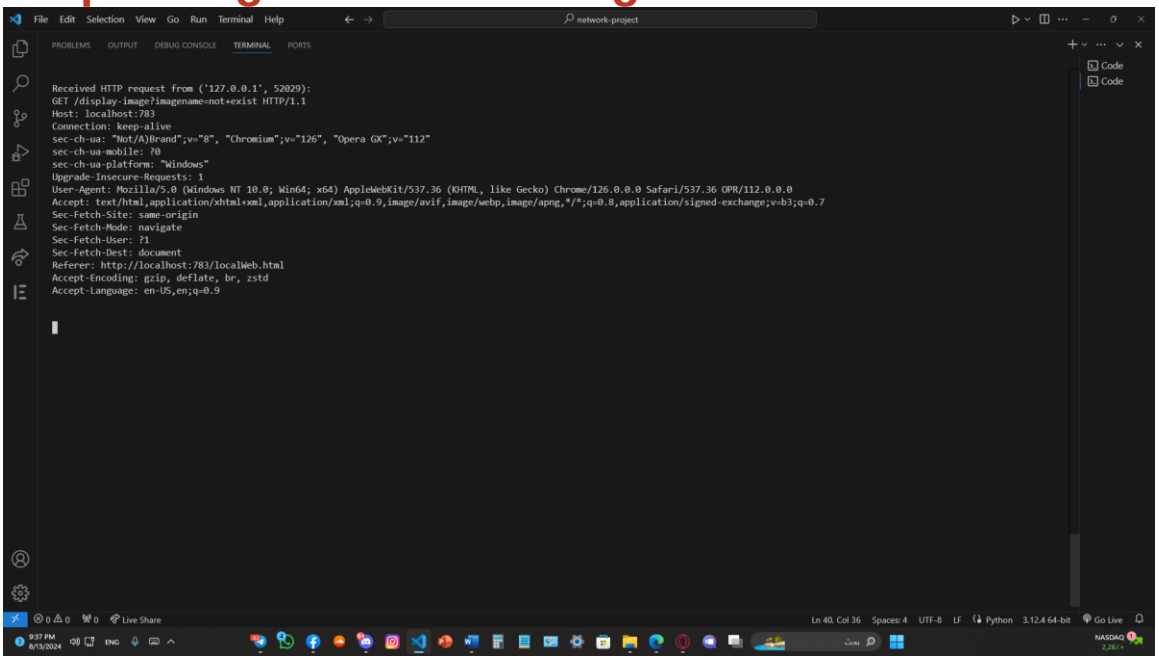
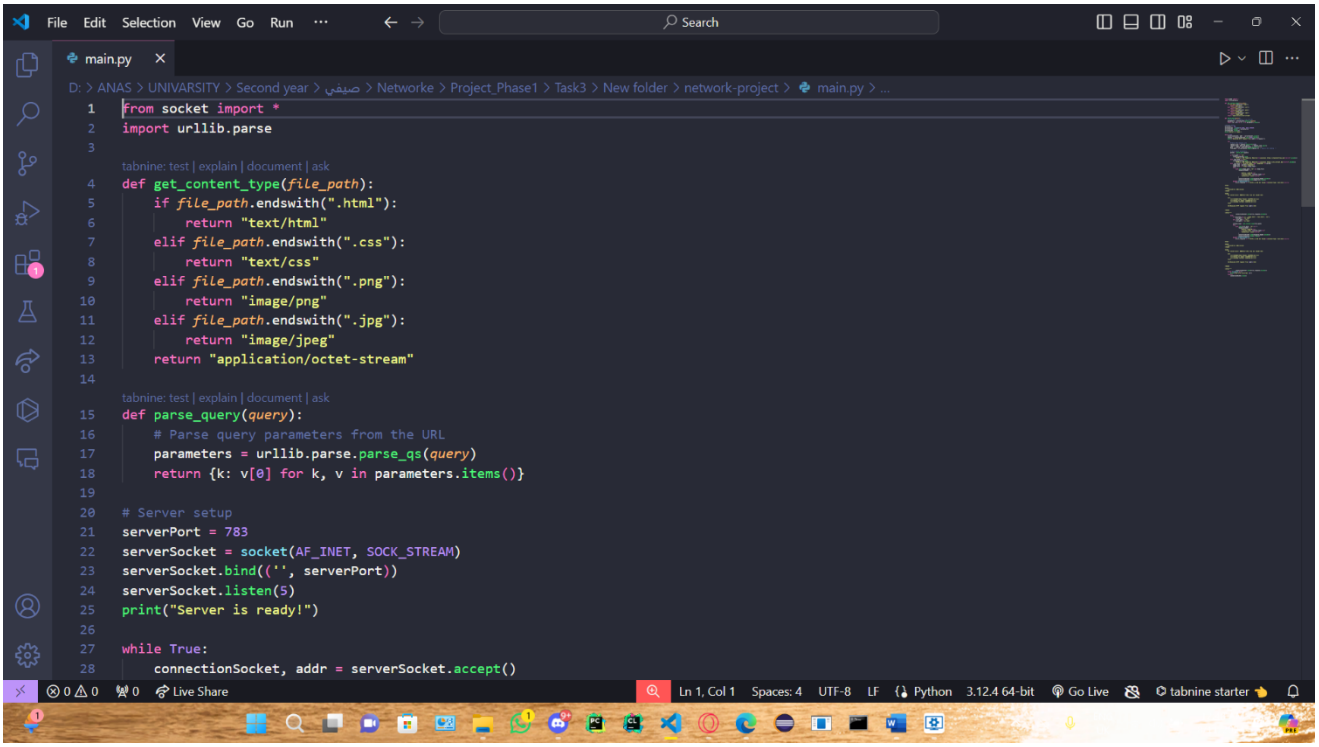


Figure 75 request for a server from our website to get a not existing image from the server

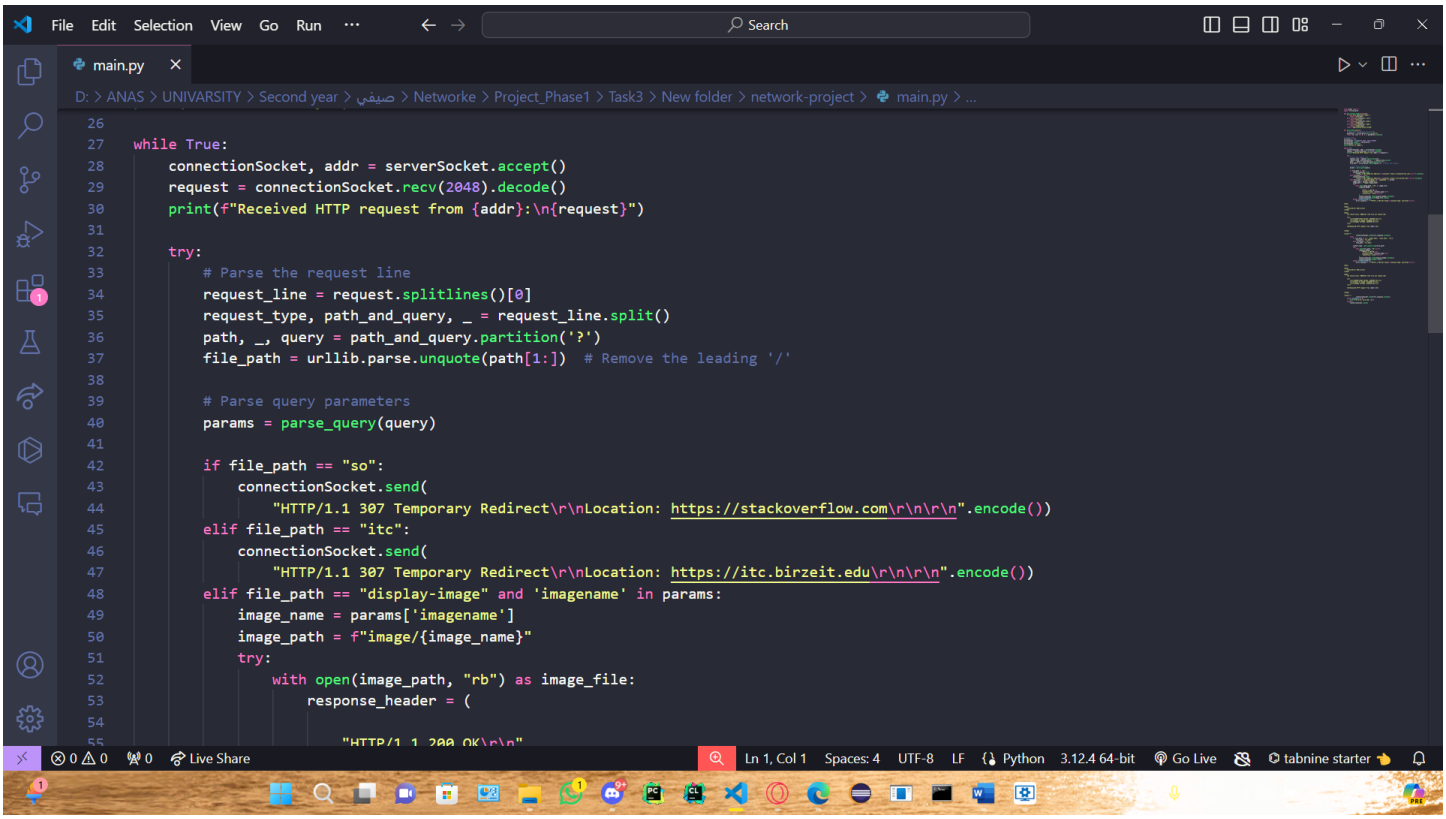
Code Screenshots:

➔ *The TCP main:*



```python
from socket import *
import urllib.parse

def get_content_type(file_path):
    if file_path.endswith(".html"):
        return "text/html"
    elif file_path.endswith(".css"):
        return "text/css"
    elif file_path.endswith(".png"):
        return "image/png"
    elif file_path.endswith(".jpg"):
        return "image/jpeg"
    return "application/octet-stream"

def parse_query(query):
    # Parse query parameters from the URL
    parameters = urllib.parse.parse_qs(query)
    return {k: v[0] for k, v in parameters.items()}

# Server setup
serverPort = 783
serverSocket = socket(AF_INET, SOCK_STREAM)
serverSocket.bind(('', serverPort))
serverSocket.listen(5)
print("Server is ready!")

while True:
    connectionSocket, addr = serverSocket.accept()
```

Figure 76 Code of TCP main



```python
while True:
    connectionSocket, addr = serverSocket.accept()
    request = connectionSocket.recv(2048).decode()
    print(f"Received HTTP request from {addr}:\n{request}")

    try:
        # Parse the request line
        request_line = request.splitlines()[0]
        request_type, path_and_query, _ = request_line.split()
        path, _, query = path_and_query.partition('?')
        file_path = urllib.parse.unquote(path[1:])  # Remove the leading '/'

        # Parse query parameters
        params = parse_query(query)

        if file_path == "so":
            connectionSocket.send(
                "HTTP/1.1 307 Temporary Redirect\r\nLocation: https://stackoverflow.com\r\n\r\n".encode())
        elif file_path == "itc":
            connectionSocket.send(
                "HTTP/1.1 307 Temporary Redirect\r\nLocation: https://itc.birzeit.edu\r\n\r\n".encode())
        elif file_path == "display-image" and 'imagename' in params:
            image_name = params['imagename']
            image_path = f"image/{image_name}"
            try:
                with open(image_path, "rb") as image_file:
                    response_header = (
```

Figure 77 Code of TCP main

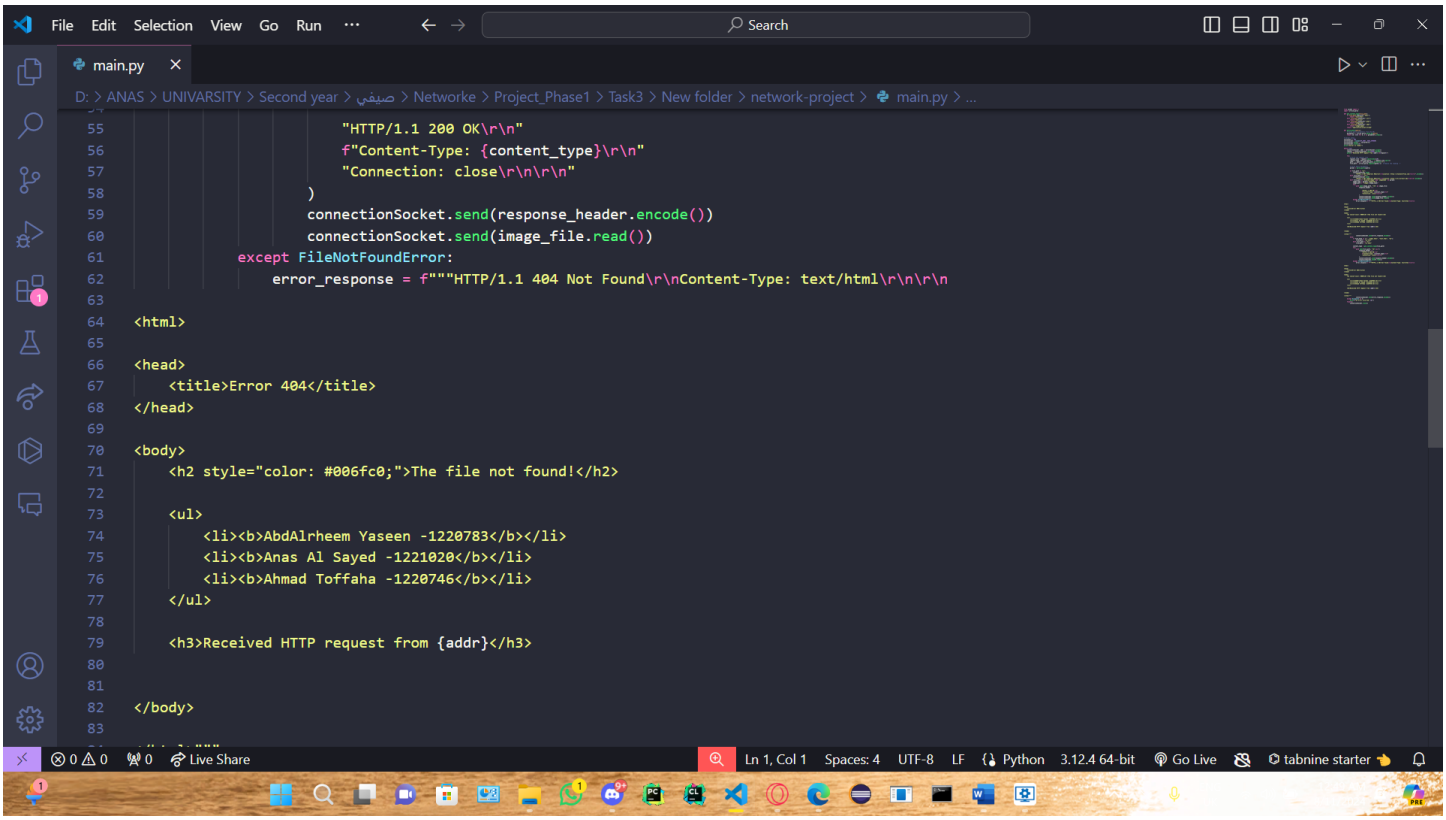


Figure 78 Code of TCP main

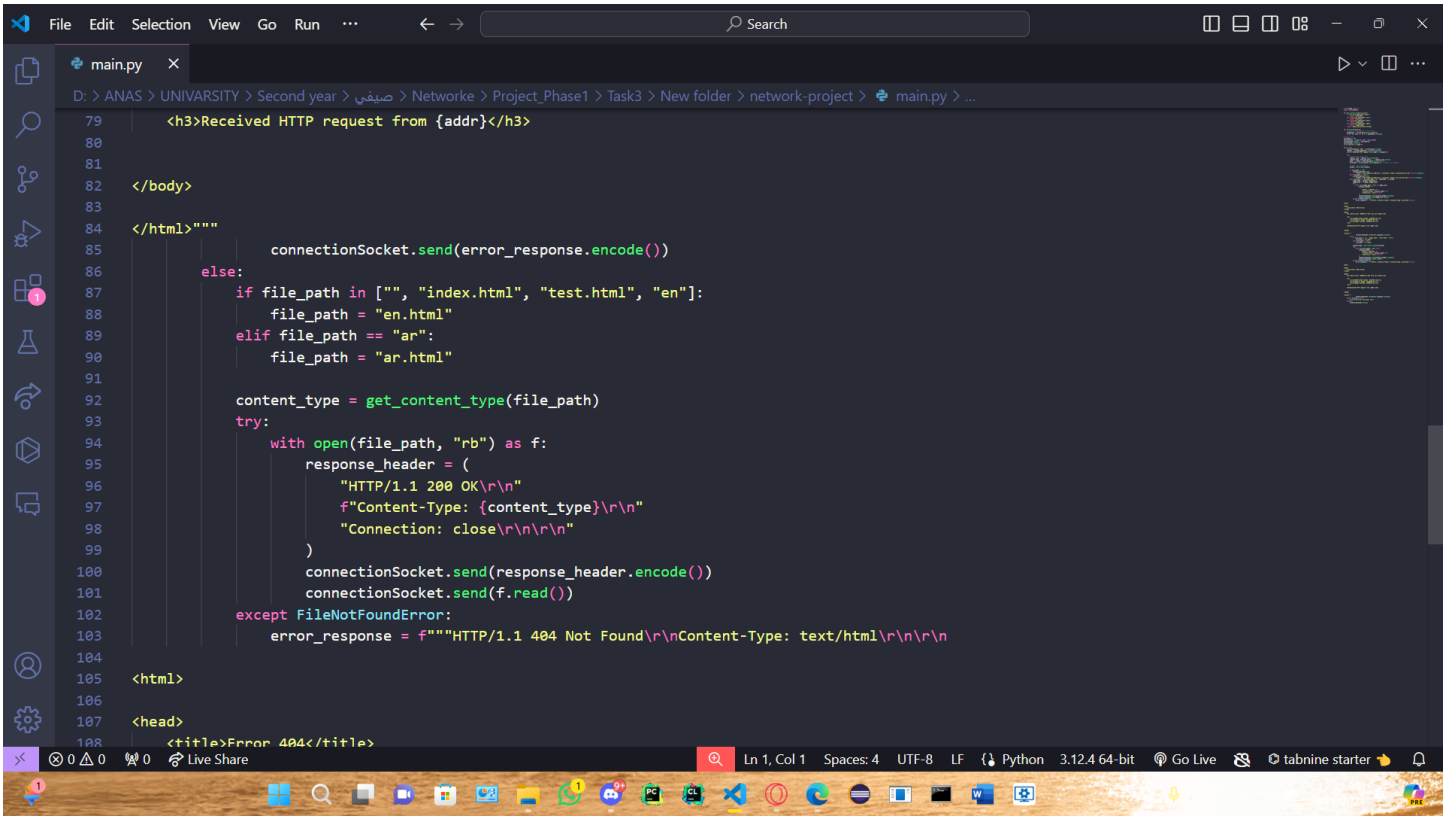


Figure 79 Code of TCP main

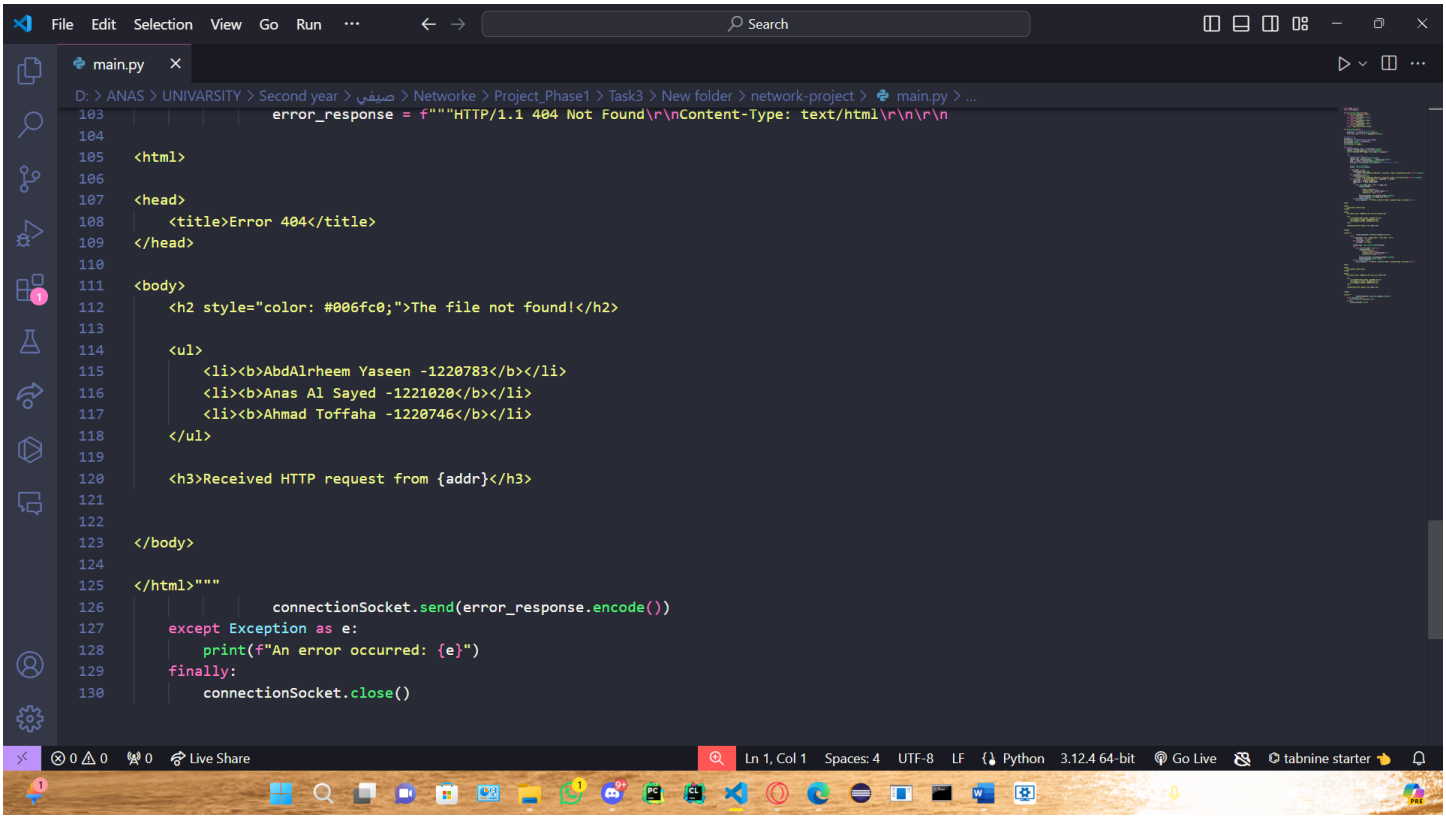


Figure 80 Code of TCP main

The server code sets up a TCP server that listens for connections on a specified port (1020). When a client connects, the server receives a string from the client. It then processes this string by replacing

The Html file:

➔ LOCAL WEB HTML:

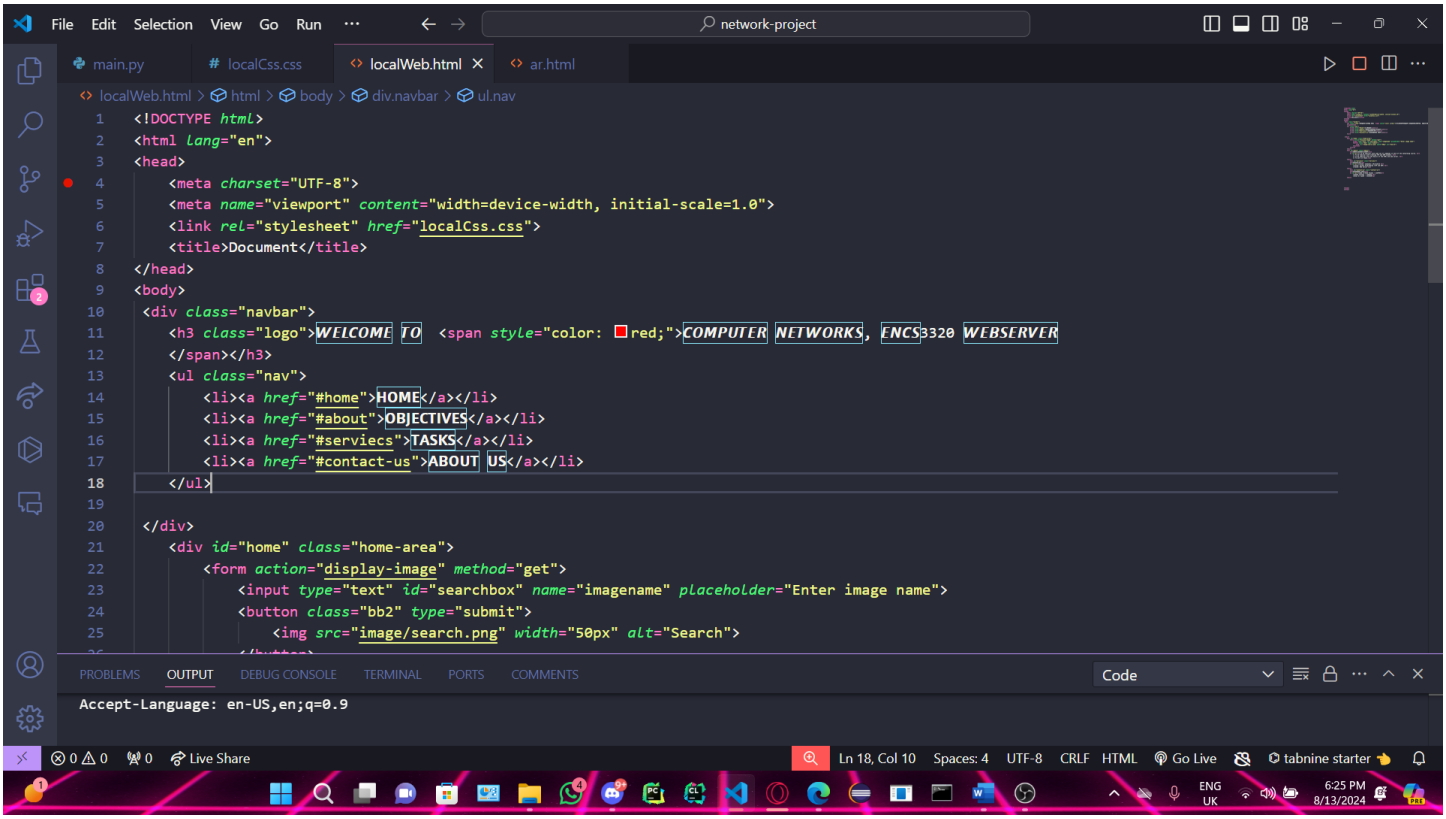


Figure 81: The local web HTML

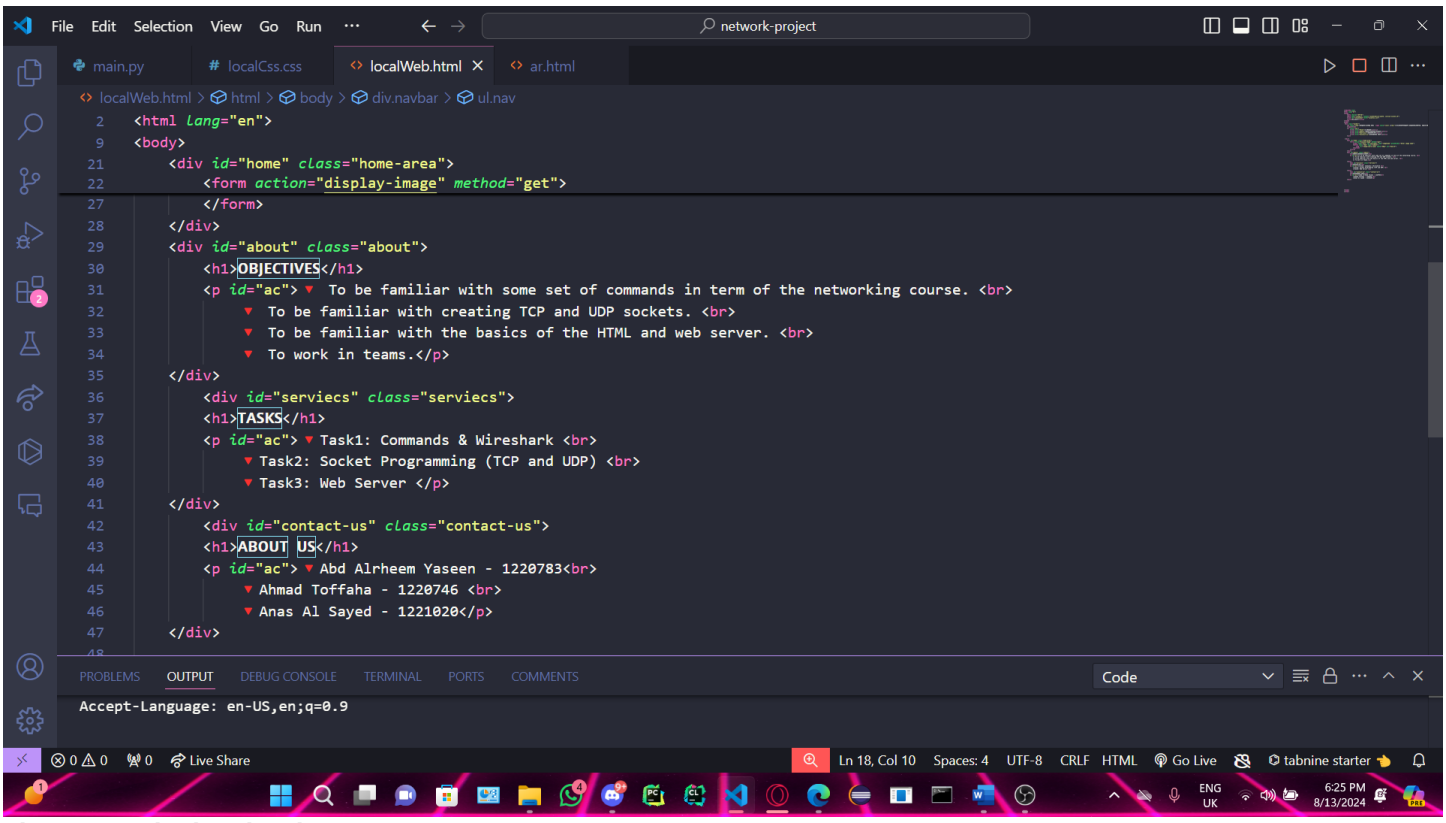


Figure 82: The local web HTML

➔ ***The en.html HTML:***

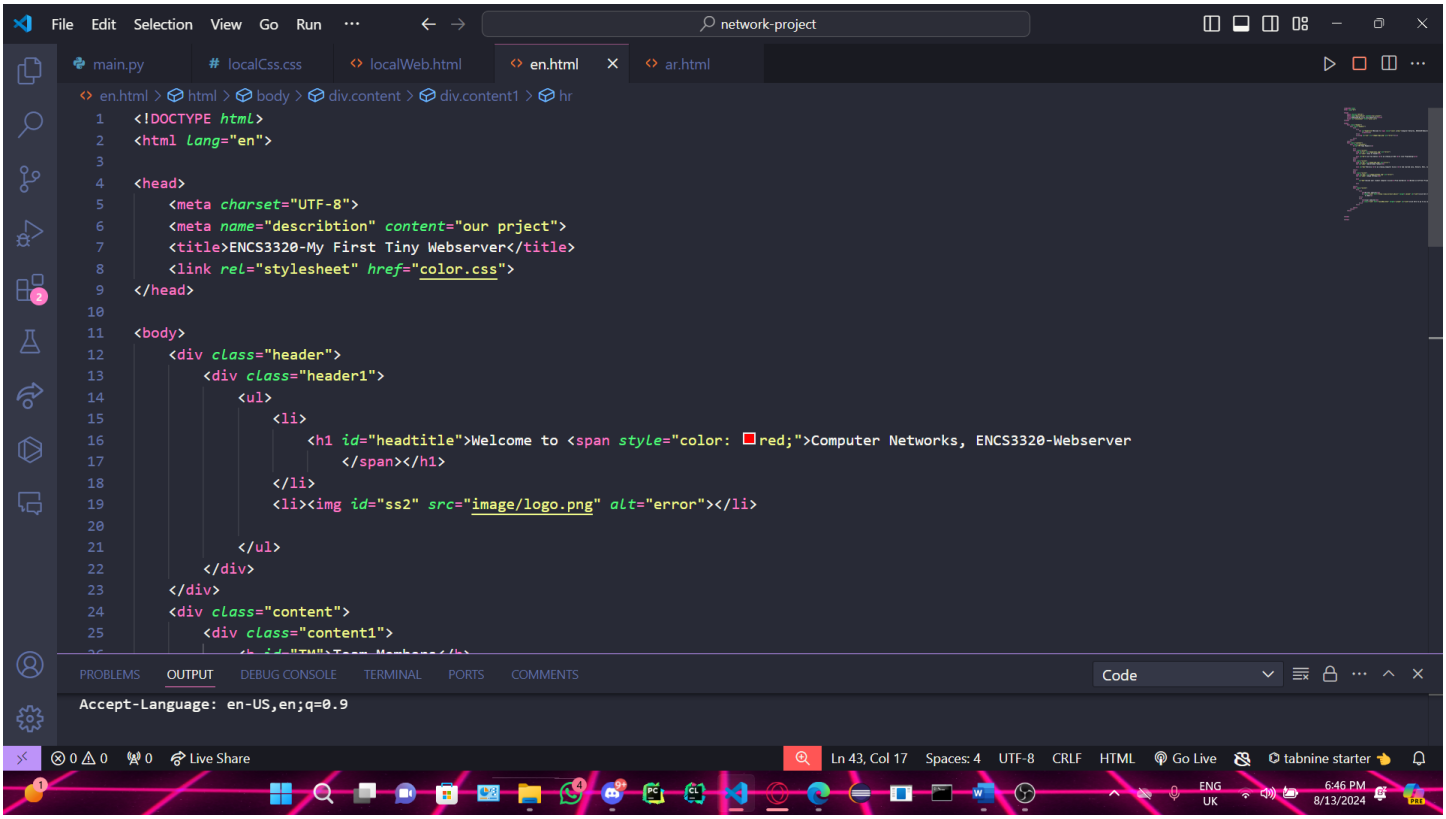


Figure 83:The en.html code

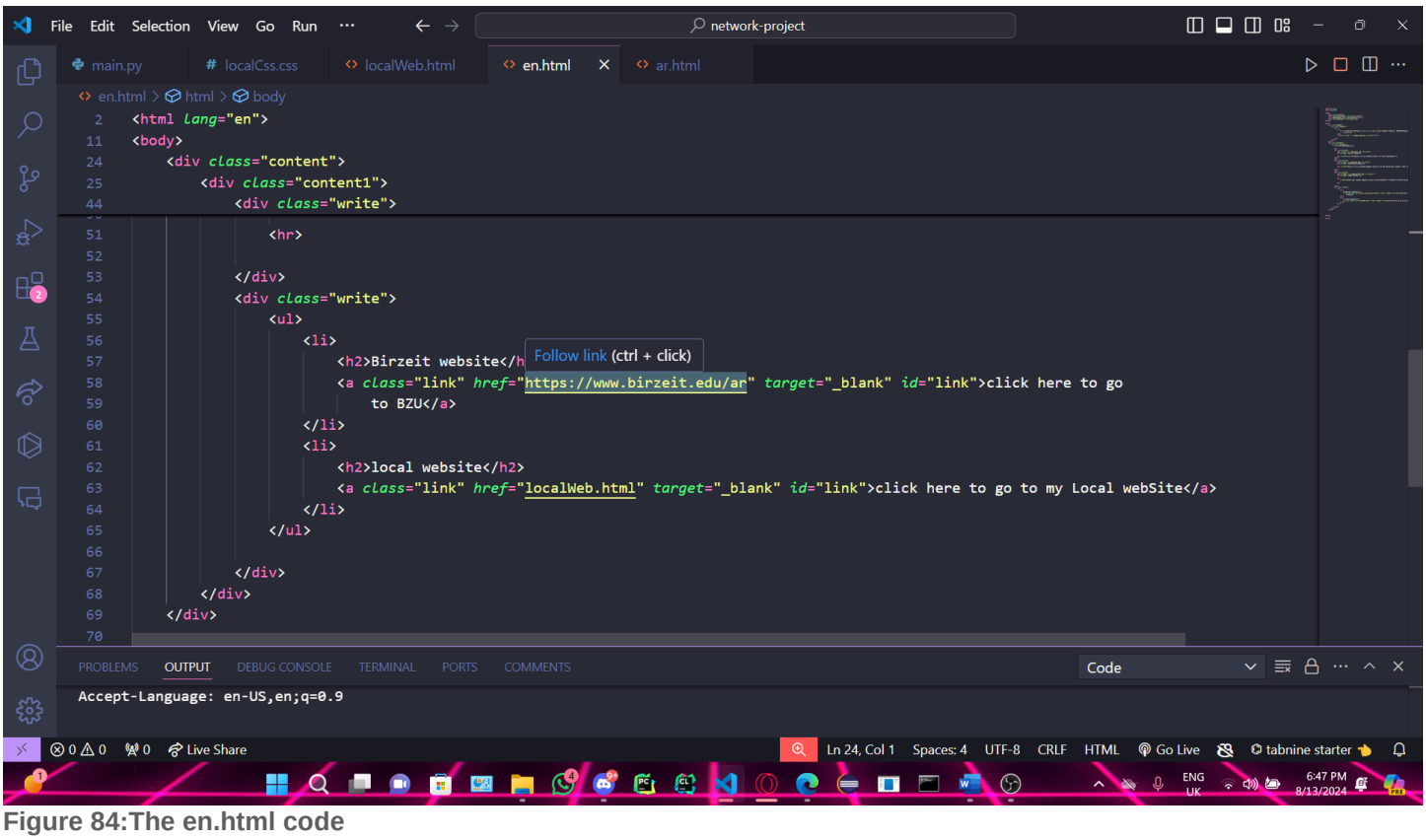


Figure 84:The en.html code

➔ ***The ar.html:***

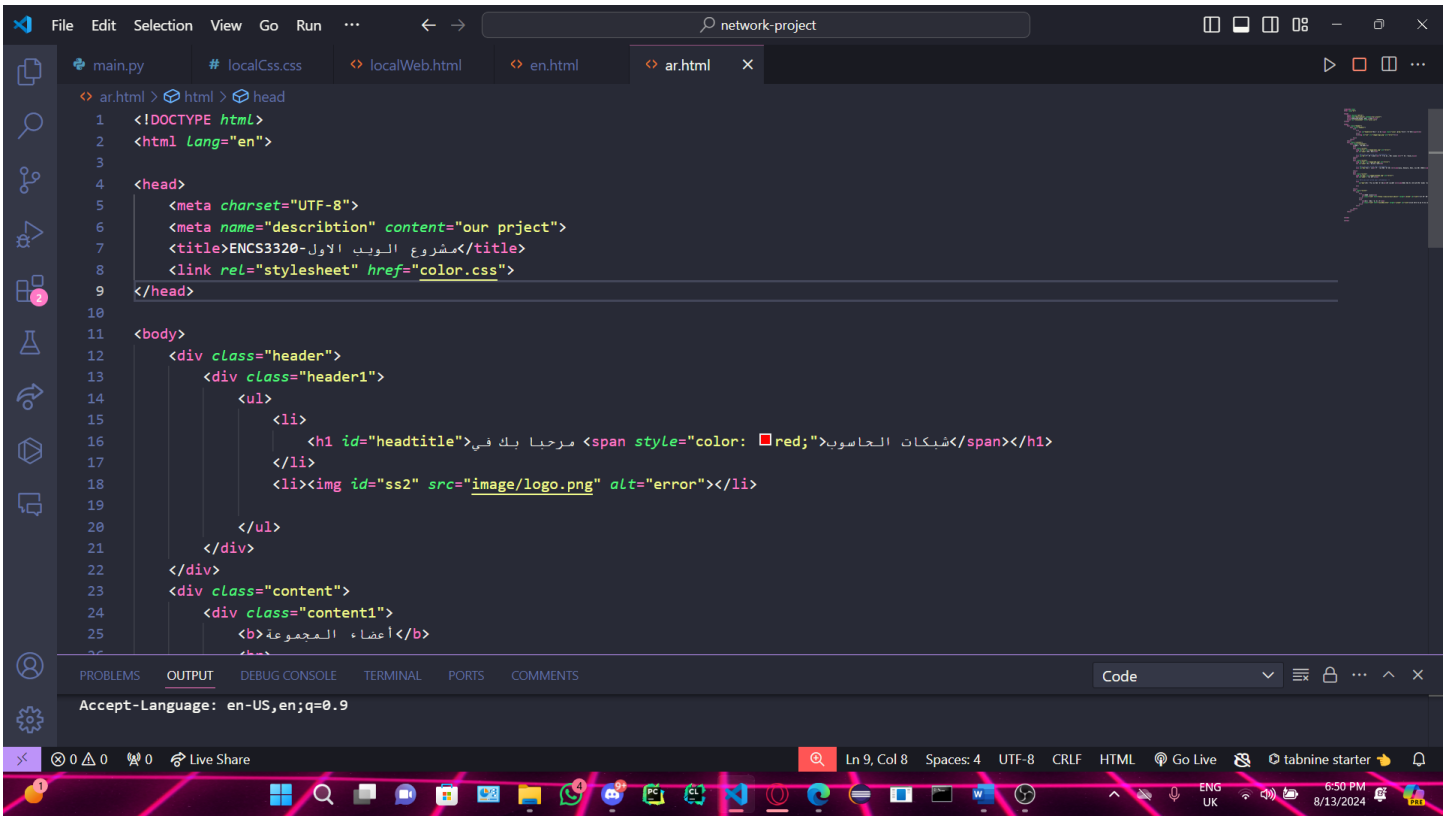


Figure 85:The ar.html code.

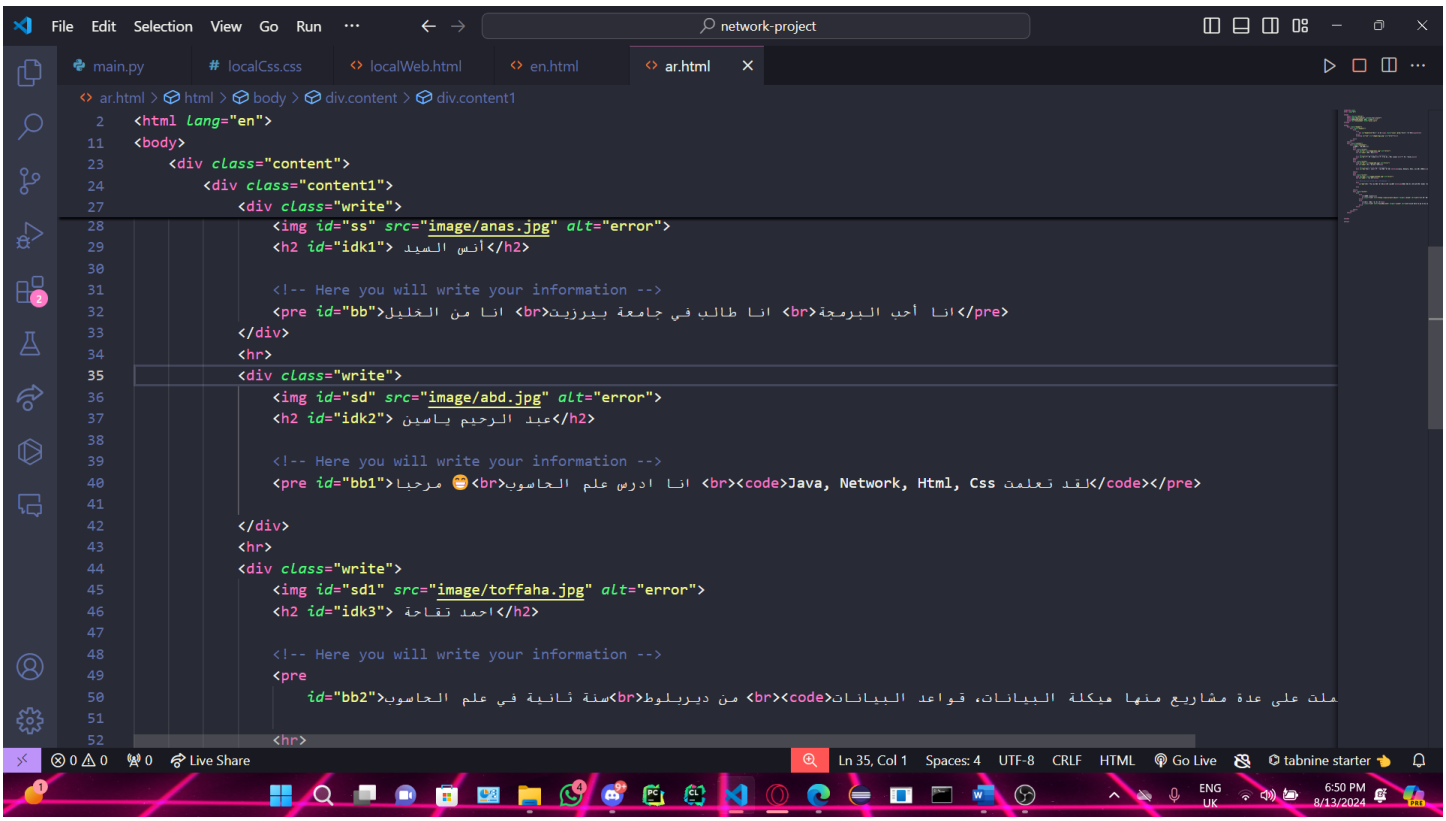


Figure 86The ar.html code.

All the html file that we use to build our project included the 2 different website (main_en.html , en ,ar) we put the base components of our website , the search and the list of image also we do some effect and we connect with the "CSS" file for each file to be more attractive.

The Css file:

➔ *The color.css :*

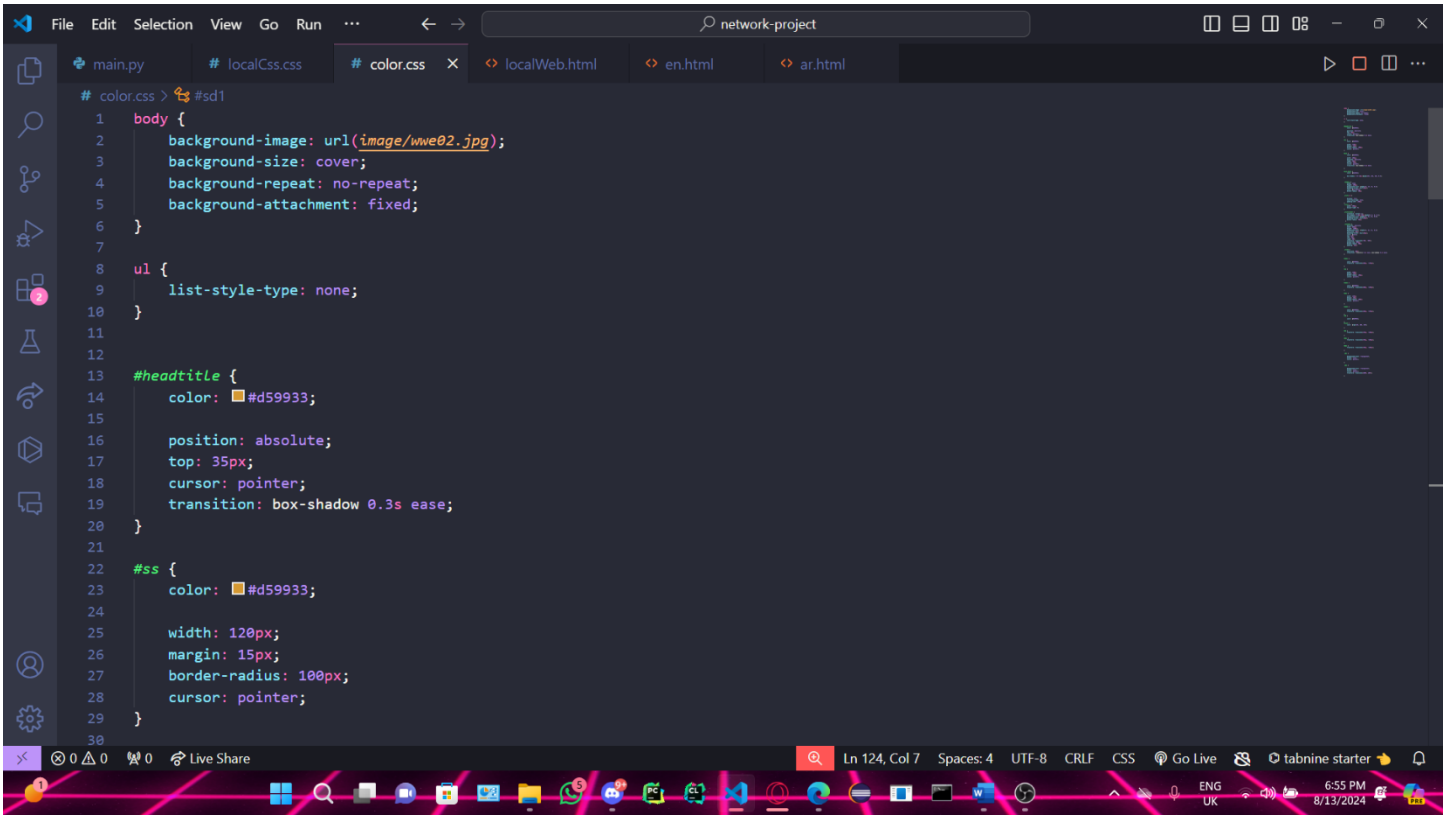


Figure 87:color.css file

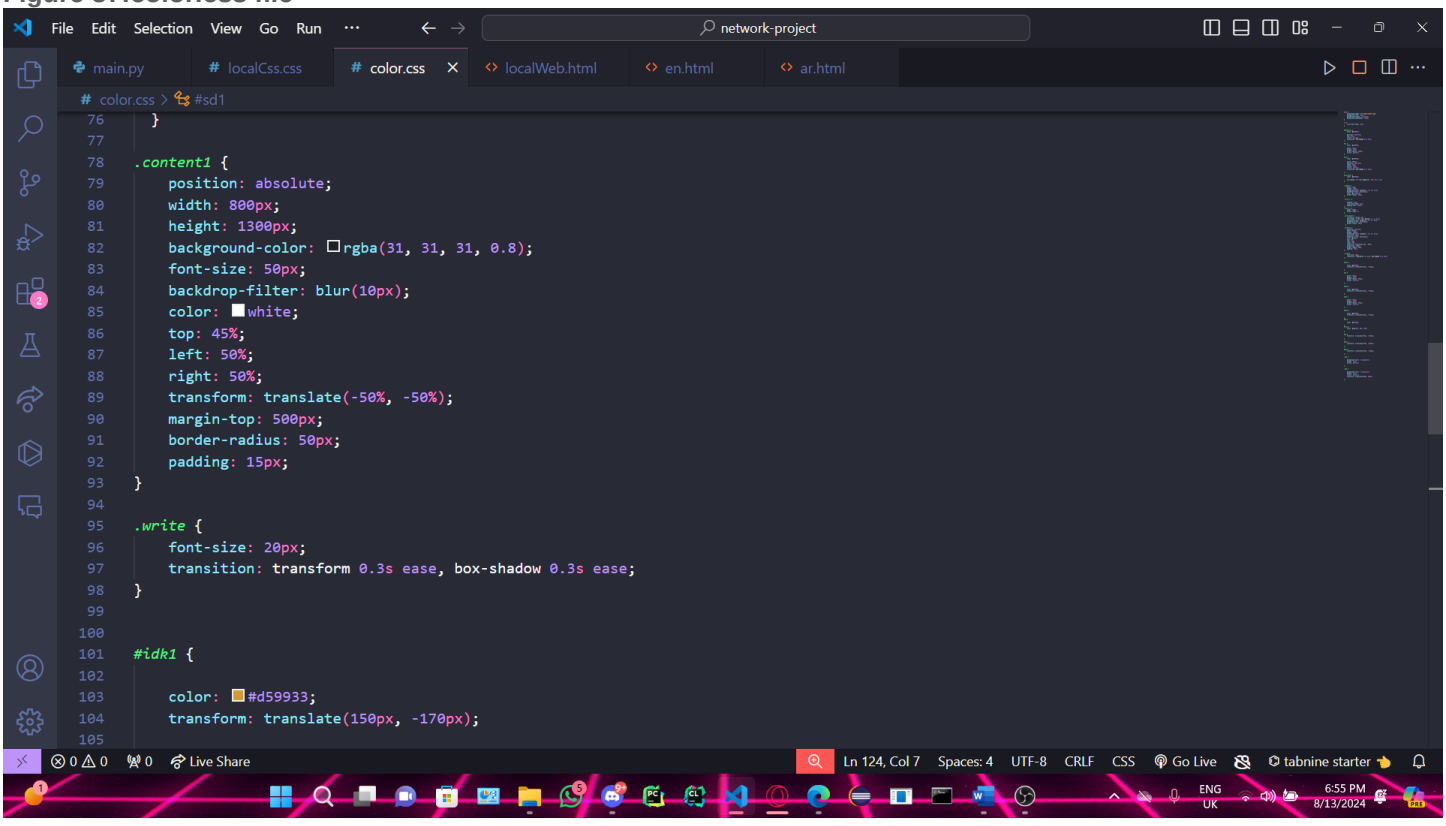


Figure 88 :color.css file

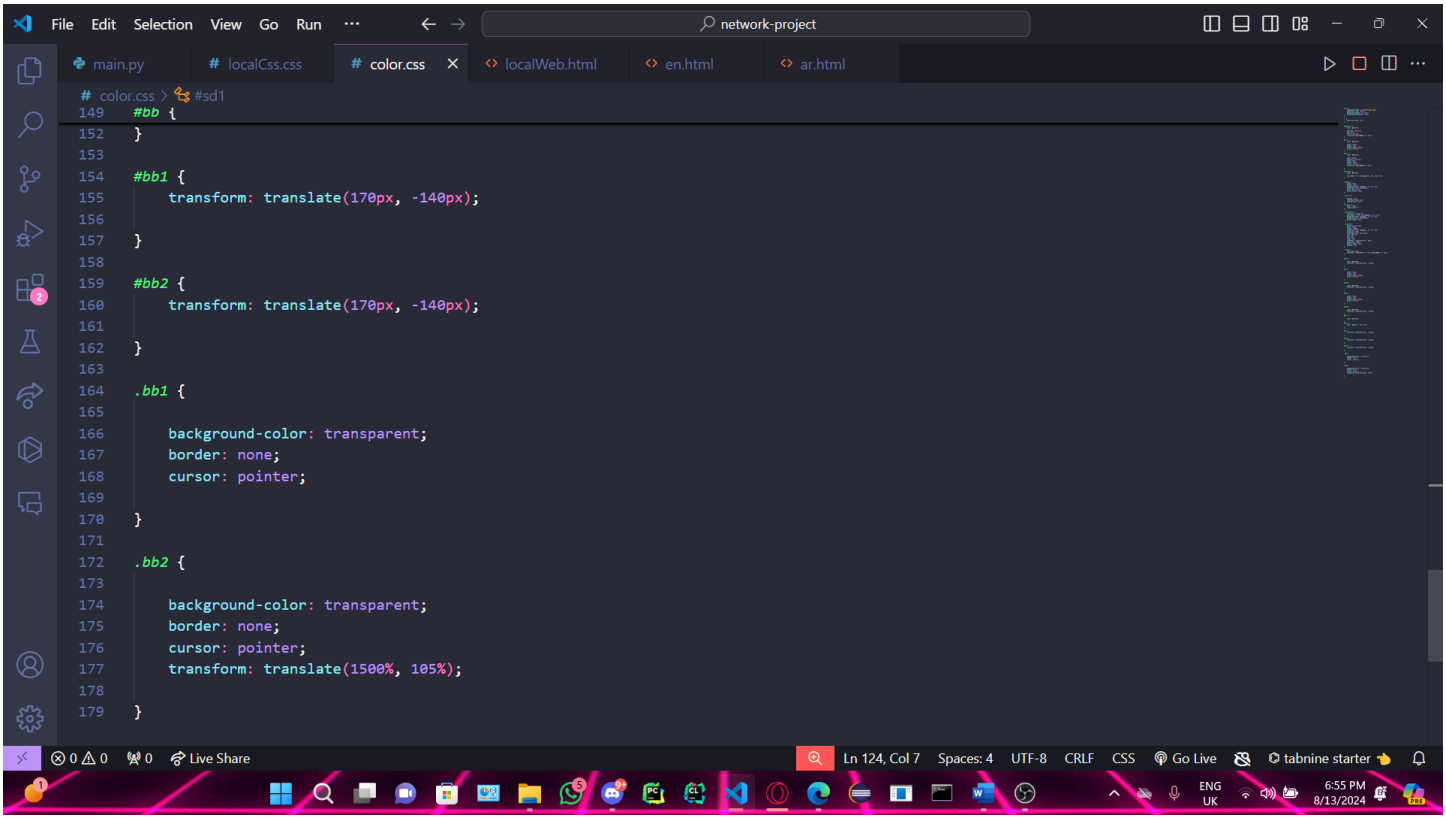


Figure 89:color.css file

➔ *The localCss.css :*

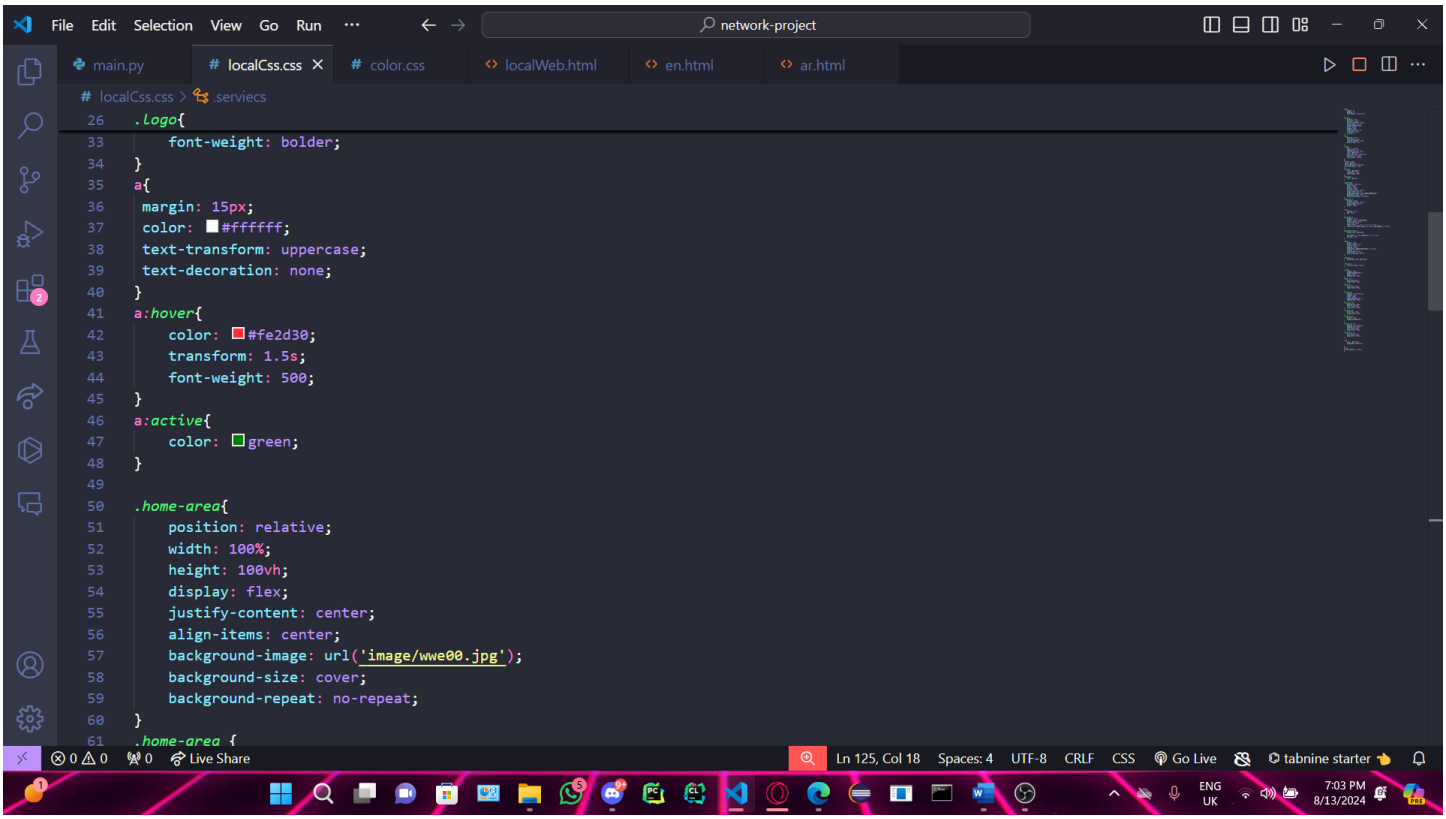


Figure 90:localCss code.

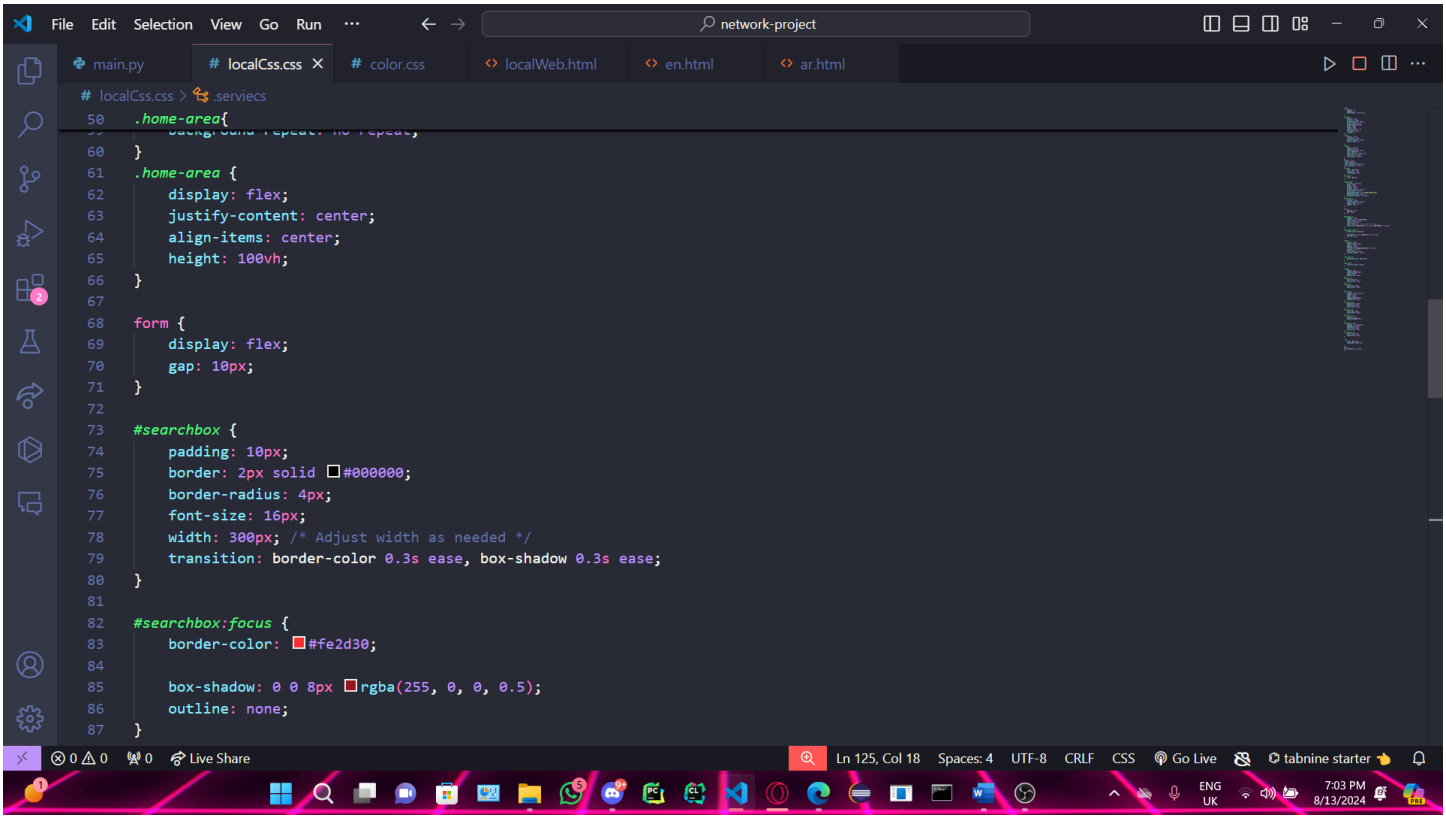


Figure 91:localCss code.

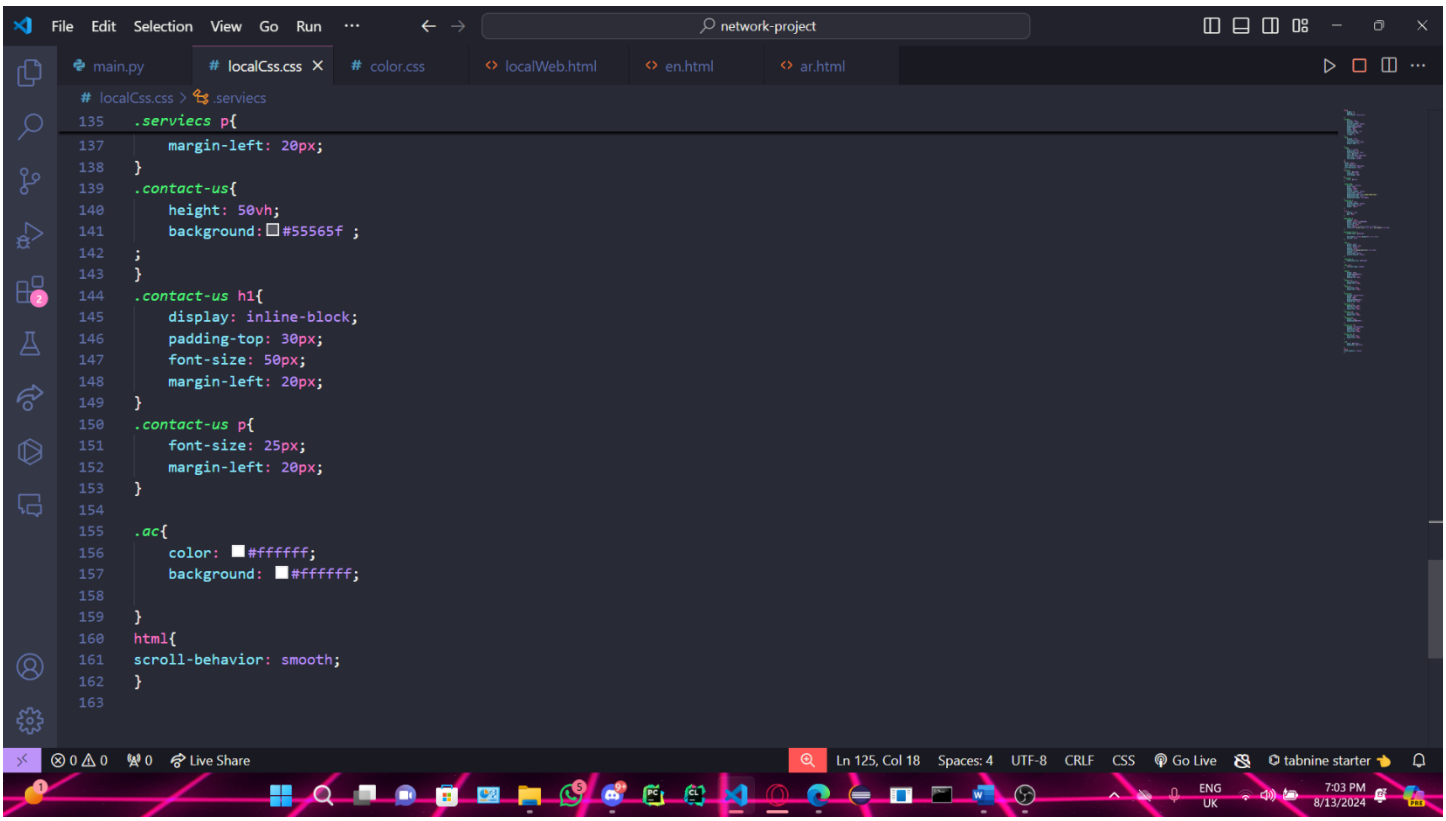


Figure 92:localCss code.

The figure from (47-50) show the code for the color.css file which is liked into the (main_en.html , en ,ar) html files and the figure from(50-53) show the code for local.css which is use to our local web .

TABLE OF FIGURE

Figure 1 use ping www.google.com 5
Figure 2 use tracert www.google.com 6
Figure 3 use nslookup www.example.com 7
Figure 4 telnet the.libraryat.whatis.edu 8
Figure 5 shows ping IP address 192.168.3.133. 9
Figure 6 the domain name " www.ox.ac.uk.". 10
Figure 7 location of the Ip address using Ip geolocation 11
Figure 8 shows a tracing route to "www.ox.ac.uk". 12
Figure 9 talking to is " myhome.mynet". 13
Figure 10 Enter port 80 for telnet 14
Figure 11 error 400 page not found 16
Figure 12 Home Page of wireshark 18
Figure 13. Start Capturing Traffic 19
Figure 14 Before applying a Filter 19
Figure 15 After applying a Filter 20
Figure 16 Stop Capturing 20
Figure 17 Inspect DNS Packets 21
Figure 18 DNS Message Analysis 21
Figure 19: Save Capture 22
Figure 20 Capture file extinction 22
Figure 21 Code of TCP Server 23
Figure 22 Code of TCP Client 24
Figure 23 Run the server python code form CMD 25
Figure 24 Run client python code form CMD 25
Figure 25 UDP client code 27
Figure 26 UDP server code 27
Figure 27 The server listens on port 28
Figure 28 The client sends a message to the server on port 746 28
Figure 29 The server receives messages from clint 1 29
Figure 30 Clint 1 messages from The server 29
Figure 31 Client 2 sends a message to the server on port 746 29
Figure 32:The server shows a list for all received massage. 31
Figure 33: Project Screenrecord: 32
Figure 34 regular request localhost:783 33
Figure 35 Default website 33
Figure 36 request for /en 34
Figure 37 the English version /en 34
Figure 38 request for /ar 35
Figure 39 the Arabic version /er 35
Figure 40 request for /index.html "default" 36
Figure 41 /index.html is shown "default" 36
Figure 42 request for /so 37
Figure 43 stack over flow website shown 37
Figure 44 request for /itc 38
Figure 45 itc website 38
Figure 46 Birzeit link at the end of the website 39
Figure 47 Birzeit website shown. 39
Figure 48 our website link at the end of the page 40
Figure 49 our website shown. 40
Figure 50 Search for anas.jpg. 41
Figure 51 jpg photo shown. 41
Figure 52 Search for logo.png. 42
Figure 53 Logo photo is shown. 42

Figure 54 search for logo.jpg which is not found 43
Figure 55 Error massage is shown. 43
Figure 60 Entering Ip address of server with port# for arabic version 45
Figure 59 Accessing our website 45
Figure 62 Entering Ip address of server with port# for StackOverFlow 45
Figure 61 Accessing our arabic version of website 45
Figure 66 Accessing the image 46
Figure 65 Searching for abd.jpg 46
Figure 69 http response 48
Figure 70 Empty terminal 48
Figure 71 request for server to get Html file 49
Figure 72 request for server to get CSS file 49
Figure 73 request for server to get logo image 49
Figure 74 request for server to get images 50
Figure 75 request for server to get Arabic HTML file 50
Figure 76 request for server to redirect to Stack Overflow 51
Figure 77 request for server to get our local website 51
Figure 78 request for a server from our website to get an existing image from the server 52
Figure 79 request for a server from our website to get a not existing image from the server 52
Figure 80 Code of TCP main 53
Figure 81 Code of TCP main 53
Figure 82 Code of TCP main 54
Figure 83 Code of TCP main 54
Figure 84 Code of TCP main 55
Figure 85: The local web HTML 56
Figure 86: The local web HTML 56
Figure 87:The en.html code 57
Figure 88:The en.html code 57
Figure 89:The ar.html code. 58
Figure 90The ar.html code. 58
Figure 91:color.css file 59
Figure 92 :color.css file 59
Figure 93:color.css file 60
Figure 94:localCss code. 60
Figure 95:localCss code. 61
Figure 96:localCss code. 61

REFERENCES:

https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/ping

https://en.wikipedia.org/wiki/Ping_(networking_utility)

https://www.ibm.com/docs/sl/aix/7.1?topic=t-traceroute-command

https://www.iplocation.net/ip-lookup

https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/telnet

https://www.wireshark.org/

https://www.geeksforgeeks.org/socket-programming-python/

https://www.w3schools.com/html/

https://www.w3schools.com/css/

https://www.whatsmyip.org/

TEAMWORK BAR: