

**Faculty of Engineering & Technology  
Electrical & Computer Engineering Department**

**ENCS3320-Computer Networks**

**Project#1**

**Socket Programming**

---

**Prepared by:**

Hala Mohammaed	1210312
Diana Naseer	1210363
Rua Srour	1221727

**Section:**

Rua & Diana 1  
Hala 2

**Instructor:**

Dr. Ibrahim Nemer

**Date:**

15/8/2024

# Abstract

The aim of this project is to be familiar with some set of commands in terms of the networking course, with creating TCP and UDP sockets, basics of the HTML and web server and to work in teams. In order that the report will be divided into three parts, the first part is about Commands and Wireshark, the second one about socket Programming (TCP and UDP) Finally the third one about Web Server. First we will start with the command and Wireshark, the commands that we will take about them are ping, tracert, nslookup, and telnet. In addition we will use WireShark to capture some messages. Second, implement socket programming for the purpose of having client-server communication that uses TCP and UDP. TCP ensures that data is transferred reliably but it does this by being connection-oriented even as faster connectionless communication is provided in UDP. In developing and testing simple network applications, this project shows the major distinctions and uses of these protocols.

# Table of Contents

<b>ABSTRACT</b> .....	I
<b>TABLE OF CONTENTS</b> .....	II
<b>LIST OF FIGURES:</b> .....	IV
<i>Teamwork:</i> .....	vii
<b>PART ONE COMMANDS AND WIRESHARK:</b> .....	1
<b>THEORY PART 1:</b> .....	1
<b>PROCEDURE PART 1:</b> .....	3
<i>ping a device in the same network, e.g. from a laptop to a smartphone.</i> .....	3
1.1.2.2 PING WWW.OX.AC.UK. ....	4
1.1.2.3 TRACERT WWW.OX.AC.UK .....	5
1.1.2.4 NSLOOKUP WWW.OX.AC.UK. ....	6
1.1.2.5 :TELNET WWW.OX.AC.UK.....	8
1.1.2.6 GIVING SOME DETAILS ABOUT AUTONOMOUS SYSTEM (AS) NUMBER, NUMBER OF IPs, PREFIXES, PEERS, .....	9
NAME OF TIER1-ISP OF WWW.OX.AC.UK. ....	9
1.1.2.7 USING WIRESHARK TO CAPTURE SOME DNS MESSAGES.....	11
<b>RESULTS AND DISCUSSIONS PART 1:</b> .....	12
1.1: PART ONE COMMANDS AND WIRESHARK:.....	12
1.2.1 THE RESULT OF PING A SMARTPHONE WITH IP = 192.168.88.2 FROM A LAPTOP AT THE SAME NETWORK: .....	12
1.2.2 THE RESULT OF PING WWW.OX.AC.UK. ....	13
1.2.3 THE RESULT OF TRACERT WWW.OX.AC.UK. ....	15
1.2.4 NSLOOKUP WWW.OX.AC.UK.....	16
1.2.6 SOME DETAILS ABOUT WWW.OX.AC.UK.....	18
1.3.1 TELNET IS NOT RECOGNIZED AS INTERNAL OR EXTERNAL COMMAND. ....	20
<b>PART 2: SOCKET PROGRAMMING (TCP AND UDP)</b> .....	23
<b>THEORY:</b> .....	23
<b>PROCEDURE:</b> .....	25
<i>TCP Protocol:</i> .....	25
TCP Client Code: .....	26
TCP Server code: .....	27
▪     UDP PROTOCOL: .....	30
UDP Server Code: .....	31

<i>UDP Client code:</i> .....	32
RESULTS AND DISCUSSIONS: .....	34
<b>3-PART THREE WEB SERVER:.....</b>	<b>42</b>
<b>3.1-THEORY AND PROCEDURE:.....</b>	<b>42</b>
<i>Web Servers</i> .....	42
<i>Socket Programming</i> .....	42
<i>HTTP Protocol and Content Types</i> .....	43
<i>Error Handling</i> .....	43
<i>Redirection with Status Codes</i> .....	43
<i>Localization Support:</i> .....	43
<b>3.2- RESULTS AND DISCUSSIONS: .....</b>	<b>58</b>
<b>REFERENCES: .....</b>	<b>63</b>

# List of Figures:

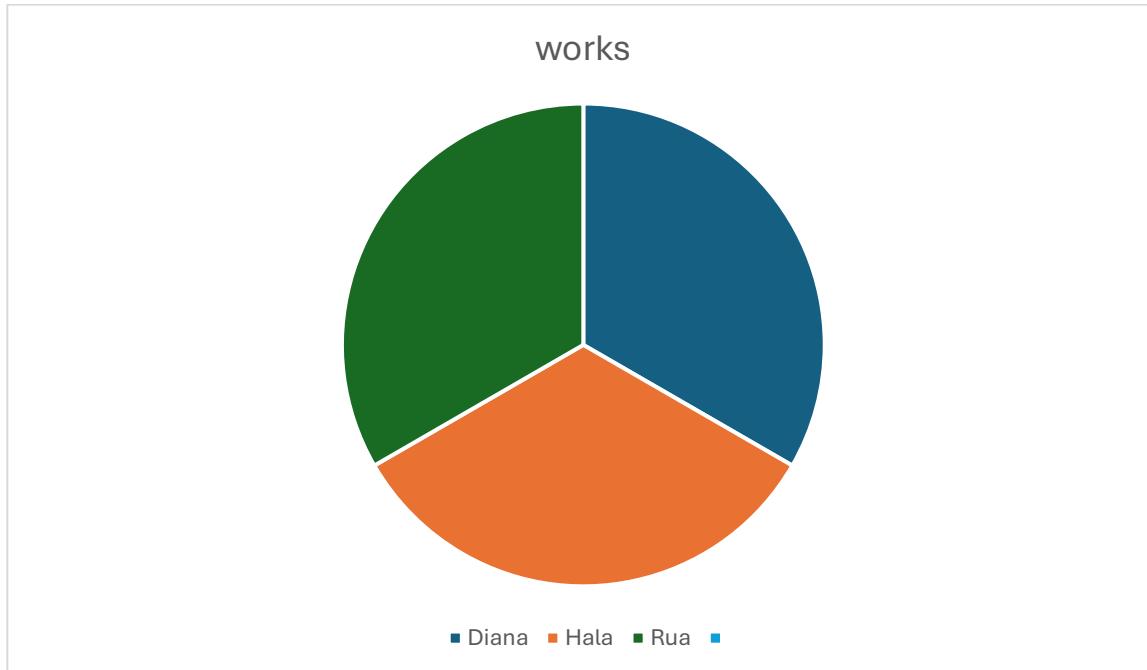
Figure 1:WireShark Interface. (Source: screenshot from my PC) .....	2
Figure 2:WireShark logo. (Source: quora).....	2
Figure 3step one to run a command. (Source: screenshot from my PC).....	3
Figure 4:step tow to run a command. (Source: screenshot from my PC) .....	3
Figure 5:step three to run a command. (Source: screenshot from my PC) .....	4
Figure 6:step three to run a command. (Source: screenshot from my PC) .....	4
Figure 7 step one to run a command. (Source: screenshot from my PC).....	5
Figure 8step tow to run a command. (Source: screenshot from my PC) .....	5
Figure 9step three to run a command. (Source: screenshot from my PC) .....	6
Figure 10 step one to run a command. (Source: screenshot from my PC).....	6
Figure 11step tow to run a command. (Source: screenshot from my PC) .....	7
Figure 12 step three to run a command. (Source: screenshot from my PC) .....	7
Figure 13 step one to run a command. (Source: screenshot from my PC).....	8
Figure 14step tow to run a command. (Source: screenshot from my PC) .....	8
Figure 15step three to run a command. (Source: screenshot from my PC) .....	9
Figure 16:BGPview interface. (Source: screenshot from my PC) .....	10
Figure 17:BGP.tools interface. (Source: screenshot from my PC) .....	10
Figure 18:Wireshark interface. (Source: screenshot from my PC) .....	11
Figure 19:the replies from the address that I tried to ping. (Source: screenshot from my PC).....	12
Figure 20: the replies from ping www.ox.ac.uk.. (Source: screenshot from my PC) .....	13
Figure 21:Location of the server with IP 104.22.48.74. (Source: IP location) .....	14
Figure 22:Location of the server with IP 104.22.48.74. (Source: IP location) .....	14
Figure 23:the replies from tracert www.ox.ac.uk.. (Source: screenshot from my PC)\.....	15
Figure 24:the concept of tracert command. (Source: cbtnuggets) .....	16
Figure 25:the reply from nslookup www.ox.ac.uk. (Source: screenshot from my PC) .....	16
Figure 26:the replies from tracert www.ox.ac.uk.. (Source: screenshot from my PC) .....	17
Figure 27:the replies from telnet www.ox.ac.uk. (Source: screenshot from my PC) .....	17
Figure 28:Information's from Bgp.view about www.ox.ac.uk server. (Source: Bgp.view ).....	18
Figure 29::Information's from Bgp.tool about www.ox.ac.uk server. (Source: Bgp.tool).....	18
Figure 30:chart of www.ox.ac.uk links. (Source: Bgp.tool) .....	19
Figure 31:DNS massages from capturing the Wi-Fi traffic. (Source: wireshark) .....	19
Figure 32:the replies from telnet www.ox.ac.uk. (Source: screenshot from my PC) .....	20

Figure 33:step one. (Source: screenshot from my PC).....	21
Figure 34:step tow. (Source: screenshot from my PC) .....	21
Figure 35:step three. (Source: screenshot from my PC). .....	22
Figure 36:step four. (Source: screenshot from my PC).....	22
Figure 37:State Diagram for Server and Client Model [12].....	23
Figure 38:TCP connections.[14] .....	24
Figure 39:UDP connections. ....	24
Figure 40:Selecte (Server) IP address from my PC. ....	25
Figure 41:TCP_Client Code.....	26
Figure 42: TCP_Server code part1.....	27
Figure 43:TCP_Server code part2.....	28
Figure 44:TCP_Server code part3.....	29
Figure 45:UDP server code part1.....	31
Figure 46:UDP server code part2.....	32
Figure 47:UDP client code. ....	32
Figure 48:run the TCP server.....	34
Figure 49: run the TCP client.....	34
Figure 50:The response in server after sending message form the client. ....	35
Figure 51:The state in client terminal after receiving the message form the server. ....	35
Figure 52:run the UDP server .....	36
Figure 53:run the UDP client. ....	36
Figure 54:run the UPP client and write message to the server.....	37
Figure 55:UDP server when have the message.....	37
Figure 56:response from the server.....	38
Figure 57:UDP client catch the response from server. ....	38
Figure 58:response again and the list of last messages from 1st client.....	39
Figure 59:Another client sends to server.....	40
Figure 60:Response to client2 and list the communication.....	40
Figure 61:addition experiment to check validity of client .....	41
Figure 62::Web Server Architecture .....	42
Figure 63:Part1 of Server code.....	44
Figure 64:Part2 of Server code.....	44
Figure 65:Part 3 of Server code.....	45
Figure 66:Part 5 of Server code.....	45

Figure 67:Handle Request.....	46
Figure 68:run the server .....	46
Figure 69:main_en Page Part1 .....	47
Figure 70:main_en Page Part2 .....	47
Figure 71:main_en Page Part3 .....	48
Figure 72:Local Page Button .....	48
Figure 73:Local Page .....	49
Figure 74:Links .....	49
Figure 75:BZU Page.....	50
Figure 76:main_ar Page Part1 .....	50
Figure 77:main_ar Page Part2 .....	51
Figure 78:main_en Code Part1.....	51
Figure 79:main_en Code Part2.....	52
Figure 80:main_en Code Part3.....	52
Figure 81:The Local Page HTML code .....	54
Figure 82:search for image that is exist (jpg).....	55
Figure 83:Result of search for image that is exist (jpg) .....	55
Figure 84:search for image that is exist (png).....	56
Figure 85:Result of search for image that is exist (png) .....	56
Figure 86:search for image that is not exist .....	57
Figure 87:Result of search for image that is not exist.....	57
Figure 88:Server Result for http://localhost:1727/main_en.html.....	58
Figure 89:Web Server Result2 .....	59
Figure 90:Web Server Result 3 .....	59
Figure 91:Web Server Result 4 .....	60
Figure 92:Error Page .....	61
Figure 93:Try a new Port .....	62
Figure 94:Test web server on another device Part1 .....	63
Figure 95:Test web server on another device Part2 .....	63
Figure 96:Test web server on another device Part3 .....	64
Figure 97::Test web server on another device Part4 .....	65

## Teamwork:

we divide the project into 3 parts, the first one for Rua, the second one for Diana, and the last part for Hala, the project depends on the participation for our parts, and about this report each of member team take her part and work on it, then all of member work on report order.



## Part one Commands and Wireshark:

### Theory part 1:

“A command, in the context of technology and computing, is an instruction given by a user to a computer or software to perform a specific task. It can be a single word, a line of code, or a series of instructions that tell the computer what to do” [1]. In this project we will be specific about four commands which are ping, tracert, nslookup, and telnet. In a short way I will explain each one. First ping is a command that use for testing the availability of the device to reach an IP address for another device or a network by sending several packets to that IP or URL and wait for a response, the response will be either the whole packet with the Round-Trip Time for those packets to return to the device or there will not be a replay. The second one is tracert, it is a command that if it is followed by the IP either destination or domain name will return the information about each loop it done, the loops contain the hoops that has been discovered and the time for each loop. It is used when the device has a connection problem to a specific device or losing a packet to know in which hope the loss or failure happened. The third one is nslookup (name server lookup) it used to return the (IP) address or domain name system (DNS) record for a specific hostname. Finally, telnet commands are used to connect to a remote computer over a network and test if this port is open. It is also useful to determine the issue if you have trouble connecting via IMAO/POP or SMTP. In addition, at the end of this part we will use Wireshark. “Wireshark is a widely used, open-source network analyzer that can capture and display real-time details of network traffic. It is particularly useful for troubleshooting network issues, analyzing network protocols and ensuring network security. Networks must be monitored to ensure smooth operations and security” [2].

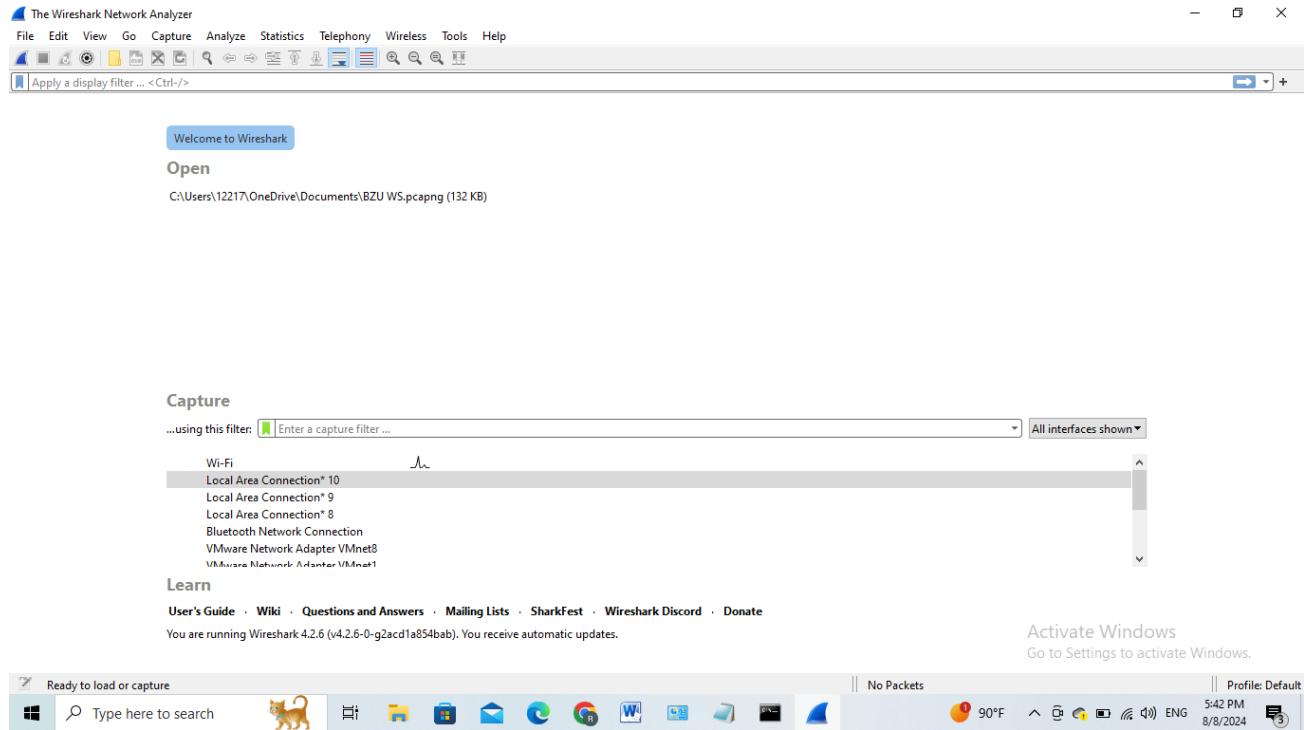


Figure 1:WireShark Interface. (Source: screenshot from my PC).



Figure 2:WireShark logo. (Source: quora).

## Procedure part 1:

First, I made sure that my PC is connected to the internet and then I run the following commands as following:  
ping a device in the same network, e.g. from a laptop to a smartphone.

1. I Clicked Start > Run.

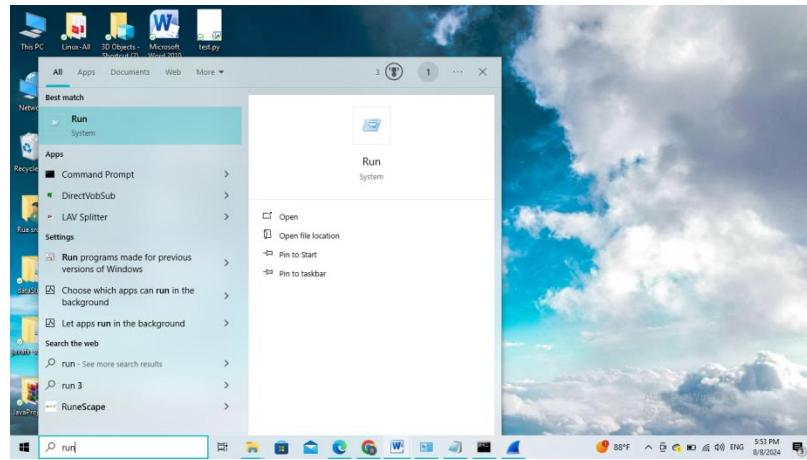


Figure 3: step one to run a command. (Source: screenshot from my PC).

2. In the run box I entered "cmd" > OK.

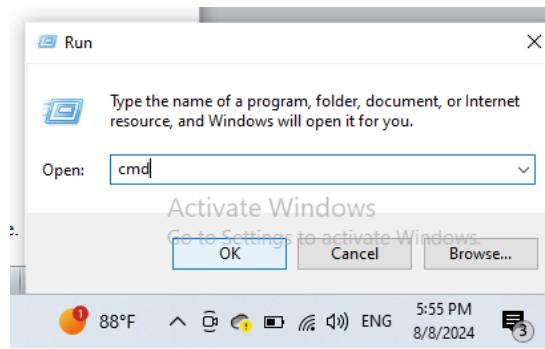


Figure 4: step two to run a command. (Source: screenshot from my PC).

1. In the command prompt I entered "ping IP for the smart phone" without quotes > press ENTER.

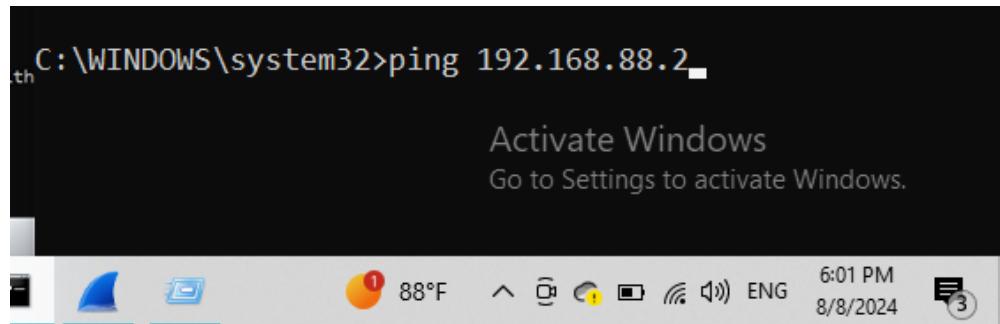


Figure 5:step three to run a command. (Source: screenshot from my PC).

2. Then a set of data that shows server internet speeds will be presented.

#### 1.1.2.2 ping [www.ox.ac.uk](http://www.ox.ac.uk).

I repeated all the steps in 1.1.2.1 but instead of entering “ping IP for the smart phone” In the command prompt, I entered “ping [www.ox.ac.uk](http://www.ox.ac.uk)” and the results will show server internet speeds

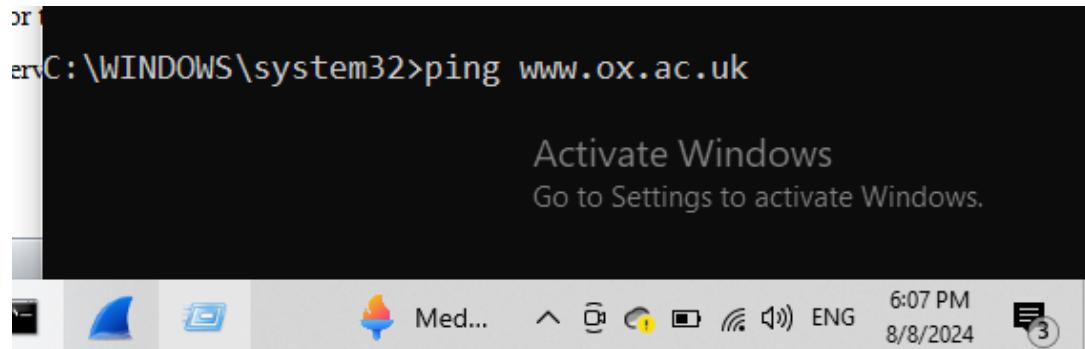


Figure 6:step three to run a command. (Source: screenshot from my PC).

### 1.1.2.3 tracert [www.ox.ac.uk](http://www.ox.ac.uk)

1. I Clicked Start > Run.

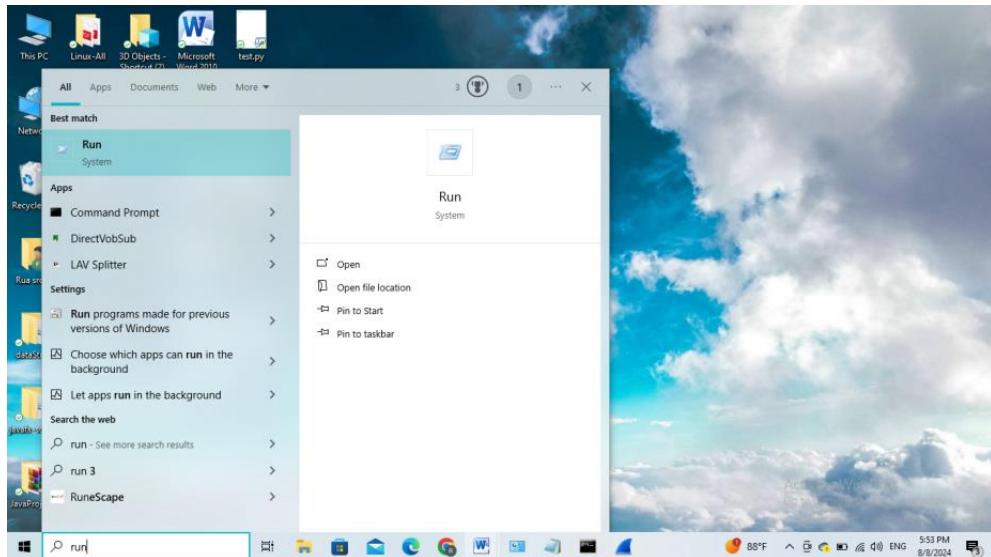


Figure 7 step one to run a command. (Source: screenshot from my PC).

2. In the run box I entered "cmd" > OK.

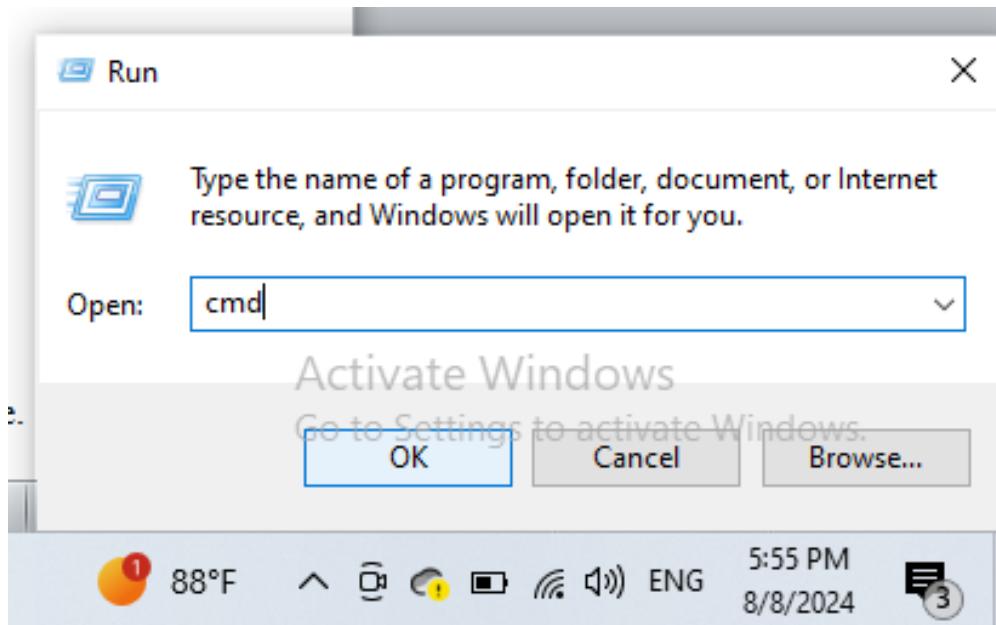


Figure 8 step tow to run a command. (Source: screenshot from my PC).

3. In the command prompt I entered "tarcert Domain Name" without quotes > press ENTER.

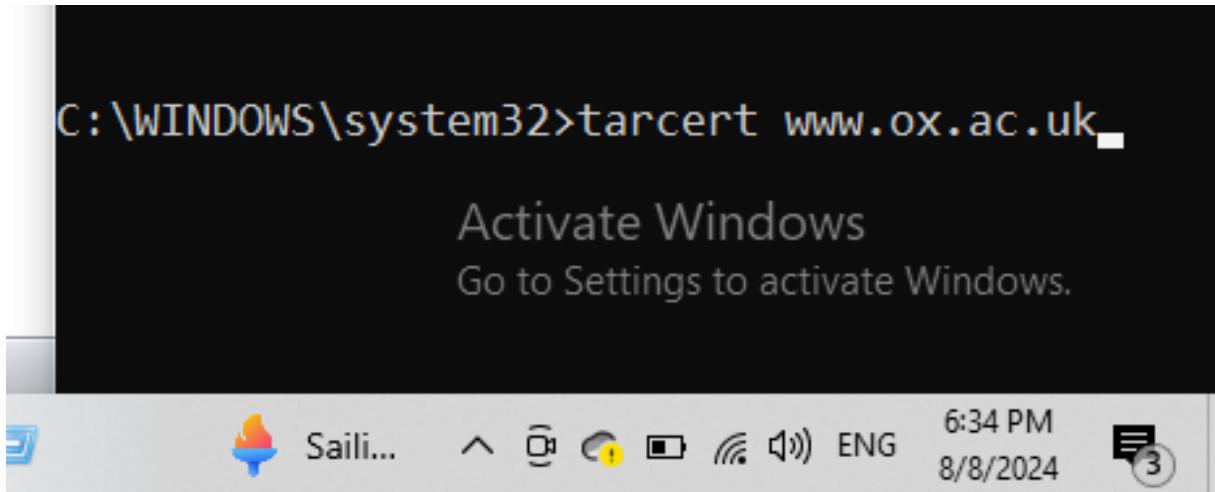


Figure 9 step three to run a command. (Source: screenshot from my PC).

4. then details about the path that a packet takes to the specified destination will be shown.

#### 1.1.2.4 nslookup www.ox.ac.uk.

1. I Clicked Start > Run.

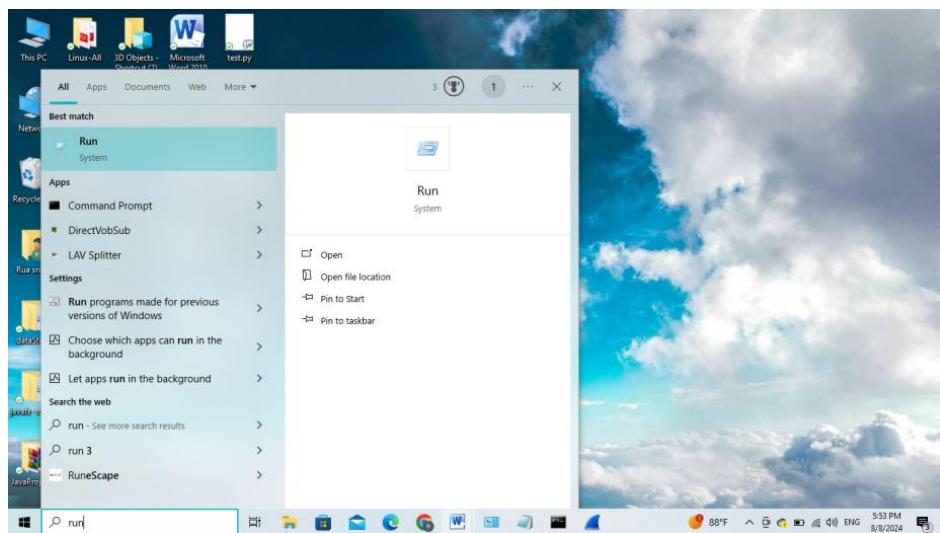


Figure 10 step one to run a command. (Source: screenshot from my PC).

1. In the run box I entered "cmd" > OK.

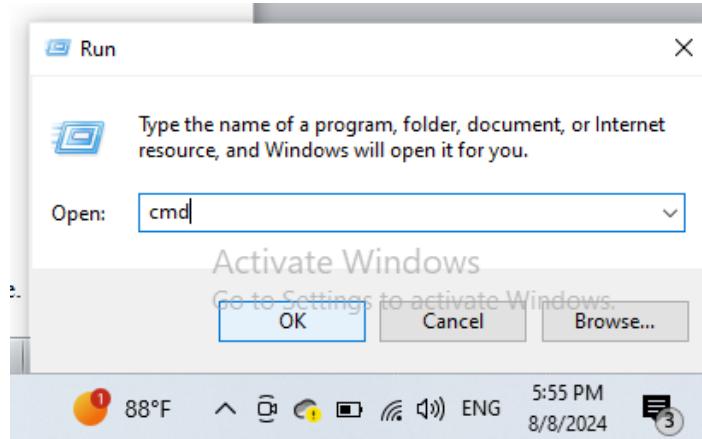


Figure 11 step tow to run a command. (Source: screenshot from my PC).

2. In the command prompt I entered "nslookup domain name" without quotes > press ENTER.

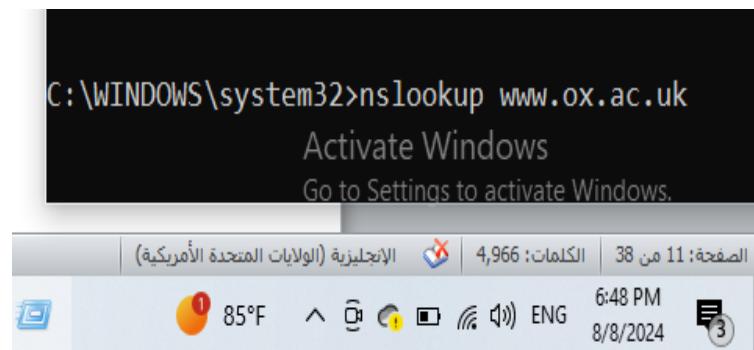


Figure 12 step three to run a command. (Source: screenshot from my PC).

3. then a set of retrieved detailed information about the specified domain will be shown.

## 1.1.2.5: telnet www.ox.ac.uk.

- I Clicked Start > Run.

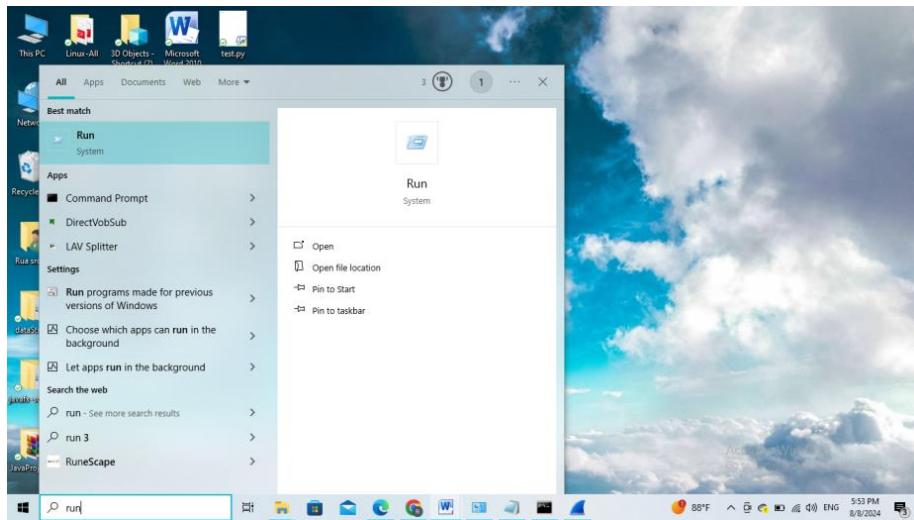


Figure 13 step one to run a command. (Source: screenshot from my PC).

- In the run box I entered "cmd" > OK.

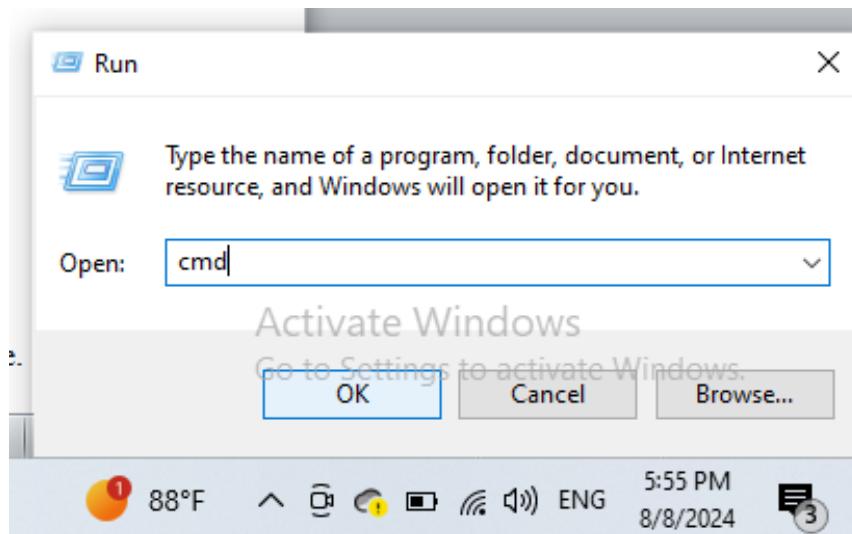


Figure 14 step tow to run a command. (Source: screenshot from my PC).

- In the command prompt I entered " telnet <target domain name> <target port>" without quotes > press ENTER.

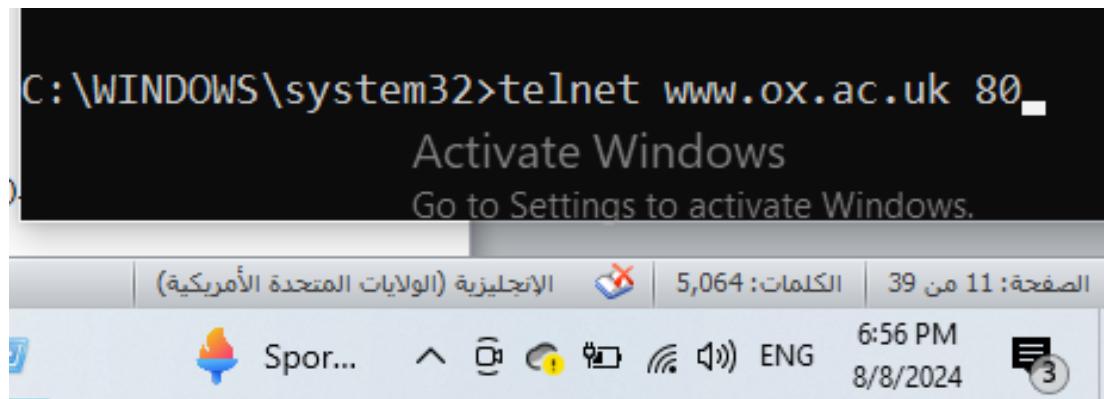


Figure 15 step three to run a command. (Source: screenshot from my PC).

4. If the ping is successful, I should receive replies from the address that I am trying to ping.

1.1.2.6 Giving some details about autonomous system (AS) number, number of IPs, prefixes, peers,

name of Tier1-ISP of [www.ox.ac.uk](http://www.ox.ac.uk).

- 1- for getting autonomous system (AS) number, number of IPs and prefixes, will use bgpview.io [3].  
The IP address is one of the IP's that I got from nslookup command.

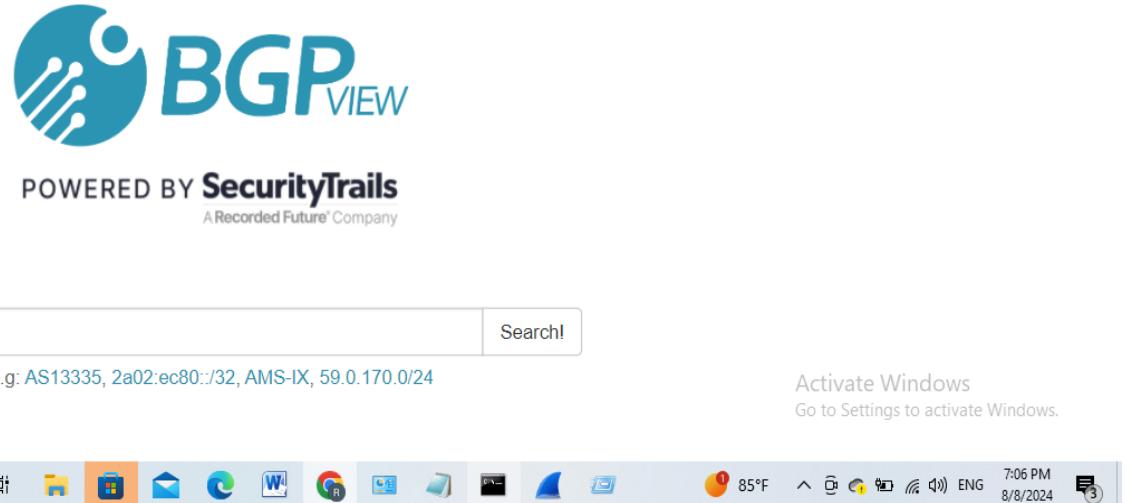


Figure 16:BGPview interface. (Source: screenshot from my PC).

2- for the number peers, name of Tier1-ISP, I will use bgp. tools.

The AS name that we got from the previous website.

Figure 17:BGP.tools interface. (Source: screenshot from my PC).

### 1.1.2.7 Using wireshark to capture some DNS messages.

The DNS messages the I captured are from my home Wi-Fi traffic.

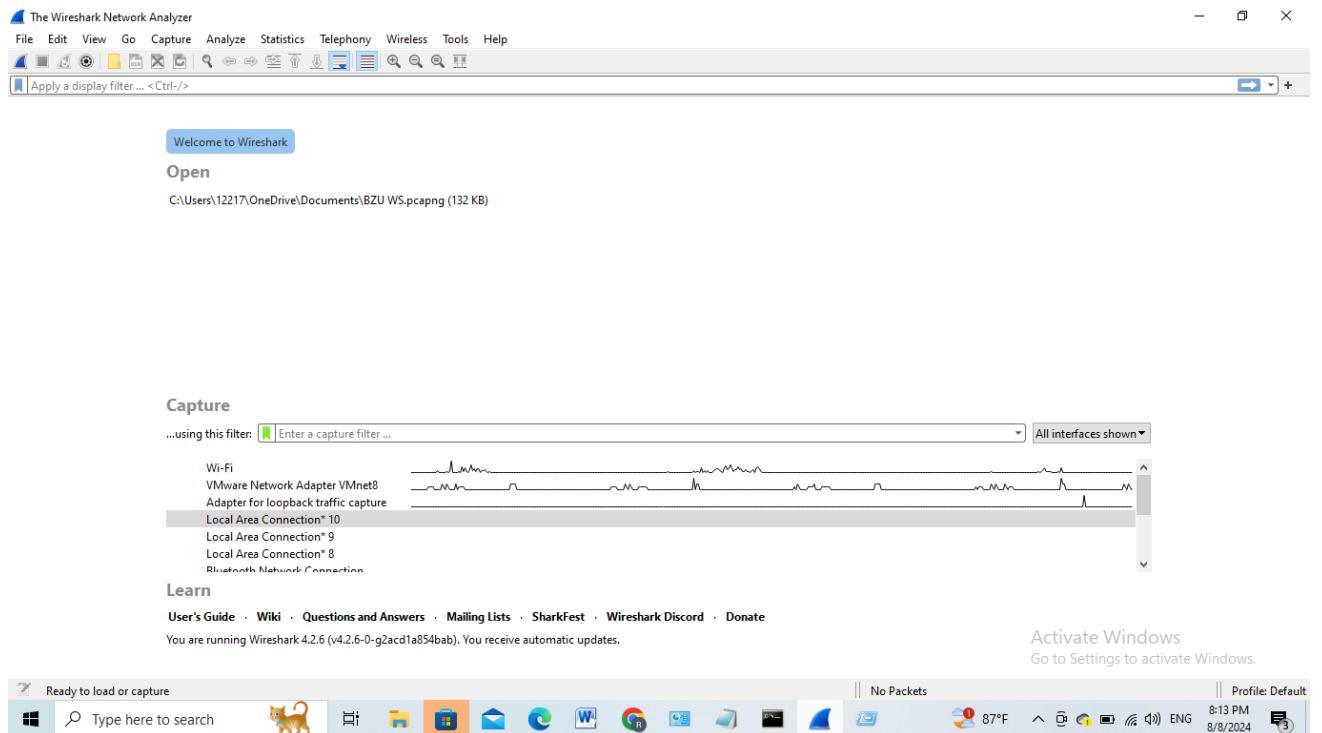
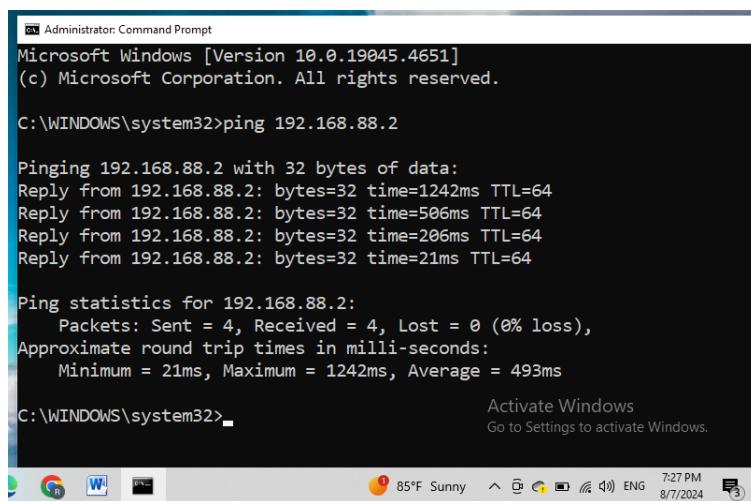


Figure 18: Wireshark interface. (Source: screenshot from my PC).

## Results and Discussions Part 1:

### 1.1: Part one Commands and Wireshark:

1.2.1 the result of ping a smartphone with IP = 192.168.88.2 from a laptop at the same network:



```
Administrator: Command Prompt
Microsoft Windows [Version 10.0.19045.4651]
(c) Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>ping 192.168.88.2

Pinging 192.168.88.2 with 32 bytes of data:
Reply from 192.168.88.2: bytes=32 time=1242ms TTL=64
Reply from 192.168.88.2: bytes=32 time=506ms TTL=64
Reply from 192.168.88.2: bytes=32 time=206ms TTL=64
Reply from 192.168.88.2: bytes=32 time=21ms TTL=64

Ping statistics for 192.168.88.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 21ms, Maximum = 1242ms, Average = 493ms

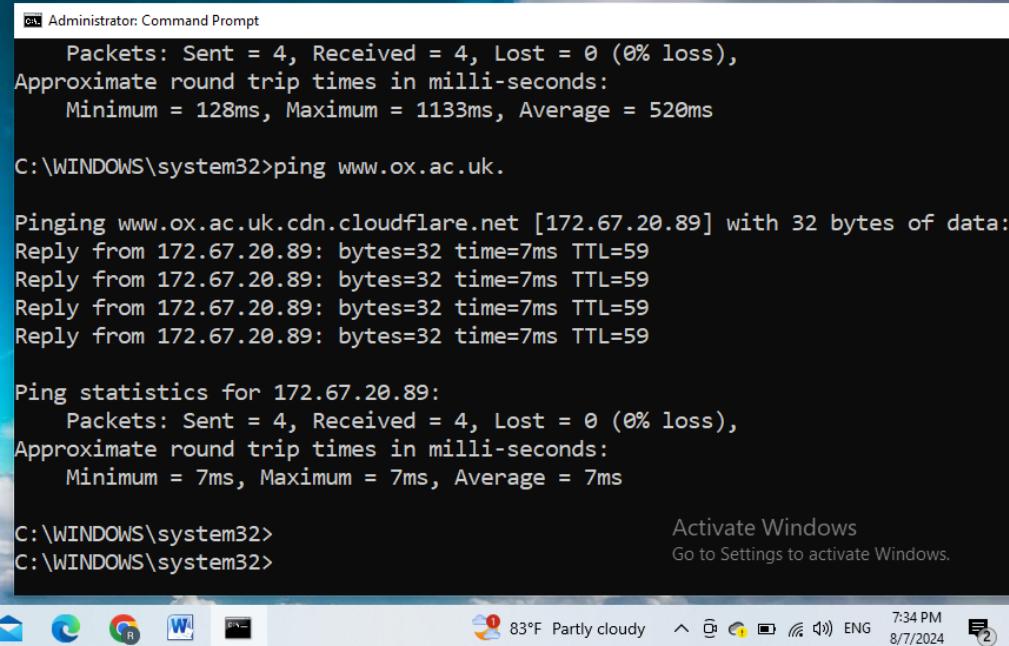
C:\WINDOWS\system32>
```

Figure 19:the replies from the address that I tried to ping. (Source: screenshot from my PC).

My Device ping the IP address of a smart phone that connect to the same network of my device

(IP =192.168.88.2) by sending four packet to this IP then wait for a replay, the replay as shown in Figure (20) has the information for each packet about the IP for the server which we want to ping, size of the packet and it is equal to 32 bytes, and what we look for it is the TTL (time to live) and it is equal to 64 msec this time mean that the time that the packet takes to trans to and from the specific IP to my device. Finally, since the lost is equal to zero and the four-packet come back that means the IP is reachable by my device by the network.

### 1.2.2 The result Of ping [www.ox.ac.uk](http://www.ox.ac.uk).



```
Administrator: Command Prompt
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 128ms, Maximum = 1133ms, Average = 520ms

C:\WINDOWS\system32>ping www.ox.ac.uk.

Pinging www.ox.ac.uk.cdn.cloudflare.net [172.67.20.89] with 32 bytes of data:
Reply from 172.67.20.89: bytes=32 time=7ms TTL=59

Ping statistics for 172.67.20.89:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 7ms, Maximum = 7ms, Average = 7ms

C:\WINDOWS\system32>
C:\WINDOWS\system32>
```

The screenshot shows a Windows Command Prompt window titled "Administrator: Command Prompt". The user has run the command "ping www.ox.ac.uk". The output shows four successful ping responses from the IP 172.67.20.89, each taking 7ms. The TTL value is 59. Below the ping results, the command "ping statistics for 172.67.20.89:" is shown, indicating 4 sent, 4 received, and 0 lost packets with an average of 7ms. The taskbar at the bottom includes icons for Mail, Internet Explorer, Google Chrome, and File Explorer, along with system status indicators like battery level, signal strength, and network connection.

Figure 20: the replies from ping [www.ox.ac.uk](http://www.ox.ac.uk). (Source: screenshot from my PC).

My Device ping the website of university of oxford by the domain name [www.ox.ac.uk](http://www.ox.ac.uk) ,by sending four packets to this domain name IP, then it waits for a replay, the replay as shown in Figure 21 has the information for each packet about the IP for the server which we want to ping, size of the packet and it is equal to 32 bytes, and what we look for it is the TTL (time to live) and it is equal to 59 msec this time mean that the time that the packet takes to trans to and from the specific domain name to my device. Finally, since the loss is equal to zero and the four-packet comes back that means the domain name IP is reachable by my device by the network. . The location of the server from where I got the first response with IP 192.168.88.1 is in United States according to [www.iplocation.net](http://www.iplocation.net) , as shown in Figure 22.

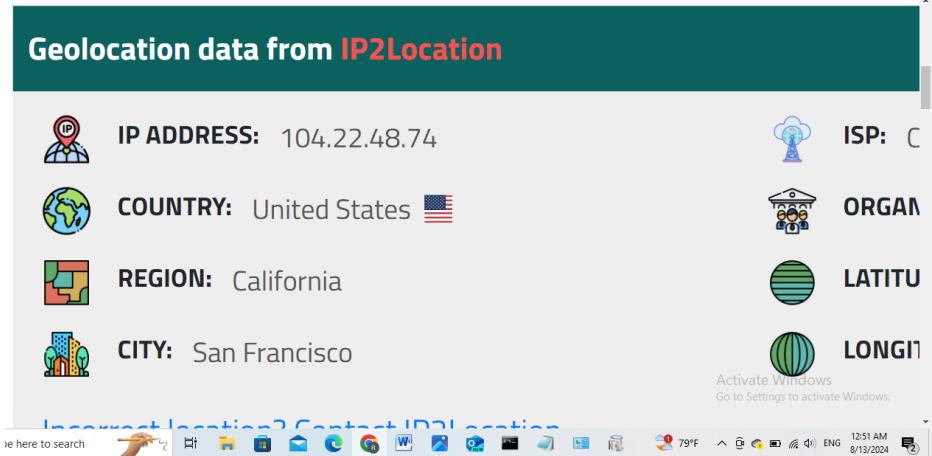


Figure 21: Location of the server with IP 104.22.48.74. (Source: IP location).

. The location of the server from where I got the first response with IP 192.168.88.1 is in Palestine according to [www.iplocation.net](http://www.iplocation.net), as shown in Figure.

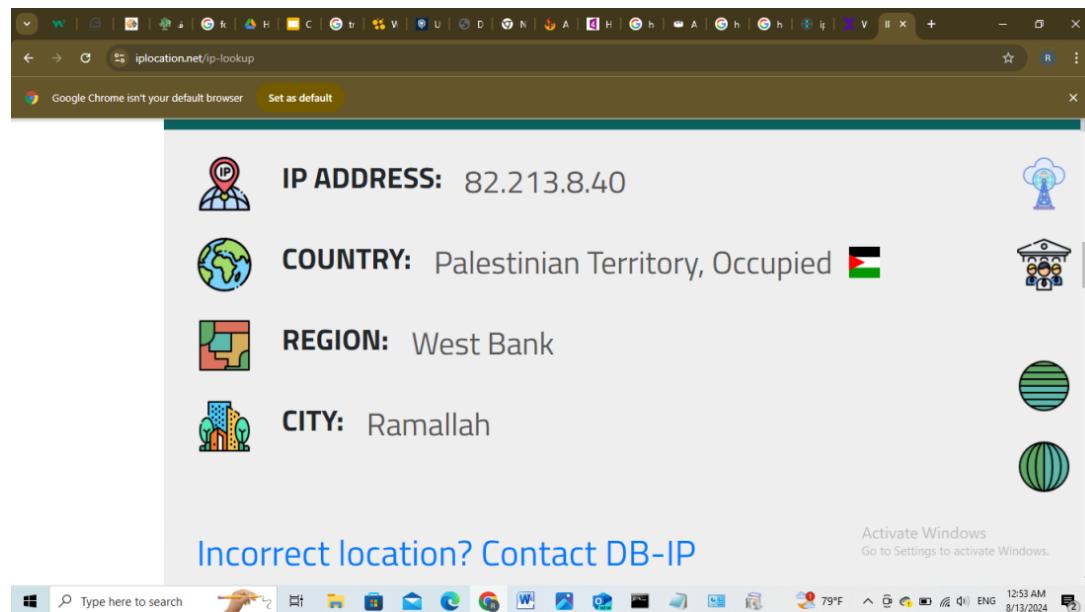
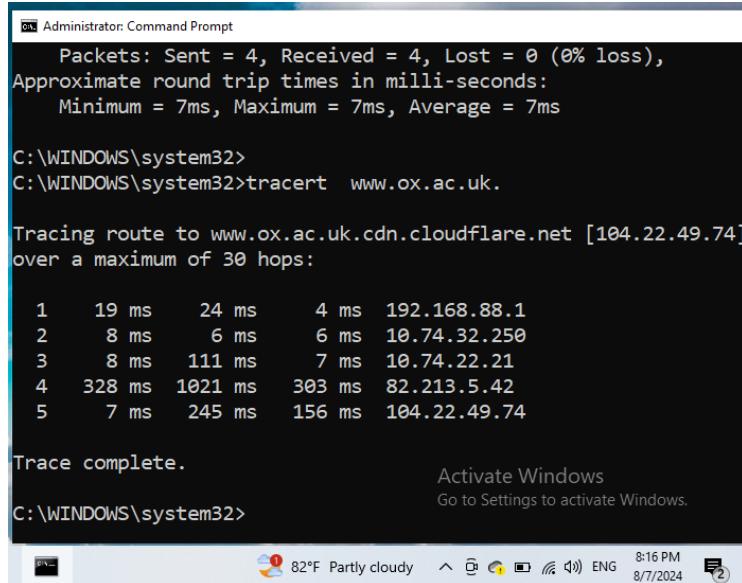


Figure 22: Location of the server with IP 104.22.48.74. (Source: IP location).

### 1.2.3 The result of tracert [www.ox.ac.uk](http://www.ox.ac.uk).



```
Administrator: Command Prompt
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 7ms, Maximum = 7ms, Average = 7ms

C:\WINDOWS\system32>
C:\WINDOWS\system32>tracert www.ox.ac.uk.

Tracing route to www.ox.ac.uk.cdn.cloudflare.net [104.22.49.74]
over a maximum of 30 hops:

 1   19 ms    24 ms      4 ms  192.168.88.1
 2     8 ms      6 ms      6 ms  10.74.32.250
 3     8 ms   111 ms      7 ms  10.74.22.21
 4   328 ms  1021 ms   303 ms  82.213.5.42
 5     7 ms   245 ms   156 ms  104.22.49.74

Trace complete.

Activate Windows
Go to Settings to activate Windows.

C:\WINDOWS\system32>
```

Figure 23:the replies from tracert [www.ox.ac.uk](http://www.ox.ac.uk). (Source: screenshot from my PC). \

The result of tracert command is similar to ping one, since it is work almost as the same way but treract provide more information about each loop .As shown in Figure 24, my device sent three packet for several hoops and in each loop it calculate the RTT time and provide me the IP of this hope. We can notice that the first packet of the fifth hop was reduced from 328ms to 7ms since the path that the packet takes in the fifth loop was congested in the fourth one. since the traffic in it reduced the packet want through it in the fifth loop since it is closer to my device.

Finally, the command provides a comment with “Trace complete “to be sure that the path to the [www.ox.ac.uk](http://www.ox.ac.uk) server is reachable with those loops and hops and there is no error with any hope. Figure 25 explain the concept of this command.

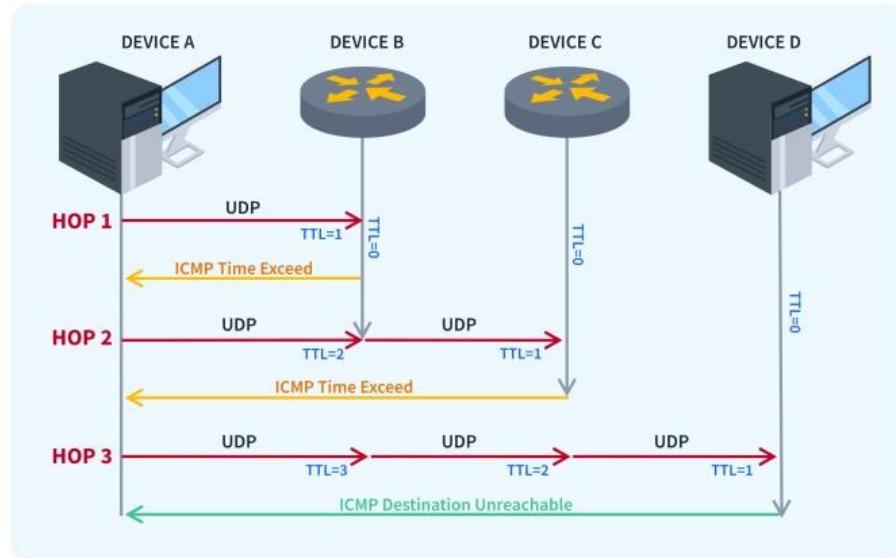


Figure 24:the concept of tracert command. (Source: cbtnuggets).

#### 1.2.4 nslookup [www.ox.ac.uk](http://www.ox.ac.uk).

```
C:\> Administrator: Command Prompt
3      8 ms    111 ms     7 ms  10.74.22.21
4    328 ms   1021 ms   303 ms  82.213.5.42
5      7 ms    245 ms   156 ms  104.22.49.74

Trace complete.

C:\> nslookup www.ox.ac.uk
Server:  UnKnown
Address:  192.168.88.1

Non-authoritative answer:
Name:   www.ox.ac.uk.cdn.cloudflare.net
Addresses:  104.22.49.74
          172.67.20.89
          104.22.48.74
Aliases:  www.ox.ac.uk

Activate Windows
Go to Settings to activate Windows.

C:\>
```

Figure 25:the reply from nslookup [www.ox.ac.uk](http://www.ox.ac.uk). (Source: screenshot from my PC).

As shown in Figure 26 nslookup command provide some information about [www.ox.ac.uk](http://www.ox.ac.uk)

Such as the name of I Don't know , the IP's of this [www.ox.ac.uk](http://www.ox.ac.uk) (IP = 104.22.48.74, 172.67.20.89, 104.22.49.74) and finally the Aliases name [www.ox.ac.uk](http://www.ox.ac.uk). “An alias is a shortcut that identifies a group of hosts, networks, or interfaces” [7].

1.2.5

telnet [www.ox.ac.uk](http://www.ox.ac.uk).

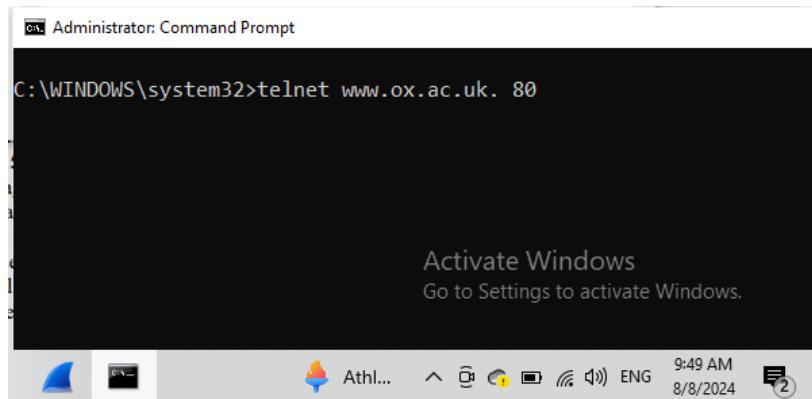


Figure 26:the replies from tracert www.ox.ac.uk.. (Source: screenshot from my PC).

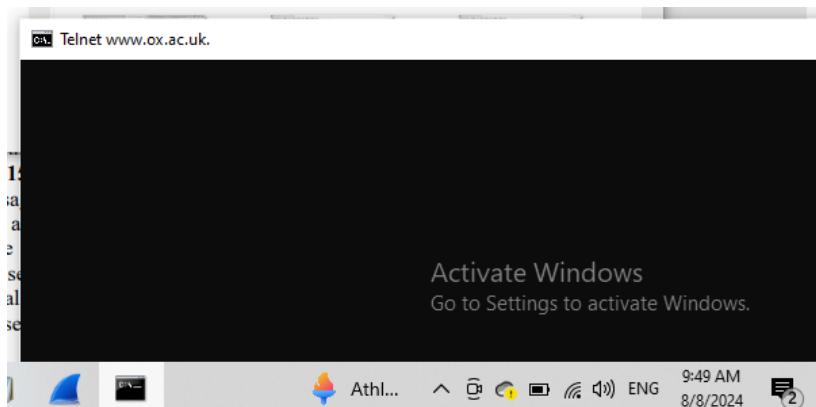


Figure 27:the replies from telnet www.ox.ac.uk. (Source: screenshot from my PC).

I used the Telnet command to connect to port 80 on a remote server of [www.ox.ac.uk](http://www.ox.ac.uk) as shown in Figure 28. To verify if the path from my computer to the [www.ox.ac.uk](http://www.ox.ac.uk) server is open over that port = 80. And the replay with blank screen means that the connection is done and success as shown in Figure 29.

## 1.2.6 some details about [www.ox.ac.uk](http://www.ox.ac.uk).

The screenshot shows the Bgpview.io interface for the IP address 104.22.49.74. At the top, it says "NO RONS FOUND". Below that is a table titled "Announced Prefixes" with two entries:

Country	Announced Prefix	Prefix Name	Prefix Description	ASN	ASN Description	ASN Name
USA	104.22.48.0/20	CLOUDFLARENET	Cloudflare, Inc.	AS13335	CLOUDFLARENET	Cloudflare, Inc.
USA	104.16.0.0/12	CLOUDFLARENET	Cloudflare, Inc.	AS13335	CLOUDFLARENET	Cloudflare, Inc.

Below the table is a section titled "RIR Allocation Summary" with the following details:

- PREFIX: 104.16.0.0/12
- GEOIP COUNTRY: USA
- IP ADDRESSES: 1,048,576
- REGIONAL REGISTRY: ARIN
- ALLOCATION STATUS: Allocated
- ALLOCATION DATE: 28<sup>th</sup> March 2014

At the bottom right of the main content area, there is a message: "Activate Windows Go to Settings to activate Windows."

The taskbar at the bottom shows various icons and the date/time: 7:12 PM 8/8/2024.

Figure 28:Information's from Bgp.view about www.ox.ac.uk server. (Source: Bgp.view ).

As shown from Figure 29 from the Bgp.view the autonomous system (AS) number is AS13335 , the number of IPs is 1,048,576 , and the prefix number for the IP 104.16.0.0 is 12, for the IP 104.22.48.74is 20.

The screenshot shows the Bgp.tools interface for AS 13335. At the top, it displays the AS number 13335 and the website <https://www.cloudflare.com>. Below that is a Cloudflare landing page with the headline "Connect, protect and build everywhere".

The main navigation menu includes "Overview", "Prefixes", "Connectivity" (which is selected), "Whois", and "IX".

Below the menu, there are three statistics boxes:

- Peers**: 2408
- Upstreams**: 294
- Downstreams**: 688 (Cone: 690)

At the bottom right, there is a message: "Activate Windows Go to Settings to activate Windows."

The taskbar at the bottom shows various icons and the date/time: 7:17 PM 8/8/2024.

Figure 29:.Information's from Bgp.tool about www.ox.ac.uk server. (Source: Bgp.tool).

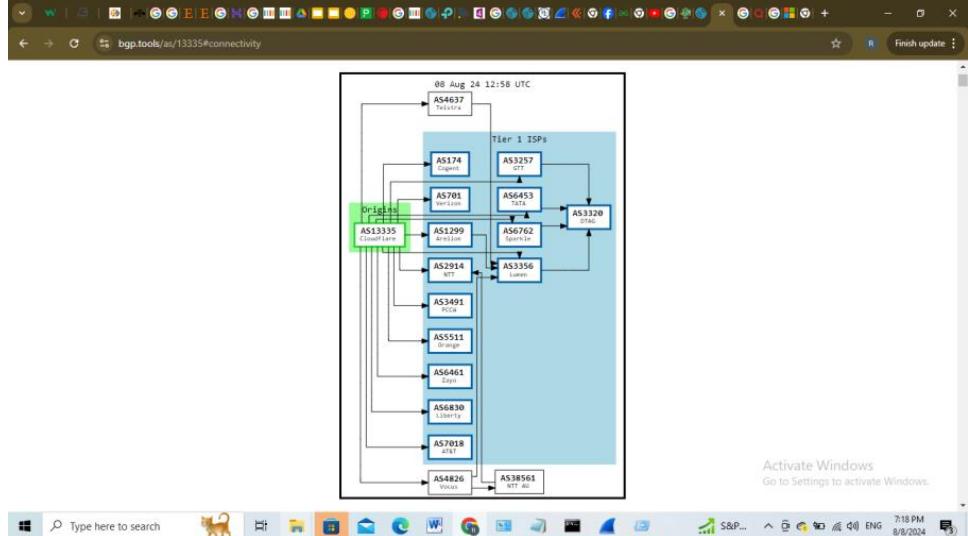


Figure 30: chart of www.ox.ac.uk links. (Source: Bgp.tool).

As shown from Figure 31 the number of peers is 2821, from Figure 32 the name of one of Tier1-ISP is AS3257.

## Using Wireshark to capture some DNS messages.

- The result of capturing my home Wi-Fi traffic is as shown in Figure 33:

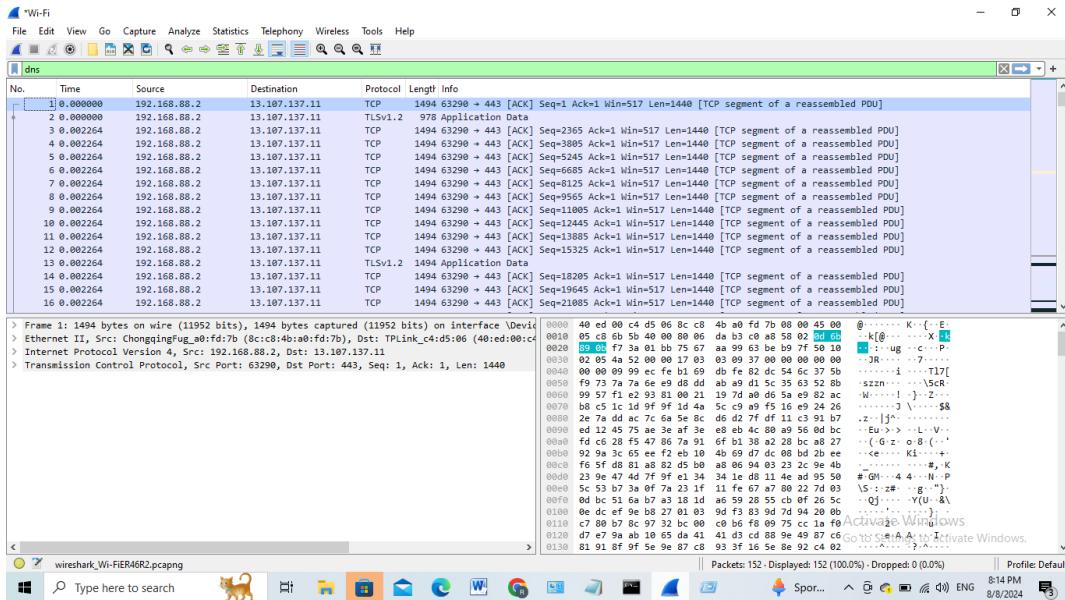
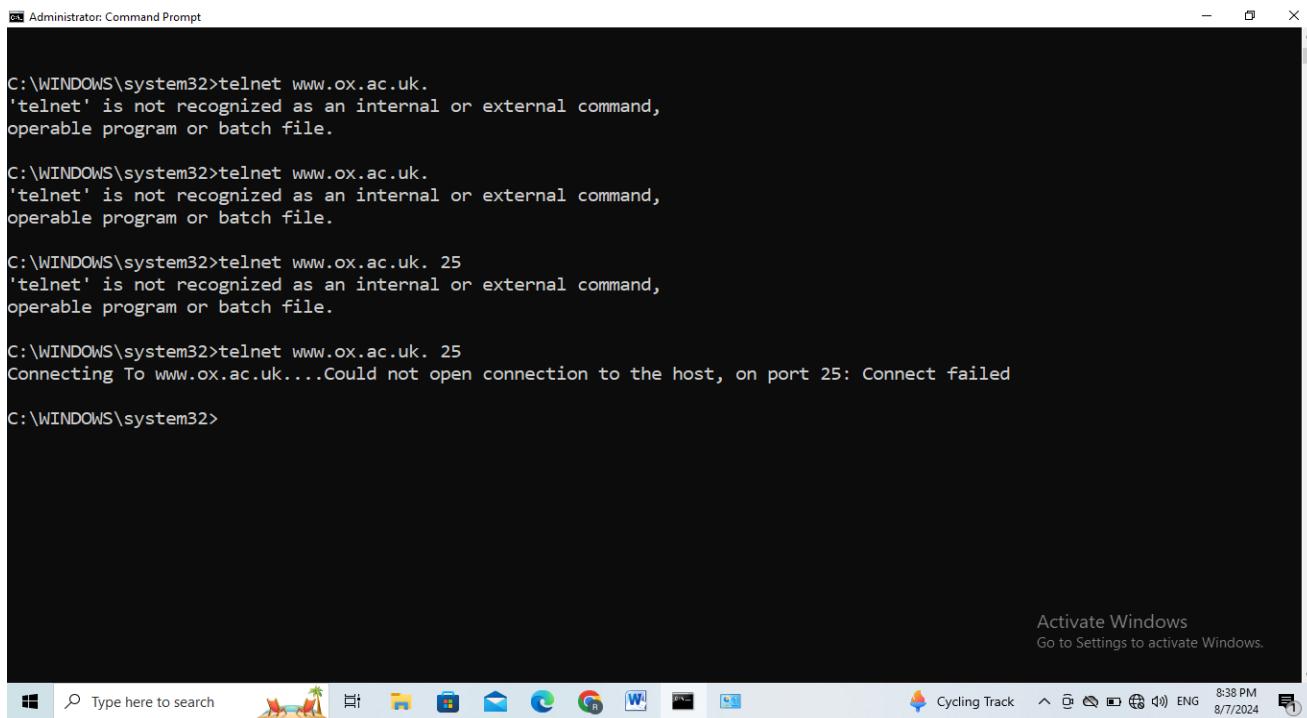


Figure 31:DNS massages from capturing the Wi-Fi traffic. (Source: wireshark).

### 1.3.1telnet is not recognized as internal or external command.

The first time I used talent command the replay was as shown in fig



```
C:\WINDOWS\system32>telnet www.ox.ac.uk.  
'telnet' is not recognized as an internal or external command,  
operable program or batch file.  
  
C:\WINDOWS\system32>telnet www.ox.ac.uk.  
'telnet' is not recognized as an internal or external command,  
operable program or batch file.  
  
C:\WINDOWS\system32>telnet www.ox.ac.uk. 25  
'telnet' is not recognized as an internal or external command,  
operable program or batch file.  
  
C:\WINDOWS\system32>telnet www.ox.ac.uk. 25  
Connecting To www.ox.ac.uk....Could not open connection to the host, on port 25: Connect failed  
C:\WINDOWS\system32>
```

Figure 32:the replies from telnet www.ox.ac.uk. (Source: screenshot from my PC).

This reply means that the command telnet is not recognized as internal or external command.

And the solution was as following by the computer setting as following steps:

- I Opened the Control Panel.
- Click on Programs & Features.
- In left bar I selected “Turn Windows features on or off”
- Found "Telnet Client" and ticked it.
- Click "OK"

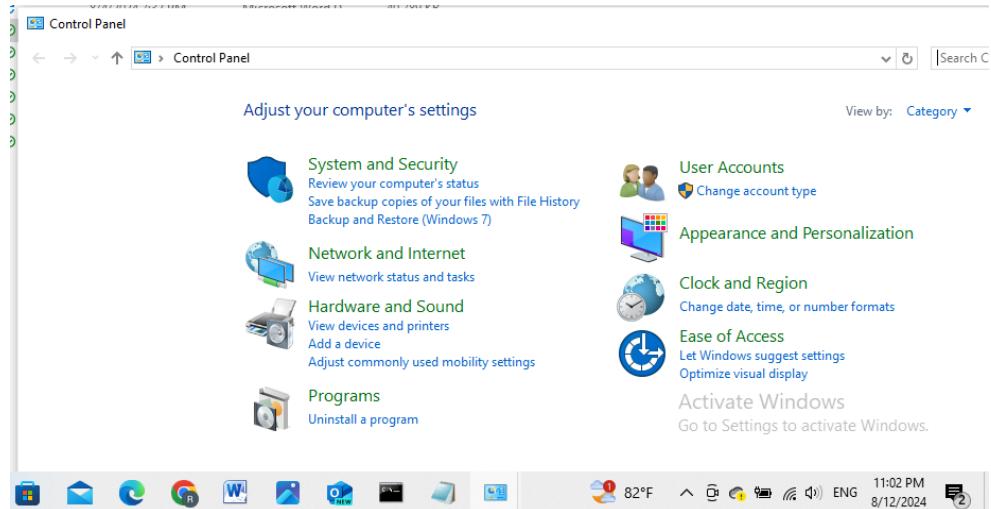


Figure 33:step one. (Source: screenshot from my PC).

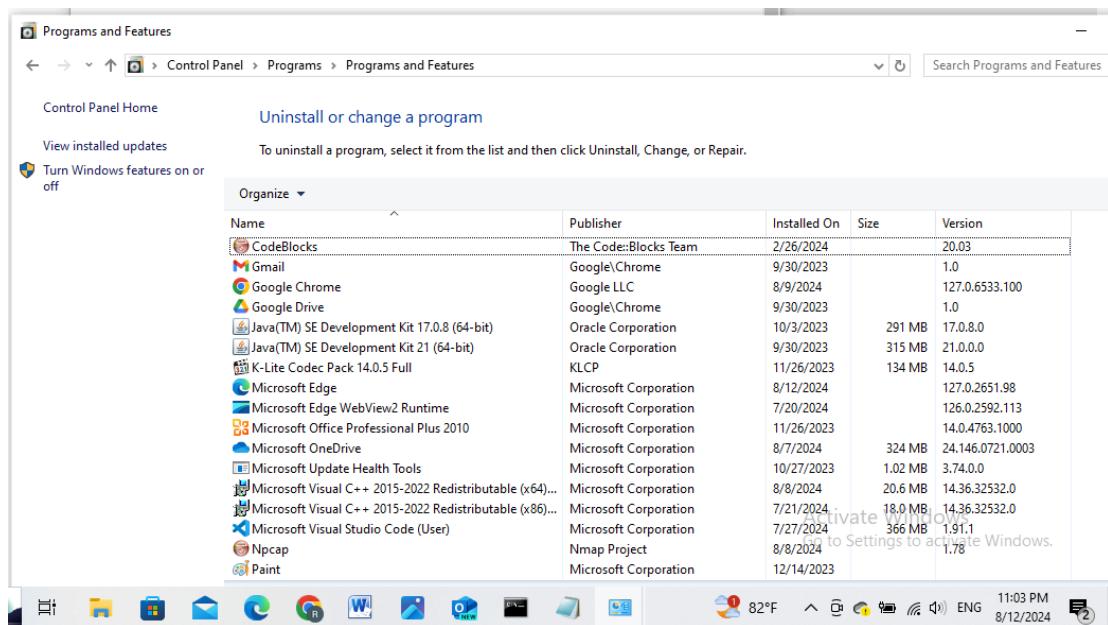


Figure 34:step tow. (Source: screenshot from my PC)

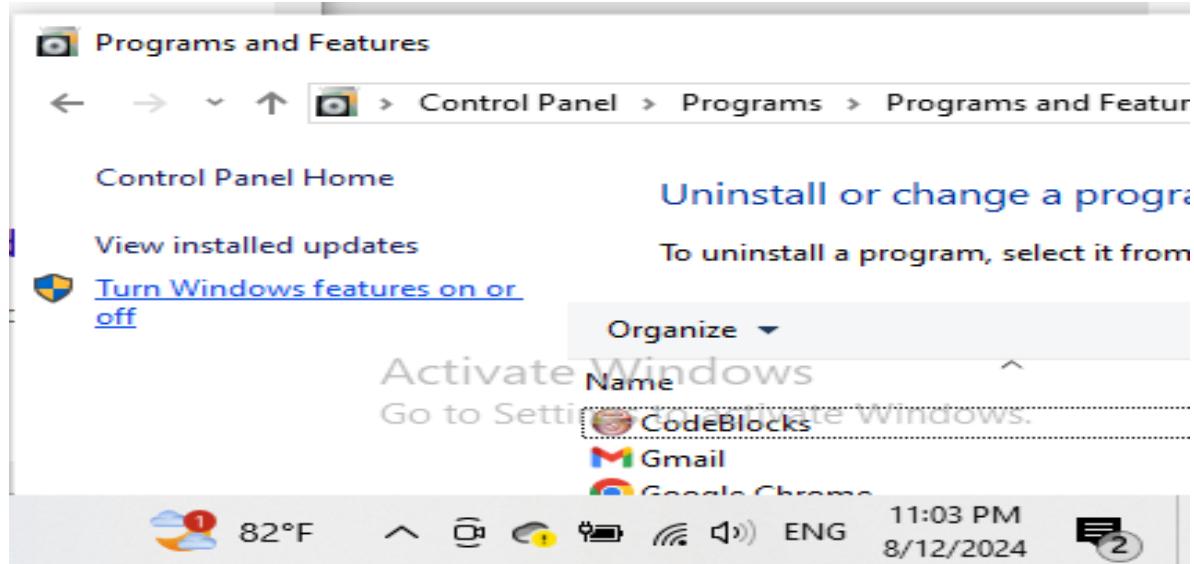


Figure 35:step three. (Source: screenshot from my PC).

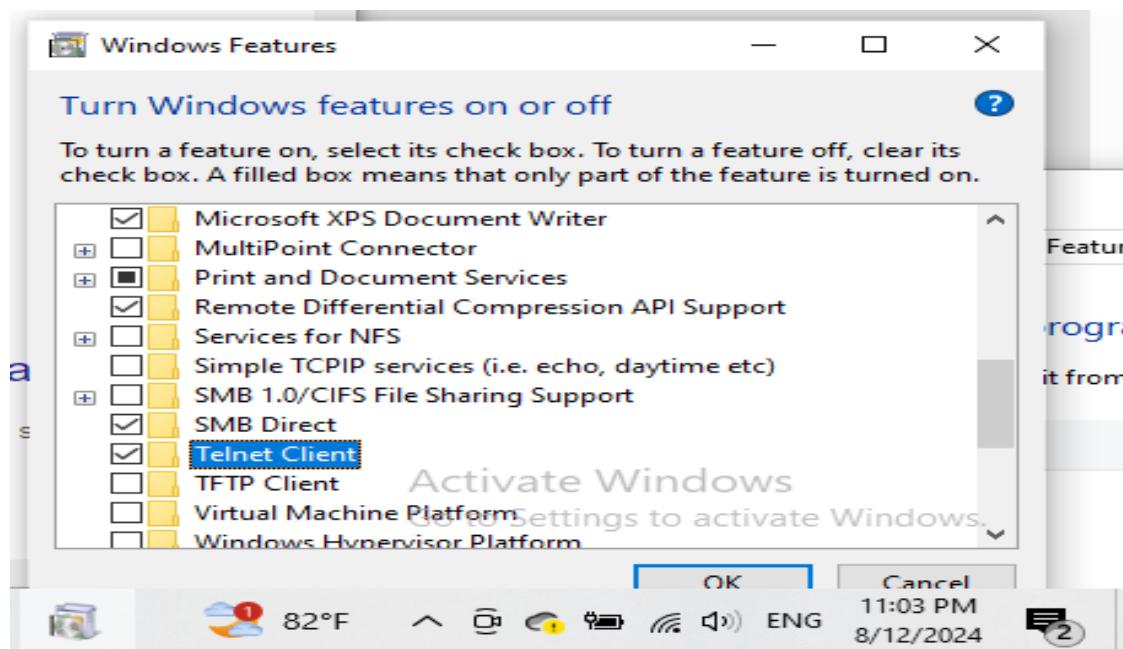


Figure 36:step four. (Source: screenshot from my PC).

## Part 2: Socket Programming (TCP and UDP)

### Theory:

Socket Programming is a way of connecting two nodes on a network to communicate with each other. One socket(node) listens on a particular port at an IP, while the other socket reaches out to the other to form a connection. The server forms the listener socket while the client reaches out to the server.[12]

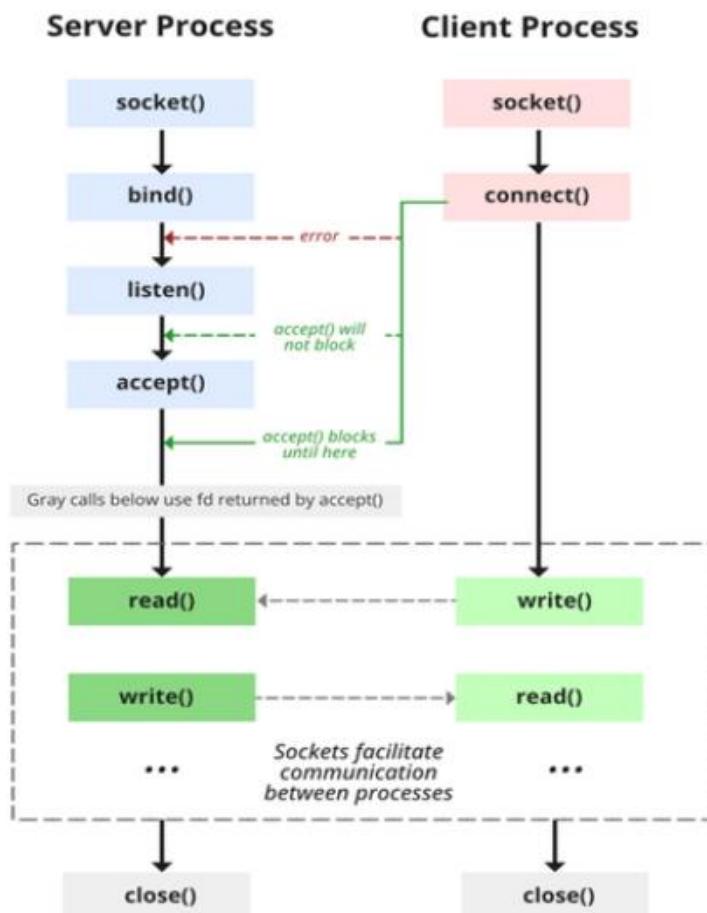
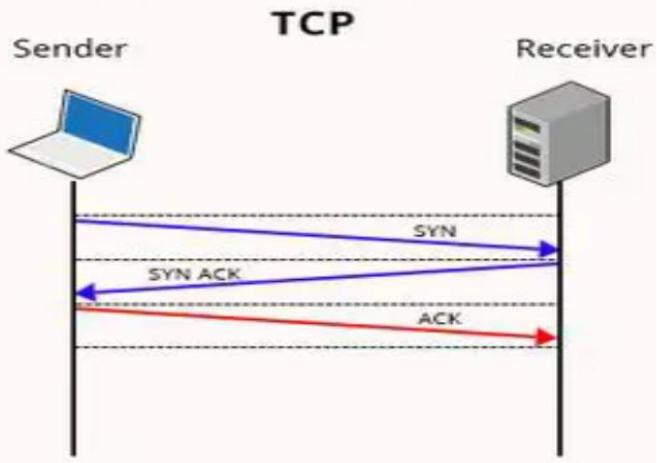


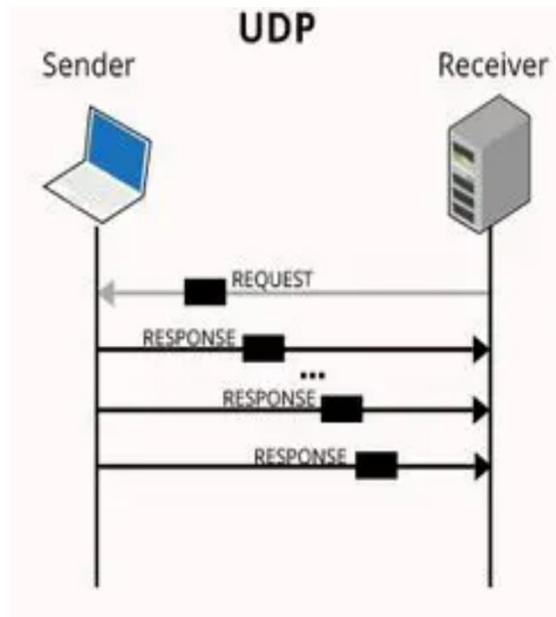
Figure 37: State Diagram for Server and Client Model [12]

Transmission Control Protocol (TCP): is a communications standard that enables application programs and computing devices to exchange messages over a network. It is designed to send packets across the internet and ensure the successful delivery of data and messages over networks.[13]



*Figure 38:TCP connections.[14]*

User Datagram Protocol (UDP): is an alternative communications protocol to Transmission Control Protocol (TCP), used primarily for starting low-latency and loss-tolerating connections between applications and the internet. UDP is also known as a “stateless” protocol, meaning it doesn’t acknowledge that the packets being sent have been received. Due to UDP working this way, it is typically used for streaming services. You may hear some breakup in the audio or see some skips in the video, but UDP transmissions prevent the stream from stopping completely. [14]



*Figure 39:UDP connections.*

## procedure:

## TCP Protocol:

- First in this part I write the TCP server code and another one for the client using python programming.
  - And depending on our ID numbers: I talk (rua ID number:1221727) => 1727 represent the port number that I use for this part.
  - As far as the IP address I talk it from my pc (that represented the server):

```
C:\Users\Diana>ipconfig
```

Using this command, I take the IP of my pc

Windows IP Configuration

Ethernet adapter Ethernet:

Media State . . . . . : Media disconnected  
Connection-specific DNS Suffix . . . . . :

Ethernet adapter Ethernet 2: From these details I take the IP address of my pc  
when the server & client devices are connected using Cable

Connection-specific DNS Suffix . . . . . :  
Link-local IPv6 Address . . . . . : fe80::a2a2:de32:1b5f:8e50%6  
**IPv4 Address** . . . . . : **192.168.56.1**  
Subnet Mask . . . . . : 255.255.255.0  
Default Gateway . . . . . :

Wireless LAN adapter Local Area Connection\* 1:

Media State . . . . . : Media disconnected  
Connection-specific DNS Suffix . . . . . :

Wireless LAN adapter Local Area Connection\* 2: From these details I take the IP address of my pc  
when the server & client devices are connected using WIFI

Media State . . . . . :  
Connection-specific DNS Suffix . . . . . :

Wireless LAN adapter Wi-Fi: and I use this IP in my code and connection

Connection-specific DNS Suffix . . . . . :  
Link-local IPv6 Address . . . . . : fe80::1fcf:10e2:e55:3532%14  
**IPv4 Address** . . . . . : **192.168.68.62**  
Subnet Mask . . . . . : 255.255.252.0  
Default Gateway . . . . . : fe80::b6b0:24ff:fec0:da69%14  
192.168.68.1

*Figure 40:Selecte (Server) IP address from my PC.*

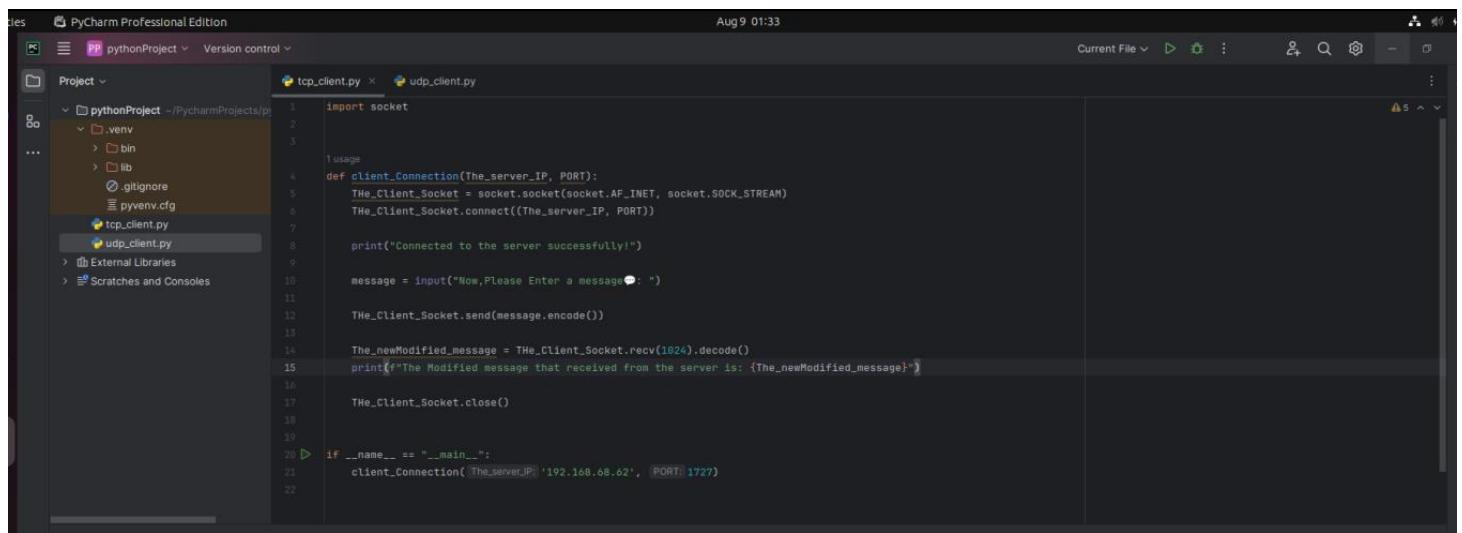
in this part I implemented two python files for (server and client):

## TCP Client Code:

In this part it will send input data from client to server, replace all the vowels (aeiou/AEIOU) with '#' in server section then it will return the string to client and print it.

Note:

I used a virtual machine (using Linux OS) as the client to send the message to the server (and the server is my PC (windows OS)).



The screenshot shows the PyCharm Professional Edition interface. The project navigation bar at the top indicates 'PyCharm Professional Edition', 'pythonProject', and 'Version control'. The current file is 'tcp\_client.py'. The code editor displays the following Python script:

```
import socket

#Usage
def Client_Connection(The_server_IP, PORT):
    THE_Client_Socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    THE_Client_Socket.connect((The_server_IP, PORT))

    print("Connected to the server successfully!")

    message = input("Now, Please Enter message: ")

    THE_Client_Socket.send(message.encode())

    The_newModified_message = THE_Client_Socket.recv(1024).decode()
    print(f"The Modified message that received from the server is: {The_newModified_message}")

    THE_Client_Socket.close()

if __name__ == "__main__":
    Client_Connection('192.168.68.62', 1727)
```

Figure 41:TCP\_Client Code.

- + Here I define function called (client connection) that accept server IP and the port as a parameters, then (*socket.AF\_INET => the socket will use IPv4 and I take it from my pc I upload image previous that define this IP, #socket.SOCK\_STREAM => specify that the socket is a TCP socket*) **then I connected this socket using these data.**
  - + I add print messages to make the client clear about the state.
  - + Then the client sends a message to the server.
  - + Then using send () function that takes the message that the client write it then encode it.
  - + It will receive the Client\_Message from the server.  
\*the 1024 is the buffer size in bytes then it will decode it
  - + It should close the socket after finishing the process.
  - + Finally, in the main I invoke the function with passes choices The selected port and the IP

## TCP Server code:

In this part it will receive input data from client, and it will replace all the vowels (aeiou/AEIOU) with '#' then it will return the string to client and print it.

The screenshot shows a PyCharm IDE interface with the following details:

- Top Bar:** Network, Version control, Current File, and other standard icons.
- Project Explorer:** Shows files: tcp\_server.py, udp\_server.py, test.py, and builtins.py.
- Status Bar:** Shows 45 errors, 9 warnings, and a timestamp of 15:34.
- Code Editor:** Displays Python code for a function named `replace_The_vowels_Function`. The code uses a loop to iterate through each character in the input message. If the character is a vowel (either lowercase or uppercase), it is replaced by a '#'. Otherwise, the character is appended as is. The modified message is then joined back together and returned.

```
import socket

#####
Codeium: Refactor | Explain | Docstring | X
Usage

def replace_The_vowels_Function(Message): # function replaces the vowels in the messages with #
    Output_String = [] # this list will contain the modified message that will
                        # be contained the messages after replacing the vowels with #
    vowels = "aeiouAEIOU" #the list of vowels in small and capital form

    #here i put 2 iterations in a loop one of it to
    for i in Message:
        if i in vowels:
            Output_String.append('#')
        else:
            Output_String.append(i)

    return ''.join(Output_String) #this will return the modified message after editing

####
```

- Bottom Bar:** Terminal, Local, and Network tabs. Network tab is selected, showing tcp\_server.py.
- Taskbar:** Shows the Start button, task switcher, and pinned application icons.
- System Tray:** Shows battery level, signal strength, and date/time (8/9/2024).

*Figure 42: TCP\_Server code part1.*

- + In this part, First I defined the function that called replace vowel's function, and take the message as a parameter, then I declared a list that called output string to store the message after changings on it (if it has vowels characters).
  - + Within the 2 iteration that iterate on the string (char by char) and search if it contains vowels characters (so it has it should replace it with '#' then add it to output list) if it not then just adds the char as it.
  - + Finally, joined this list and returned it when this function invoked.

The screenshot shows a code editor window with several tabs at the top: 'tcp\_server.py' (active), 'udp\_server.py', 'test.py', and 'builtins.py'. The main pane displays Python code for a TCP server. The code includes comments explaining the use of various socket parameters and the handling of client connections. The code editor has a dark theme with syntax highlighting. The bottom status bar shows the file path 'Network > tcp\_server.py', the current line '15:1', encoding 'CRLF', character set 'UTF-8', and spaces '4 spaces', along with the Python version 'Python 3.11 (Network)' and the date/time '1:57 AM 8/9/2024'.

```
Codeium: Refactor | Explain | X
1 usage
def Starting_Of_The_Server(Port):
    """these informations and the used lines here i learn it
    from video & i will provide the references in report "https://www.youtube.com/watch?v=3QiPPX-KeSc" """
    #socket.AF_INET => the socket will use IPv4 and i take it from my pc.
    #socket.SOCK_STREAM => specifies that the socket is a TCP socket.
    # "0.0.0.0" makes the server listen to all network interfaces
    server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server_socket.bind(('0.0.0.0', Port))
    server_socket.listen(1)

    print(f"The Server being run on port {Port}") # "" to know that the server is
                                                # running on this port ""

    while True:
        client_socket, addr = server_socket.accept() #
        print(f"This Connection is from {addr}")#it will print the address of the client

        Client_Message= client_socket.recv(1024).decode() # it will receive the Client_Message from the client
                                                        #the 1024 is the buffer size in bytes then i decode it
        print(f"The Data received from the client is : {Client_Message}") # to print the Client_Message from the client
```

Figure 43:TCP\_Server code part2.

These information I learned it from a specified videos in YouTube (I will provide the links in references part) + from chapter 2 slides.

First in starting the connection function I declared the socket and pass some parameters ,”  
“socket.AF\_INET” => the socket will use IPv4 and I take it from my pc(192.186.68.62).  
“socket.SOCK\_STREAM” => specify that the socket is a TCP socket.

“0.0.0.0” makes the server listen to all network interfaces

- After this it will take the client socket and the address using accept () func.
- Finally, it will receive the Client\_Message from the client

\*the 1024 is the buffer size in bytes then it will decode it.

A screenshot of a Windows operating system desktop. In the foreground, a code editor window is open, showing Python code for a TCP server. The code defines a function `Starting_Of_The_Server` that prints a message from the client, checks for an empty Client\_Message, replaces vowels in the message, and sends it back to the client. It also includes a main block that prints a welcome message and calls the function with port 1727. The code editor interface includes tabs for `tcp_server.py`, `udp_server.py`, `test.py`, and `builtins.py`. The background shows a taskbar with various pinned icons, including Microsoft Office applications like Word, Excel, and PowerPoint, as well as other utilities and browser icons.

```
19     def Starting_Of_The_Server(Port):
20         print(f"The Data Received From the client is : {Client_Message}") # to print the Client_Message from the client
21
22         if not Client_Message: #check the Client_Message if it is not empty then break
23             break
24
25         modified_Message = replace_The_vowels_Function(Client_Message) #calling the replace vowels function
26
27         client_socket.send(modified_Message.encode())#sending the modified message to the client
28
29
30         client_socket.close()
31
32 #####
33 #this is the main function:
34 if __name__ == "__main__":
35     print("Welcome To The Server")
36     Starting_Of_The_Server(1727) #calling the function with the specified port
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
```

Figure 44:TCP\_Server code part3.

## ■ UDP Protocol:

- ❖ First in this part I write the UDP server code and another one for the client using python programming.
- ❖ And depending on our ID numbers: I talk (rua ID number:1221727) => **1727** represent the port number that I use for this part.
- ❖ As far as the IP address I talk it from my pc (that represented the server):

```
C:\Users\Diana>ipconfig           Using this command, I take the IP of my pc

Windows IP Configuration

Ethernet adapter Ethernet:
  Media State . . . . . : Media disconnected
  Connection-specific DNS Suffix . . . . . : From these details I take the IP address of my pc
                                                when the server & client devices are connected using Cable
Ethernet adapter Ethernet 2:          From these details I take the IP address of my pc
  Connection-specific DNS Suffix . . . . . : fe80::a2a2:de32:1b5f:8e50%6
  Link-local IPv6 Address . . . . . : 192.168.56.1
  IPv4 Address . . . . . : 192.168.56.1
  Subnet Mask . . . . . : 255.255.255.0
  Default Gateway . . . . . : 255.255.255.0

Wireless LAN adapter Local Area Connection* 1:
  Media State . . . . . : Media disconnected
  Connection-specific DNS Suffix . . . . . :

Wireless LAN adapter Local Area Connection* 2:
  Media State . . . . . : Media disconnected
  Connection-specific DNS Suffix . . . . . : From these details I take the IP address of my pc
                                                when the server & client devices are connected using WIFI
Wireless LAN adapter Wi-Fi:          and I use this IP in my code and connection
  Connection-specific DNS Suffix . . . . . :
  Link-local IPv6 Address . . . . . : fe80::1fcf:10e2:e55:3532%14
  IPv4 Address . . . . . : 192.168.68.62
  Subnet Mask . . . . . : 255.255.252.0
  Default Gateway . . . . . : fe80::b6b0:24ff:fecc:da69%14
                                192.168.68.1
=====
```

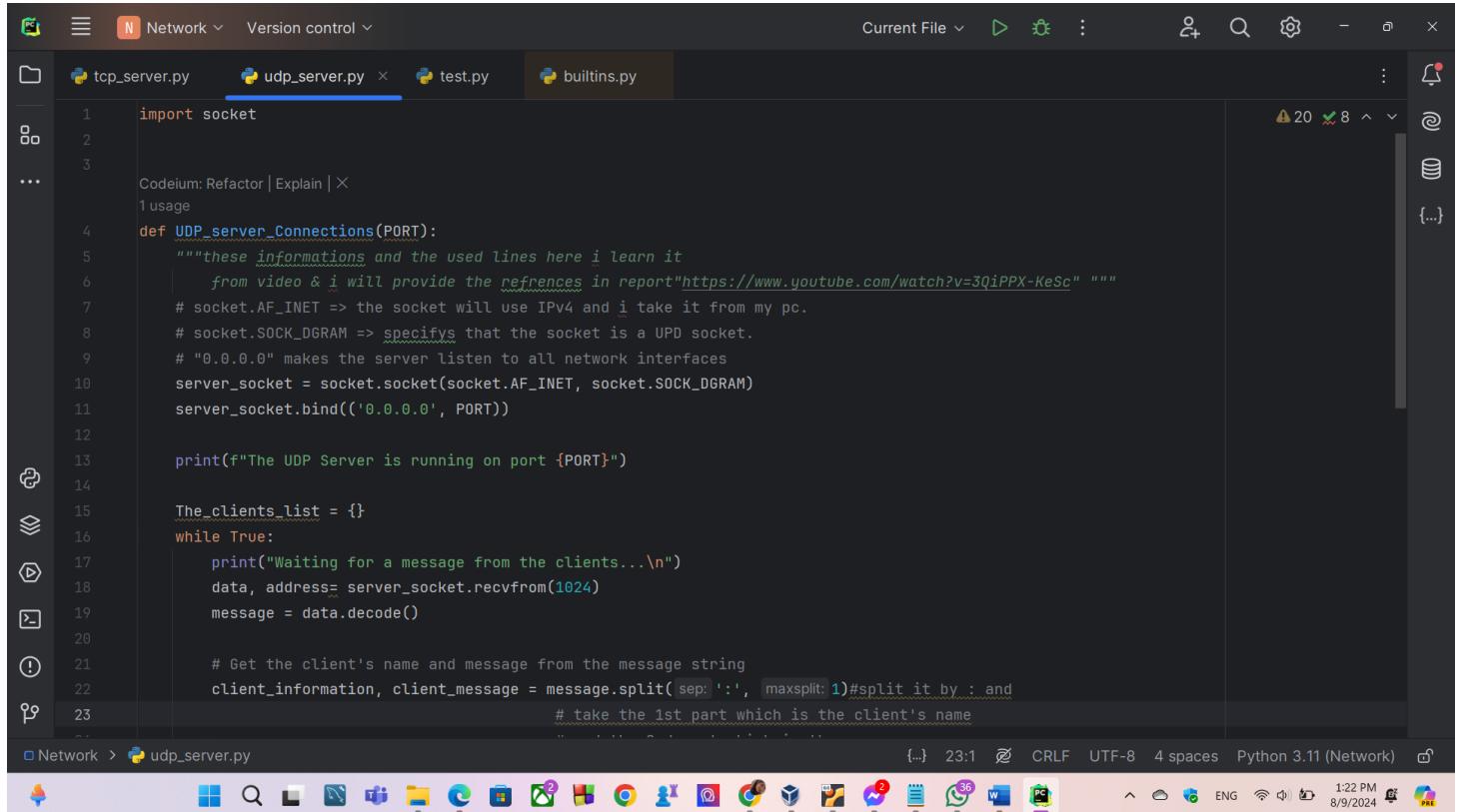
in this part I implemented two python files for (server and client):

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.

- ❖ Note: I used a virtual machine (using Linux OS) as the client to send the message to the server (and the server is my PC (windows OS)).

## UDP Server Code:

Here the server starts working in selected port then waits for the clients to start sending a message, then after the client sends a message, the server can replay this message, and can also show the list of the last communication of client messages.



```
import socket

# socket.AF_INET => the socket will use IPv4 and I take it from my pc.
# socket.SOCK_DGRAM => specifies that the socket is a UDP socket.
# "0.0.0.0" makes the server listen to all network interfaces
server_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
server_socket.bind(('0.0.0.0', PORT))

print(f"The UDP Server is running on port {PORT}")

clients_list = {}
while True:
    print("Waiting for a message from the clients...\n")
    data, address= server_socket.recvfrom(1024)
    message = data.decode()

    # Get the client's name and message from the message string
    client_information, client_message = message.split( sep=':', maxsplit=1) #split it by : and
                                                                # take the 1st part which is the client's name
    print(f"Client Name: {client_information} - Message: {client_message}
```

Figure 45: UDP server code part1.

These information I learned it from a specified videos in YouTube (I will provide the links in references part) and from slides.

First in starting the connection function I declared the socket and pass some parameters , “socket.AF\_INET” => the socket will use IPv4 and I take it from my pc(192.186.68.62). “socket.SOCK\_DGRAM” => specify that the socket is a UDP socket.

“0.0.0.0” makes the server listen to all network interfaces

- After this it will take the client socket and the address using accept () func.
- Finally, it will receive the Client\_Message from the client

\*the 1024 is the buffer size in bytes then it will decode it.

Here it will list the last messages from all clients that contact this server func().

The screenshot shows the PyCharm IDE interface with the following details:

- Project:** Network
- Code Editor:** The active tab is `udp_server.py`. The code implements a UDP server that receives messages from clients, prints them, sends a reply back, and then prints all client communications.

```
3 def UDP_server_Connections(PORT):
4     try:
5         client_information, client_message = message.split(sep=':', maxsplit=1)
6     except ValueError:
7         print(f"Invalid message format: {message}")
8         continue
9
10    # Update the client's last message
11    The_clients_list[client_information] = client_message
12
13    # Print the message from the client
14    print(f"Message from {client_information}: {client_message}")
15
16    # Get a reply from the server to send back to the client
17    The_Server_reply = input(f"Enter your message to {client_information}: ")
18    server_socket.sendto(The_Server_reply.encode(), address)
19    print(f"Sent reply to {client_information}: {The_Server_reply}")
20
21    # Print messages from all clients
22    print("\nMessages from all client communications:\n")
23    for client_info, client_msg in The_clients_list.items():
24        print(f"Message from {client_info}: {client_msg}")
25
26 if __name__ == "__main__":
27
```

- Terminal:** Shows the command `Network > udp_server.py`.
- System Tray:** Shows the date and time as 8/9/2024 6:49 PM.

Figure 46: UDP server code part2.

## UDP Client code:

The screenshot shows the PyCharm IDE interface with the following details:

- Project:** pythonProject
- Code Editor:** The active tab is `udp_client.py`. The code implements a UDP client that sends messages to a server and receives responses.

```
1 import socket
2
3 Usage
4 def UDP_client_Connections(ServerIP, PORT, clientID):
5     client_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
6     while True:
7         message = input(f"Enter your message to the server (Client {clientID}): ")
8         formatted_message = f"Client{clientID}:{message}" # to format the message as Client1:hello
9         client_socket.sendto(formatted_message.encode(), (ServerIP, PORT))#to send the message to the server
10        print("Waiting for a message from the server...\n")
11        data, address = client_socket.recvfrom(1024)
12        print(f"Received message from server: {data.decode()}")
13
14        continue_chat = input("Do you want to send another message? (Y/N): ").strip().lower()
15        if continue_chat != 'Y' and continue_chat != 'y':
16            break
17
18    client_socket.close()
19
20 if __name__ == "__main__":
21     server_ip = '192.168.68.62'
22     port = 1727 #the port that the server is running on
23     client_id = input("Enter your client ID (e.g :1,2,...):")
24     UDP_client_Connections(server_ip, port, client_id)
```

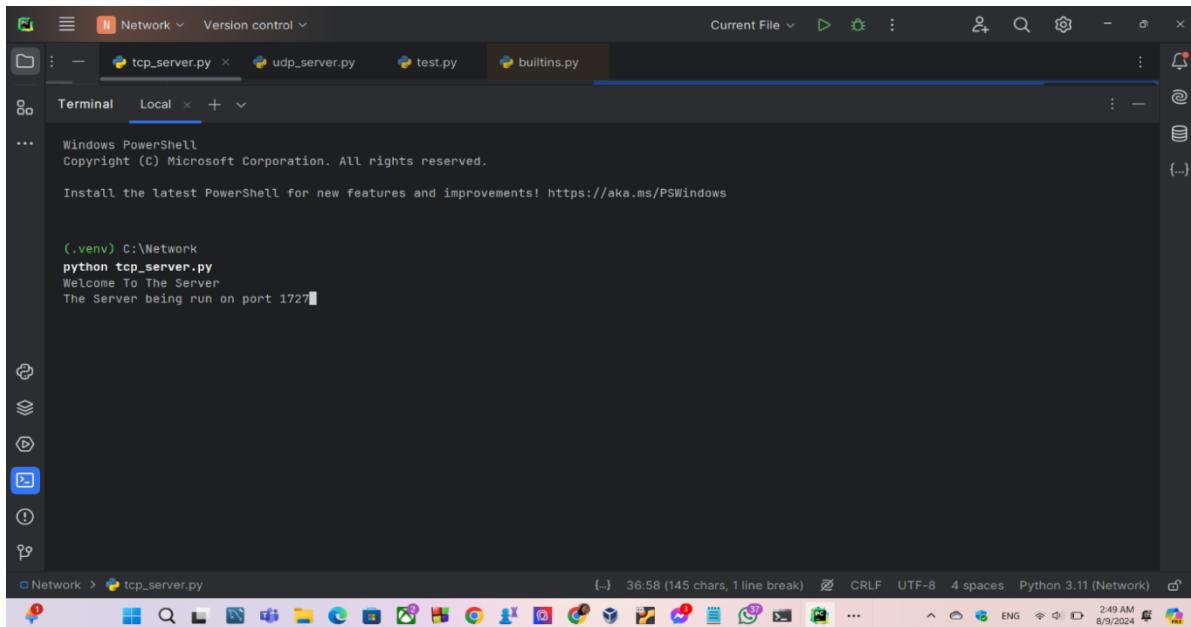
Figure 47: UDP client code.

- + Here I define function called (udp client connection) that accept server IP and the port as a parameters, then (*socket.AF\_INET => the socket will use IPv4 and I take it from my pc I upload image previous that define this IP, #socket.SOCK\_DGRAM => specify that the socket is a UDP socket*) ***then I connected this socket using these data.***
  - + *I add print messages to make the client clear about the state.*
  - + *Then the client sends a message to the server.*
  - + *Then using sendto () function that takes the message that the client write it then encode it.*
  - + It will receive the Client\_Message from the server.  
 \*the 1024 is the buffer size in bytes then it will decode it.
  - + I added a print message to ask the client if he/she wants to continue this conversation.
  - + *It should close the socket after finishing the process.*
  - + *Finally, in the main I invoke the function with passes choices The selected port and the IP*
  
- ✓ Each client chooses a specific id to make the server clear about which client connected with it and can determine the last message from all clients that connected with the server.

## Results and Discussions:

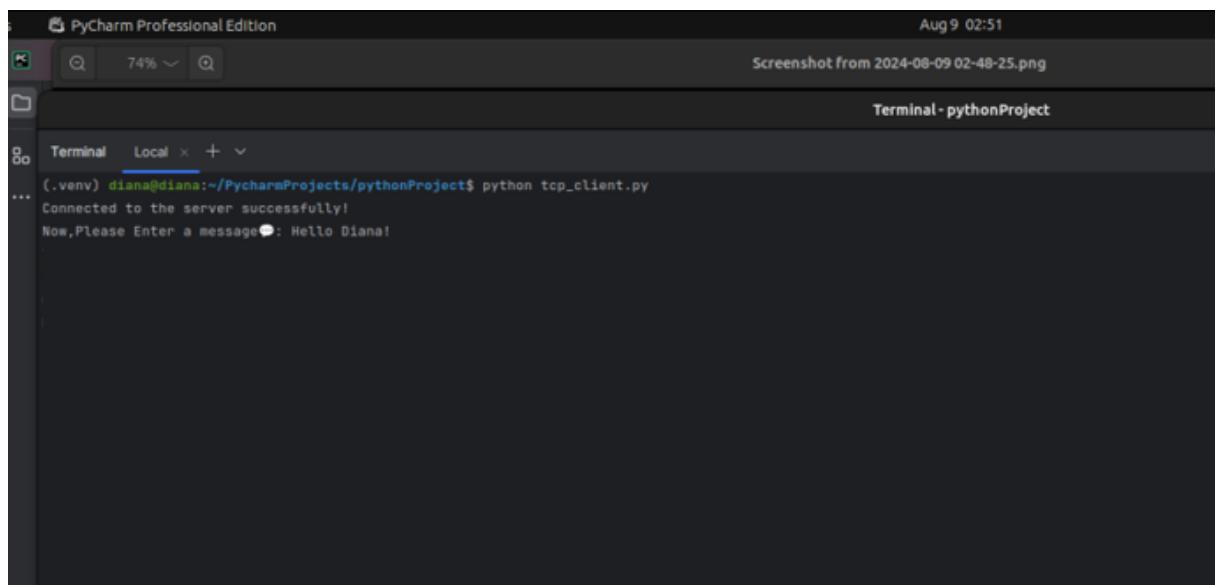
### Screenshots of Part2\_1:

Here when I run server code in terminal, the server starts working in our specified port and waits for the message from the client.



A screenshot of the PyCharm IDE interface. The top navigation bar shows 'Network' and 'Version control'. Below it, there's a file list with 'tcp\_server.py', 'udp\_server.py', 'test.py', and 'builtins.py'. A terminal window is open, titled 'Terminal' (Local). The terminal output shows a Windows PowerShell prompt, followed by the command 'python tcp\_server.py', and the server's response: 'Welcome To The Server' and 'The Server being run on port 1727'. The bottom status bar indicates the file is 36:58 (145 chars, 1 line break), encoding is CRLF, and the Python version is 3.11 (Network).

Figure 48: run the TCP server.



A screenshot of the PyCharm IDE interface, specifically the terminal window titled 'Terminal - pythonProject'. The terminal shows a user session in a virtual environment: '(.venv) diana@diana:~/PycharmProjects/pythonProject\$ python tcp\_client.py'. The client connects to the server successfully and prompts the user to enter a message. The user types 'Hello Diana!' and presses Enter. The terminal shows the message being sent.

Figure 49: run the TCP client.

Here when I run client code in terminal, the client gives message to tell that its connected successfully with server then it will write the message and send it to the server.

A screenshot of the Visual Studio Code interface. The top bar shows 'Network' and 'Version control'. The left sidebar has icons for file explorer, terminal, and snippets. The main area shows a terminal window titled 'Terminal Local'. The terminal output is as follows:

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

(.venv) C:\Network
python tcp_server.py
Welcome To The Server
The Server being run on port 1727
This Connection is from ('192.168.68.62', 53172)
The Data received from the client is : Hello Diana!
```

*Figure 50:The response in server after sending message form the client.*

Here it will get a message that gives us the connection work in which IP, and the address of the client that sent this message to the server. (Here the sent message it will enter the server code functions: which all vowels char will be replaced with '#' symbol and will send again to the client in this form.).

The screenshot shows the PyCharm Professional Edition interface. The top bar displays "PyCharm Professional Edition" and the date "Aug 9 02:48". The left sidebar includes project navigation, file status, and a search bar. The main area features a terminal window titled "Terminal - pythonProject". The terminal output shows the execution of a Python script named "tcp\_client.py". The user connects to a server and sends a message, receiving a modified response.

```
(.venv) diana@diana:~/PycharmProjects/pythonProject$ python tcp_client.py
Connected to the server successfully!
Now,Please Enter a message: Hello Diana!
The Modified message that received from the server is: H#ll# D##n#!
```

*Figure 51: The state in client terminal after receiving the message from the server.*

Here after receiving the modified message after changing (if it has vowels char), and it will appear in the terminal.

## Screenshots of Part2\_2:

Here when I run server code in terminal, the server starts working in our specified port and waits for the message from the client:

A screenshot of a Windows terminal window titled "Terminal". The window shows the command "python udp\_server.py" being run in a virtual environment (.venv) located at C:\Network. The output indicates that the UDP Server is running on port 1727 and is waiting for messages from clients. The terminal interface includes a file explorer sidebar, a status bar at the bottom, and a taskbar at the very bottom.

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

(.venv) C:\Network
python udp_server.py
The UDP Server is running on port 1727
Waiting for a message from the clients...
```

Figure 52:run the UDP server

A screenshot of a PyCharm Professional Edition terminal window titled "Terminal - pythonProject". The terminal shows the command "python udp\_client.py" being run in a virtual environment (.venv) located at PycharmProjects/pythonProject. The user is prompted to enter their client ID (e.g., :1,2,...) and a message to the server (Client 1). The terminal interface includes a file explorer sidebar, a status bar at the bottom, and a taskbar at the very bottom.

```
(.venv) diana@diana:~/PycharmProjects/pythonProject$ python udp_client.py
Enter your client ID (e.g.:1,2,...):1
Enter your message to the server (Client 1):
```

Figure 53:run the UDP client.

Here when I run client code in terminal, it should choose an ID then send to the server, then waits the server until send replay to this message.

```
(.venv) diana@diana:~/PycharmProjects/pythonProject$ python udp_client.py
Enter your client ID (e.g :1,2,...):1
Enter your message to the server (Client 1): Hello from client1
Waiting for a message from the The servr...
```

Figure 54:run the UPP client and write message to the server.

Here client 1 sends a message to the server and waits for the server to send the reply.

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

(.venv) C:\Network
python udp_server.py
The UDP Server is running on port 1727
Waiting for a message from the clients...

Message from Client1:Hello from client1
Enter your message to Client1:
```

Figure 55:UDP server when have the message.

Here the server catches the message from client1 and now it should write reply to this client.

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

(.venv) C:\Network
python udp_server.py
The UDP Server is running on port 1727
Waiting for a message from the clients...

Message from Client1:Hello from client1
Enter your message to Client1: Hey from the server!

Messages from all client communications:

Message from Client1: Hello from client1
Waiting for a message from the clients...
```

Figure 56:response from the server.

```
Terminal - pythonProject
Terminal Local (2) + ▾
(.venv) diana@diana:~/PycharmProjects/pythonProject$ python udp_client.py
Enter your client ID (e.g :1,2,...):1
Enter your message to the server (Client 1): Hello from client1
Waiting for a message from the The servr...

Received message from server: Hey from the server!
Do you want to send another message? (Y/N): y
Enter your message to the server (Client 1): How are u?
Waiting for a message from the The servr...
```

Figure 57:UDP client catch the response from server.

Here the client catches the message from server and (should answer the question if s/he need to continue this conversation if yes it will continue if not it will terminate this conv with the server and if run again and another client sent messages the server still in work and will save the last messages from all clients.).

The screenshot shows a Windows terminal window titled "Terminal Local (2)". The terminal output is as follows:

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

(.venv) C:\Network
python udp_server.py
The UDP Server is running on port 1727
Waiting for a message from the clients...

Message from Client1:Hello from client1
Enter your message to Client1: Hey from the server!

Messages from all client communications:
Message from Client1: Hello from client1
Waiting for a message from the clients...
Message from Client1:How are u?
Enter your message to Client1: good
Messages from all client communications:
Message from Client1: How are u?
Waiting for a message from the clients...
```

The terminal window has a dark theme. The status bar at the bottom shows "Network > udp\_server.py" and other system information like battery level, signal strength, and date/time.

Figure 58: response again and the list of last messages from 1st client.

Here the client sends another message, and the server replies to it then the client can continue the chat or not.

If the client chooses not to continue the conversation, then another client can send to the same server but still if client1 and 2 are still in connection then can a lot of clients send to the same server.

```
(.venv) diana@diana:~/PycharmProjects/pythonProject$ python udp_client.py
Enter your client ID (e.g :1,2,...):1
Enter your message to the server (Client 1): Hello from client1
Waiting for a message from the The servr...
Received message from server: Hey from the server!
Do you want to send another message? (Y/N): y
Enter your message to the server (Client 1): How are u?
Waiting for a message from the The servr...
Received message from server: good
Do you want to send another message? (Y/N): n
(.venv) diana@diana:~/PycharmProjects/pythonProject$ python udp_client.py
Enter your client ID (e.g :1,2,...):2
Enter your message to the server (Client 2): Hi from client2
Waiting for a message from the The servr...
```

Figure 59:Another client sends to server.

```
Message from Client1: hi from client1
Enter your message to Client1: hi
Sent reply to Client1: hi

Messages from all client communications:

Message from Client1: hi from client1
Waiting for a message from the clients...

Message from Client1: How are u?
Enter your message to Client1: good
Sent reply to Client1: good

Messages from all client communications:

Message from Client1: How are u?
Waiting for a message from the clients...

Message from Client2: Hi from client2
Enter your message to Client2: hi from server!
Sent reply to Client2: hi from server!

Messages from all client communications:

Message from Client1: How are u?
Message from Client2: Hi from client2
Waiting for a message from the clients...
```

Figure 60:Response to client2 and list the communication.

Here the server receives message from another client2 then the server replies to it (with list of all last messages from all clients shows to the server.)

```

ubuntu [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities PyCharm Professional Edition Aug 9 19:14
Terminal Local + v
Terminal - pythonProject
Do you want to send another message? (Y/N): ^C
Client shutting down...
(.venv) diana@diana-OptiPlex-5070:~/PycharmProjects/pythonProject$ python udp_client.py
Enter your client ID (e.g. 1,2,...): 1
Enter your message to the server (Client 1): hi
Waiting for a message from the server...
Received message from server: hi1
Do you want to send another message? (Y/N): y
Enter your message to the server (Client 1): kefak
Waiting for a message from the server...
Received message from server: good
Do you want to send another message? (Y/N): n
(.venv) diana@diana-OptiPlex-5070:~/PycharmProjects/pythonProject$ python udp_client.py
Enter your client ID (e.g. 1,2,...): 2
Enter your message to the server (Client 2): hi2
Waiting for a message from the server...
Received message from server: ok
Do you want to send another message? (Y/N): n
(.venv) diana@diana-OptiPlex-5070:~/PycharmProjects/pythonProject$ python udp_client.py
Enter your client ID (e.g. 1,2,...): 1
Enter your message to the server (Client 1): hello again
Waiting for a message from the server...
Received message from server: n
Do you want to send another message? (Y/N):

```

Figure 61: addition experiment to check validity of client.

Here I try to send from 2 clients, and I check if I can send message from client1 then from client2, then back to send from client1, look in the list of commendations update truly so the connection work will.

# 3-Part three Web Server:

## 3.1-Theory and procedure:

### Web Servers

A web server consists of computer software and supporting hardware that may receive requests sent over HTTP or HTTPS, the secure variation of the network protocol designed for distributing web content. When a user agent—typically a web browser or web crawler—uses HTTP to request a web page or other resource, the server replies with either the resource's content or an error message. This begins communication. If set up properly, a web server can also receive, and store resources delivered by the user agent.[15]

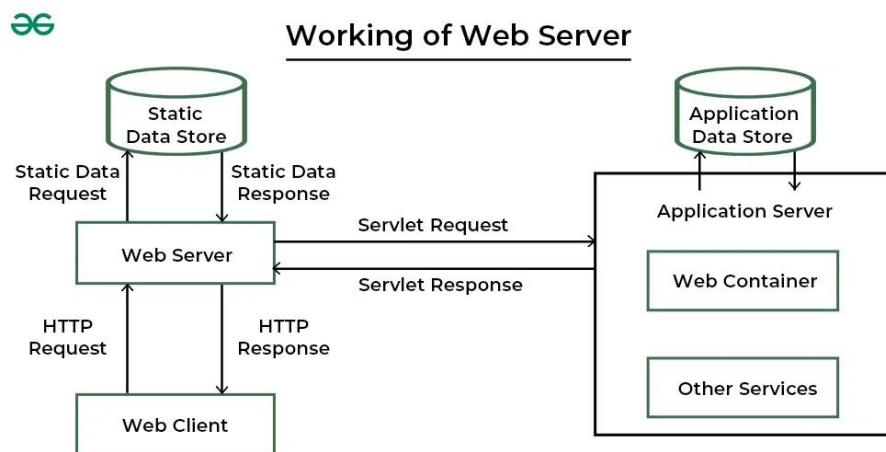


Figure 62::Web Server Architecture

### Socket Programming

In order to do activities on socket objects (such as connect, receive, send, and listen), we must first instantiate them in our code. Sockets are just special objects that we build in our software and have unique techniques for handling network data and connections. The network stack, a unique component of the operating system kernel that is in charge of overseeing network operations, is what those methods actually call your operating system kernel.[16]

## HTTP Protocol and Content Types

The resource's media type is indicated by the Content-Type header. The media type is a string that is sent with the file that describes its format. An image file, for instance, might have a media type of image/png, image/jpg, etc. [17]

In response, it provides the client with information about the kind of returned material. The kind of content that needs to load on the computer is discovered by the browser. When a browser receives a file's byte stream through the Content-type header, it performs a process known as MIME sniffing, in which it examines the stream it is receiving and loads the contents appropriately.[17]

## Error Handling

Programming and data processing both require error handling. It deals with how an application or system reacts to abnormalities or unforeseen circumstances while it is being used. Error handling in data processing might involve handling missing, corrupted, or inconsistent data.[18]

## Redirection with Status Codes

The numerical codes that a client, like a web browser, receives from the server to indicate the request's status are known as HTTP status codes. Based on the code's initial digit, we can group them. 3xx is the category that includes the redirection status codes. The client should route their request to the new location as indicated by these codes, which signal that a requested resource has been relocated. Redirection status codes come in a variety of forms, including 302, 307, and 308. Every one of these has a distinct function and a set of guidelines. It's critical to comprehend these codes in order to preserve the website's integrity and offer a positive customer experience.[19]

## Localization Support:

The purpose of localization services is to modify goods, information, and services to meet the functional, linguistic, and cultural needs of a particular target market or audience. In order to make sure that the product or content resonates with the local audience, reflects their customs, preferences, and cultural subtleties, and functions well within their particular region, the localization process entails adjusting several parts beyond basic translation.[20]

The screenshot shows the Microsoft Visual Studio Code interface with the Python file 'Server.py' open. The code implements a simple HTTP server using sockets. It listens on port 1727, receives requests from clients, extracts the requested path, handles it, and then closes the connection. A 'Developer PowerShell' terminal window is also visible at the bottom.

```
1  from socket import socket, AF_INET, SOCK_STREAM
2  import os
3  import urllib.parse
4
5  def main():
6      port = 1727
7      server_socket = socket(AF_INET, SOCK_STREAM)
8      server_socket.bind(('', port))
9      server_socket.listen(1)
10
11     print("Server listening on port:", port)
12
13     while True:
14         client_socket, address = server_socket.accept()
15         print("Client connected from:", address)
16
17         request = client_socket.recv(1024).decode()
18         start = request.find('/')
19         end = request.find(' ', start)
20         if end != -1:
21             request_path = request[start:end]
22         else:
23             request_path = request[start:]
24
25         print("Request path:", request_path)
26
27         handle_request(request_path, client_socket)
28
29         client_socket.close()
30         print("Connection closed")
31
32     def handle_request(request_path, client_socket):
33
34 100%  ✓ No issues found
```

Figure 63:Part1 of Server code

The screenshot shows the Microsoft Visual Studio Code interface with the Python file 'Server.py' open. This part of the code defines the 'handle\_request' function. It checks if the request path starts with '/image'. If so, it extracts the image name from query parameters, determines the content type based on the file extension (png, jpg, jpeg), and sends the file content back to the client. If the path does not start with '/image', it returns a 404 Not Found response. A 'Developer PowerShell' terminal window is also visible at the bottom.

```
31
32     def handle_request(request_path, client_socket):
33         base_directory = 'C:/Users/halaa/Desktop/NetworksProject'
34         content_type = "text/html"
35
36         if request_path.startswith('/image'):
37             # Extract image_name from query parameters
38             parsed_url = urllib.parse.urlparse(request_path)
39             query_params = urllib.parse.parse_qs(parsed_url.query)
40             image_name = query_params.get('image_name', [None])[0]
41
42             if image_name:
43                 file_path = os.path.join(base_directory, image_name)
44                 if os.path.exists(file_path):
45                     # Determine content type
46                     if file_path.lower().endswith('.png'):
47                         content_type = 'image/png'
48                     elif file_path.lower().endswith('.jpg') or file_path.lower().endswith('.jpeg'):
49                         content_type = 'image/jpeg'
50                     else:
51                         content_type = 'application/octet-stream'
52
53                     with open(file_path, 'rb') as file:
54                         data = file.read()
55                         client_socket.sendall(
56                             b"HTTP/1.1 200 OK\r\n"
57                             b"Content-Type: " + content_type.encode() + b"\r\n"
58                             b"Content-Length: " + str(len(data)).encode() + b"\r\n\r\n" + data
59                         )
60                     return
61                     send_not_found(client_socket)
62                 else:
```

Figure 64:Part2 of Server code

```

58         b"Content-Length: " + str(len(data)).encode() + b"\r\n\r\n" + data
59     )
60     return
61   send_not_found(client_socket)
62 else:
63   # Handle other types of requests
64   if request_path in ['/', '/main_en.html', '/en']:
65     file_path = os.path.join(base_directory, 'main_en.html')
66   elif request_path in ['/ar', '/main_ar.html']:
67     file_path = os.path.join(base_directory, 'main_ar.html')
68   elif request_path.endswith('.html'):
69     file_path = os.path.join(base_directory, request_path.lstrip('/'))
70   if not os.path.exists(file_path):
71     file_path = os.path.join(base_directory, 'main_en.html')
72   elif request_path.endswith('.css'):
73     file_path = os.path.join(base_directory, request_path.lstrip('/'))
74   if not os.path.exists(file_path):
75     file_path = os.path.join(base_directory, 'style.css')
76   content_type = 'text/css'
77 else:
78   file_path = os.path.join(base_directory, request_path.lstrip('/'))
79   if not os.path.exists(file_path):
80     send_not_found(client_socket)
81   return
82
83 if file_path:
84   try:
85     with open(file_path, 'rb') as file:
86       data = file.read()
87       client_socket.sendall(
88         b"HTTP/1.1 200 OK\r\n"
89         b"Content-Type: " + content_type.encode() + b"\r\n"
90         b"Content-Length: " + str(len(data)).encode() + b"\r\n\r\n" + data
91   )
92 except IOError:
93   send_not_found(client_socket)
94 def send_not_found(client_socket):
95   html_response = (
96     b"<html>"
97     b"<head><title>Error 404</title></head>"
98     b"<body>"
99     b"<h1>Error 404</h1>"
100    b"<p style='color: blue;'>The file is not found</p>"
101    b"<p><strong>Hala Mohammed - 1210312</strong></p>"
102    b"<p><strong>Diana Naseer - 1210363</strong></p>"
103    b"<p><strong>Rua Sour - 1221727</strong></p>"
104    b"<p><strong>Client IP and Port:</strong> " + client_socket.getpeername()[0].encode() + b" : " + str(client_socket.getpeername()[1]).encode() + b"</p>"
105    b"</body>"
106  )
107
108  client_socket.sendall(
109    b"HTTP/1.1 404 Not Found\r\n"
110    b"Content-Type: text/html\r\n"
111    b"Content-Length: " + str(len(html_response)).encode() + b"\r\n\r\n" + html_response
112  )
113
114 if __name__ == "__main__":
115   main()

```

Developer PowerShell

Ready

25°C سماء صافية

Search

halaa (untrusted)

8:21 PM 8/14/2024

Figure 65: Part 3 of Server code

```

84   try:
85     with open(file_path, 'rb') as file:
86       data = file.read()
87       client_socket.sendall(
88         b"HTTP/1.1 200 OK\r\n"
89         b"Content-Type: " + content_type.encode() + b"\r\n"
90         b"Content-Length: " + str(len(data)).encode() + b"\r\n\r\n" + data
91   )
92 except IOError:
93   send_not_found(client_socket)
94 def send_not_found(client_socket):
95   html_response = (
96     b"<html>"
97     b"<head><title>Error 404</title></head>"
98     b"<body>"
99     b"<h1>Error 404</h1>"
100    b"<p style='color: blue;'>The file is not found</p>"
101    b"<p><strong>Hala Mohammed - 1210312</strong></p>"
102    b"<p><strong>Diana Naseer - 1210363</strong></p>"
103    b"<p><strong>Rua Sour - 1221727</strong></p>"
104    b"<p><strong>Client IP and Port:</strong> " + client_socket.getpeername()[0].encode() + b" : " + str(client_socket.getpeername()[1]).encode() + b"</p>"
105    b"</body>"
106  )
107
108  client_socket.sendall(
109    b"HTTP/1.1 404 Not Found\r\n"
110    b"Content-Type: text/html\r\n"
111    b"Content-Length: " + str(len(html_response)).encode() + b"\r\n\r\n" + html_response
112  )
113
114 if __name__ == "__main__":
115   main()

```

File Edit View Git Project Analyze Tools Extensions Window Help | Search | Solution1

Sign in GitHub Copilot

mySiteSTID.html main\_ar.html style.css main\_en.html

91% No issues found

Developer PowerShell

Item(s) Saved

26°C سماء صافية

Search

halaa (untrusted)

8:06 PM 8/14/2024

Figure 66: Part 5 of Server code

This is a Python program for a simple web server via socket programming to handle incoming HTTP requests. The program listens for client connections on port 1727. Immediately after a client connects, it reads the request, identifies the path of the requested resource, and delivers its content using the handle\_request function. This function serves different file types such as HTML, CSS, and images from a given directory. If it is a request for an image, it will extract the image name in the URL and serve it if available. It serves up a custom 404 Not Found page in case it is a resource not found for smooth user experience.

```
# Handle other types of requests
if request_path in [ '/', '/main_en.html', '/en' ]:
    file_path = os.path.join(base_directory, 'main_en.html')
```

Figure 67:Handle Request

This part of the code checks for requests for one of the following : `^` , `/main\_en.html` , or `/en` , for instance, `localhost:1727/^` or `localhost:1727/en` . If one has one of these paths, then it sends in response one and the same file - `main\_en.html` , with `Content-Type` in the headers.

First, we run the server code. It appears on the terminal that it is listening to the port.

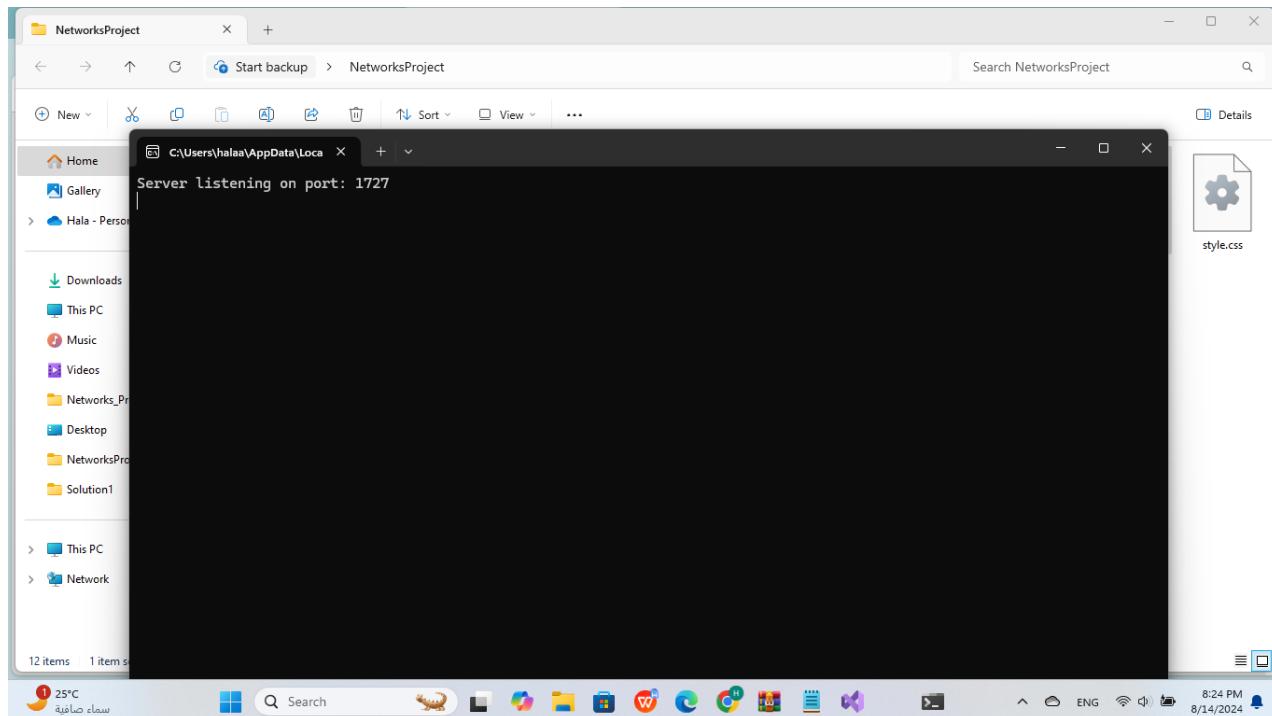


Figure 68:run the server

Second, we go to the browser and search for [http://localhost:1727/main\\_en.html](http://localhost:1727/main_en.html) and the following appears:

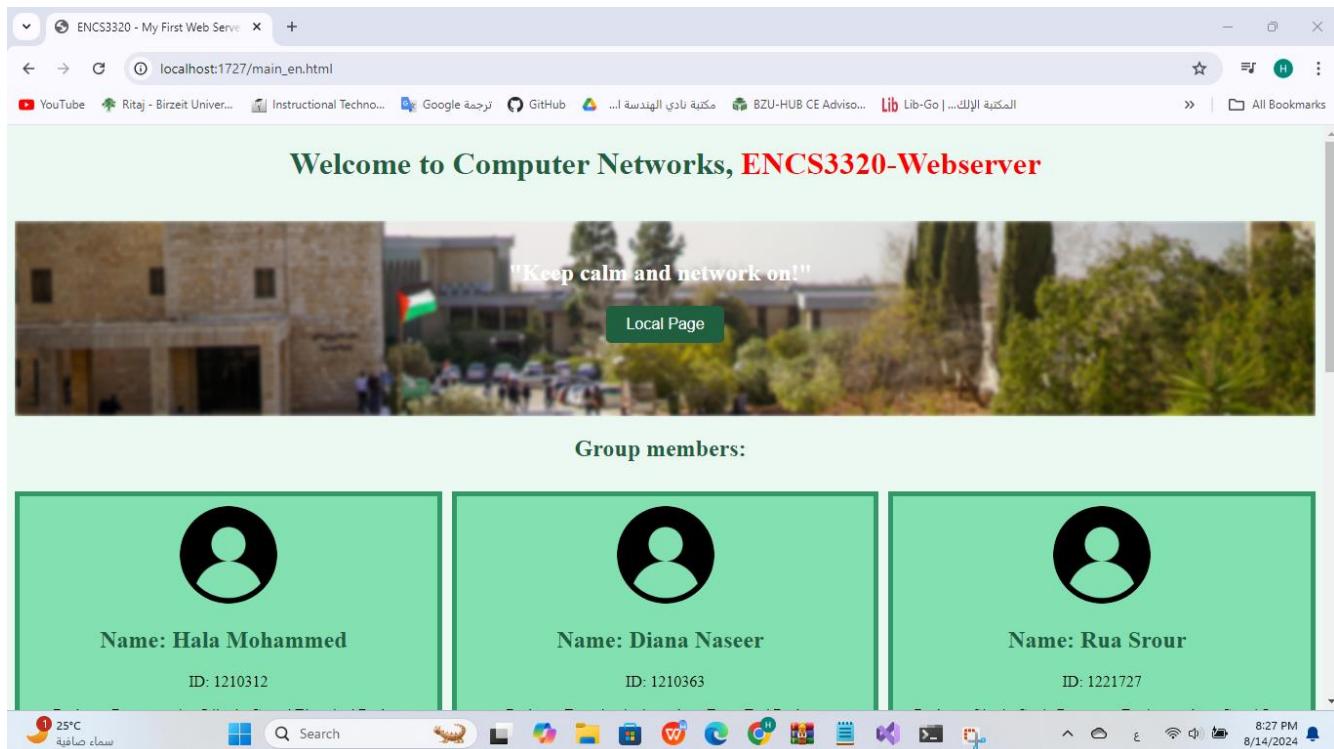


Figure 69:main\_en Page Part1

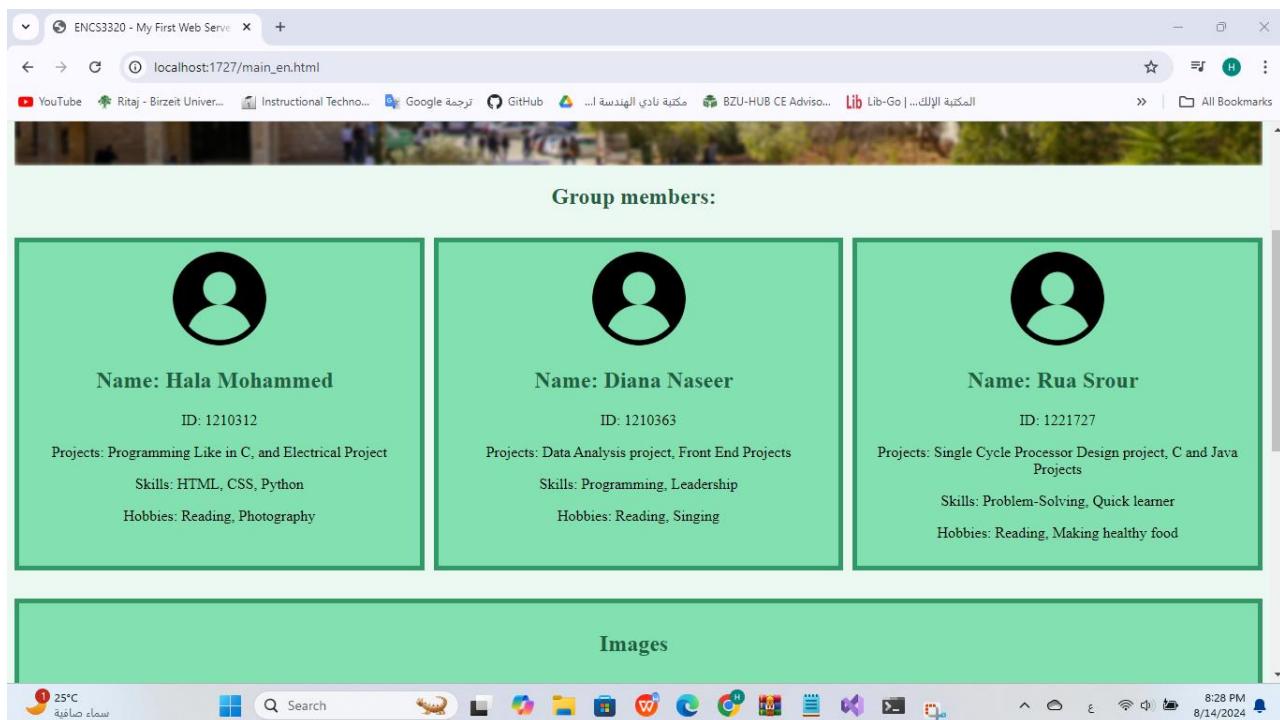


Figure 70:main\_en Page Part2

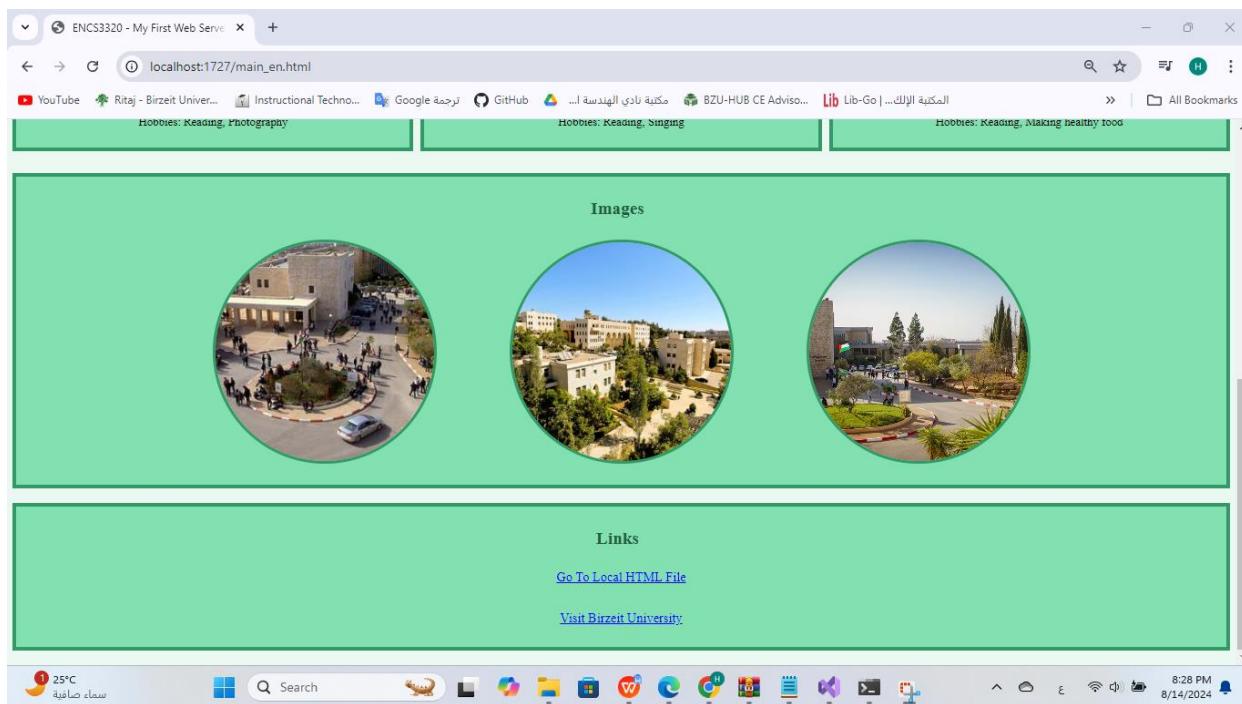


Figure 71:main\_en Page Part3

This button takes us to the local page.

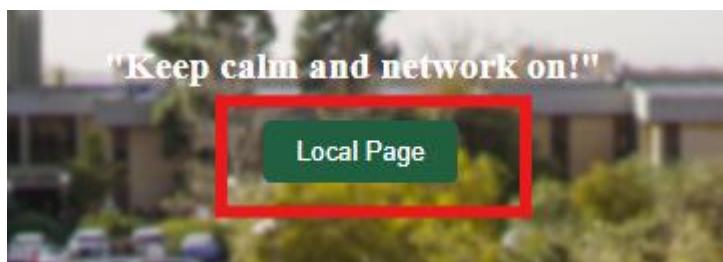


Figure 72:Local Page Button

This is the local page.

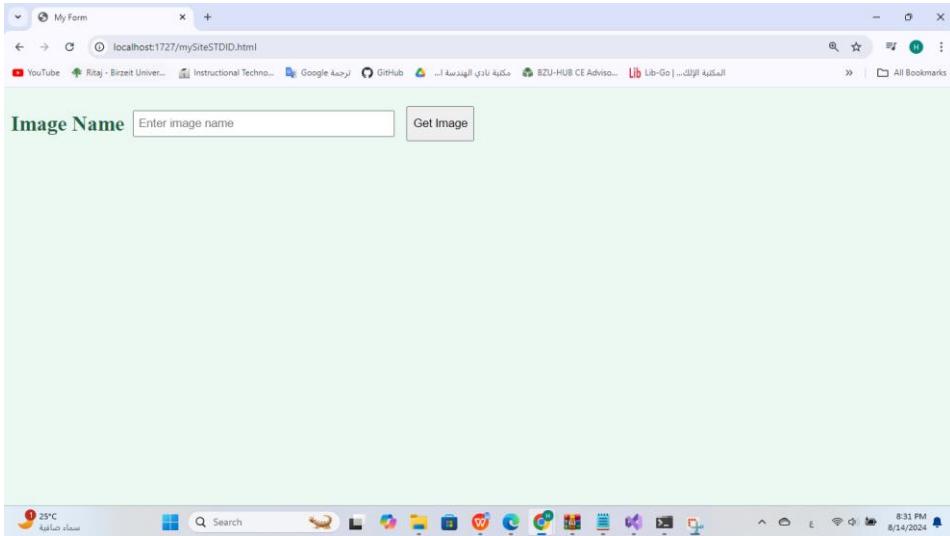


Figure 73:Local Page

Here we find two links, the first takes us to the local page like the previous button.



Figure 74:Links

The second link takes us to the next page.

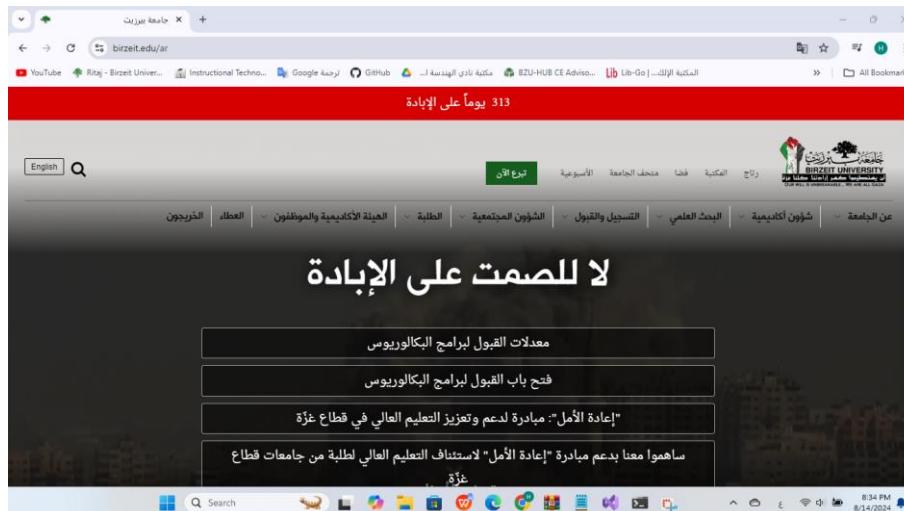


Figure 75:BZU Page

Also, after running the server, we can request the page in Arabic [http://localhost:1727/main\\_ar.html](http://localhost:1727/main_ar.html) and it will appear as the English page with all its details, with only changing the language.

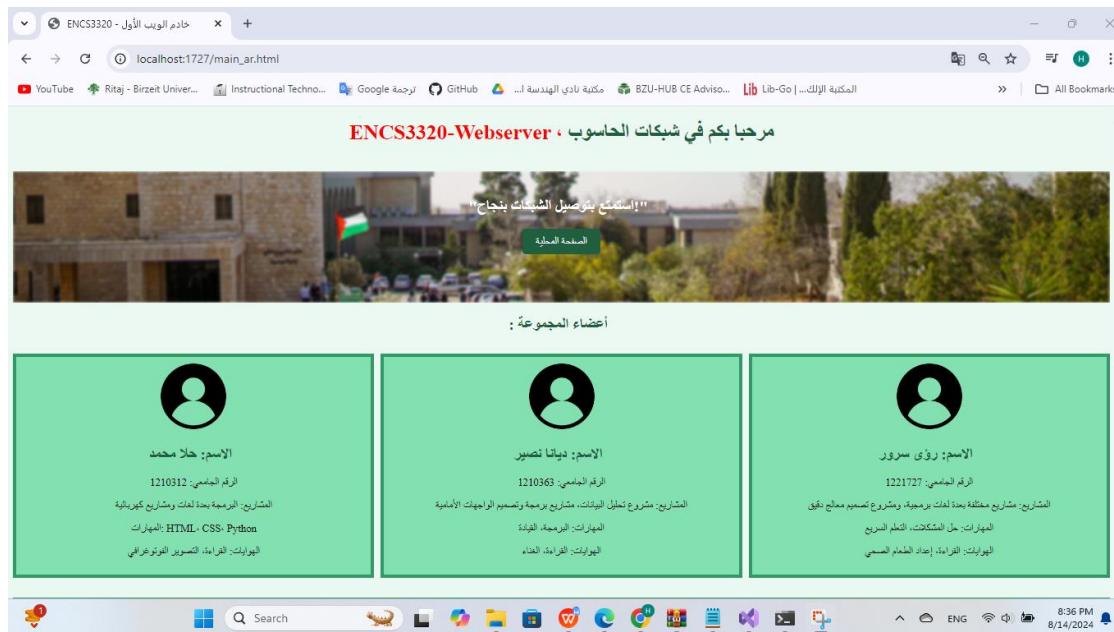


Figure 76:main\_ar Page Part1

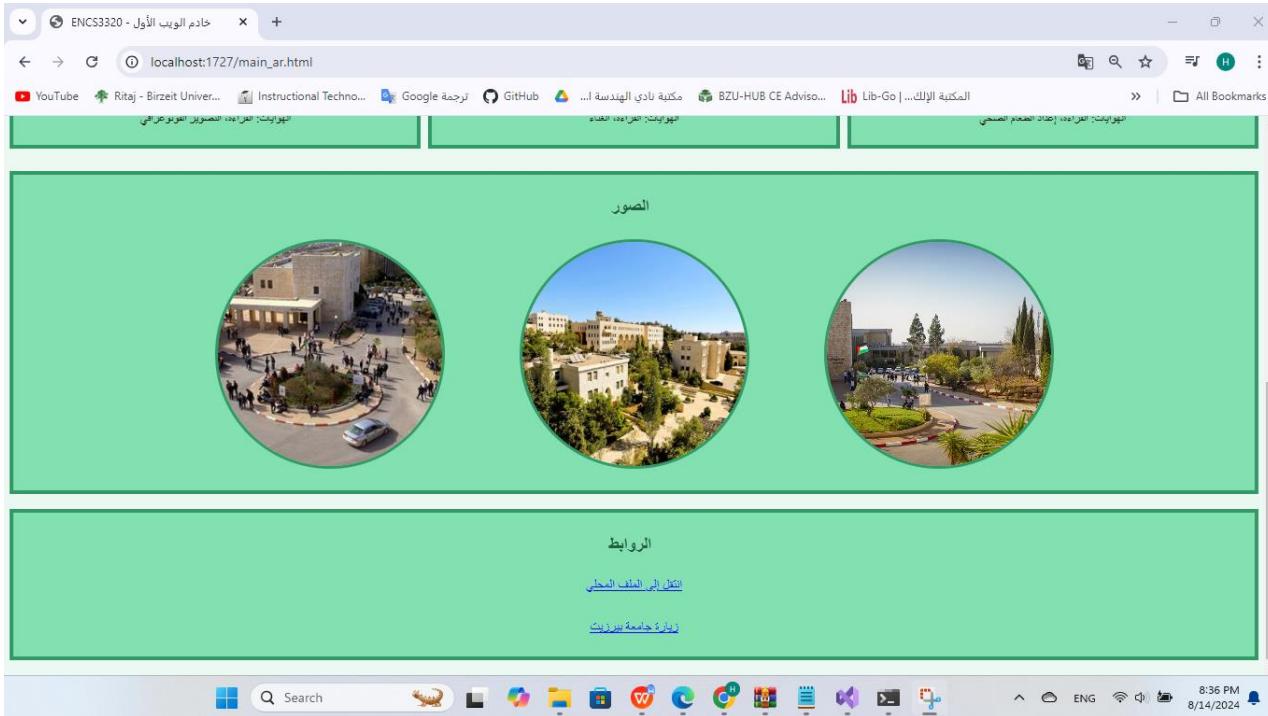


Figure 77:main\_ar Page Part2

The previous two pages and the local page were written using HTML and CSS was used to arrange the page layout.

```

File Edit View Git Project Analyze Tools Extensions Window Help | Search | Solution1
Server.py mySiteSTID0.html main_ar.html style.css main_en.html
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="UTF-8">
5     <meta name="viewport" content="width=device-width, initial-scale=1.0">
6     <title>ENCS3320 - My First Web Server</title>
7     <link rel="stylesheet" type="text/css" href="style.css">
8   </head>
9   <body>
10    <h1>Welcome to Computer Networks, <span style="color:red;">ENCS3320-Webserver</span></h1>
11    <br>
12    <div class="box_2">
13      <h2 style="color: white;">Keep calm and network on!</h2>
14      <button onclick="window.location.href='mySiteSTID0.html'">Local Page</button>
15    </div>
16    <div class="group-members-heading">
17      <h2>Group members:</h2>
18    </div>
19    <div class="box-info">
20      <div class="rec-user">
21        
22        <div>
23          <h3>Name: Hala Mohammed</h3>
24          <p>ID: 1210312</p>
25          <p>Projects: Programming Like in C, and Electrical Project</p>
26          <p>Skills: HTML, CSS, Python</p>
27          <p>Hobbies: Reading, Photography</p>
28        </div>
29      </div>
30    </div>
31    <div class="box">
32      
33    </div>
34  </div>
35  <div>
36    No issues found
37  </div>
Developer PowerShell
Ready

```

Figure 78:main\_en Code Part1

File Edit View Git Project Analyze Tools Extensions Window Help Search Solution1

Server.py mySiteSTDID.html main\_ar.html style.css main\_en.html

```

29      <p>Hobbies: Reading, Photography</p>
30    </div>
31    <div class="box">
32      
33      <h2>Name: Diana Naseer</h2>
34      <p>ID: 1210363</p>
35      <p>Projects: Data Analysis project, Front End Projects</p>
36      <p>Skills: Programming, Leadership</p>
37      <p>Hobbies: Reading, Singing</p>
38    </div>
39    <div class="box">
40      
41      <h2>Name: Rua Srour</h2>
42      <p>ID: 1221727</p>
43      <p>Projects: Single Cycle Processor Design project, C and Java Projects</p>
44      <p>Skills: Problem-Solving, Quick learner</p>
45      <p>Hobbies: Reading, Making healthy food</p>
46    </div>
47  </div>
48
49  <div class="full-width">
50    <h2>Images</h2>
51    <br>
52    <div class="image-container">
53      
54      
55      
56    </div>
57    <br>
58  </div>
59
60  <div class="full-width">
61    <h2>Links</h2>
62    <p><a href="mySiteSTDID.html">Go To Local HTML File</a></p>
63    <p><a href="https://www.birzeit.edu/">Visit Birzeit University</a></p>
64  </div>
65
66</body>
67</html>
```

100% No issues found Ln: 28 Ch: 45 SPC CRLF

Developer PowerShell

Ready Search ENG 8:40 PM 8/14/2024 halaa (untrusted)

Figure 79:main\_en Code Part2

File Edit View Git Project Analyze Tools Extensions Window Help Search Solution1

Server.py mySiteSTDID.html main\_ar.html style.css main\_en.html

```

54      
55      
56    <br>
57  </div>
58
59  <div class="full-width">
60    <h2>Links</h2>
61    <p><a href="mySiteSTDID.html">Go To Local HTML File</a></p>
62    <p><a href="https://www.birzeit.edu/">Visit Birzeit University</a></p>
63  </div>
64
65</body>
66</html>
```

100% No issues found Ln: 28 Ch: 45 SPC CRLF

Developer PowerShell

Ready Search ENG 8:40 PM 8/14/2024 halaa (untrusted)

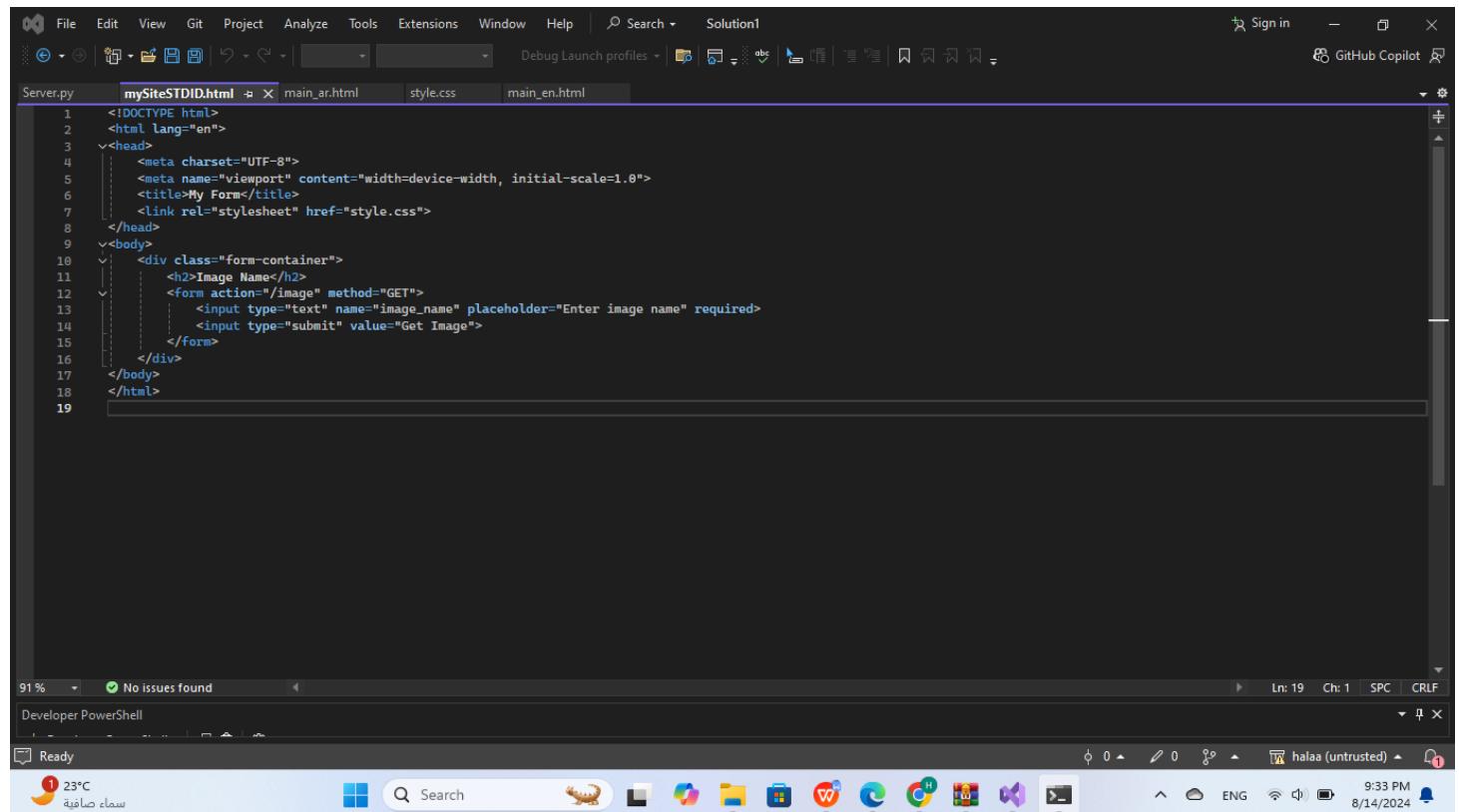
Figure 80:main\_en Code Part3

The HTML file is describe a web page for Computer Networks project . The first description is the title of the page, which is "ENCS3320 - My First Web Server". The main content in the body includes a heading for welcoming the user to the course, displaying the course name in red, and a quotation related to motivation with a button. Upon clicking the latter, the user will be able to access some HTML file available locally `mySiteSTDID.html`. Also It contains a section to our group members , which shows photos, names, IDs, projects, skills, and hobbies; also, it contains a part that represents some pictures of Birzeit University and another part that contains the local HTML file and the website of Birzeit University.

The `<meta charset="UTF-8">' tag will let the server know that the webpage is in UTF-8 encoding, supporting a wide range of characters and symbols from most languages. This helps text display correctly across different browsers and devices. The class `<meta name="viewport" content="width=device-width, initial-scale=1.0">` sets the value of width to the width of the devices and sets the initial zoom level to 1.0. This attribute does, more or less, what one might expect from its name: it changes the mobile devices' scaling of a webpage. Thus, this will help develop a responsive page to offer an excellent user experience since the page is going to be viewable and well-formatted on all screen sizes.

The Arabic version of the Main page is written exactly like the English version with the language changed.

## The Local Page HTML code



A screenshot of Microsoft Visual Studio Code showing the HTML code for a local page. The code is contained in a file named 'mySiteSTDID.html'. The code includes a DOCTYPE declaration, an HTML tag with a lang attribute of 'en', a head section with meta tags for charset ('UTF-8') and viewport ('width=device-width, initial-scale=1.0'), a title ('My Form'), and a link to a stylesheet ('style.css'). The body section contains a form for searching an image, with a class attribute of 'form-container'. The form has a heading ('Image Name'), a text input field ('image\_name') with a placeholder ('Enter image name' and 'required' attribute), and a submit button ('Get Image'). The code is numbered from 1 to 19.

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>My Form</title>
<link rel="stylesheet" href="style.css">
</head>
<body>
<div class="form-container">
<h2>Image Name</h2>
<form action="/image" method="GET">
<input type="text" name="image_name" placeholder="Enter image name" required>
<input type="submit" value="Get Image">
</form>
</div>
</body>
</html>
```

Figure 81: The Local Page HTML code

On the page <http://localhost:1727/mySiteSTDID.html> the purpose is for the user to search for an image with several different extensions and if it is not there or an error occurs, an error message will certainly be displayed.

If I search for image that is exist (jpg)

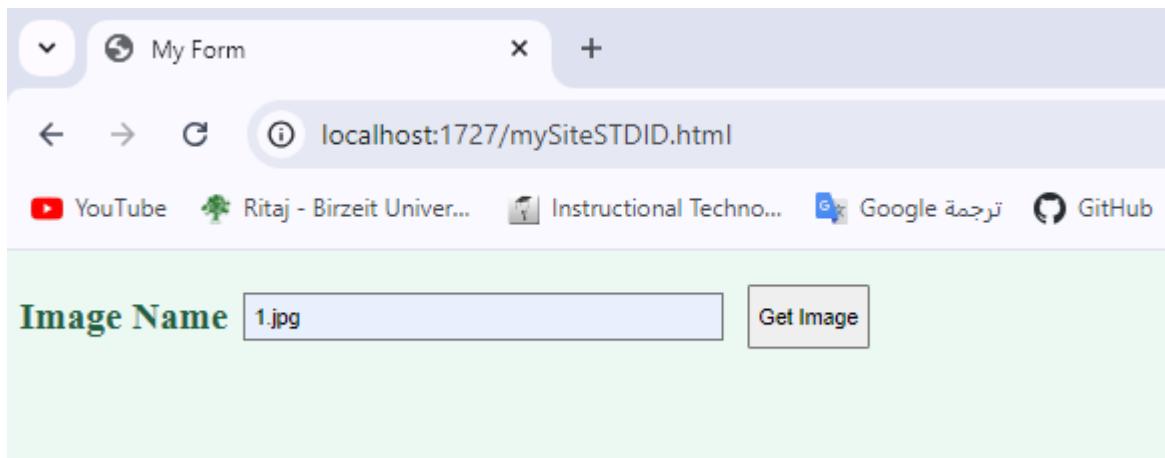


Figure 82:search for image that is exist (jpg)

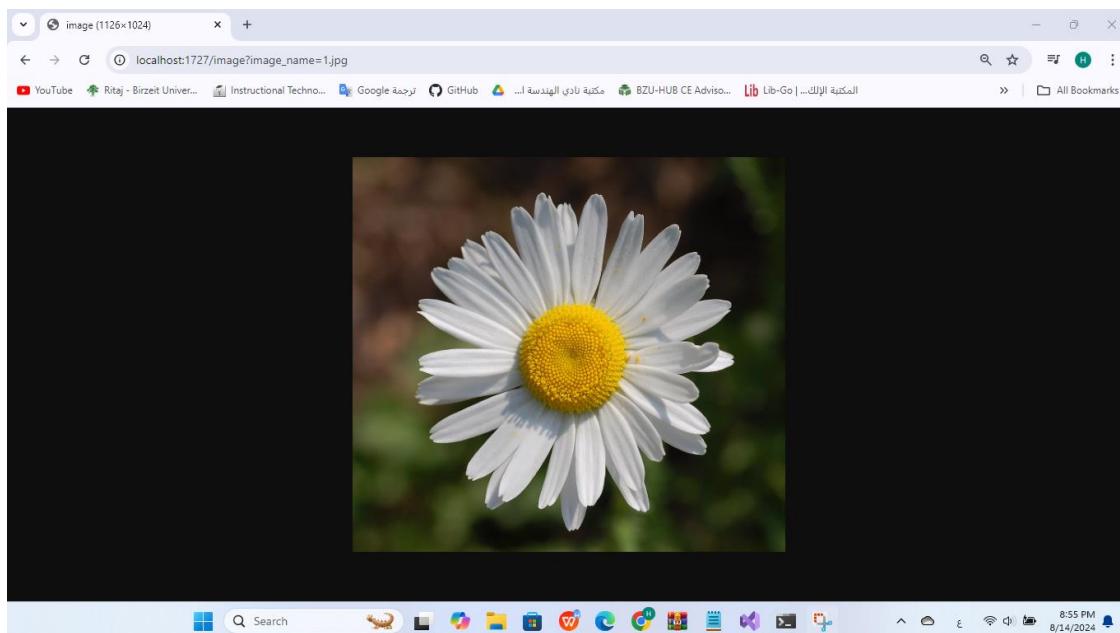


Figure 83:Result of search for image that is exist (jpg)

If I search for image that is exist (png)

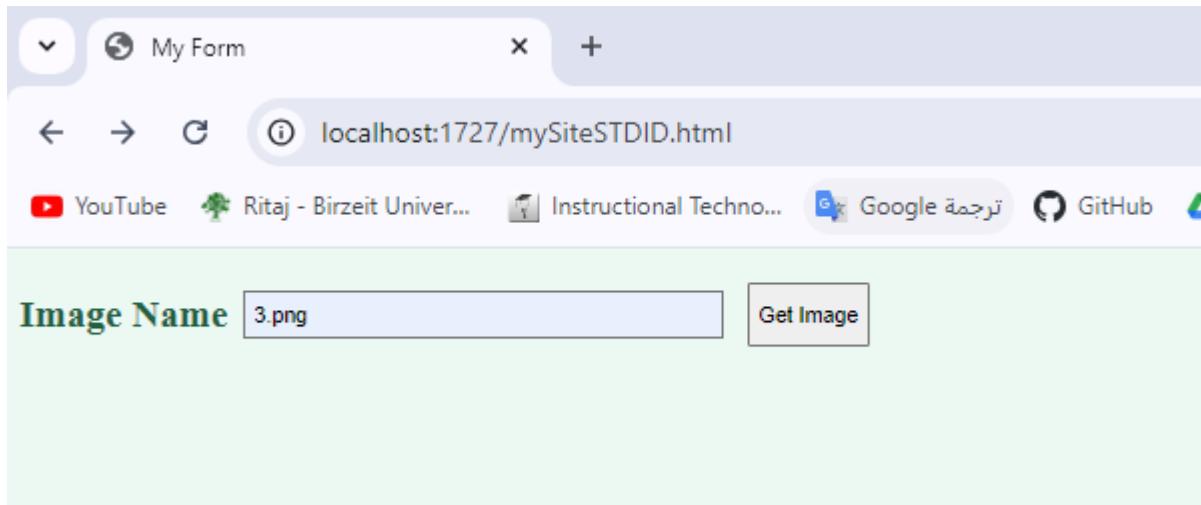


Figure 84:search for image that is exist (png)

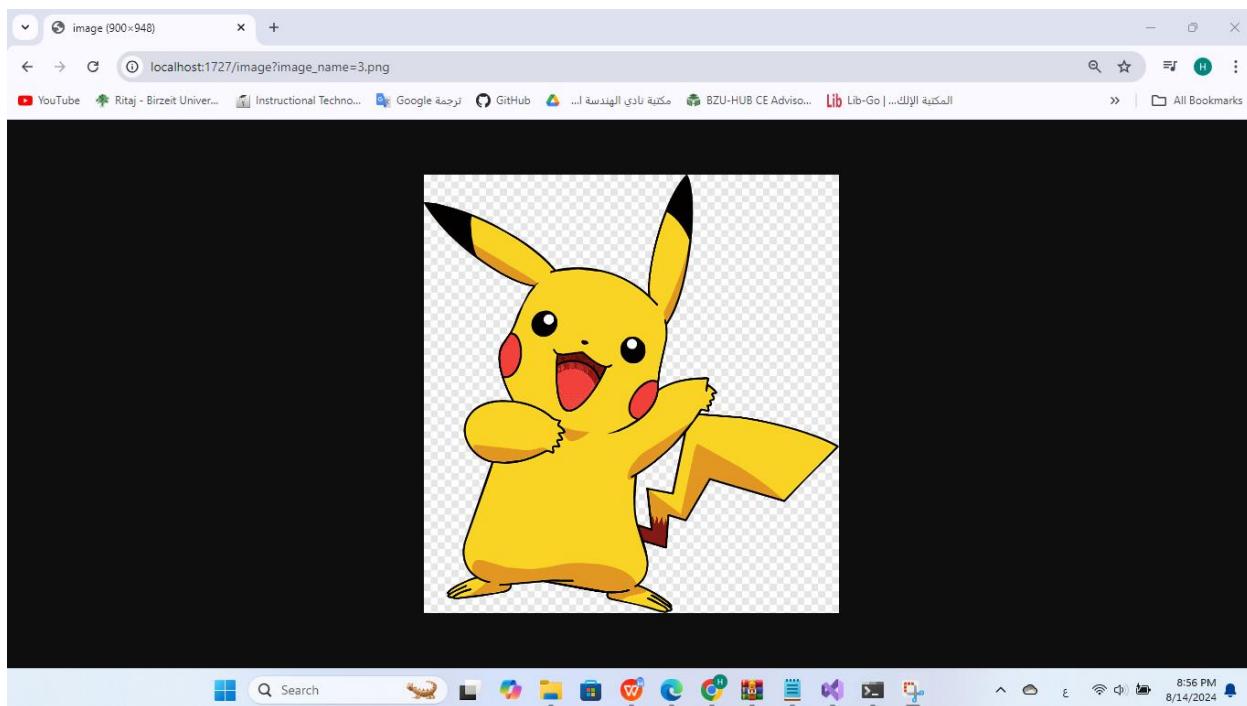


Figure 85:Result of search for image that is exist (png)

If I search for image that is not exist

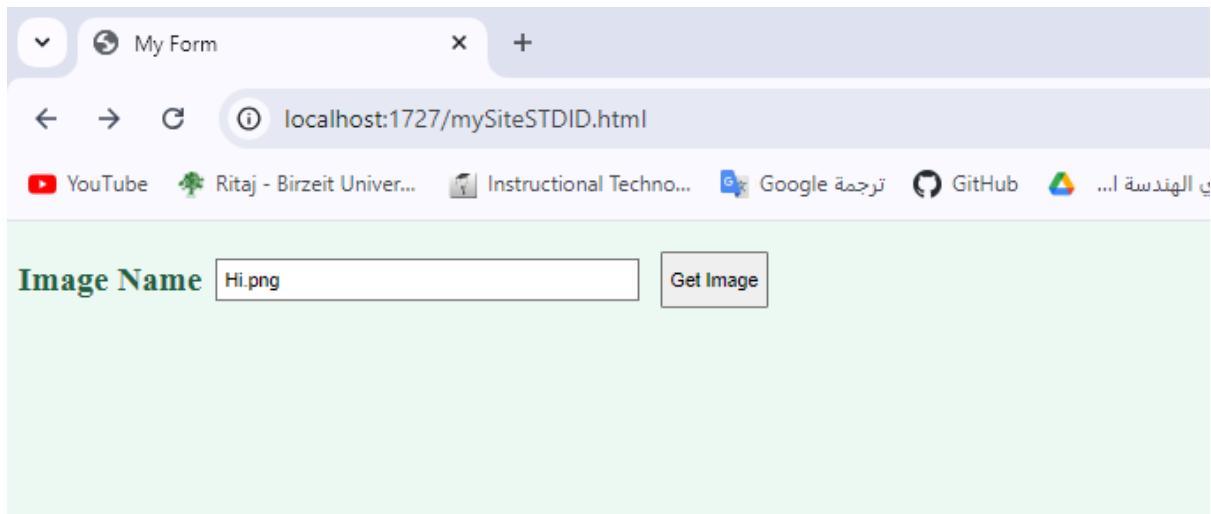


Figure 86:search for image that is not exist



## Error 404

The file is not found

Hala Mohammed - 1210312

Diana Naseer - 1210363

Rua Srour - 1221727

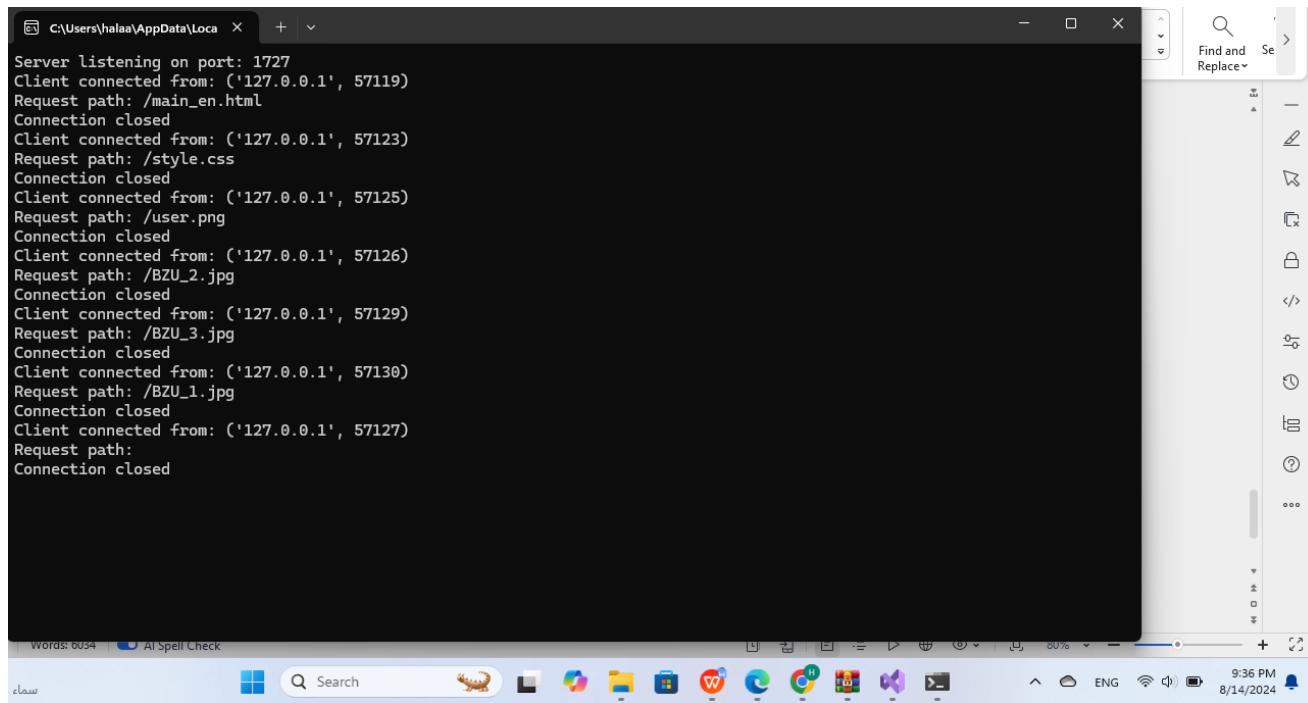
Client IP and Port: 127.0.0.1 : 56825



Figure 87:Result of search for image that is not exist

### 3.2- Results and Discussions:

When I ask for [http://localhost:1727/main\\_en.html](http://localhost:1727/main_en.html)



A screenshot of a code editor window displaying a log of server requests. The log entries are as follows:

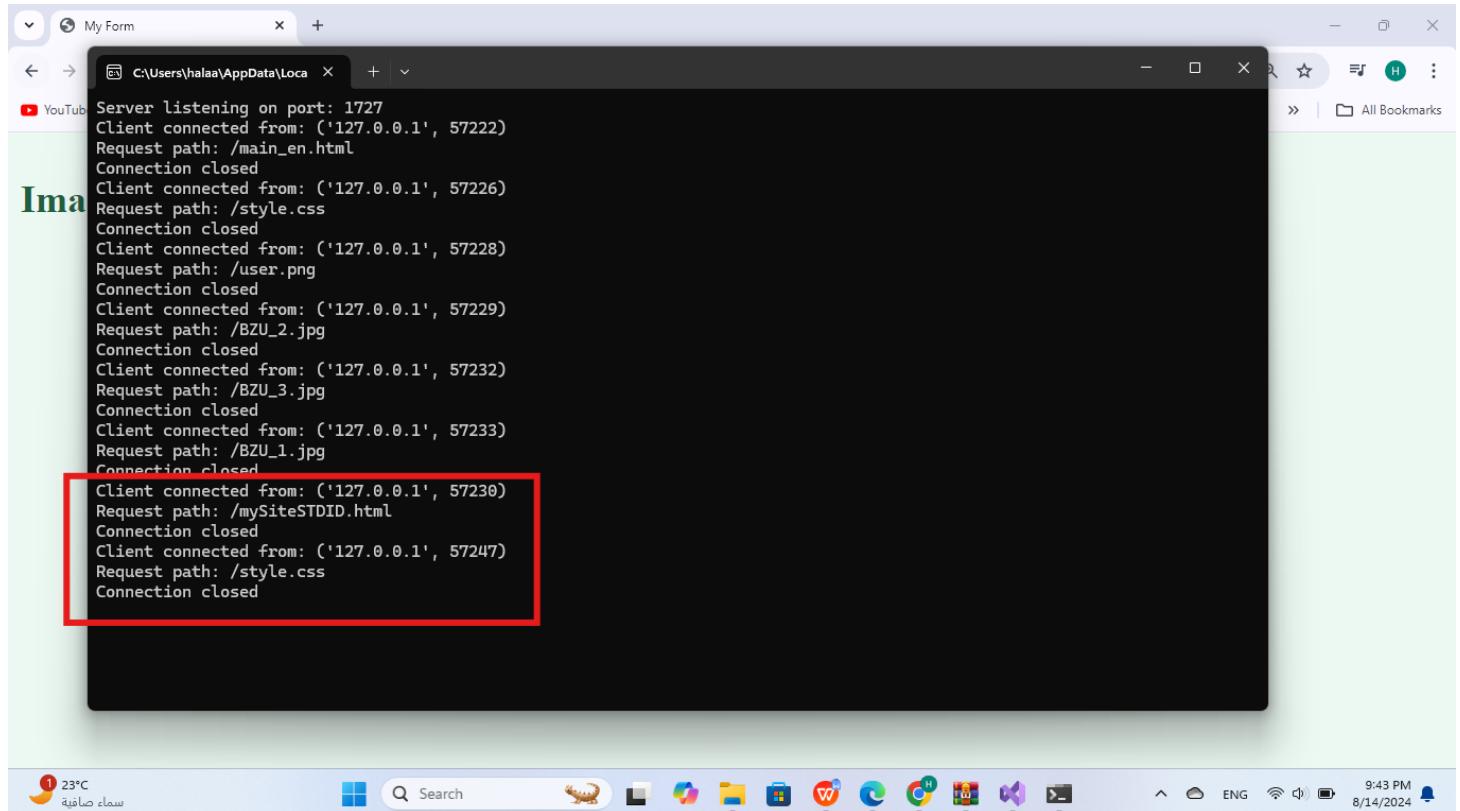
```
Server listening on port: 1727
Client connected from: ('127.0.0.1', 57119)
Request path: /main_en.html
Connection closed
Client connected from: ('127.0.0.1', 57123)
Request path: /style.css
Connection closed
Client connected from: ('127.0.0.1', 57125)
Request path: /user.png
Connection closed
Client connected from: ('127.0.0.1', 57126)
Request path: /BZU_2.jpg
Connection closed
Client connected from: ('127.0.0.1', 57129)
Request path: /BZU_3.jpg
Connection closed
Client connected from: ('127.0.0.1', 57130)
Request path: /BZU_1.jpg
Connection closed
Client connected from: ('127.0.0.1', 57127)
Request path:
Connection closed
```

The window has a dark theme and includes a toolbar with various icons and a status bar at the bottom showing the date and time.

Figure 88: Server Result for [http://localhost:1727/main\\_en.html](http://localhost:1727/main_en.html)

These log entries show that some client is connecting from 127.0.0.1 and sending various requests for resources on the server. The IP address and the port number of the client are maintained for each of the connections, and it places the requested particular resource and serves it back, such as main\_en.html, style.css and various image files, including user.png and others. Here, the server looks for the appropriate file in its directory to each request and sends it back to the client. The "Connection closed" notification indicates that a server closes a connection after serving each request. For [http://localhost:1727/main\\_ar.html](http://localhost:1727/main_ar.html) is the same

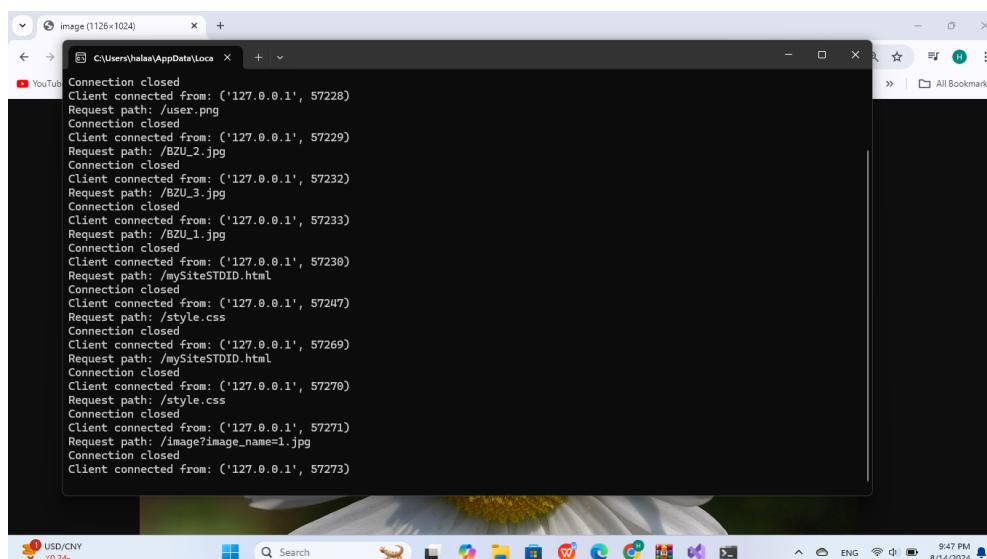
When I ask for <http://localhost:1727/mySiteSTDID.html> OR <https://www.birzeit.edu/ar>



```
Server listening on port: 1727
Client connected from: ('127.0.0.1', 57222)
Request path: /main_en.html
Connection closed
Client connected from: ('127.0.0.1', 57226)
Request path: /style.css
Connection closed
Client connected from: ('127.0.0.1', 57228)
Request path: /user.png
Connection closed
Client connected from: ('127.0.0.1', 57229)
Request path: /BZU_2.jpg
Connection closed
Client connected from: ('127.0.0.1', 57232)
Request path: /BZU_3.jpg
Connection closed
Client connected from: ('127.0.0.1', 57233)
Request path: /BZU_1.jpg
Connection closed
Client connected from: ('127.0.0.1', 57230)
Request path: /mySiteSTDID.html
Connection closed
Client connected from: ('127.0.0.1', 57247)
Request path: /style.css
Connection closed
```

Figure 89: Web Server Result2

When I search for exist image



```
Connection closed
Client connected from: ('127.0.0.1', 57228)
Request path: /user.png
Connection closed
Client connected from: ('127.0.0.1', 57229)
Request path: /BZU_2.jpg
Connection closed
Client connected from: ('127.0.0.1', 57232)
Request path: /BZU_3.jpg
Connection closed
Client connected from: ('127.0.0.1', 57233)
Request path: /BZU_1.jpg
Connection closed
Client connected from: ('127.0.0.1', 57230)
Request path: /mySiteSTDID.html
Connection closed
Client connected from: ('127.0.0.1', 57247)
Request path: /style.css
Connection closed
Client connected from: ('127.0.0.1', 57269)
Request path: /mySiteSTDID.html
Connection closed
Client connected from: ('127.0.0.1', 57270)
Request path: /style.css
Connection closed
Client connected from: ('127.0.0.1', 57271)
Request path: /image?image_name=1.jpg
Connection closed
Client connected from: ('127.0.0.1', 57273)
```

Figure 90: Web Server Result 3

When I search for not exist image

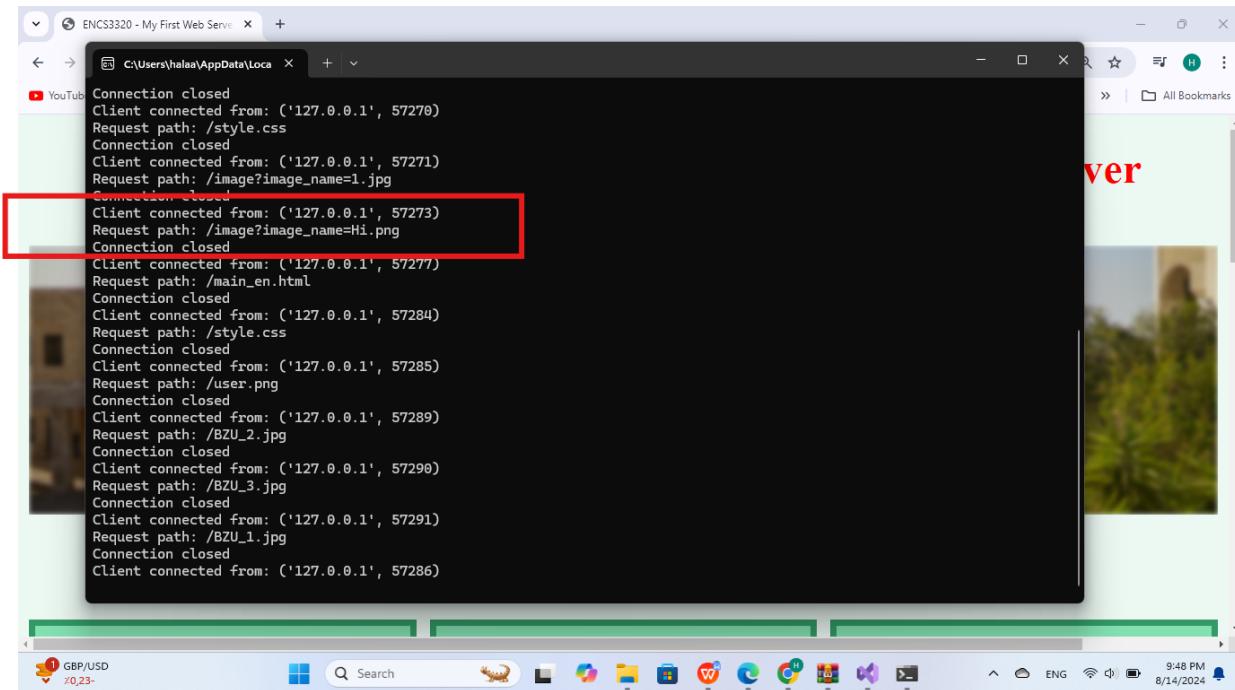
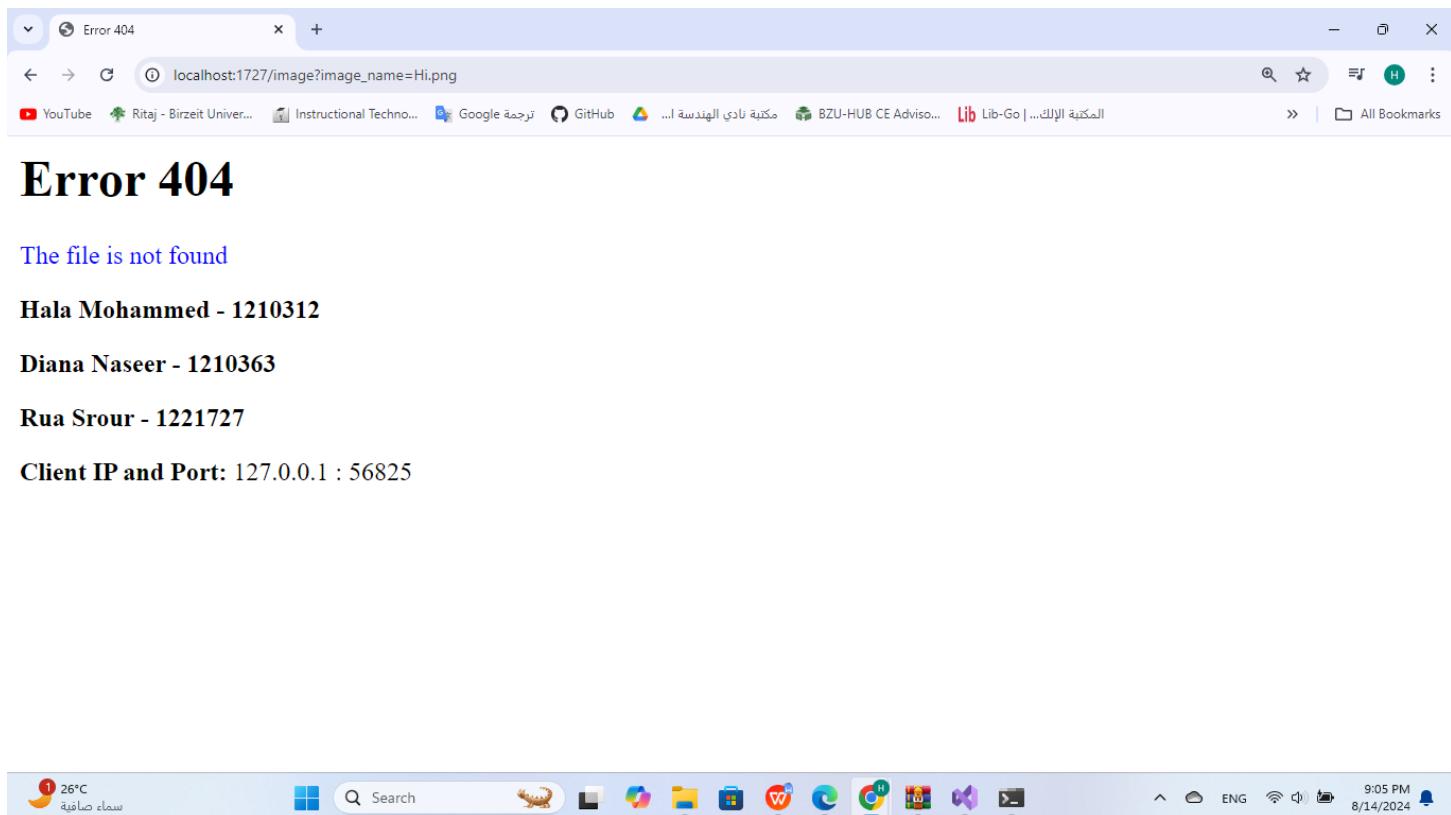


Figure 91:Web Server Result 4

This log shows that a connection from the client at 127.0.0.1 is made to the server, meaning that these are local requests. It is supposed to fetch back and serve an image file with the name `1.jpg` upon receiving a request for `/image? image\_name=1.jpg` from the initial connection. Following this, it responds to the request and closes the connection. The second connection shows a similar request for `/image? image\_name=Hi.png`, asking the server to return the image file `Hi.png`. Once this request is processed, the connection is also closed.



*Figure 92:Error Page*

The error page displaying "Client IP and Port: 127.0.0.1 : 57317" gives details about the source of the request. It is evident that it originated from the local machine by noticing the IP address `127.0.0.1`. That is, the request came from the same computer on which the server is running. The port number `57317` gets dynamically allocated to the clients' machine for this particular connection.

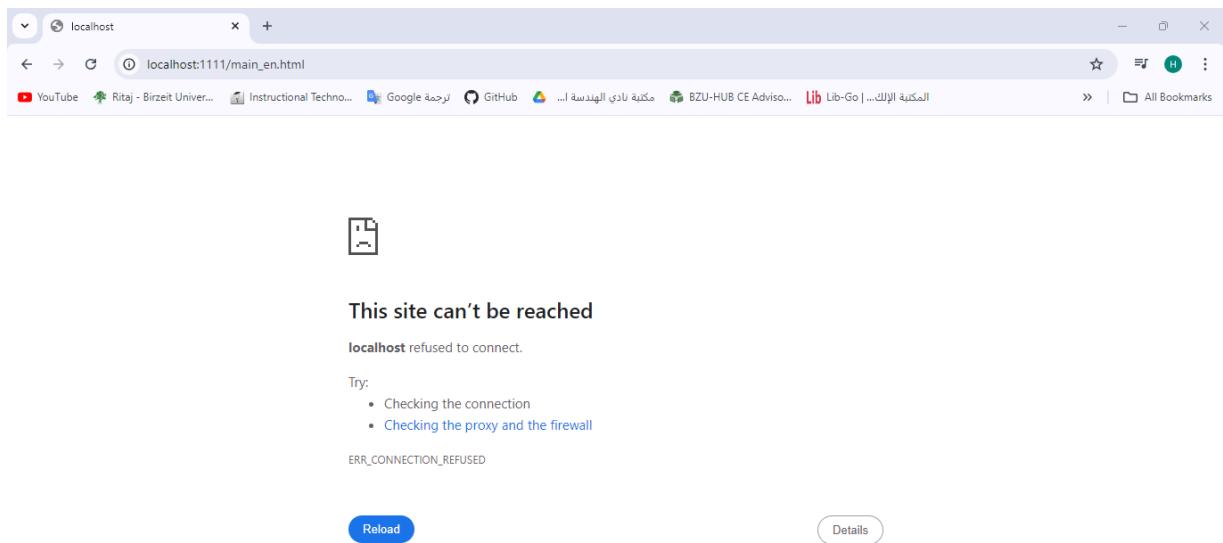
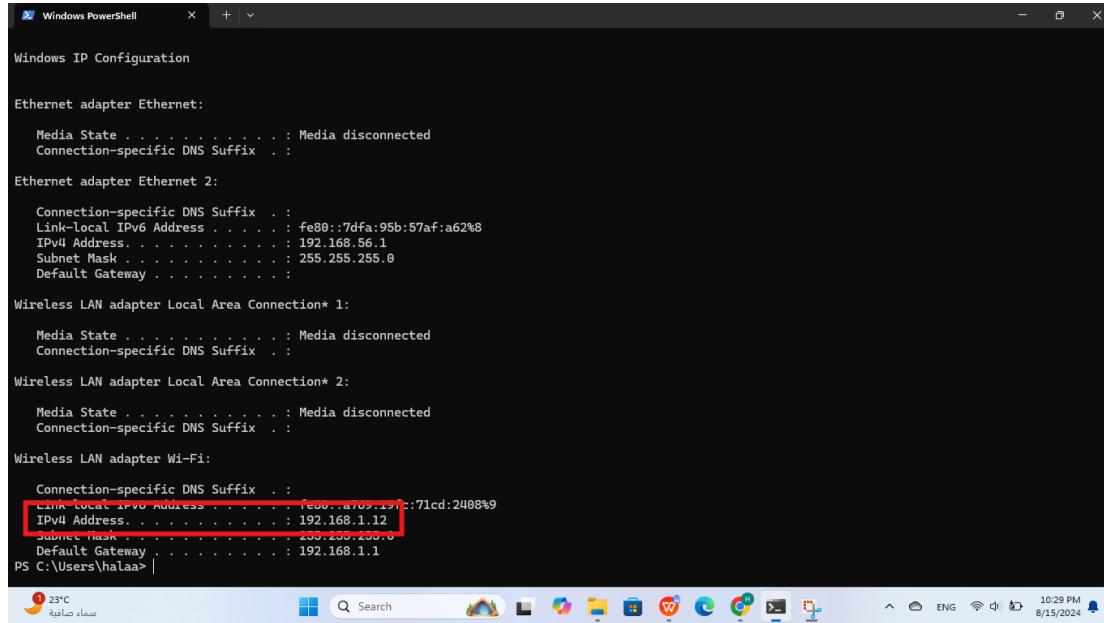


Figure 93: Try a new Port

Now, in the case of setting the server port, one should remember that the declared port number within the server code is very critical for proper communication. For example, in our case server code was set to listen in a port 1727, but here we are trying to access using port 1111 similar to `http://localhost:1111/main_en.html`. The connection failed. This will not work because a server does not have an instruction to listen on port 1111; it, instead, is awaiting requests on port 1727.

## Test web server on another device

First we need the IPv4 for the first device, we can get it by searching for ipconfig on the terminal.



```
Windows PowerShell

Windows IP Configuration

Ethernet adapter Ethernet:
  Media State . . . . . : Media disconnected
  Connection-specific DNS Suffix . :

Ethernet adapter Ethernet 2:
  Connection-specific DNS Suffix . :
  Link-local IPv6 Address . . . . . : fe80::7dfa:95b:57af:a62%1
  IPv4 Address . . . . . : 192.168.56.1
  Subnet Mask . . . . . : 255.255.255.0
  Default Gateway . . . . . :

Wireless LAN adapter Local Area Connection* 1:
  Media State . . . . . : Media disconnected
  Connection-specific DNS Suffix . :

Wireless LAN adapter Local Area Connection* 2:
  Media State . . . . . : Media disconnected
  Connection-specific DNS Suffix . :

Wireless LAN adapter Wi-Fi:
  Connection-specific DNS Suffix . :
  Link-local IPv6 Address . . . . . : fe80::a769:95ff%1:71cd:2408%9
  IPv4 Address . . . . . : 192.168.1.12
  Subnet Mask . . . . . : 255.255.255.0
  Default Gateway . . . . . : 192.168.1.1
PS C:\Users\halaa>
```

Figure 94: Test web server on another device Part1

Then we run the server code

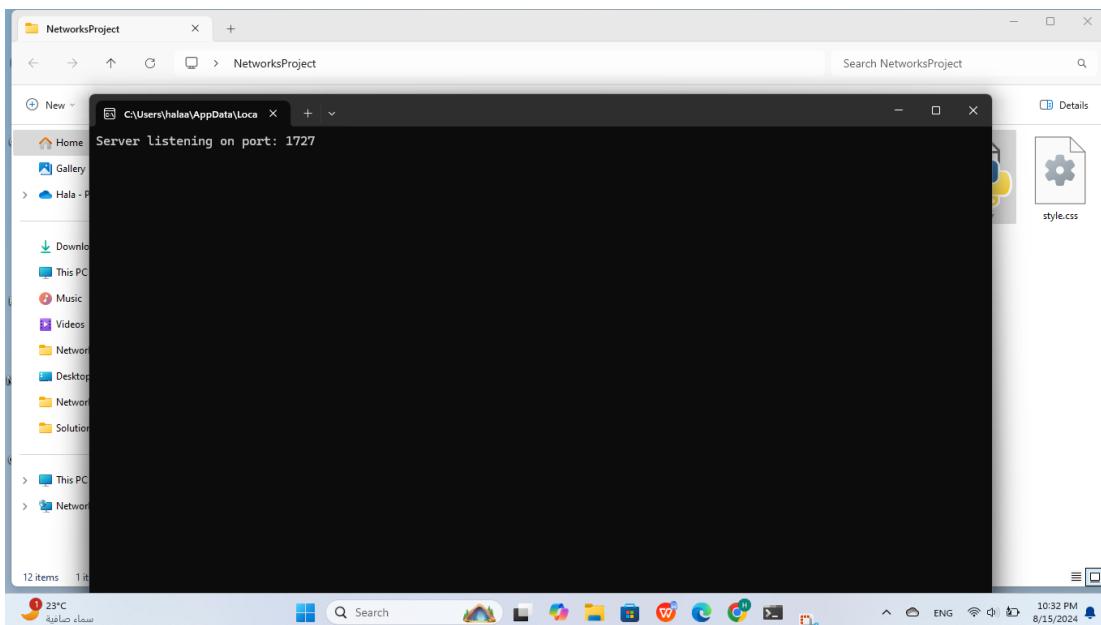


Figure 95: Test web server on another device Part2

Then we search for the page on the other device using the port number and the IPv4 number, with making sure that the two devices are connected to the same Internet network.

[http://192.168.1.12:1727/main\\_en.html](http://192.168.1.12:1727/main_en.html)

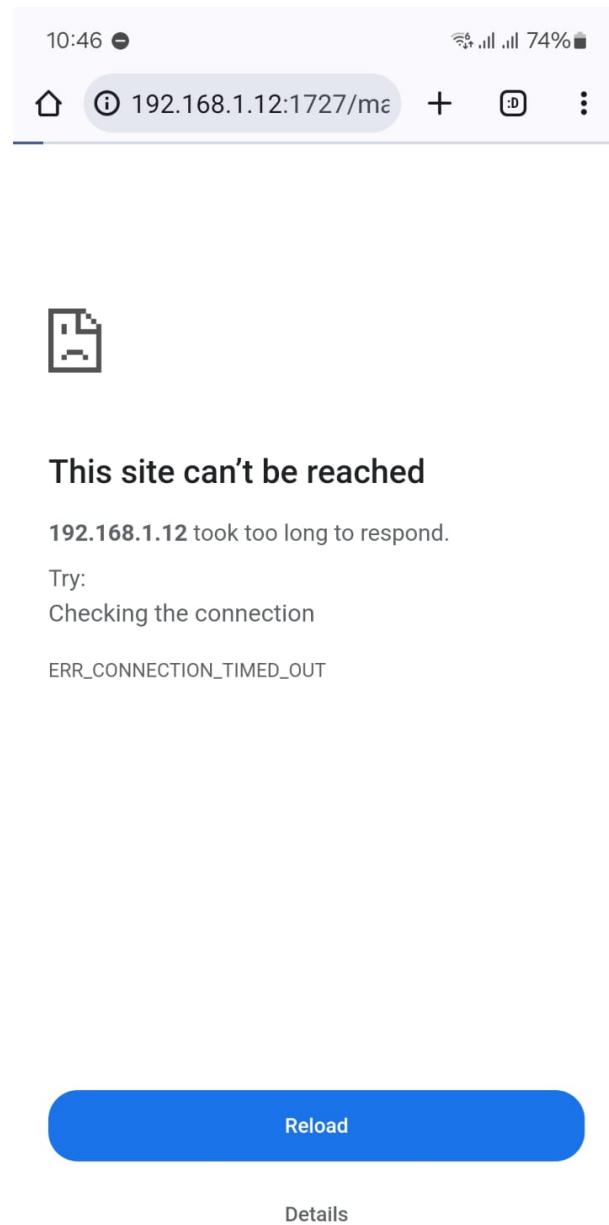


Figure 96: Test web server on another device Part3

The page did not load and was tested on several other devices with the same result .

I searched for a solution to this problem and make sure that the firewall on my computer is not blocking incoming connections on port 1727

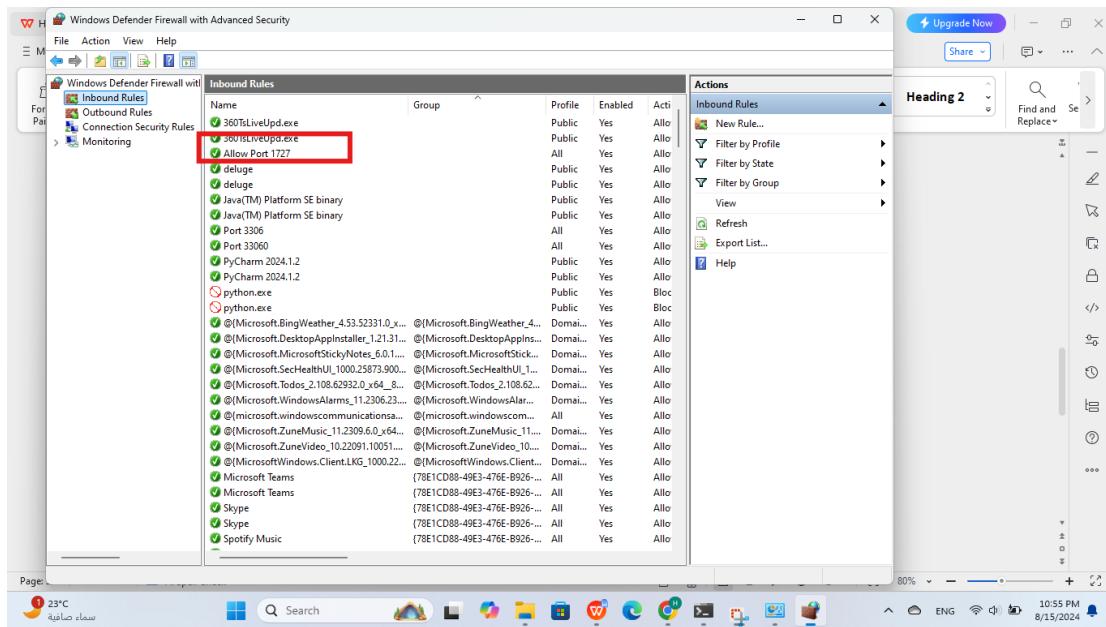


Figure 97::Test web server on another device Part4

I tried more than once and it didn't work):

## References:

[1]: lenovo.

<https://www.lenovo.com/in/en/glossary/command/?orgRef=https%253A%252F%252Fwww.google.com%252F>

[Accessed on 4 August 2024 at 10:01 PM].

=====

[2]: tech target. [online image]

[Accessed on 8 August 2024 at 5:37PM].

<https://www.techtarget.com/whatis/definition/Wireshark#:~:text=Wireshark%20is%20a%20widely%20used,ensure%20smooth%20operations%20and%20security.>

=====

[3]: Quora [online image]

[Accessed on 8 August 2024 at 5:46PM].

<https://www.quora.com/Why-is-Wireshark-useful>

=====

[4]: BGP view.

[Accessed on 8 August 2024 at 7:10 PM].

<https://bgpview.io/>

=====

[5]: BGP tools. [online image]

[Accessed on 8 August 2024 at 8:05 PM].

<https://bgp.tools/>

=====

[6] cbtnuggets. [online image]

[Accessed on 8 August at 8:45 PM].

<https://www.cbtnuggets.com/blog/technology/networking/traceroute-command-in-linux>

=====

[7]: watch guard. [online image]

[Accessed on 8 August at 10:24 PM].

[https://www.watchguard.com/help/docs/fireware/12/en-us/Content/en-US/policies/aliases\\_about\\_c.html#:~:text=An%20alias%20is%20a%20shortcut,can%20also%20create%20new%20aliases.](https://www.watchguard.com/help/docs/fireware/12/en-us/Content/en-US/policies/aliases_about_c.html#:~:text=An%20alias%20is%20a%20shortcut,can%20also%20create%20new%20aliases.)

=====

[8]: IP location. [online image]

[Accessed on 13 August at 1:16 AM].

[https://www.iplocation.net/ip-lookup#google\\_vignette](https://www.iplocation.net/ip-lookup#google_vignette)

=====

[9]: Socket programming: <https://www.youtube.com/watch?v=3QiPPX-KeSc&t=1133s>

[Accessed on 10 August at 1:12 PM].

=====

[10]: Previous Video for socket preprogramming (for Dr. Ibrahim Nimer):

[https://drive.google.com/drive/u/0/folders/1JvfyNheYXJHFmg1mpP\\_3-ePPmLlx2478](https://drive.google.com/drive/u/0/folders/1JvfyNheYXJHFmg1mpP_3-ePPmLlx2478)

[Accessed on 11 August at 9:15 PM].

=====

[11]: another video for socket programming that it helps me in my code:

[https://www.youtube.com/watch?v=bwTAVGg\\_kVs](https://www.youtube.com/watch?v=bwTAVGg_kVs)

[Accessed on 8 August at 8:15 AM].

=====

[12]: Socket programming Definition:

<https://www.geeksforgeeks.org/socket-programming-cc/>

[Accessed on 13 August at 2:15 AM].

=====

[13]: TCP definition:

<https://www.fortinet.com/resources/cyberglossary/tcp-ip>

[Accessed on 13 August at 3:21 PM].

=====

[14]: UDP definition:

<https://cyberhoot.com/cybrary/transmission-control-protocol-tcp/>

[Accessed on 13 August at 3:21 PM].

=====

[15] :Web Server

[https://en.wikipedia.org/wiki/Web\\_server](https://en.wikipedia.org/wiki/Web_server)

[Accessed on 14 August at 7:18 PM].

=====

[16]: Socket Programming

<https://www.datacamp.com/tutorial/a-complete-guide-to-socket-programming-in-python>

[Accessed on 14 August at7:37PM].

---

[17]: HTTP Protocol and Content Types

<https://www.geeksforgeeks.org/http-headers-content-type/>

[Accessed on 14 August at7:47PM].

---

[18] : Error Handling

<https://www.dremio.com/wiki/error-handling/>

[Accessed on 14 August at7:52PM].

---

[19] : Redirection with Status Codes

<https://www.baeldung.com/cs/redirection-status-codes>

[Accessed on 14 August at8:02PM].

---

[20] : **Localization Support:**

<https://poeditor.com/blog/localization-services/>

[Accessed on 14 August at8:14PM].