



Department of Electrical and Computer Engineering

ENCS3320 - Computer Networks

Summer Semester 2023/2024

Project#1: Socket Programming

Prepared by:

Mouath Masalmah - 1220179

Zaid Mousa - 1221833

Bahaa Bani Shamsaa - 1220252

Instructor: Dr.Ibrahim Nemer

Section: SECTION_1

Table of Contents

Table of Figures	4
Theory:	6
Task 1: Commands & Wireshark	6
1. Ping:	6
2. Tracert:	6
3. Nslookup:	6
4. Telnet:	6
Task 2: Socket Programming (TCP and UDP)	6
11. TCP Client-Server Application:	6
12. UDP Client-Server Application:	6
Task 3: Web Server	7
15. Web Server Application:	7
Procedure:	8
Task 1: Commands & Wireshark	8
5. Ping a device in the same network:	8
6. Ping www.ox.ac.uk:	8
7. Tracert www.ox.ac.uk:	8
8. Nslookup www.ox.ac.uk:	8
9. Telnet www.ox.ac.uk:	8
10. Wireshark DNS Capture:	8
Task 2: Socket Programming (TCP and UDP)	8
13. TCP Client-Server:	8
14. UDP Client-Server:	8
Task 3: Web Server	9
16. Web Server Implementation:	9
17. HTML and CSS:	9
18. Testing and Documentation:	9
Result&Discussion	10
Task1: Commands & Wireshark	10
Part 1: In your words, what are ping, tracert, nslookup, and telnet?	10
Part 2: Make sure that your computer is connected to the internet and then run the following commands	11

Part 3: Give some details about autonomous system (AS) number, number of IPs, prefixes, peers, name of Tier1-ISP of www.ox.ac.uk .	17
Task 2:	23
Part 1:	23
Part 2:	28
Task 3:	33
1-	33
2.	50
3.	51
4.	52
5.	53
6.	54
7 and 8:	55
9.	59
10.	61
11.	62
Problems and challenges:	63
Work Done:	64
References	66

Table of Figures

FIGURE 1: PINGING MY NETWORK	11
FIGURE 2: PINGING WWW.OX.AC.UK	12
FIGURE 3:LOCATION SERVER RESPONSE	13
FIGURE 4: TRACERT WWW.OX.AC.UK	14
FIGURE 5: NSLOOKUP WWW.OX.AC.UK	15
FIGURE 6: TELNET WWW.OX.AC.UK	16
FIGURE 7: DETAILS ABOUT TIER1 ISP FOR WWW.OX.AC.UK	17
FIGURE 8: DNS ON WIRE SHARK	18
FIGURE 9: DETAILS FOR ONE OF THE PACKETS	18
FIGURE 10: FRAME DETAILS ON THE PACKET	19
FIGURE 11: ETHERNET II DETAILS	19
FIGURE 12: DETAILS FOR THE SOURCE AND DESTINATION FOR IT	20
FIGURE 13: USER DATAGRAM PROTOCOL	20
FIGURE 14: DOMAIN NAME SYSTEM	21
FIGURE 15: THE PACKET THAT WE SHOW THE DETAILS ABOUT IT	21
FIGURE 16: SERVER CODE	23
FIGURE 17: CLIENT CODE	24
FIGURE 18: FIRST SERVER OUTPUT MESSAGE	25
FIGURE 19: CLIENT OUTPUT MESSAGE	26
FIGURE 20: SECOND SERVER OUTPUT	27
FIGURE 21: SERVER CODE FOR PART2	28
FIGURE 22: CLIENT 2 CODE	29
FIGURE 23: CLIENT 1 CODE	29
FIGURE 24:STARTING THE SERVER	30
FIGURE 25: CLIENT 1 MESSAGE	31
FIGURE 26: CLIENT 2 MESSAGE	31
FIGURE 27: SERVER RECEIVE CLIENT 1 MESSAGE	32
FIGURE 28: SERVER RECEIVE THE CLIENT 2 MESSAGE	32
FIGURE 29:REQUEST /	33
FIGURE 30: INDEX.HTML REQUEST	34
FIGURE 31: MAIN_EN.HTML REQUEST	35
FIGURE 32: TITLE OF THE PAGE	36
FIGURE 33:BROWSER TITLE	36
FIGURE 34: WELCOME MESSAGE	37
FIGURE 35:BROWSER WELCOME	37
FIGURE 36:GROUP MEMBERS	38
FIGURE 37: BROWSER GROUP MEMBERS	38
FIGURE 38: INFO ABOUT GROUP MEMBERS	39
FIGURE 39:BROWSER INFO GROUP MEMBERS	39
FIGURE 40: TABLE 1	40
FIGURE 41: TABLE 2	40
FIGURE 42: TABLE 3	41
FIGURE 43: CSS FOR TABLE 1	41
FIGURE 44: CSS FOR TABLE 2	42

FIGURE 45: CSS FOR TABLE 3	42
FIGURE 46:BROWSER FOR TABLE 1	43
FIGURE 47: BROWSER FOR TABLE 2	43
FIGURE 48:PNG AND JPG 1	44
FIGURE 49:PNG AND JPG 2	44
FIGURE 50:PNG AND JPG 3	45
FIGURE 51:PNG AND JPG 4	45
FIGURE 52:CSS PNG AND JPG 1	46
FIGURE 53:CSS PNG AND JPG 2	46
FIGURE 54:CSS PNG AND JPG 3	47
FIGURE 55:CSS PNG AND JPG 4	47
FIGURE 56:BROWSER FOR PNG AND JPG 1	48
FIGURE 57:BROWSER FOR PNG AND JPG 2	48
FIGURE 58: LINK TO LOCAL HTML AND BIRZEIT WEB	49
FIGURE 59:BROWSER FOR LINK TO LOCAL HTML AND BIRZEIT WEB	49
FIGURE 60: /AR REQUEST	50
FIGURE 61: MAIN_AR.HTML REQUEST	50
FIGURE 62: MAIN_EN.HTML REQUEST	51
FIGURE 63: STYLE.CSS REQUEST	52
FIGURE 64: MOUATH.PNG REQUEST	53
FIGURE 65: ZAID.JPG REQUEST	54
FIGURE 66: SERVER 1	55
FIGURE 67: SERVER 2	55
FIGURE 68: SERVER 3	56
FIGURE 69: SERVER 4	56
FIGURE 70: MYSITESTDID.HTML	57
FIGURE 71: BROWESR FOR MYSITESTDID.HTML	57
FIGURE 72: IMAGE BROWSER FOR MYSITESTDID.HTML	58
FIGURE 73: /SO REQUEST	59
FIGURE 74: SOLUTION FOR /SO	59
FIGURE 75: /ITC REQUEST	60
FIGURE 76: SOLUTION FOR /ITC	60
FIGURE 77: WRONG IMAGE	61
FIGURE 78: ERROR.HTML BROWSER	61
FIGURE 79: HTTP REQUEST ON TERMINAL	62

Theory:

Task 1: Commands & Wireshark

1. Ping:

- Ping is a network diagnostic tool used to test the reachability of a host on an IP network. It works by sending ICMP Echo Request packets to the target host and waiting for an Echo Reply. This helps in measuring the round-trip time and identifying network issues.

2. Tracert:

- Tracert (or traceroute) is a utility that traces the path that a packet takes from the source to the destination across multiple routers. It helps in identifying the route, delays, and any points of failure along the path.

3. Nslookup:

- Nslookup is used to query the Domain Name System (DNS) to obtain domain name or IP address mappings. It's a helpful tool for troubleshooting DNS-related issues and understanding the DNS structure.

4. Telnet:

- Telnet is a protocol that allows you to connect to another machine on the network, typically for managing servers or network devices. It's an older protocol and lacks security, making it less commonly used today.

Task 2: Socket Programming (TCP and UDP)

11. TCP Client-Server Application:

- TCP is a connection-oriented protocol that ensures reliable data transmission between client and server. In this task, the client sends a string to the server, which processes it (replacing vowels with #) and sends it back.

12. UDP Client-Server Application:

- UDP is a connectionless protocol that allows fast data transmission with no guarantee of delivery. The server listens on a specific port, and clients send messages to it. The server keeps track of the last message received from each client.

Task 3: Web Server

15. Web Server Application:

- A web server listens for HTTP requests from clients (browsers) and serves them content based on the request. The server needs to handle various content types (HTML, CSS, images) and redirect or return error pages as necessary.

Procedure:

Task 1: Commands & Wireshark

5. **Ping a device in the same network:**
 - Connect your laptop and smartphone to the same Wi-Fi network.
 - Open the command prompt on your laptop.
 - Type ping <smartphone IP> to test connectivity.
6. **Ping www.ox.ac.uk:**
 - In the command prompt, type ping www.ox.ac.uk.
 - Note the response times and analyze the location based on the IP.
7. **Tracert www.ox.ac.uk:**
 - Type tracert www.ox.ac.uk in the command prompt.
 - Observe the hops and response times along the route to Oxford University.
8. **Nslookup www.ox.ac.uk:**
 - Type nslookup www.ox.ac.uk in the command prompt.
 - Record the IP address returned and verify its ownership.
9. **Telnet www.ox.ac.uk:**
 - Type telnet www.ox.ac.uk in the command prompt.
 - Attempt to establish a connection and note the result.
10. **Wireshark DNS Capture:**
 - Open Wireshark and start capturing packets.
 - Perform some DNS queries (e.g., visit a website).
 - Stop the capture and analyze the DNS messages.
 - Take screenshots showing the time and date.

Task 2: Socket Programming (TCP and UDP)

13. **TCP Client-Server:**
 - Write a server program that listens on a port.
 - Write a client program that connects to the server and sends a string.
 - The server replaces vowels with # and sends the modified string back.
 - The client prints the received string.
14. **UDP Client-Server:**
 - Write a UDP server that listens on a port.
 - Write UDP clients that send messages to the server.

- The server logs the messages and the clients print their received messages.
- Test the communication between multiple clients and the server.

Task 3: Web Server

16. Web Server Implementation:

- Write a server program that listens on a specific port.
- Handle different URL requests by serving the appropriate HTML or CSS files.
- Implement logic to return custom error pages for invalid requests.
- Include images, links, and redirects as specified.
- Test the server by making various requests from a browser (e.g., <http://localhost:1515/en>).

17. HTML and CSS:

- Create HTML files (main_en.html, main_ar.html, etc.) that include information about the team, images, and links.
- Style the pages using CSS, ensuring the layout is visually appealing.
- Store all files in the appropriate directories for the server to access.

18. Testing and Documentation:

- Run the server and access it through a browser.
- Document the theory, procedure, and results with explanations and screenshots.

Result&Discussion

Task1: Commands & Wireshark

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

Ping: is a network tool that tests the possibility of sending messages between devices by transmitting small data packets. It also measures the time taken for these packets to be sent and received, helping to identify any network issues that need to be resolved.

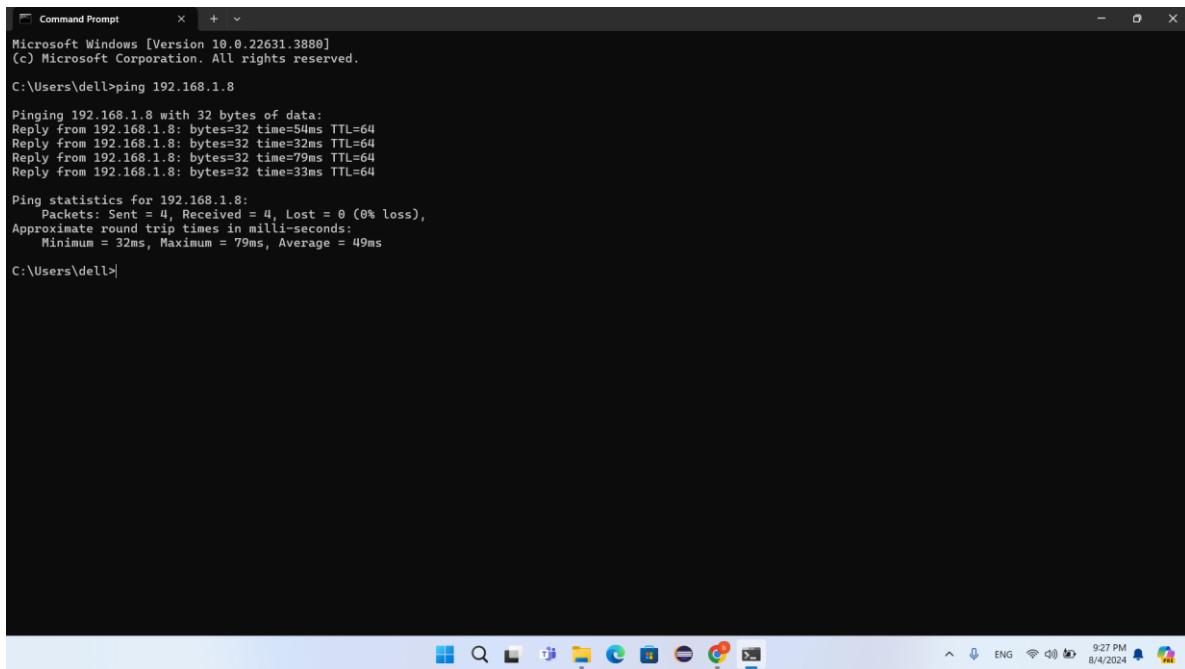
Tracert: It is a network tool that displays the path that data takes to reach a destination by showing all the routers it passes through. It is useful for seeing the path and checking for any network delays along the way.

Nslookup: It is a command line tool used in network administration to find the domain name or IP address associated with a website. It helps diagnose DNS problems by examining and retrieving DNS records.

Telnet: It is a protocol that allows users to connect to remote computers over the Internet or a local network, allowing them to interact with the remote device as if they were directly in front of it, issuing commands and controlling the system through a text interface.

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

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



```
Command Prompt
Microsoft Windows [Version 10.0.22631.3888]
(c) Microsoft Corporation. All rights reserved.

C:\Users\dell>ping 192.168.1.8

Pinging 192.168.1.8 with 32 bytes of data:
Reply from 192.168.1.8: bytes=32 time=54ms TTL=64
Reply from 192.168.1.8: bytes=32 time=32ms TTL=64
Reply from 192.168.1.8: bytes=32 time=79ms TTL=64
Reply from 192.168.1.8: bytes=32 time=33ms TTL=64

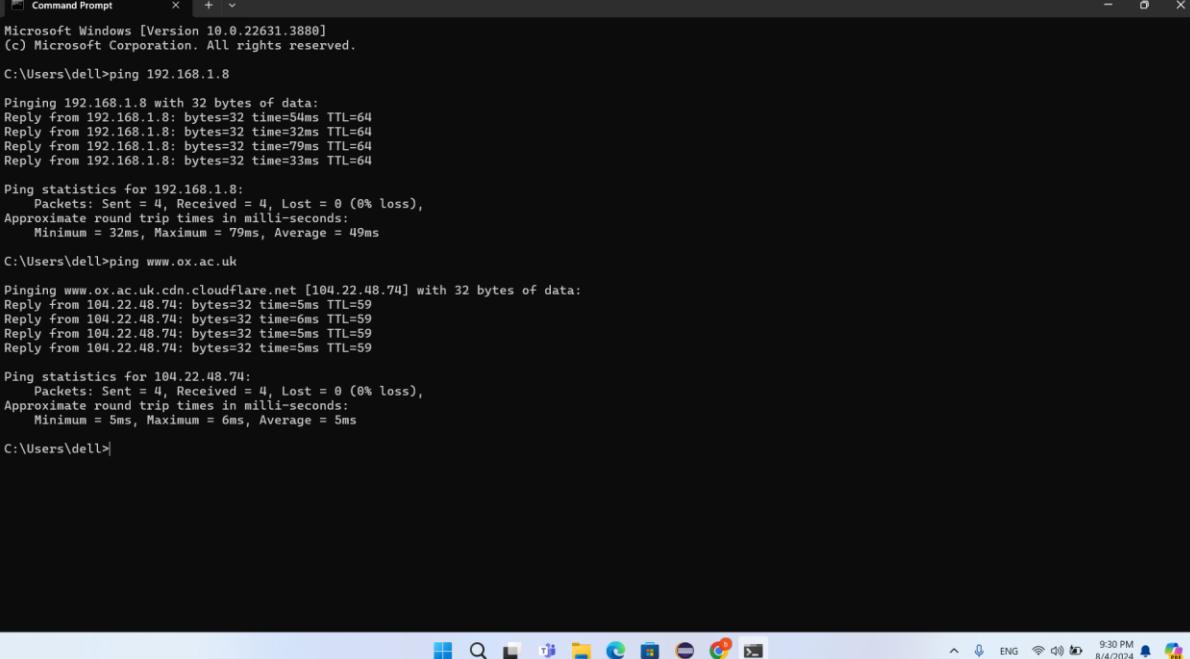
Ping statistics for 192.168.1.8:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 32ms, Maximum = 79ms, Average = 49ms

C:\Users\dell>
```

Figure 1: pinging my network

Here when we ping a device in the same network, my laptop sends small data packets to your smartphone to check if it's online. If the smartphone is connected, it replies, and we get the response time, showing that the device is reachable.

Point B: ping www.ox.ac.uk.



```
Microsoft Windows [Version 10.0.22631.3880]
(c) Microsoft Corporation. All rights reserved.

C:\Users\dell>ping 192.168.1.8

Pinging 192.168.1.8 with 32 bytes of data:
Reply from 192.168.1.8: bytes=32 time=54ms TTL=64
Reply from 192.168.1.8: bytes=32 time=32ms TTL=64
Reply from 192.168.1.8: bytes=32 time=79ms TTL=64
Reply from 192.168.1.8: bytes=32 time=33ms TTL=64

Ping statistics for 192.168.1.8:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 32ms, Maximum = 79ms, Average = 49ms

C:\Users\dell>ping www.ox.ac.uk

Pinging www.ox.ac.uk.cdn.cloudflare.net [104.22.48.74] with 32 bytes of data:
Reply from 104.22.48.74: bytes=32 time=5ms TTL=59
Reply from 104.22.48.74: bytes=32 time=6ms TTL=59
Reply from 104.22.48.74: bytes=32 time=5ms TTL=59
Reply from 104.22.48.74: bytes=32 time=5ms TTL=59

Ping statistics for 104.22.48.74:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 5ms, Maximum = 6ms, Average = 5ms

C:\Users\dell>
```

Figure 2: pinging www.ox.ac.uk

When we ping `www.ox.ac.uk`, my computer sends data packets to the Oxford University website's server. The server replies if it's online, and you see the response time. This helps to check if the website is reachable and how quickly it responds.

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

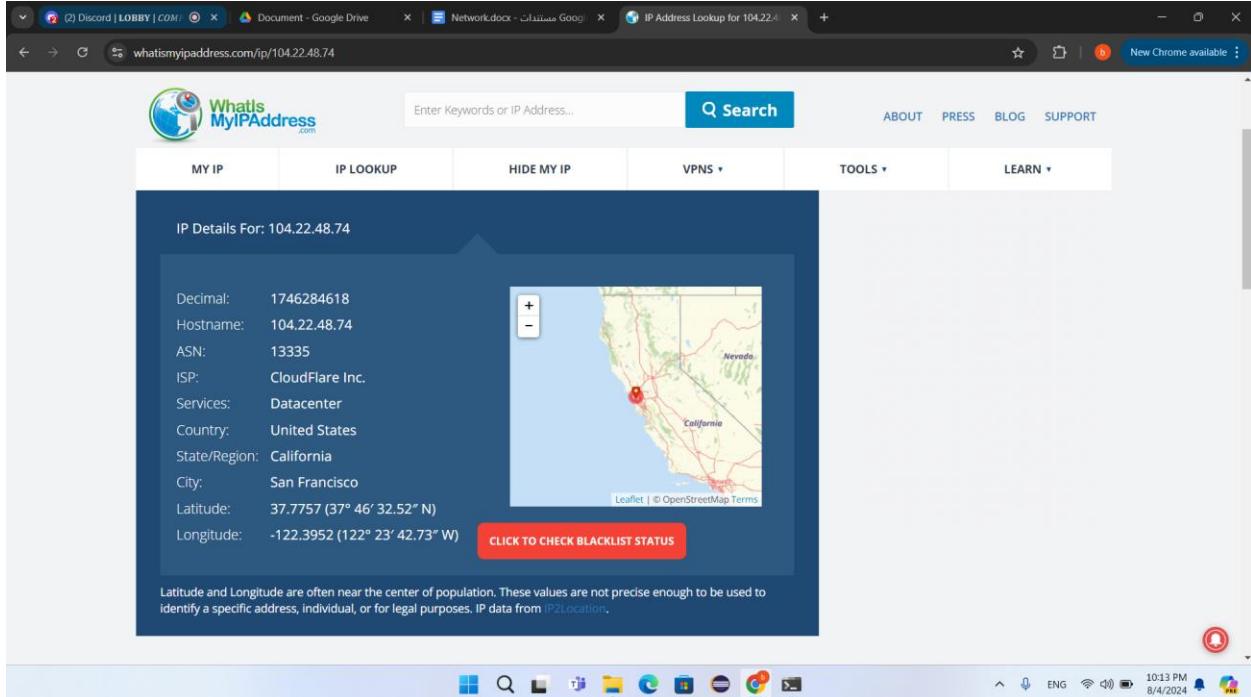
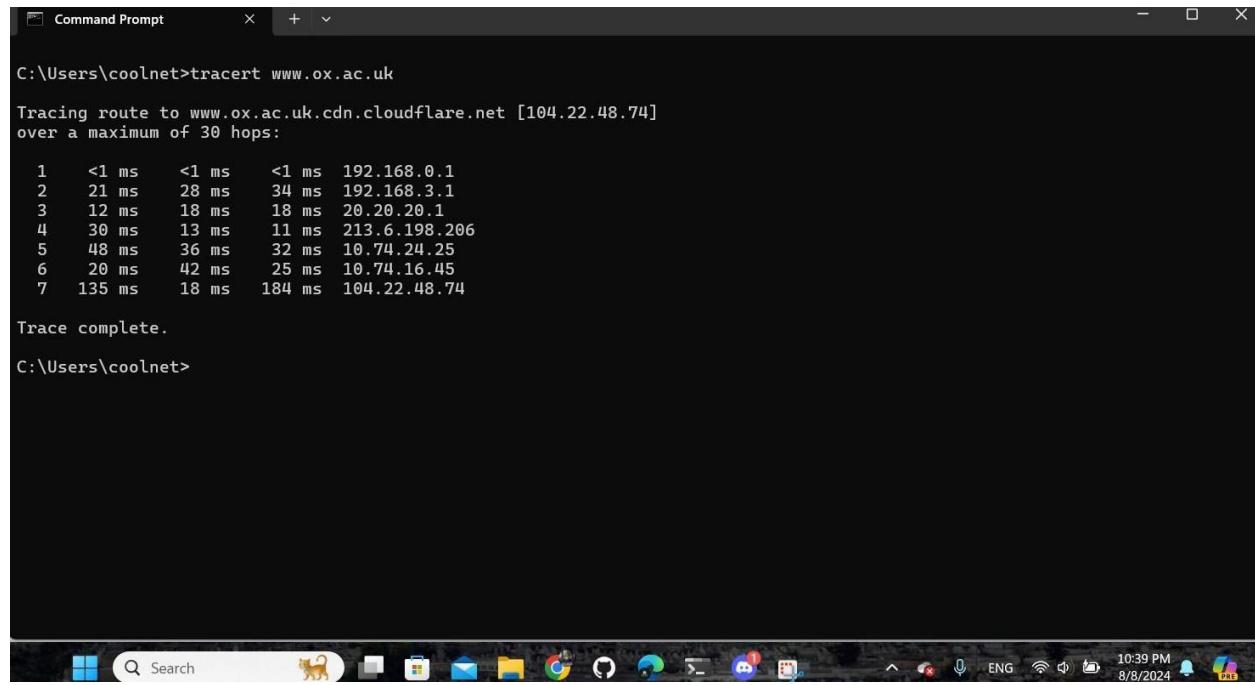


Figure 3:location server response

From the ping results alone, we typically cannot directly determine the exact physical location of the server. The ping command shows me the IP address of the server and the round-trip time (RTT), but it doesn't provide geographic information.

However, we can use the IP address obtained from the ping results and perform an IP geolocation lookup to estimate the server's location. This method isn't perfectly accurate, but it gives a general idea of where the server might be located.

Point D: tracert www.ox.ac.uk.



```
Command Prompt

C:\Users\coolnet>tracert www.ox.ac.uk
Tracing route to www.ox.ac.uk.cdn.cloudflare.net [104.22.48.74]
over a maximum of 30 hops:
1 <1 ms <1 ms 192.168.0.1
2 21 ms 28 ms 34 ms 192.168.3.1
3 12 ms 18 ms 18 ms 20.20.20.1
4 30 ms 13 ms 11 ms 213.6.198.206
5 48 ms 36 ms 32 ms 10.74.24.25
6 20 ms 42 ms 25 ms 10.74.16.45
7 135 ms 18 ms 184 ms 104.22.48.74

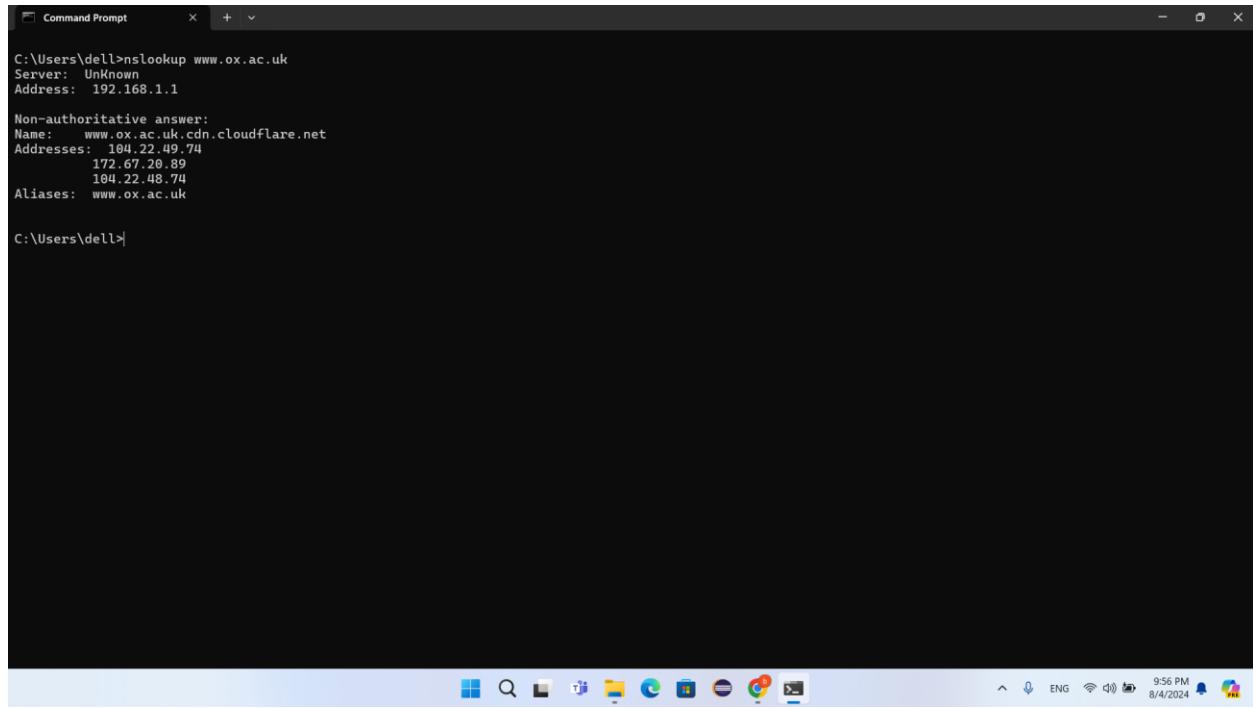
Trace complete.

C:\Users\coolnet>
```

Figure 4: tracert www.ox.ac.uk

That shows me the path my data takes from my device to Oxford University's server, listing each router (hop) along the way and how long it takes to reach each one.

Point E: nslookup *www.ox.ac.uk*.



```
C:\Users\dell>nslookup www.ox.ac.uk
Server: UnKnown
Address: 192.168.1.1

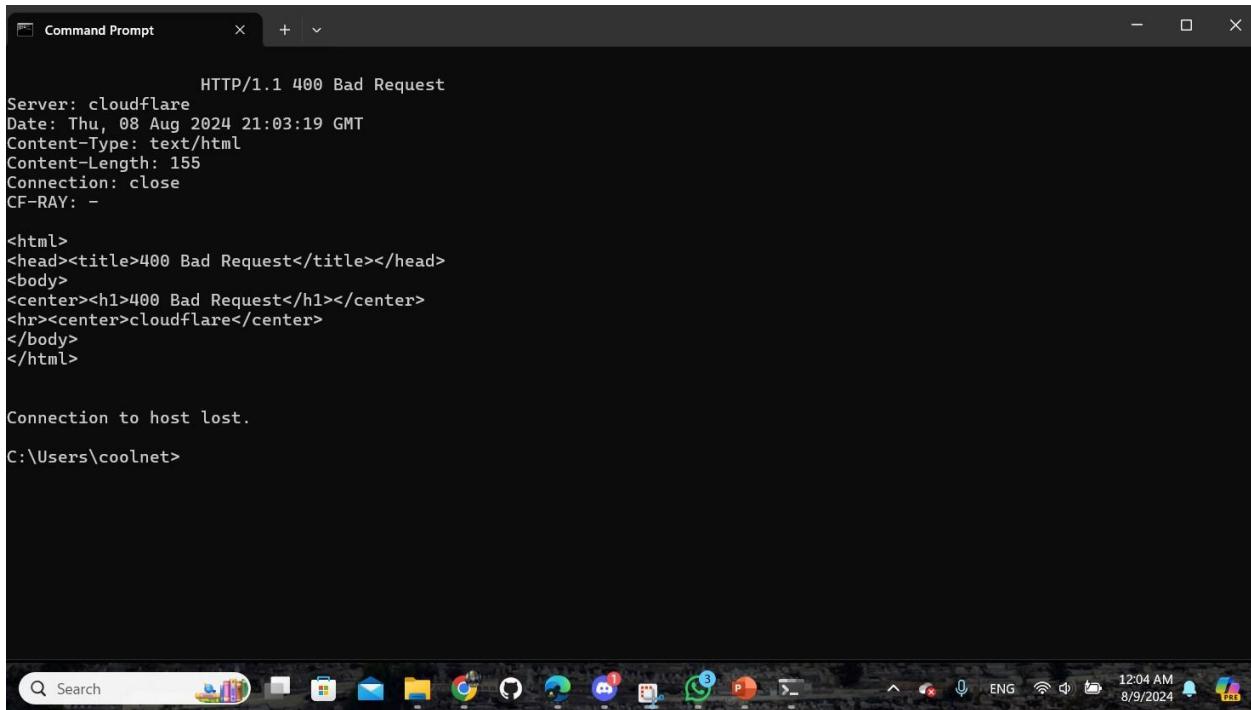
Non-authoritative answer:
Name: www.ox.ac.uk.cdn.cloudflare.net
Addresses: 104.22.49.74
          172.67.20.89
          104.22.48.74
Aliases: www.ox.ac.uk

C:\Users\dell>
```

Figure 5: nslookup *www.ox.ac.uk*

It retrieves the IP address of the Oxford University website by querying a DNS server.

The point F: telnet www.ox.ac.uk.



```
HTTP/1.1 400 Bad Request
Server: cloudflare
Date: Thu, 08 Aug 2024 21:03:19 GMT
Content-Type: text/html
Content-Length: 155
Connection: close
CF-RAY: -1000000000000000000

<html>
<head><title>400 Bad Request</title></head>
<body>
<center><h1>400 Bad Request</h1></center>
<hr><center>cloudflare</center>
</body>
</html>

Connection to host lost.

C:\Users\coolnet>
```

Figure 6: telnet www.ox.ac.uk

The "Bad Request" error happens because Telnet sends an incomplete or improperly formatted request that the server can't understand. The server is reachable, but the request isn't correct.

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

The screenshot shows a search result for the IP address 104.22.48.74. At the top, there is a banner with the text "Unlock the Intelligence Handbook: Your Guide to Cyber Threat Intelligence" and a "Get it for Free" button. Below the banner, the IP address is displayed with a question mark icon and the text "NO RDNS FOUND".

Announced Prefixes

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

RIR Allocation Summary

PREFIX: 104.16.0.0/12
GEOIP COUNTRY: US
IP ADDRESSES: 1,048,576

REGIONAL REGISTRY: ARIN
ALLOCATION STATUS: Allocated
ALLOCATION DATE: 28th March 2014



Figure 7: Details about tier1 ISP for www.ox.ac.uk

Here we use this website to find all details we need about "www.ox.ac.uk" that we need.

Part 4: Wireshark

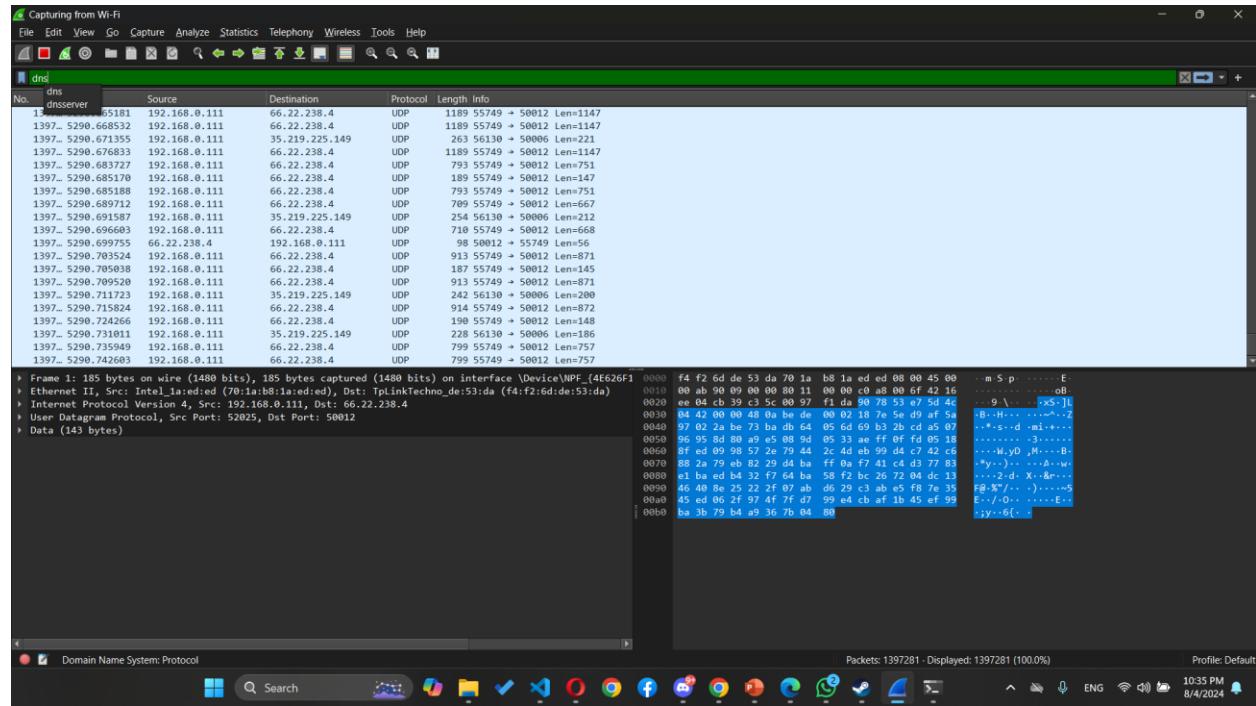


Figure 8: dns on wire shark

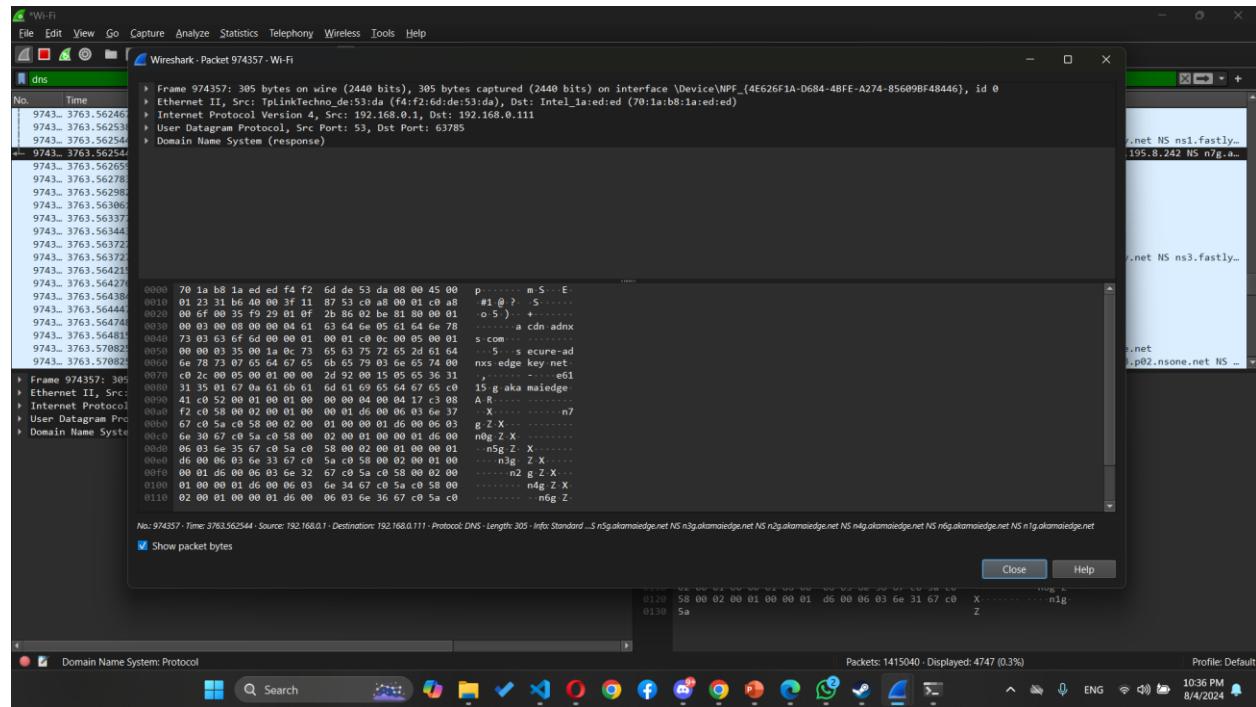


Figure 9: details for one of the packets

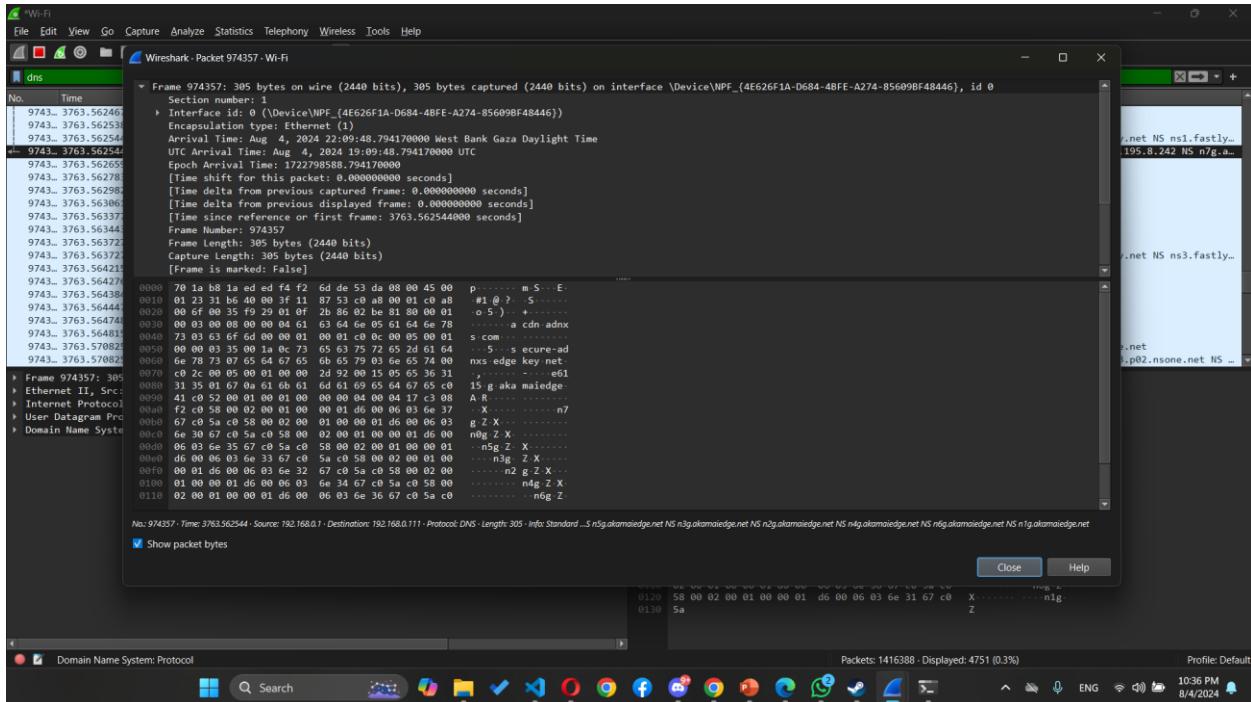


Figure 10: frame details on the packet

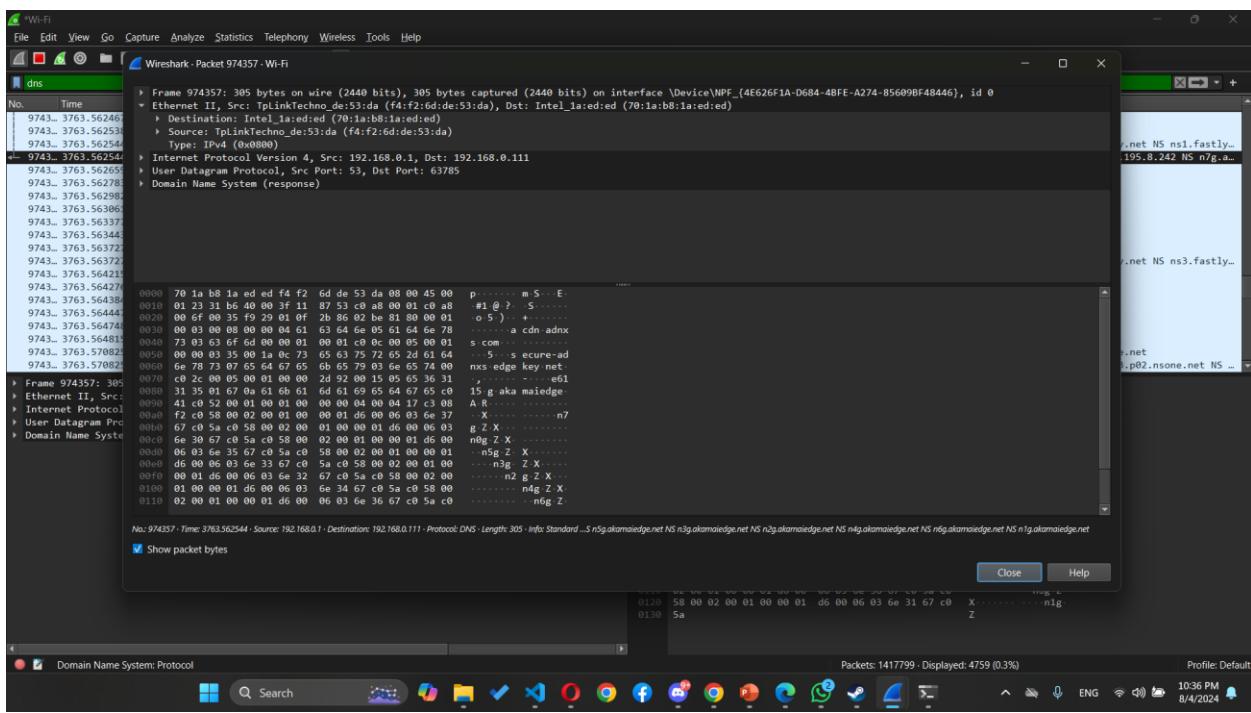


Figure 11: ethernet II details

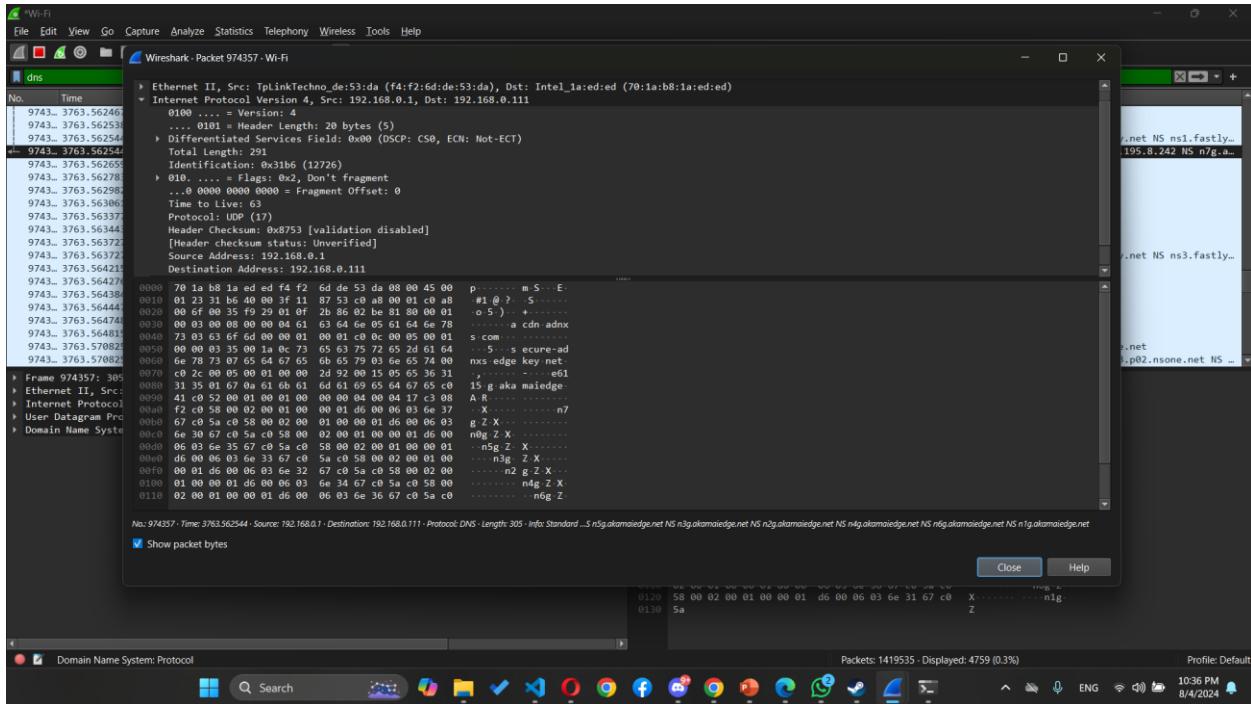


Figure 12: details for the source and destination for it

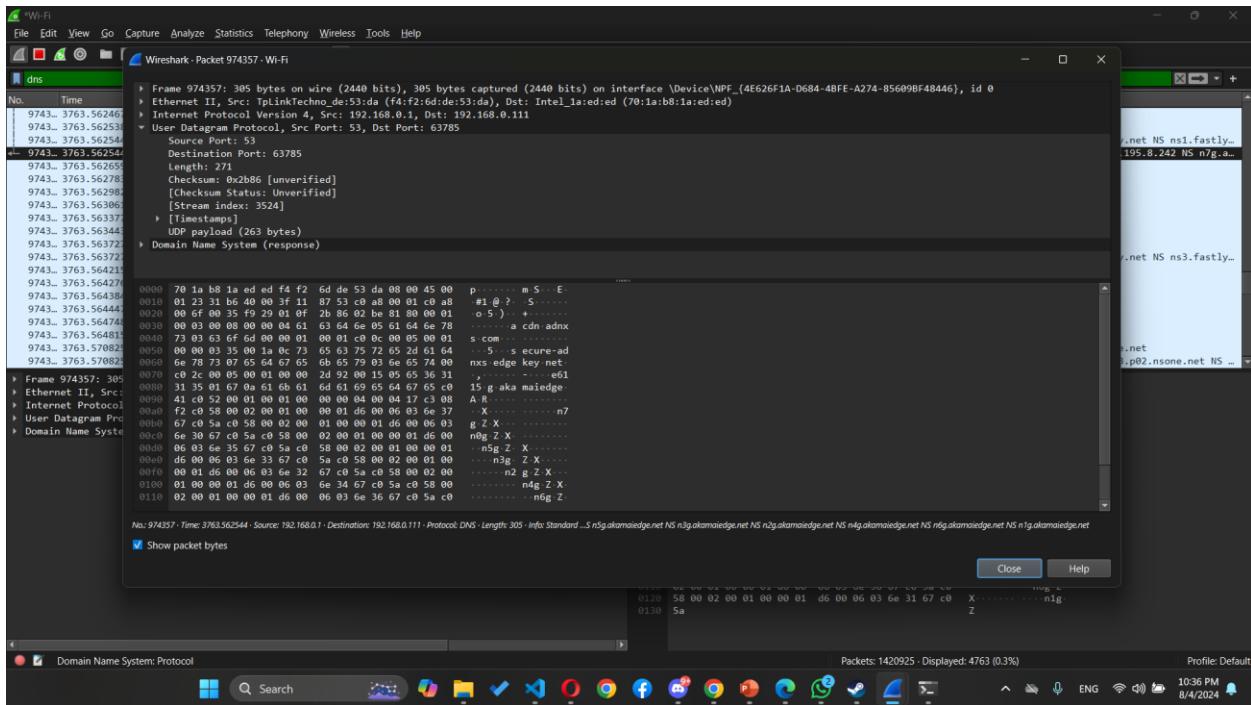


Figure 13: user datagram protocol

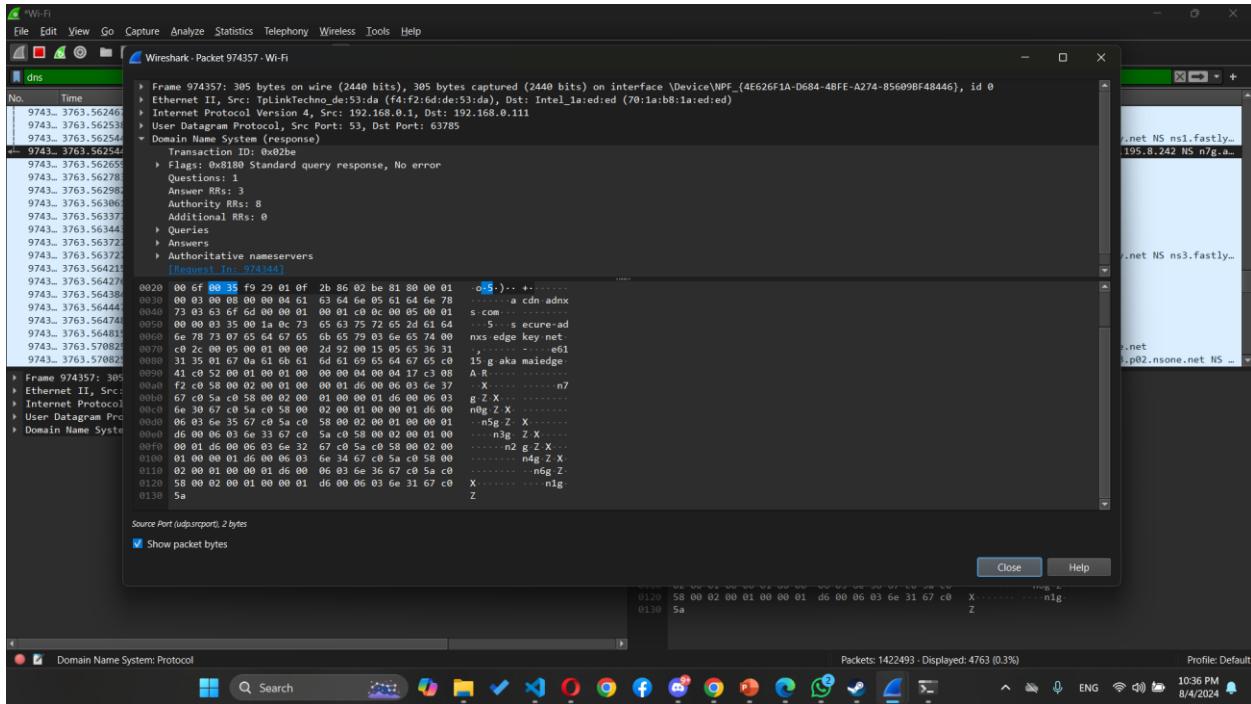


Figure 14: domain name system

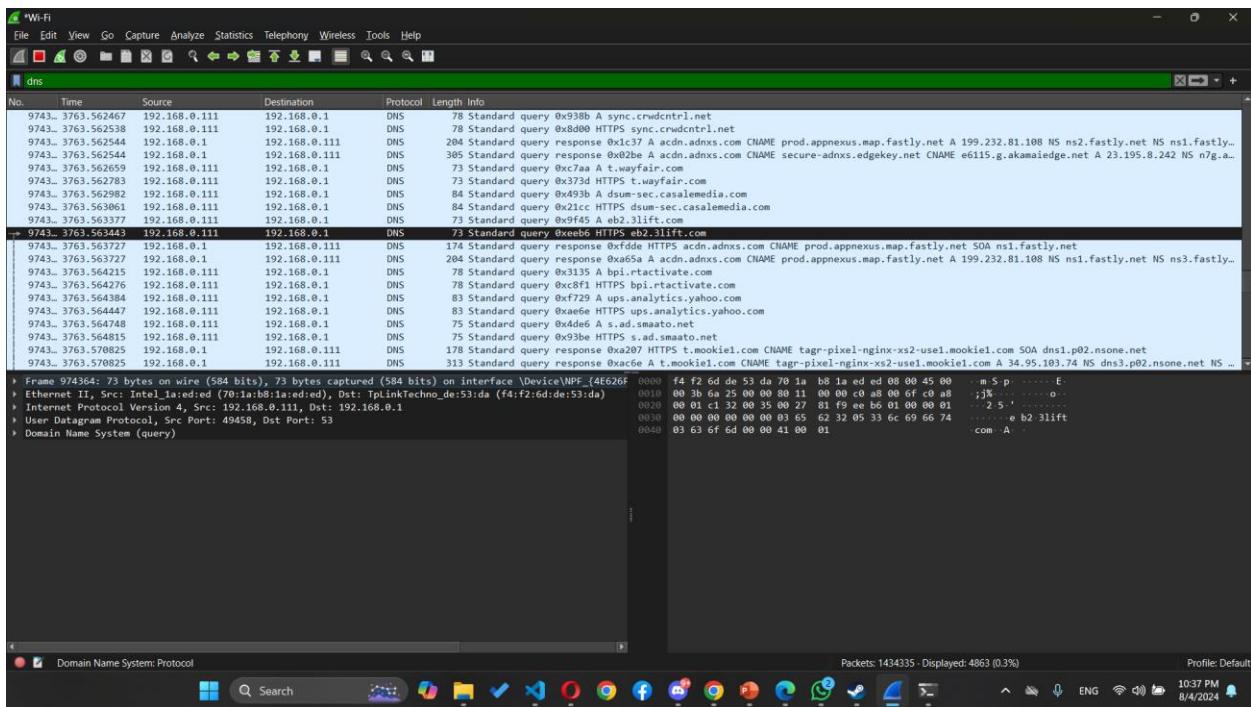


Figure 15: The packet that we show the details about it

- Capturing DNS Messages:
- 1. Start Wireshark: Open Wireshark, choose your network interface, and begin capturing packets.
- 2. Filter DNS Traffic: Type `dns` in the display filter to show only DNS packets.
-
- Examining DNS Queries:
- Query Packets: Displays DNS queries where the client requests the IP address for a domain name from the DNS server.
-
- Analyzing DNS Responses:
- Response Packets: Shows DNS responses containing the IP address corresponding to the requested domain.
-
- Detailed Packet Analysis:
- Packet Details: Click on a DNS packet to see details such as:
- Transaction ID: Matches queries to responses.
- Flags: Indicates if the packet is a query or response and other info.
- Questions: Domain name requested.
- Answers: IP address provided by the server.
- Interpreting Data:
- Hex Data: The raw packet data is shown in hexadecimal at the bottom, with Wireshark translating it for easier understanding.
- Common DNS Operations:
- Standard Query: Client request for a domain's IP address.
- Response: DNS server's reply with the resolved IP address.

Task 2:

Part 1:

- The Server Code:

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows files client1.py and server.py.
- Code Editor:** Displays the Python server code. The code creates a socket, binds it to localhost port 1183, and listens for connections. It then enters a loop where it receives a message from a client, converts vowels to '#', and sends the modified message back to the client. Finally, it prints a message about the connection being closed and exits the loop.
- Terminal:** Shows the command PS C:\Users\coolnet\Desktop\network project\task2\1> & "c:\Users\coolnet\AppData\Local\Programs\Python\Python39\python.exe" 'c:\Users\coolnet\.vscode\extensions\ms-python.python.debug-2024.10.0-win32-x64\bundle\libs\debug\adapter\..\..\debug\launcher' '51375' '--' 'c:\Users\coolnet\Desktop\network project\task2\1\client.py'. It also prompts the user to enter their message.
- Status Bar:** Shows the current file is client1.py, line 3, column 18, spaces: 4, encoding: UTF-8, CRLF, Python 3.9.13 64-bit, and the date/time: 8/13/2024 11:29 PM.

Figure 16: server code

The server code simply opens a connection and waits for only one client. After that, the server receives a message from the client, converts the upper and lower case vowels to #, then sends it back to the client and stops the connection and print message about this.

- The Client Code:

The screenshot shows the Visual Studio Code interface. The left sidebar has 'client1.py' and 'server.py' listed. The main editor area contains the following Python code:

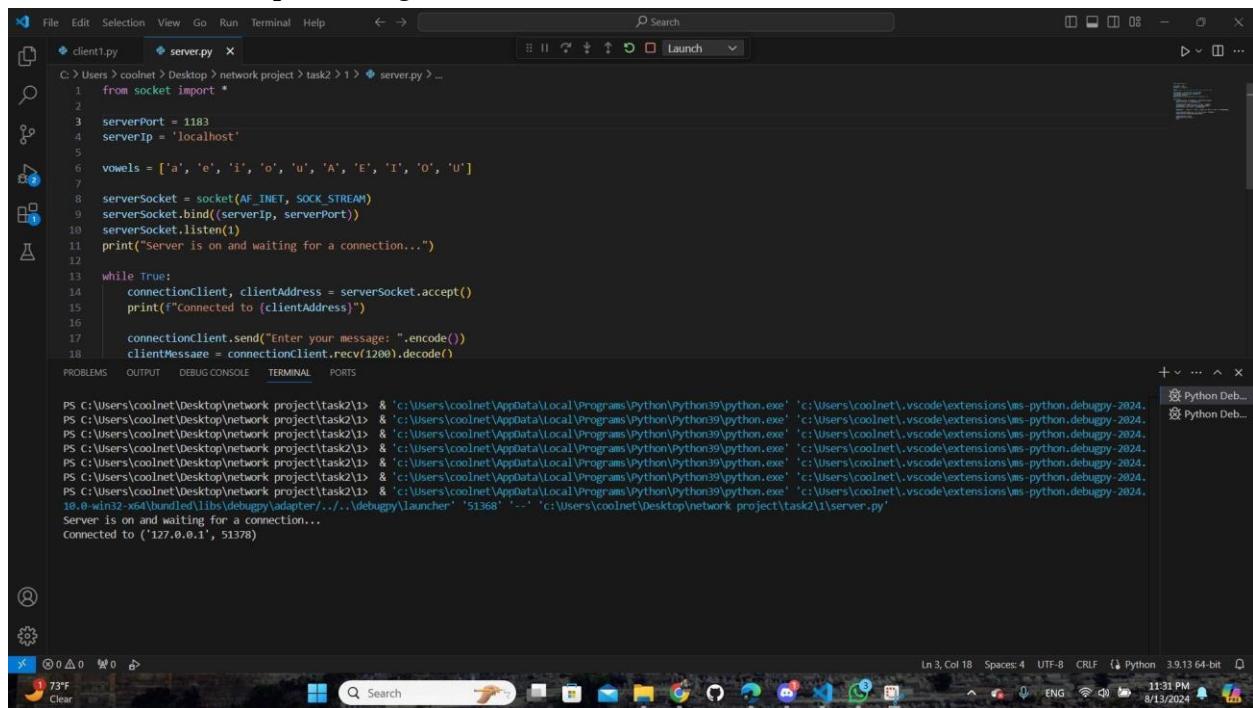
```
1  from socket import *
2
3  serverIp = 'localhost'
4  serverPort = 1183
5
6  clientSocket = socket(AF_INET, SOCK_STREAM)
7  clientSocket.connect((serverIp, serverPort))
8
9  receivedData = clientSocket.recv(1200).decode()
10 print(receivedData)
11
12 sentData = input(' ')
13 clientSocket.send(sentData.encode())
14
15 print(clientSocket.recv(1200).decode())
16 print(clientSocket.recv(1200).decode())
17
18 clientSocket.close()
19
```

The terminal tab at the bottom shows the command run: PS C:\Users\coolnet\Desktop\network project\task2\ & 'c:\Users\coolnet\AppData\Local\Programs\Python\Python39\python.exe' 'c:\Users\coolnet\.vscode\extensions\ms-python.debugpy-2024.10.0-win32-x64\bundled\libs\debugpy\adapter'...'\debugpy\launcher' "51375" ... 'c:\Users\coolnet\Desktop\network project\task2\1\client1.py'. Below the terminal, the status bar displays: Enter your message: 73°F Clear, Search, and various system icons.

Figure 17: client code

The client code simply sends a connection request to the server. After the server accepts the connection, the client sends a message to the server, which the server returns in the new format, prints it, and then disconnects from the server.

- The First Server Output Message:



```
C:\> Users > coolnet > Desktop > network project > task2 > 1 > server.py > ...
1   from socket import *
2
3   serverPort = 1183
4   serverIp = 'localhost'
5
6   vowels = ['a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U']
7
8   serverSocket = socket(AF_INET, SOCK_STREAM)
9   serverSocket.bind((serverIp, serverPort))
10  serverSocket.listen(1)
11  print("Server is on and waiting for a connection...")
12
13 while True:
14     connectionClient, clientAddress = serverSocket.accept()
15     print(f"Connected to {clientAddress}")
16
17     connectionClient.send("Enter your message: ".encode())
18     clientMessage = connectionClient.recv(1200).decode()

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\coolnet\Desktop\network\project\task2\1> 8 'c:\Users\coolnet\AppData\Local\Programs\Python\Python39\python.exe' 'c:\Users\coolnet\.vscode\extensions\ms-python.debugpy-2024.10.0-win32-x64\bundles\libs\debugpy\adapter\..\..\debugpy\launcher' '51368' -- 'c:\Users\coolnet\Desktop\network\project\task2\1\server.py'
Server is on and waiting for a connection...
Connected to ('127.0.0.1', 51378)
```

Figure 18: first server output message

The server prints a message that it is up and running and is waiting for a client to connect to it (one client, no more). After connecting to that client, it prints the client's IP and its port to confirm that it is connected.

- The Client Output Message:

The screenshot shows the Visual Studio Code interface. In the top left, there are two tabs: 'client1.py' and 'server.py'. The 'client1.py' tab is active, displaying the following Python code:

```
1  from socket import *
2
3  serverIp = 'localhost'
4  serverPort = 1183
5
6  clientSocket = socket(AF_INET, SOCK_STREAM)
7  clientSocket.connect((serverIp, serverPort))
8
9  receivedData = clientSocket.recv(1200).decode()
10 print(receivedData)
11
12 sentData = input('')
13 clientSocket.send(sentData.encode())
14
15 print(clientSocket.recv(1200).decode())
16 print(clientSocket.recv(1200).decode())
17
18 clientSocket.close()
```

Below the code editor is the terminal pane, which shows the execution of the script and its output:

```
PS C:\Users\cooInet\Desktop\network project\task2> 8 'c:\Users\cooInet\AppData\Local\Programs\Python\Python39\python.exe' 'c:\Users\cooInet\.vscode\extensions\ms-python.debugpy-2024.10.8-win32-x64\bundled\libs\debugpy\adapter'... \debugpy\launcher' '51375' - - 'c:\Users\cooInet\Desktop\network project\task2\1\client1.py'
Enter your message:
Ali Abdullah Abu Al Reesh
This is the new message:
#I# #d#lll#h #B# # I Raash
PS C:\Users\cooInet\Desktop\network project\task2>
```

The terminal also displays system status information at the bottom right, including the date and time.

Figure 19: Client output message

After connecting to the server, the client is asked to enter a sentence to be sent to the server, which in turn converts the vowels to # and then returns it to the client, and the final sentence is printed as we see above.

- The Second Server Output Message:

The screenshot shows a Microsoft Windows desktop environment. In the center is a terminal window from Visual Studio Code. The terminal tab is active, showing the command PS C:\Users\coolnet\Desktop\network project\task2\1>. Below the tab, there's a scrollable text area containing Python code and its execution output. The code is as follows:

```
C:\> Users > coolnet > Desktop > network project > task2 > 1 > client.py > ...
1  from socket import *
2
3  serverIp = 'localhost'
4  serverPort = 1183
5
6  clientSocket = socket(AF_INET, SOCK_STREAM)
7  clientSocket.connect((serverIp, serverPort))
8
9  receivedData = clientSocket.recv(1200).decode()
10 print(receivedData)
11
12 sentData = input(' ')
13 clientSocket.send(sentData.encode())
14
15 print(clientSocket.recv(1200).decode())
16 print(clientSocket.recv(1200).decode())
17
18 clientSocket.close()

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
```

The output section of the terminal shows the execution of the script and the resulting interaction:

```
PS C:\Users\coolnet\Desktop\network project\task2\1> 8 'c:\Users\coolnet\AppData\Local\Programs\Python\Python39\python.exe' 'c:\Users\coolnet\.vscode\extensions\ms-python.python.debug-2024.0.1\Python\Debug\adapter\..\..\debugpy\launcher' '51368' '--' 'c:\Users\coolnet\Desktop\project\task2\1\server.py'
Server is on and waiting for a connection...
Connected to ('127.0.0.1', 51378)
Message from client: Ali Abdullah Abu Al Reesh
Connection closed.
PS C:\Users\coolnet\Desktop\network project\task2\1>
```

At the bottom of the terminal window, status indicators show the file is saved (green dot), the Python extension is active, and the version is 3.13 64-bit. The system tray shows the date and time as 8/13/2024 at 11:34 PM.

Figure 20: Second server output

In this image, the server shows the sentence it received from the client by printing it, then performs the necessary operations on it, sends it, then stops the connection and prints the sentence "Connection closed."

Part 2:

- The Server Code:

The screenshot shows a code editor window with the following Python code for a UDP server:

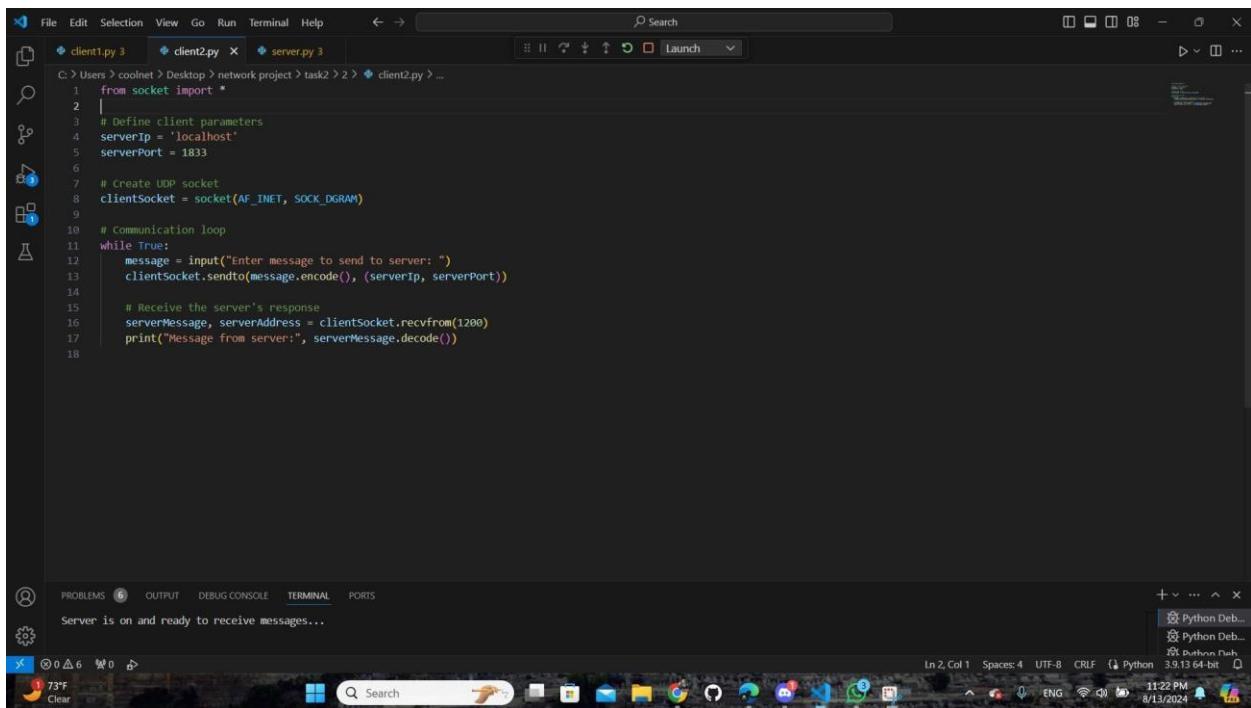
```
C:\> Users > coolnet > Desktop > network project > task2 > 2 > server.py 3 ...
1  from socket import *
2  import threading
3
4  # Define server parameters
5  serverPort = 1833
6  serverIp = 'localhost'
7
8  # Create UDP socket
9  serverSocket = socket(AF_INET, SOCK_DGRAM)
10 serverSocket.bind((serverIp, serverPort))
11 print("Server is on and ready to receive messages...")
12
13 # Dictionary to keep track of the last message from each client
14 clients = {}
15
16 def handle_client(clientMessage, clientAddress):
17     clientNumber = clients.get(clientAddress, len(clients) + 1)
18     clients[clientAddress] = clientNumber
19
20     print(f"\nMessage from client {clientNumber}:")
21     print(clientMessage.decode())
22
23     # Prompt server user to send a message back to the client
24     messageToClient = input("Enter your message to client " + str(clientNumber) + ": ")
25     serversocket.sendto(messageToClient.encode(), clientAddress)
26
27 while True:
28     clientMessage, clientAddress = serverSocket.recvfrom(1200)
29     threading.Thread(target=handle_client, args=(clientMessage, clientAddress)).start()
30
```

The code defines a UDP server that binds to port 1833 on the localhost. It uses a dictionary to track messages from each client. For each client, it prints the message, prompts the server user to enter a response, and then sends it back to the client. The server runs in a loop, receiving messages and handling them in parallel threads.

Figure 21: server code for part2

This image shows the server code, which is different from the previous task, because in this task the server will open a connection with the rest of the clients, and the server can respond to messages (it works as a client, not a server).

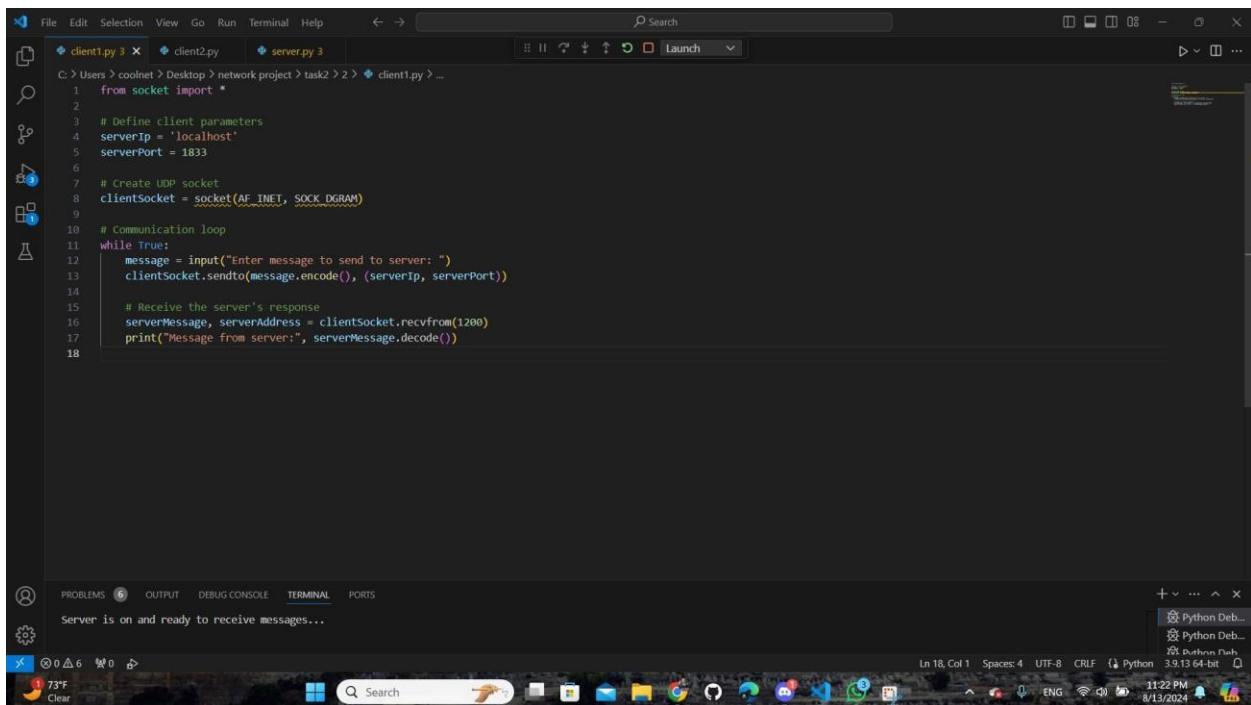
- The Client1 && Client2 Code:



```
C:\Users\coonet\Desktop\network project\task2>2> client2.py > ...
1 from socket import *
2
3 # Define client parameters
4 serverIp = 'localhost'
5 serverPort = 1833
6
7 # Create UDP socket
8 clientSocket = socket(AF_INET, SOCK_DGRAM)
9
10 # Communication loop
11 while True:
12     message = input("Enter message to send to server: ")
13     clientSocket.sendto(message.encode(), (serverIp, serverPort))
14
15     # Receive the server's response
16     serverMessage, serverAddress = clientSocket.recvfrom(1280)
17     print("Message from server:", serverMessage.decode())
18
```

The screenshot shows a code editor window with three tabs: client1.py, client2.py (which is the active tab), and server.py. The code in client2.py is a UDP client that sends messages to a local server and prints the received responses. The code editor interface includes a search bar, file navigation, and a terminal-like status bar at the bottom.

Figure 22: client 2 code



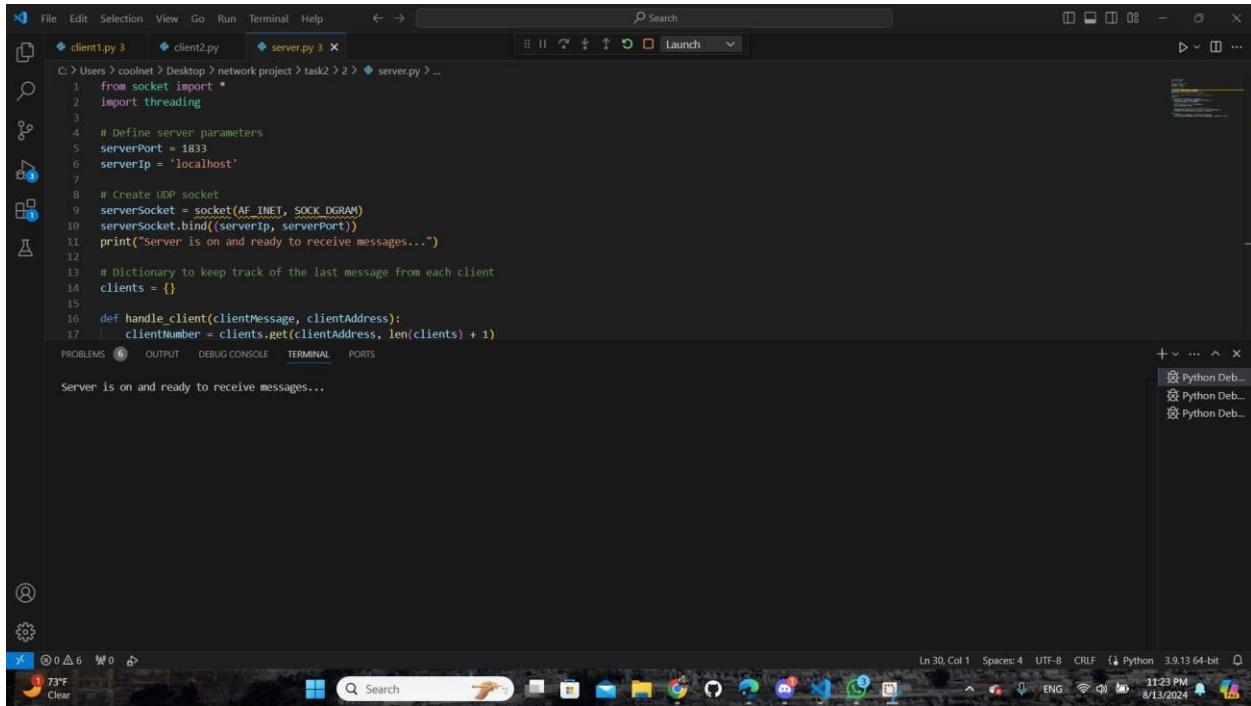
```
C:\Users\coonet\Desktop\network project\task2>2> client1.py > ...
1 from socket import *
2
3 # Define client parameters
4 serverIp = 'localhost'
5 serverPort = 1833
6
7 # Create UDP socket
8 clientSocket = socket(AF_INET, SOCK_DGRAM)
9
10 # Communication loop
11 while True:
12     message = input("Enter message to send to server: ")
13     clientSocket.sendto(message.encode(), (serverIp, serverPort))
14
15     # Receive the server's response
16     serverMessage, serverAddress = clientSocket.recvfrom(1280)
17     print("Message from server:", serverMessage.decode())
18
```

The screenshot shows a code editor window with three tabs: client1.py (active), client2.py, and server.py. The code in client1.py is a UDP client that sends messages to a local server and prints the received responses. The code editor interface includes a search bar, file navigation, and a terminal-like status bar at the bottom.

Figure 23: client 1 code

At the same time, the client, after connecting to the server, can send and receive (it works as a server and a client at the same time), and the client cannot send to other clients.

- The First Server Output Message:



The screenshot shows a code editor interface with several tabs open. The active tab is 'server.py' containing the following Python code:

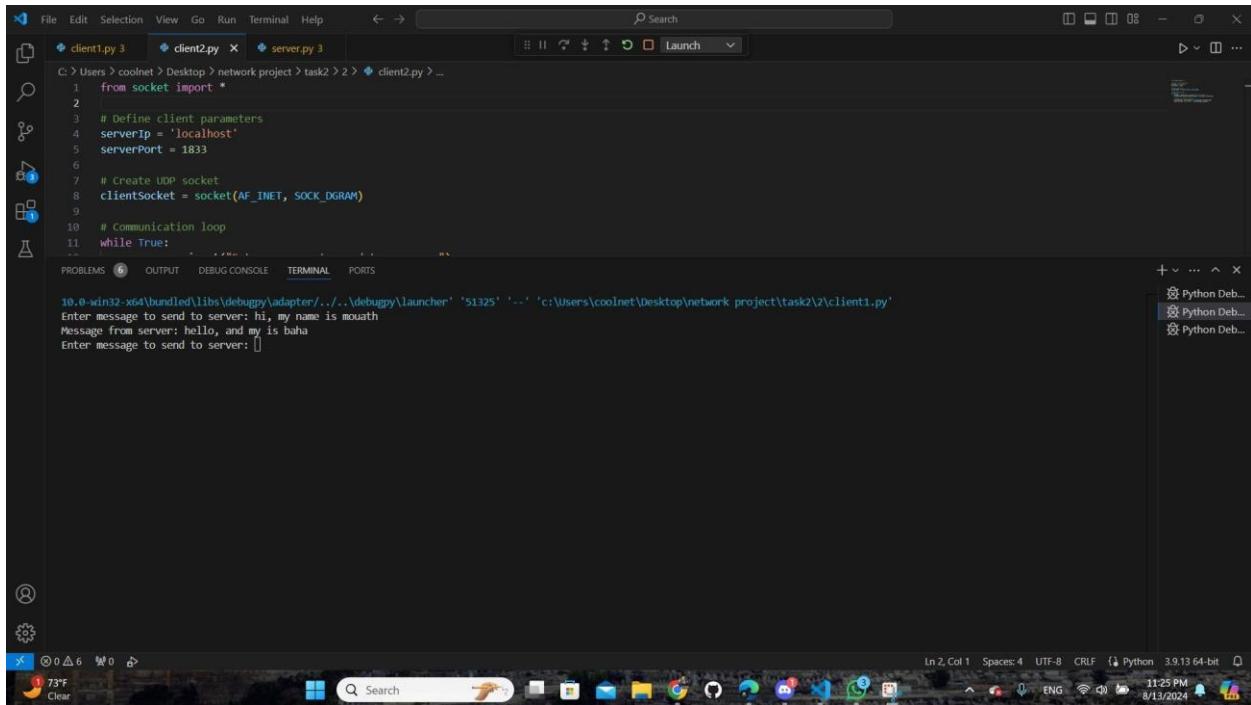
```
C:\> Users > coolnet > Desktop > network project > task2 > 2 > server.py > ...
1  from socket import *
2  import threading
3
4  # Define server parameters
5  serverPort = 1833
6  serverIp = 'localhost'
7
8  # Create UDP socket
9  serverSocket = socket(AF_INET, SOCK_DGRAM)
10 serverSocket.bind((serverIp, serverPort))
11 print("Server is on and ready to receive messages...")
12
13 # Dictionary to keep track of the last message from each client
14 clients = {}
15
16 def handle_client(clientMessage, clientAddress):
17     clientNumber = clients.get(clientAddress, len(clients) + 1)
```

Below the code, the terminal pane displays the output: "Server is on and ready to receive messages...".

Figure 24:starting the server

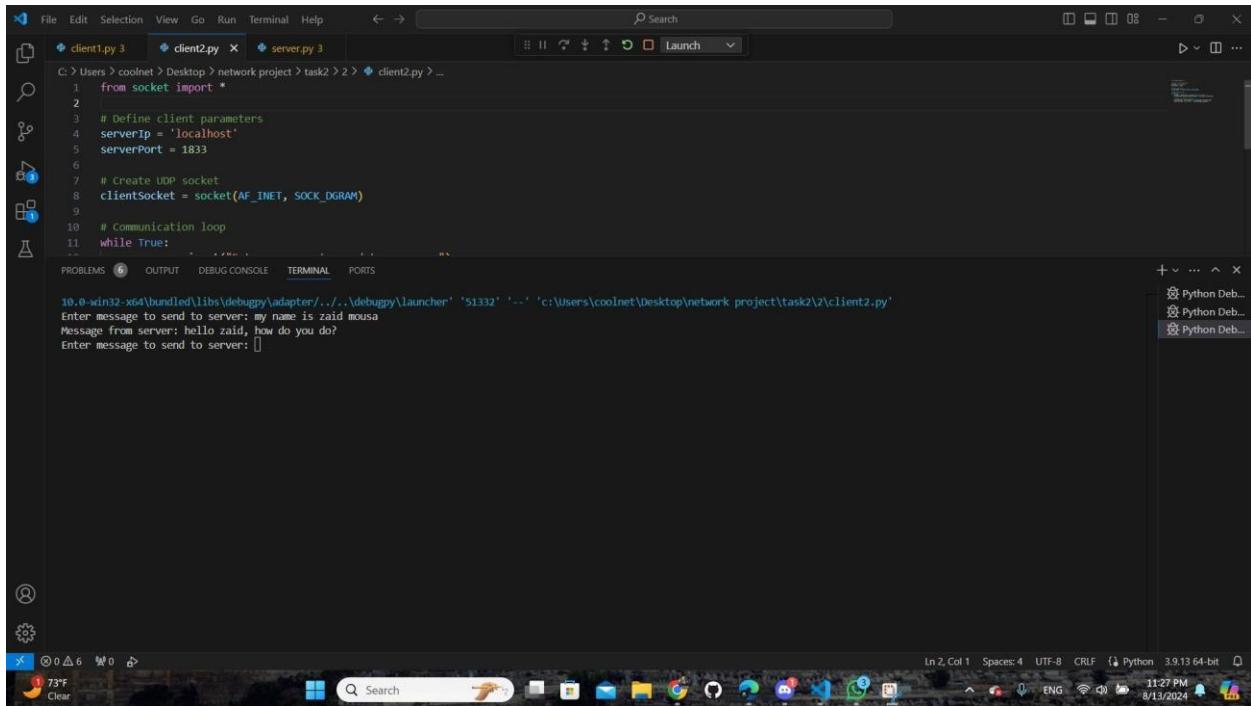
After the server is started and configured to connect to more than one client, it prints a message that it is ready to connect with clients.

- The Client Output:



```
C:\Users\coonet>Desktop>network project>task2>2>client1.py > ...
1  from socket import *
2
3  # Define client parameters
4  serverIp = 'localhost'
5  serverPort = 1833
6
7  # Create UDP socket
8  clientSocket = socket(AF_INET, SOCK_DGRAM)
9
10 # Communication loop
11 while True:
12
13     Enter message to send to server: hi, my name is mouatsh
14     Message from server: hello, and my is baha
15     Enter message to send to server: [ ]
```

Figure 25: client 1 message

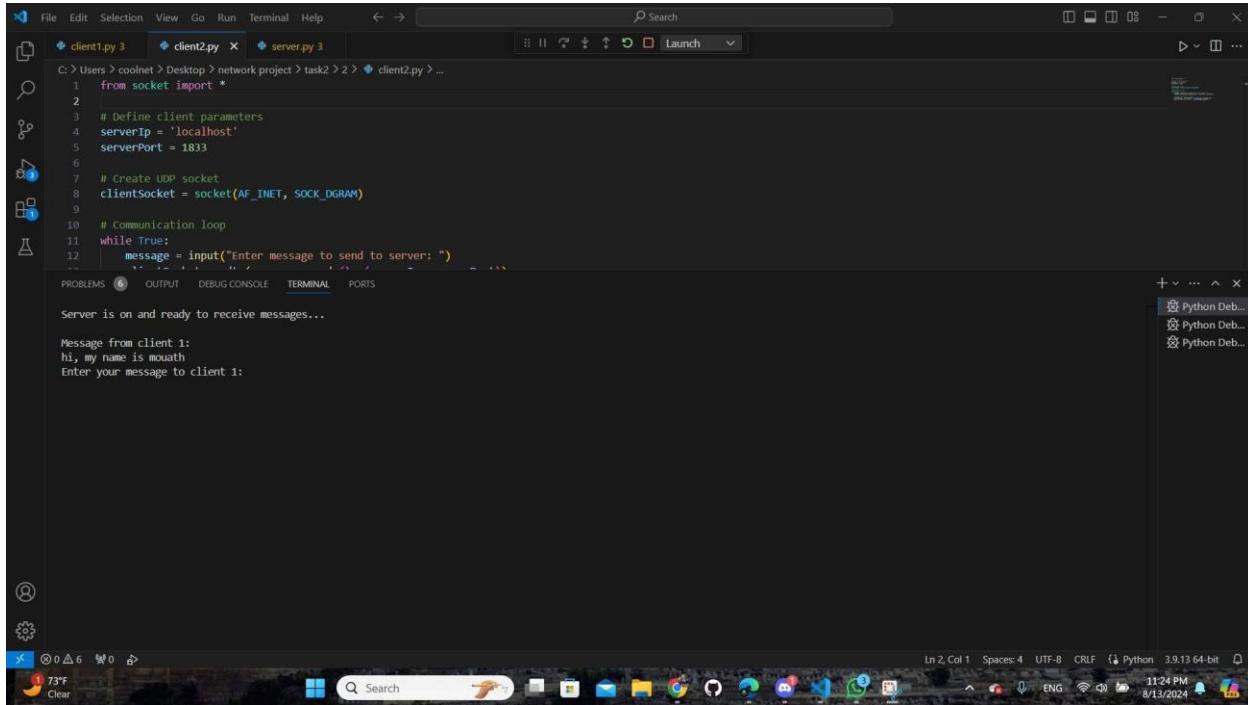


```
C:\Users\coonet>Desktop>network project>task2>2>client2.py > ...
1  from socket import *
2
3  # Define client parameters
4  serverIp = 'localhost'
5  serverPort = 1833
6
7  # Create UDP socket
8  clientSocket = socket(AF_INET, SOCK_DGRAM)
9
10 # Communication loop
11 while True:
12
13     Enter message to send to server: my name is zaid mousa
14     Message from server: hello zaid, how do you do?
15     Enter message to send to server: [ ]
```

Figure 26: client 2 message

After the client connects to the server, it requests the server's message, and when it arrives there, the server responds to the message (exchanging messages like Messenger, etc.).

- The Second Server Output Message:



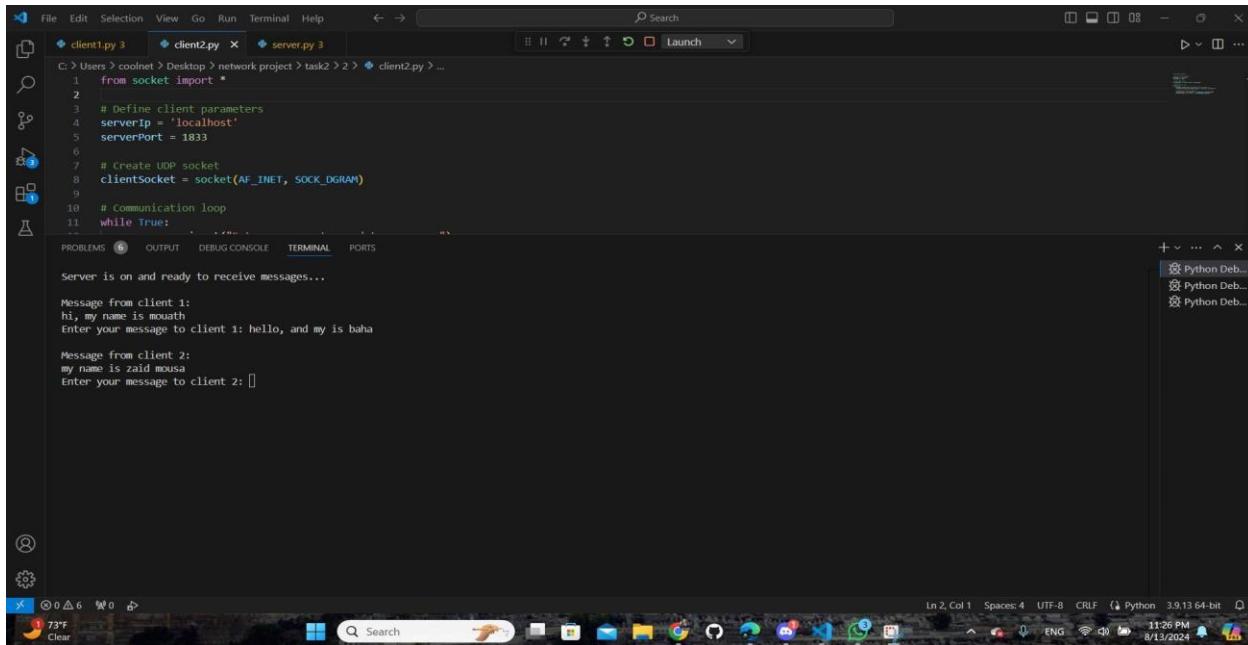
The screenshot shows a code editor interface with three tabs: client1.py, client2.py, and server.py. The server.py tab is active, displaying the following code:

```
C:\> Users > coolnet > Desktop > network project > task2 > 2 > client2.py > ...
1  from socket import *
2
3  # Define client parameters
4  serverIp = 'localhost'
5  serverPort = 1833
6
7  # Create UDP socket
8  clientSocket = socket(AF_INET, SOCK_DGRAM)
9
10 # Communication loop
11 while True:
12     message = input("Enter message to send to server: ")
```

Below the code, the terminal output shows:

```
Server is on and ready to receive messages...
Message from client 1:
hi, my name is mouat
Enter your message to client 1:
```

Figure 27: server receive client 1 message



The screenshot shows a code editor interface with three tabs: client1.py, client2.py, and server.py. The server.py tab is active, displaying the same code as Figure 27. The terminal output shows messages from both client 1 and client 2:

```
Server is on and ready to receive messages...
Message from client 1:
hi, my name is mouat
Enter your message to client 1: hello, and my is baha

Message from client 2:
my name is zaid mousa
Enter your message to client 2: []
```

Figure 28:server receive the client 2 message

After the message arrives from the client, it is printed who is the client who sent the message and what is the message sent, and he can reply to the sent message, but the client who sent the message cannot resend it unless the server replies to him.

Task 3:

1-

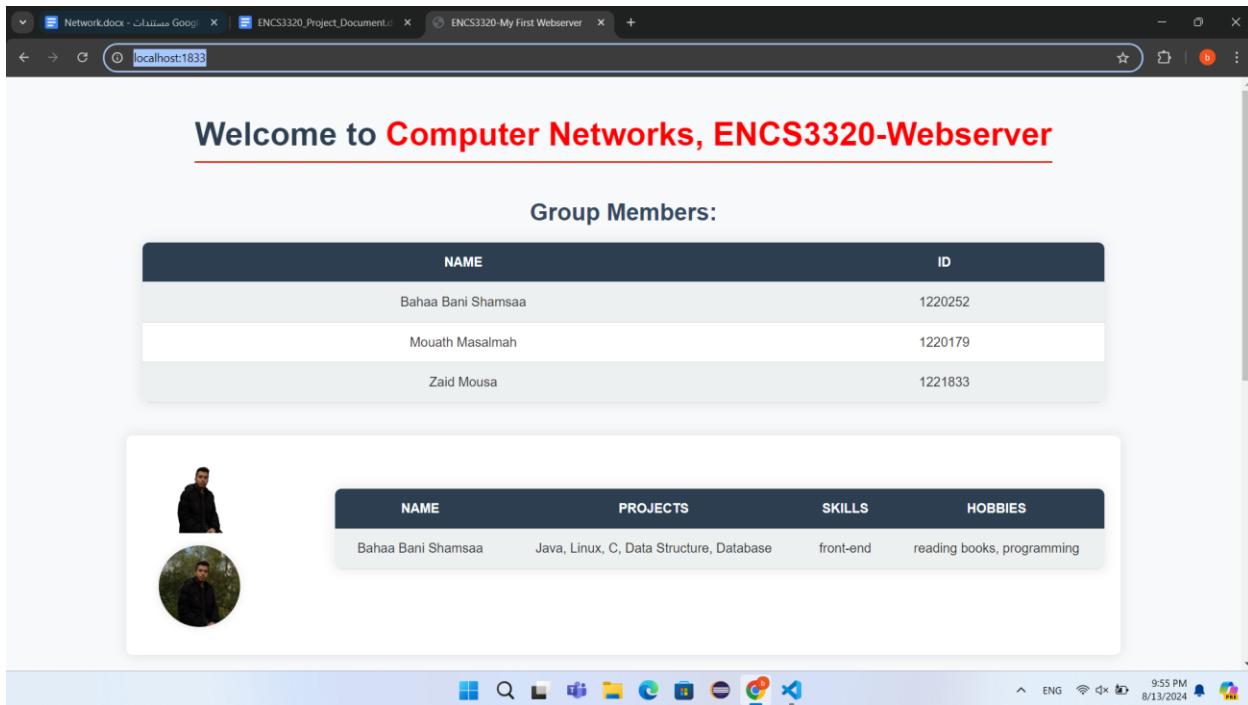


Figure 29:request /

As we see here when I type link <http://localhost:1833/> it goes to the english page

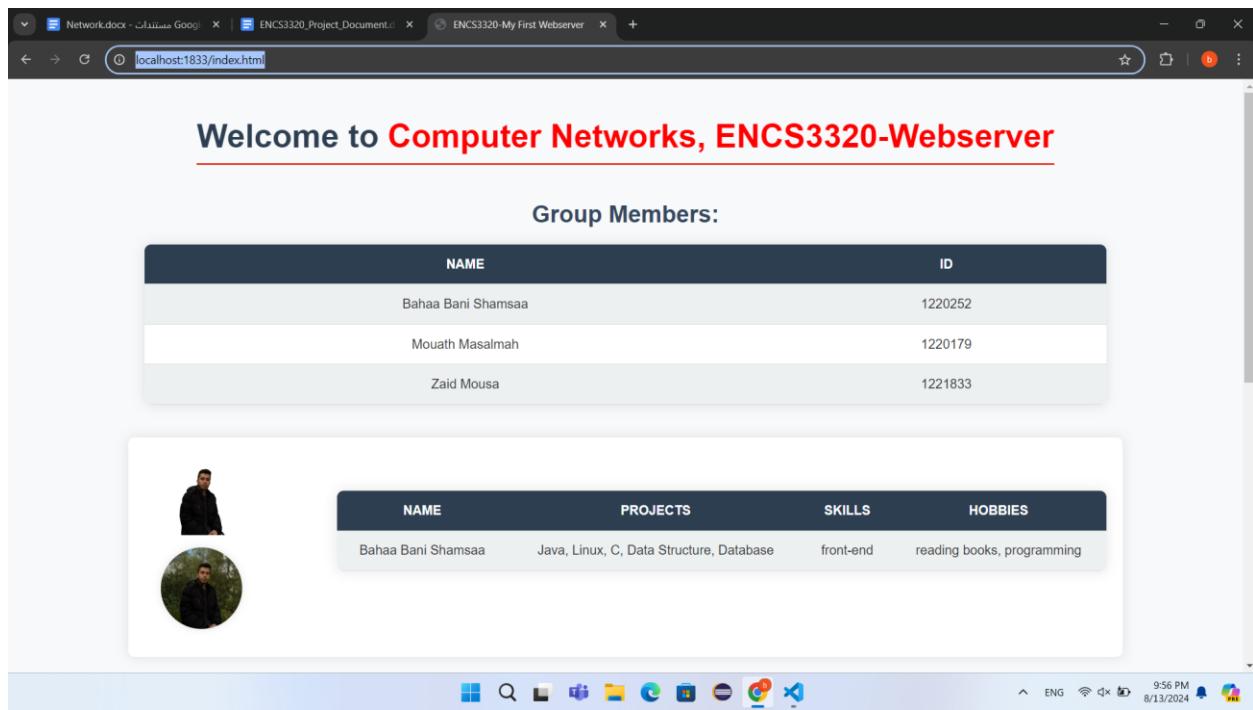


Figure 30: index.html request

As we see here when I type link <http://localhost:1833/index.html> it goes to the english page

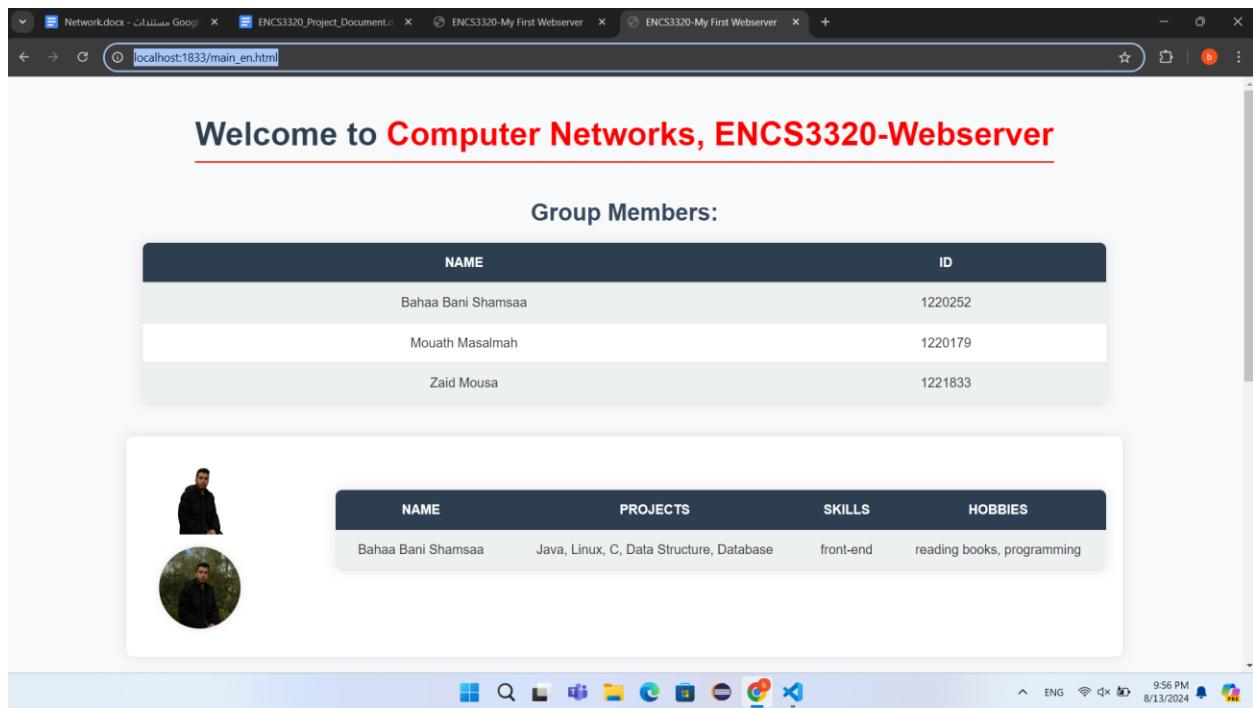


Figure 31: main_en.html request

As we see here when I type link http://localhost:1833/main_en.html it goes to the english page

a.

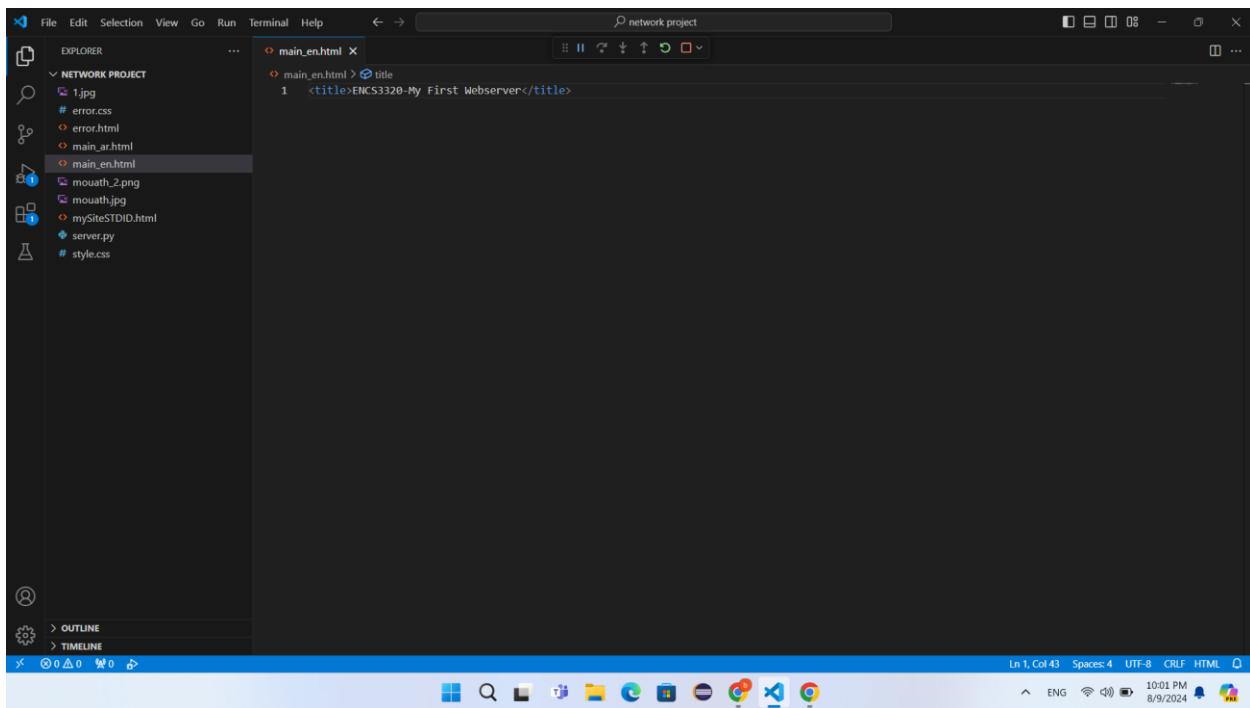


Figure 32: title of the page

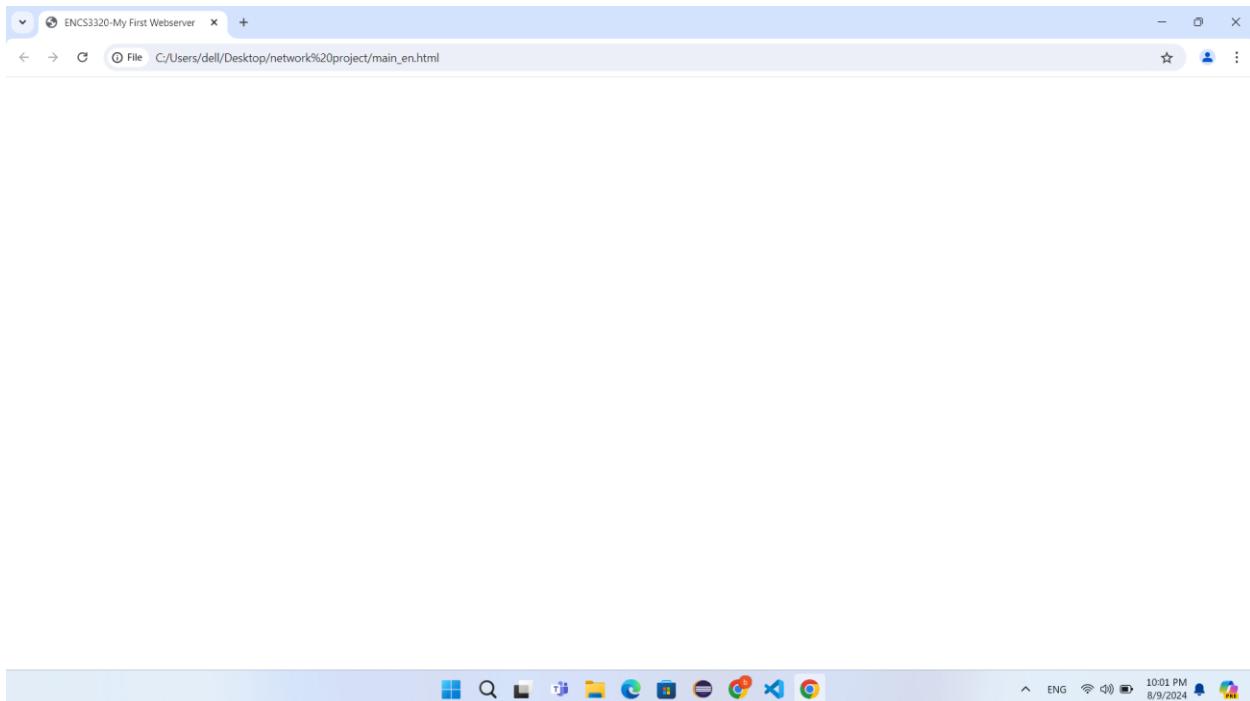


Figure 33: browser title

Here we put the title ENCS-My First Webserver

b.

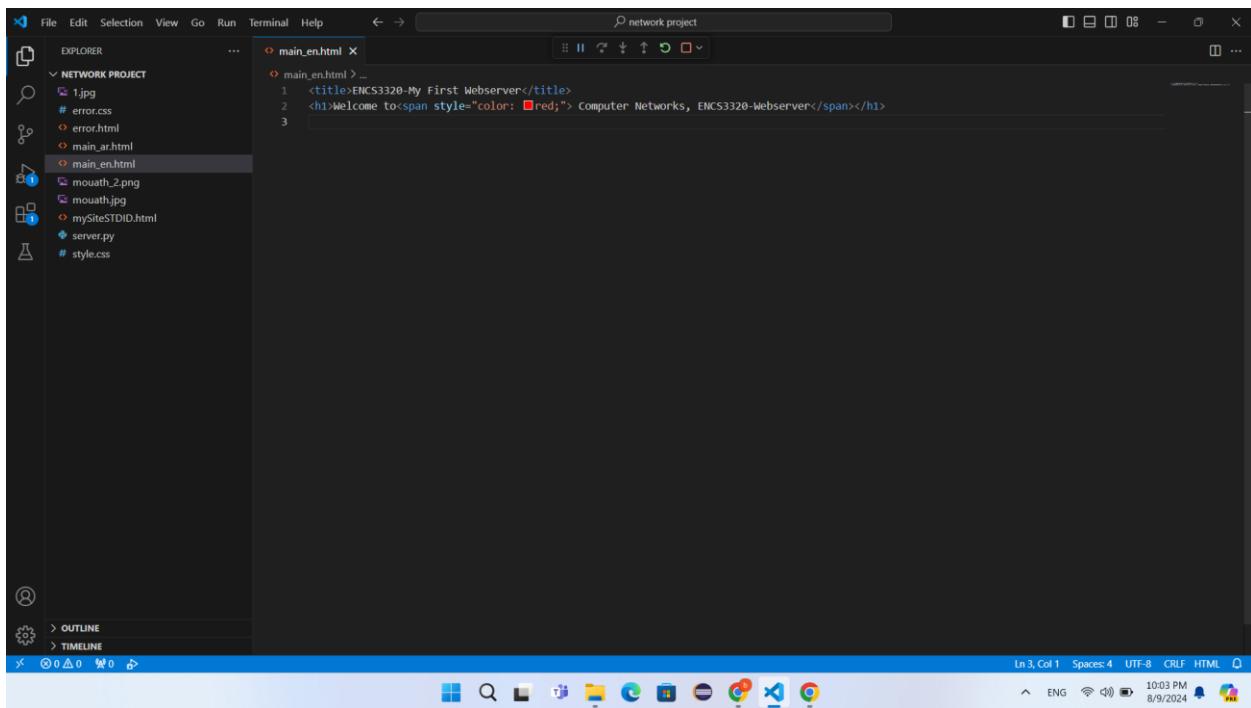


Figure 34: welcome message

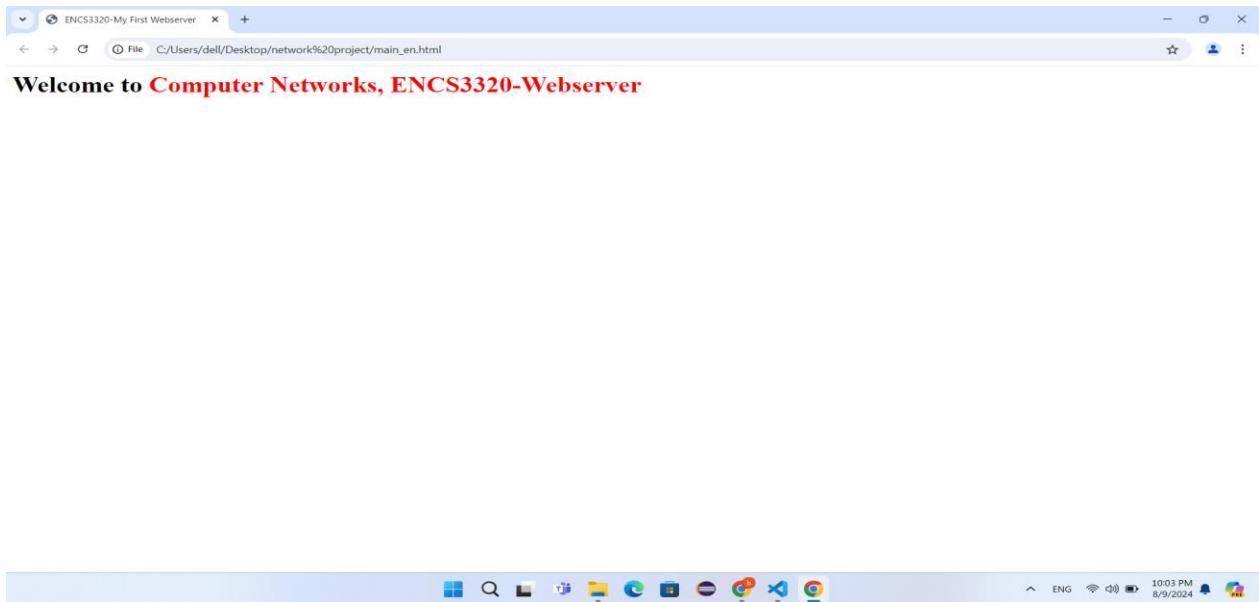


Figure 35: browser welcome

Here I use span to make the words with different colors

c.

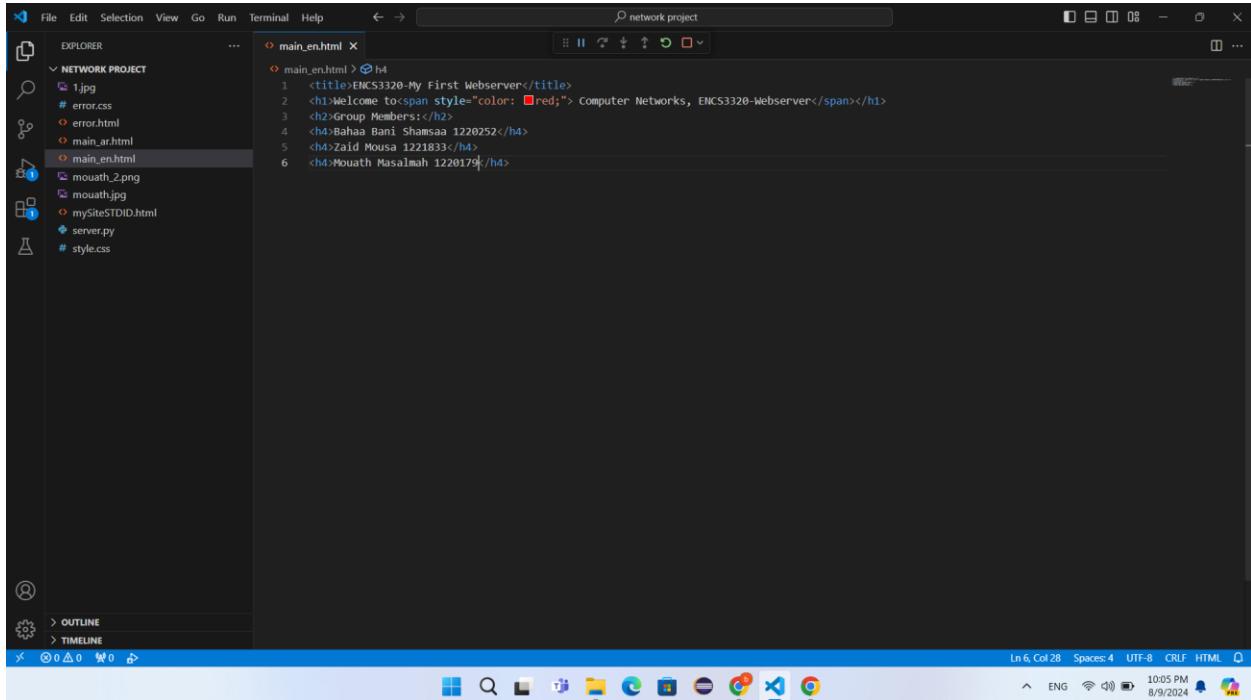


Figure 36:group members

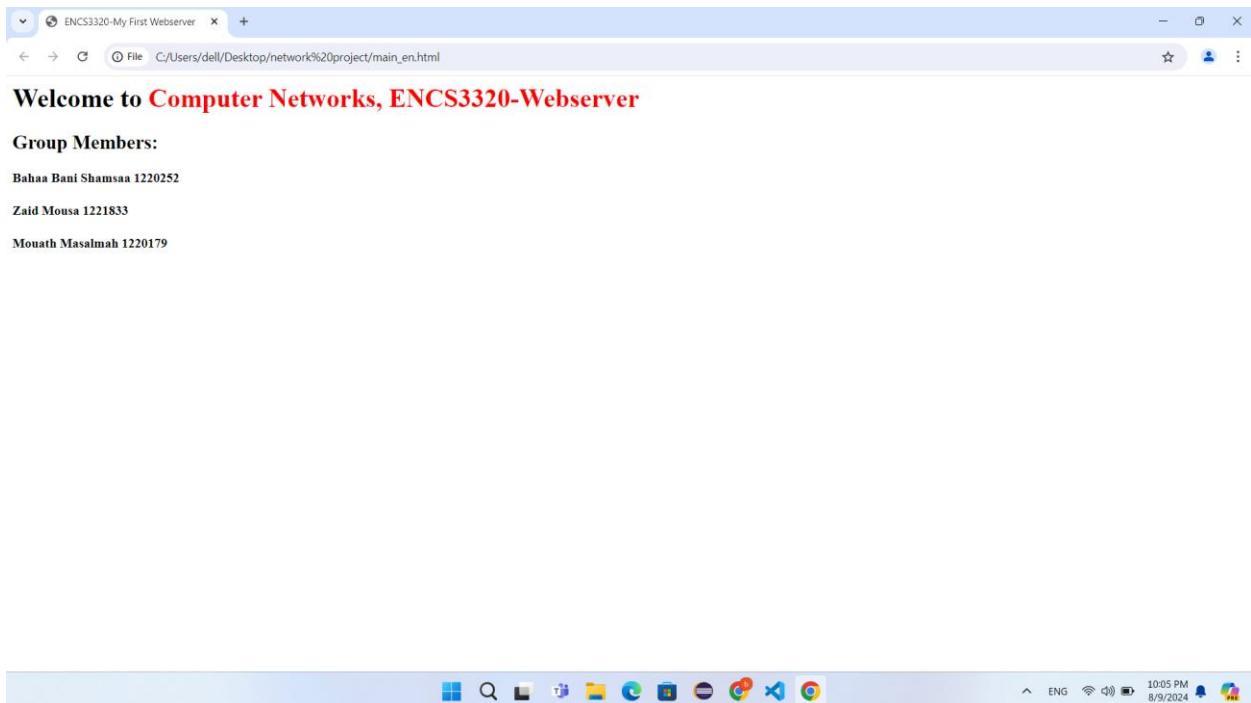
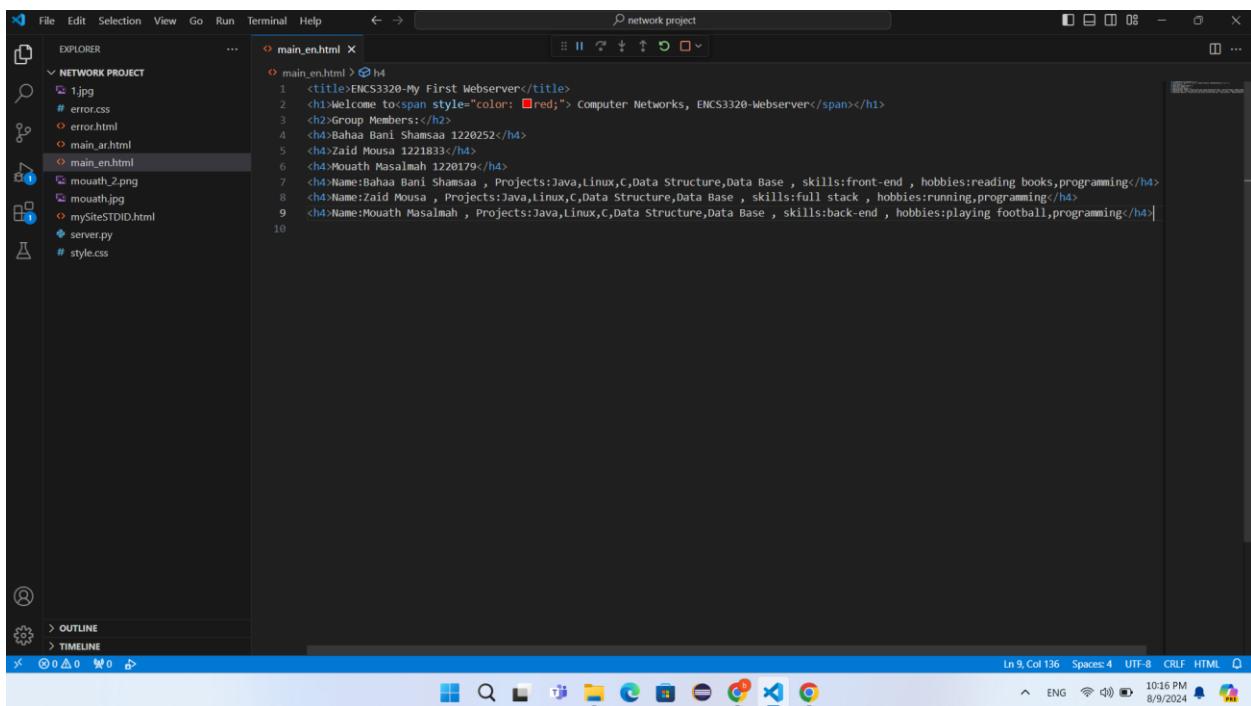


Figure 37: browser group members

Here we type the group members names with the numbers

d.



The screenshot shows a code editor window with the following details:

- File Path:** network project\main_en.html
- Content:**

```
<title>ENCS3320-My First Webserver</title>
<h1>Welcome to<span style="color: red;> Computer Networks, ENCS3320-Webserver:</span></h1>
<h2>Group Members:</h2>
<h3>Bahaa Bani Shamsaa 1220252</h3>
<h3>Zaid Mousa 1221833</h3>
<h3>Mouath Masalmah 1220179</h3>
Name:Bahaa Bani Shamsaa , Projects:Java,Linux,C,Data Structure,Data Base , skills:front-end , hobbies:reading books,programming
Name:Zaid Mousa , Projects:Java,Linux,C,Data Structure,Data Base , skills:full stack , hobbies:running,programming
Name:Mouath Masalmah , Projects:Java,Linux,C,Data Structure,Data Base , skills:back-end , hobbies:playing football,programming
```
- Tools Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help, a search bar, and icons for file operations.
- Status Bar:** Ln 9, Col 136, Spaces: 4, UTF-8, CRLF, HTML, and a date/time stamp: 8/9/2024 10:16 PM.

Figure 38: info about group members

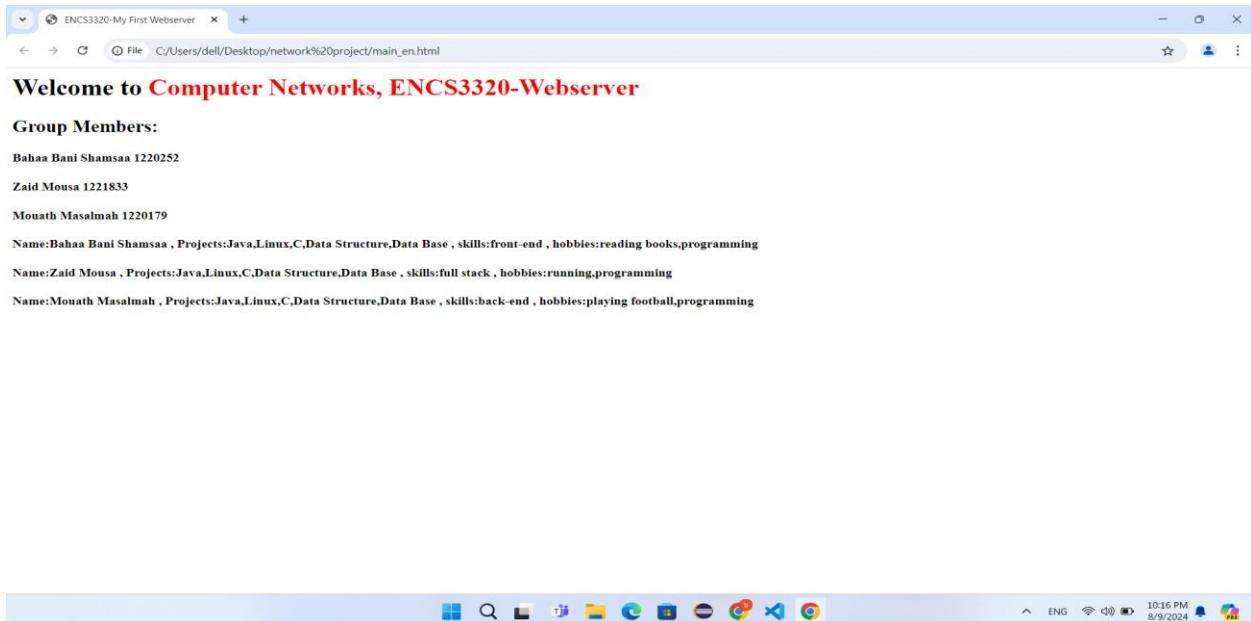
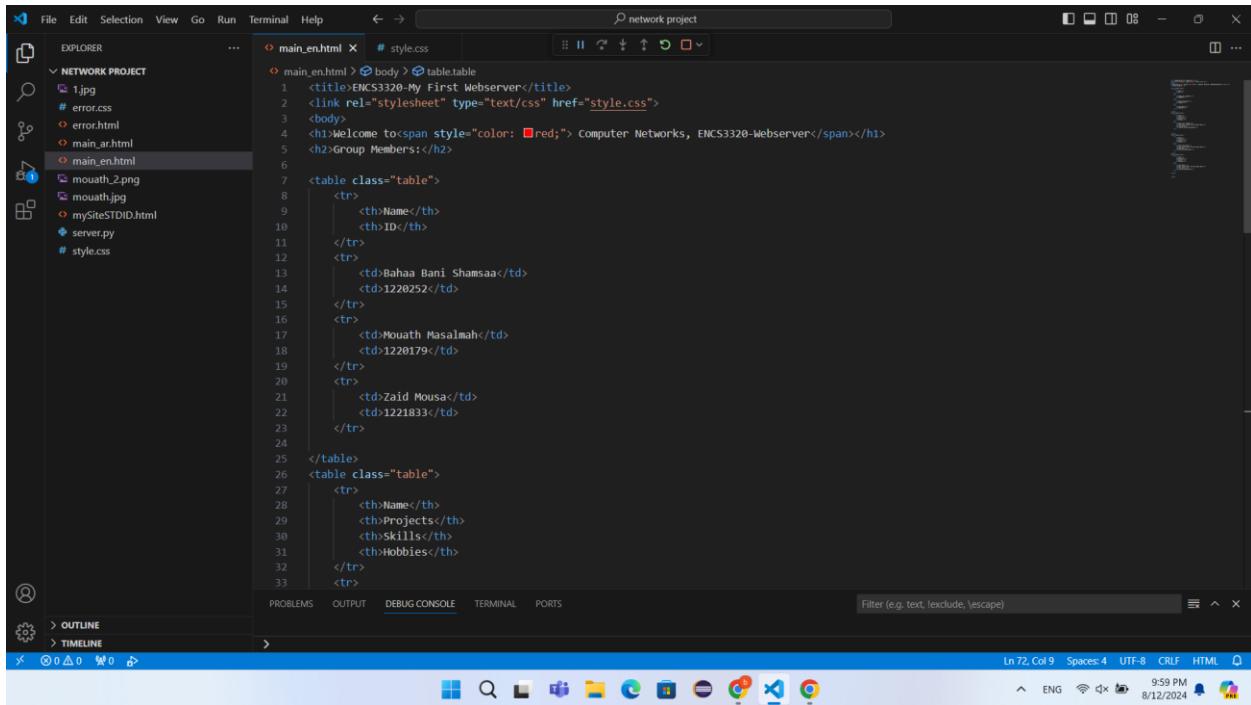


Figure 39:browser info group members

Here we type the names, projects, skills and hobbies

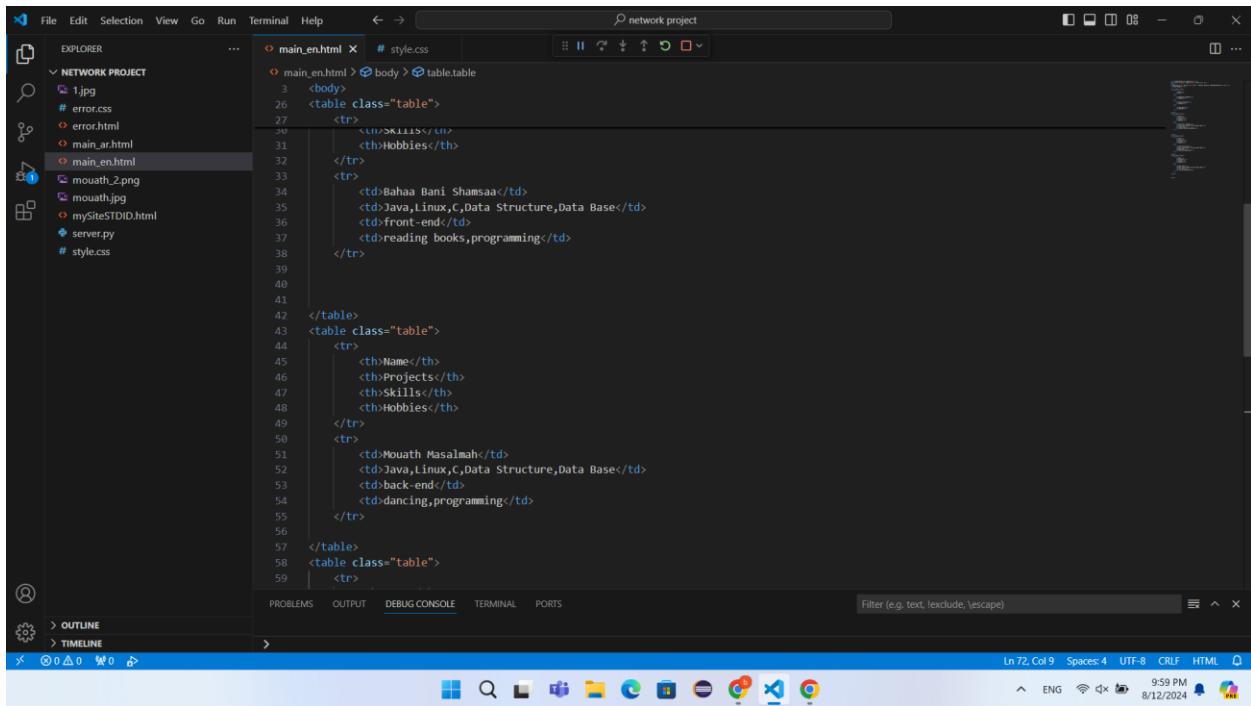
e and f:



```
<main_en.html> # style.css
<body>
    <table>
        <tr>
            <th>Name</th>
            <th>ID</th>
        </tr>
        <tr>
            <td>Bahaa Bani Shamsaa</td>
            <td>1220252</td>
        </tr>
        <tr>
            <td>Moush Masalmah</td>
            <td>1220179</td>
        </tr>
        <tr>
            <td>Zaid Mousa</td>
            <td>1221833</td>
        </tr>
    </table>
    <table>
        <tr>
            <th>Name</th>
            <th>Projects</th>
            <th>Skills</th>
            <th>Hobbies</th>
        </tr>
        <tr>
            <td>Bahaa Bani Shamsaa</td>
            <td>Java,Linux,C,Data Structure,Data Base</td>
            <td>front-end</td>
            <td>reading books,programming</td>
        </tr>
        <tr>
            <td>Moush Masalmah</td>
            <td>Java,Linux,C,Data Structure,Data Base</td>
            <td>back-end</td>
            <td>dancing,programming</td>
        </tr>
    </table>

```

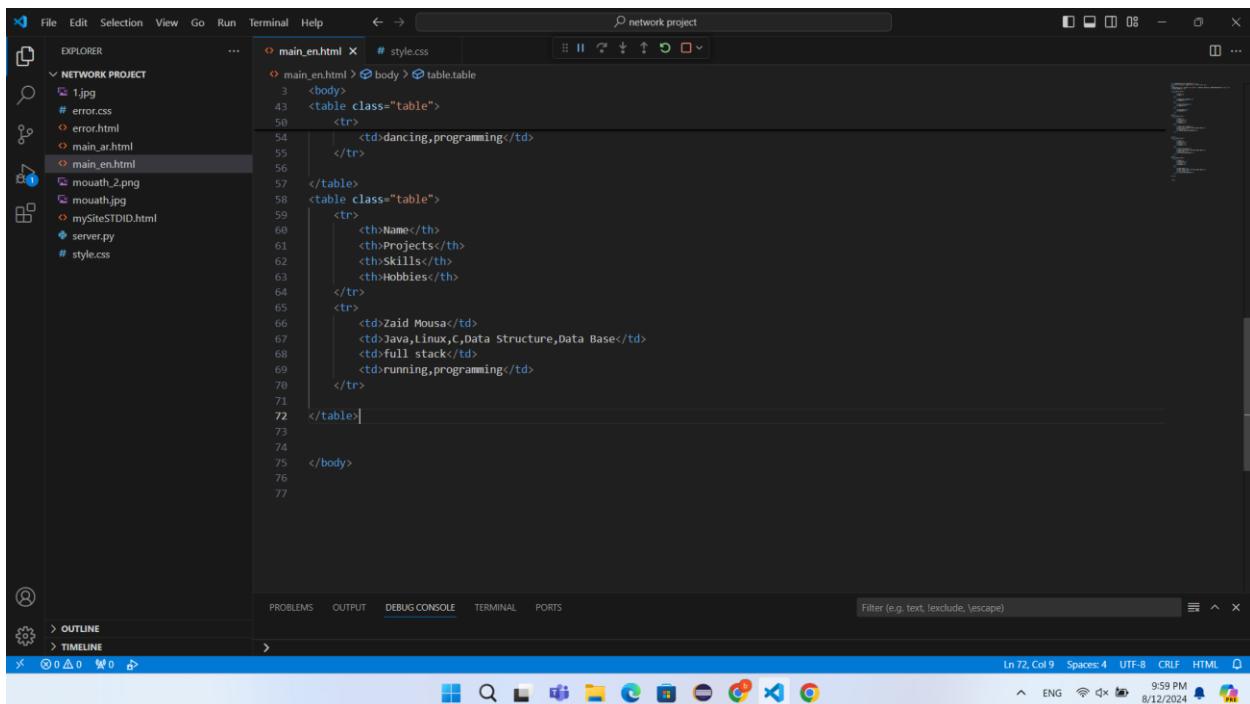
Figure 40: table 1



```
<main_en.html> # style.css
<body>
    <table>
        <tr>
            <th>Name</th>
            <th>Projects</th>
            <th>Skills</th>
            <th>Hobbies</th>
        </tr>
        <tr>
            <td>Bahaa Bani Shamsaa</td>
            <td>Java,Linux,C,Data Structure,Data Base</td>
            <td>front-end</td>
            <td>reading books,programming</td>
        </tr>
        <tr>
            <td>Moush Masalmah</td>
            <td>Java,Linux,C,Data Structure,Data Base</td>
            <td>back-end</td>
            <td>dancing,programming</td>
        </tr>
    </table>
    <table>
        <tr>
            <th>Name</th>
            <th>Projects</th>
            <th>Skills</th>
            <th>Hobbies</th>
        </tr>
        <tr>
            <td>Zaid Mousa</td>
            <td>Java,Linux,C,Data Structure,Data Base</td>
            <td>back-end</td>
            <td>dancing,programming</td>
        </tr>
    </table>
    <table>
        <tr>
            <th>Name</th>
            <th>ID</th>
        </tr>
        <tr>
            <td>Bahaa Bani Shamsaa</td>
            <td>1220252</td>
        </tr>
        <tr>
            <td>Moush Masalmah</td>
            <td>1220179</td>
        </tr>
    </table>

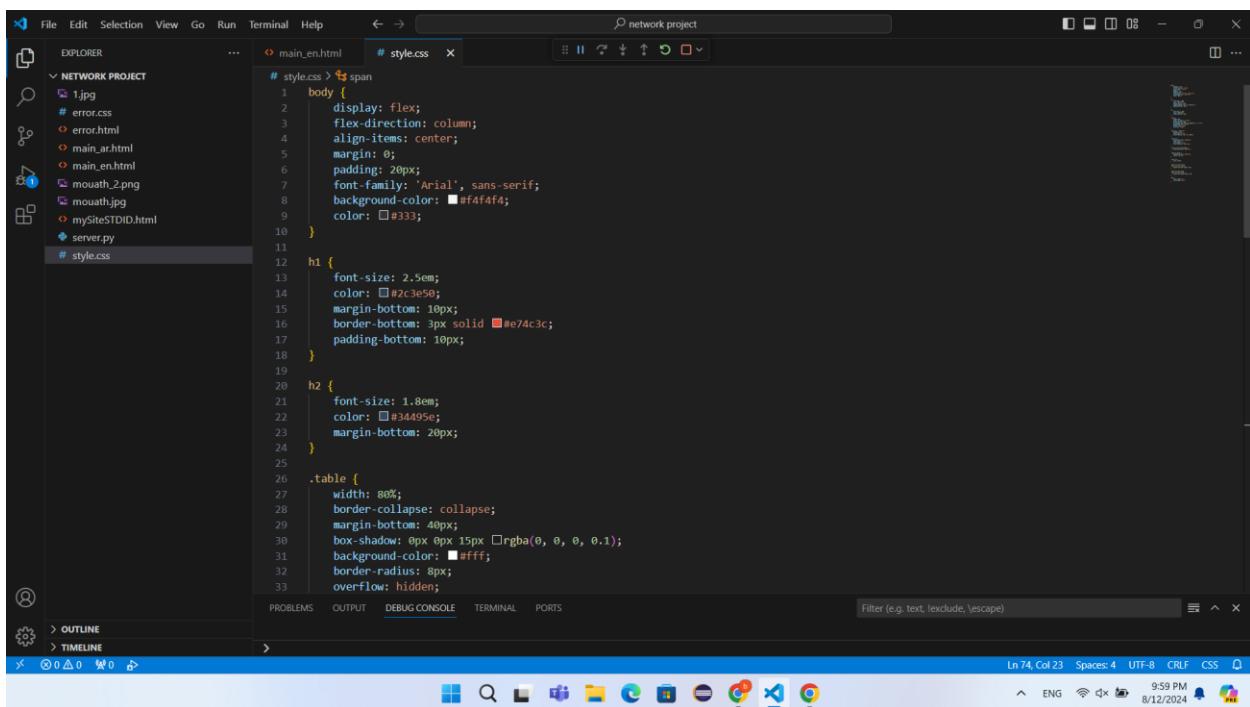
```

Figure 41: table 2



```
<body>
  <table class="table">
    <tr>
      <td>dancing,programming</td>
    </tr>
  </table>
  <table class="table">
    <tr>
      <th>Name</th>
      <th>Projects</th>
      <th>Skills</th>
      <th>Hobbies</th>
    </tr>
    <tr>
      <td>Zaid Mousa</td>
      <td>Java,Linux,C,Data Structure,Data Base</td>
      <td>full stack</td>
      <td>running,programming</td>
    </tr>
  </table>
</body>
```

Figure 42: table 3



```
# style.css > span
body {
  display: flex;
  flex-direction: column;
  align-items: center;
  margin: 0;
  padding: 20px;
  font-family: 'Arial', sans-serif;
  background-color: #f4f4f4;
  color: #333;
}

h1 {
  font-size: 2.5em;
  color: #2c3e50;
  margin-bottom: 10px;
  border-bottom: 3px solid #e74c3c;
  padding-bottom: 10px;
}

h2 {
  font-size: 1.8em;
  color: #34495e;
  margin-bottom: 20px;
}

.table {
  width: 80%;
  border-collapse: collapse;
  margin-bottom: 40px;
  box-shadow: 0px 0px 15px rgba(0, 0, 0, 0.1);
  background-color: #fff;
  border-radius: 8px;
  overflow: hidden;
```

Figure 43: css for table 1

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows a "NETWORK PROJECT" folder containing files like 1.jpg, #error.css, error.html, main_ar.html, main_en.html, mouath_2.png, mouath.jpg, mySiteSTDID.html, server.py, and #style.css.
- Editor:** Displays the content of the #style.css file, specifically the styles for table 2.

```
# style.css > span
26 .table {
27   background-color: #fff;
28   border-radius: 8px;
29   overflow: hidden;
30 }
31
32 .table th, .table td {
33   padding: 15px;
34   text-align: center;
35   border-bottom: 1px solid #ddd;
36 }
37
38 .table th {
39   background-color: #34495e;
40   color: #fff;
41   font-weight: bold;
42   text-transform: uppercase;
43 }
44
45 .table tr:nth-child(even) {
46   background-color: #f2f2f2;
47 }
48
49 .table tr:hover {
50   background-color: #f9c5c5;
51   cursor: pointer;
52 }
53
54 .table td {
55   color: #555;
56 }
57
58 .table th:first-child,
59 .table th:last-child,
```
- Bottom Bar:** Includes tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS. It also shows the current line (Ln 74, Col 23), spaces (Spaces: 4), encoding (UTF-8), and file type (CRLF, CSS).

Figure 44: css for table 2

2

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows a "NETWORK PROJECT" folder containing files like 1.jpg, #error.css, error.html, main_ar.html, main_en.html, mouath_2.png, mouath.jpg, mySiteSTDID.html, server.py, and #style.css.
- Editor:** Displays the content of the #style.css file, specifically the styles for table 3.

```
# style.css > span
53 .table tr:hover {
54   cursor: pointer;
55 }
56
57 .table td {
58   color: #555;
59 }
60
61 .table th:first-child,
62 .table th:last-child {
63   border-top-left-radius: 8px;
64 }
65
66 .table th:last-child,
67 .table td:last-child {
68   border-top-right-radius: 8px;
69 }
70
71 span {
72   color: #e74c3c;
73   font-weight: bold;
74 }
75
76
77
```
- Bottom Bar:** Includes tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS. It also shows the current line (Ln 74, Col 23), spaces (Spaces: 4), encoding (UTF-8), and file type (CRLF, CSS).

Figure 45: css for table 3

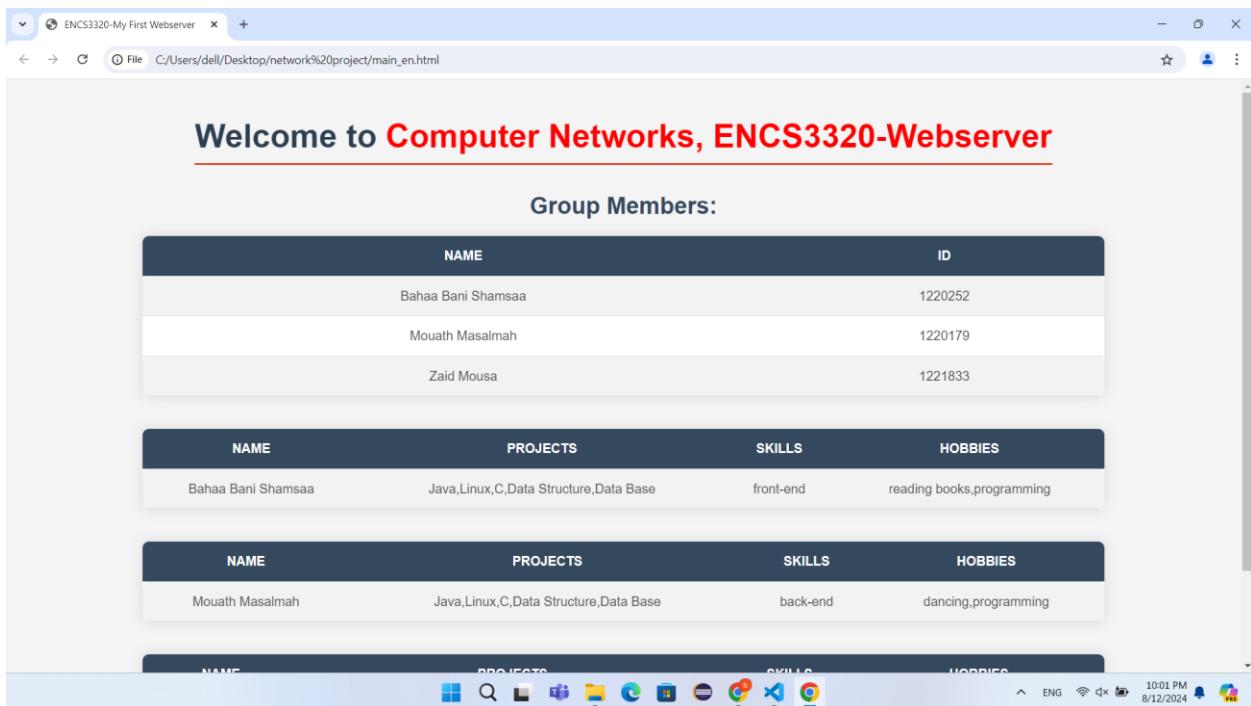


Figure 46: browser for table 1

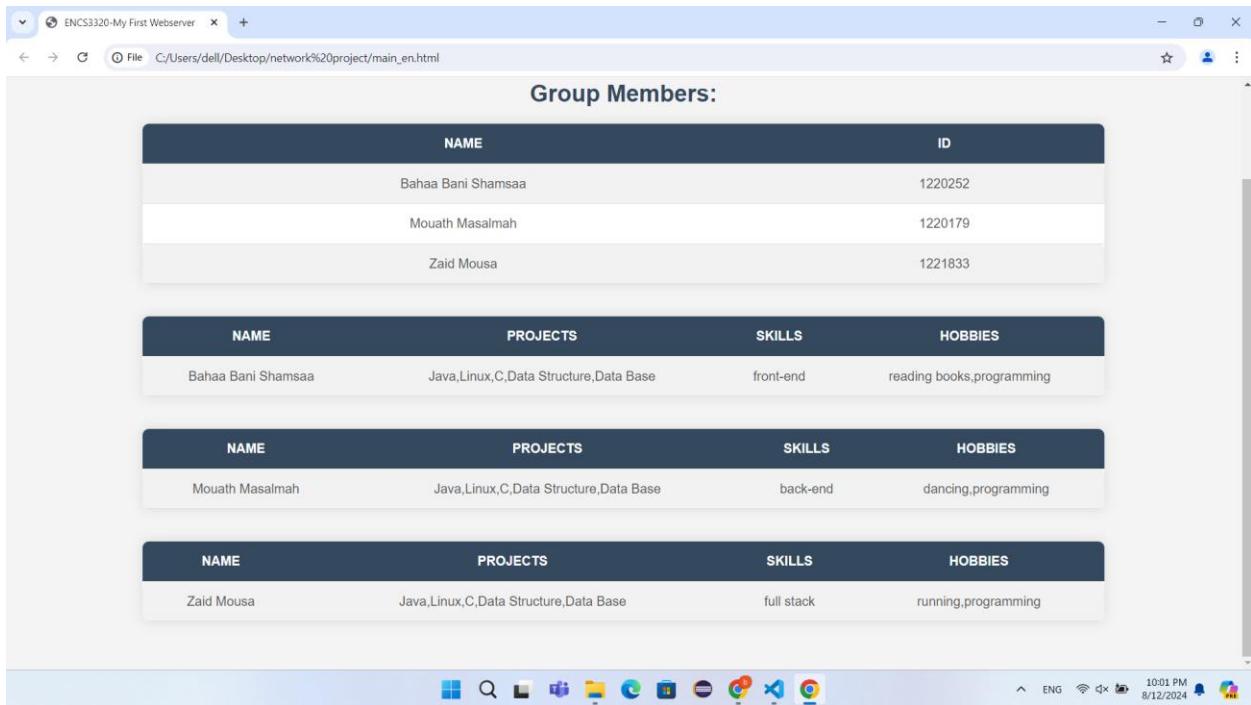
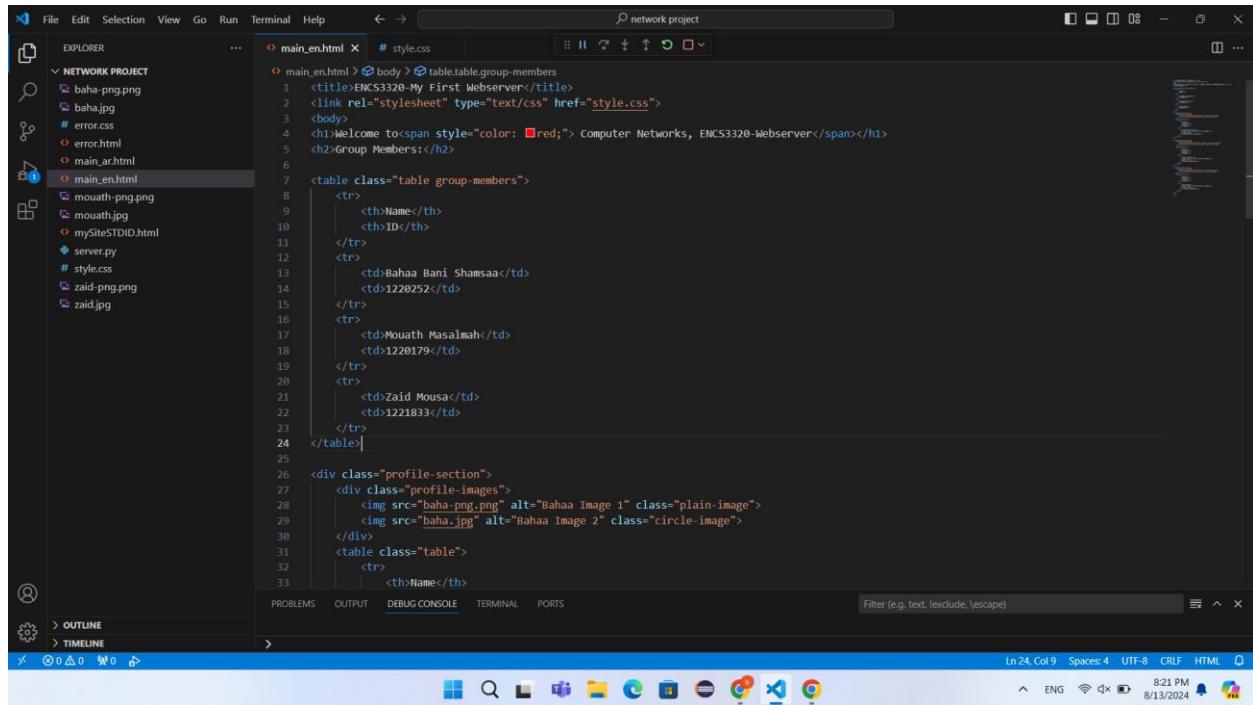


Figure 47: browser for table 2

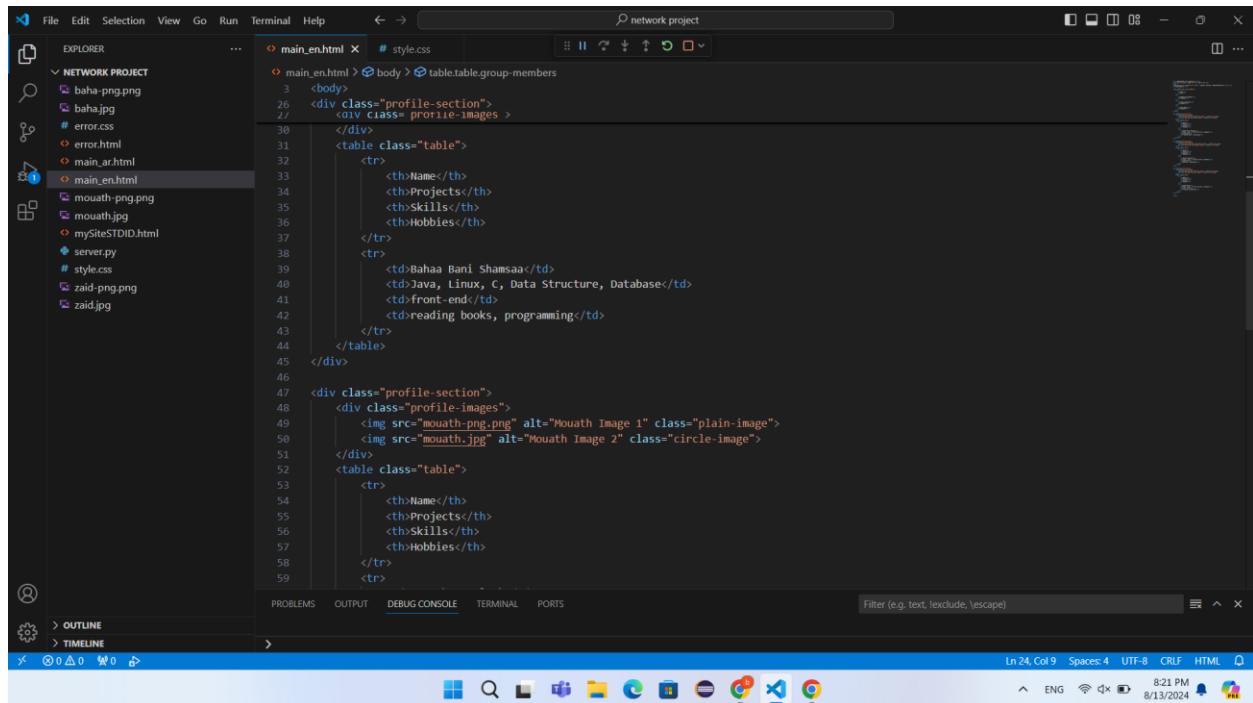
Here we divide the group members names and IDs in table and each member with his information in tables, and we use a good CSS to make the page look nice, we use things like hover, background, border

h.



```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8" />
        <title>ENCS3320-My First Webserver</title>
        <link rel="stylesheet" type="text/css" href="style.css" />
    </head>
    <body>
        <h1>Welcome to<span style="color: red;> Computer Networks, ENCS3320-Webserver</span></h1>
        <h2>Group Members:</h2>
        <table class="table group-members">
            <tr>
                <th>Name</th>
                <th>ID</th>
            </tr>
            <tr>
                <td>Baha Bani Shamsaa</td>
                <td>1220252</td>
            </tr>
            <tr>
                <td>Mouath Masalmah</td>
                <td>1220179</td>
            </tr>
            <tr>
                <td>Zaid Mousa</td>
                <td>1221833</td>
            </tr>
        </table>
        <div class="profile-section">
            <div class="profile-images">
                
                
            </div>
            <table class="table">
                <tr>
                    <th>Name</th>
                </tr>
```

Figure 48:png and jpg 1



```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8" />
        <title>ENCS3320-My First Webserver</title>
        <link rel="stylesheet" type="text/css" href="style.css" />
    </head>
    <body>
        <div class="profile-section">
            <div class="profile-images">
                
                
            </div>
            <table class="table">
                <tr>
                    <th>Name</th>
                    <th>Projects</th>
                    <th>Skills</th>
                    <th>Hobbies</th>
                </tr>
                <tr>
                    <td>Baha Bani Shamsaa</td>
                    <td>Java, Linux, C, Data Structure, Database</td>
                    <td>front-end</td>
                    <td>reading books, programming</td>
                </tr>
            </table>
        </div>
        <div class="profile-section">
            <div class="profile-images">
                
                
            </div>
            <table class="table">
                <tr>
                    <th>Name</th>
                    <th>Projects</th>
                    <th>Skills</th>
                    <th>Hobbies</th>
                </tr>
```

Figure 49:png and jpg 2

The screenshot shows the Visual Studio Code interface with the 'main_en.html' file open in the editor. The code displays a profile section with a table and two image placeholders. The table rows contain fields for Name, Projects, Skills, and Hobbies, with sample data for 'Mouath Masalmah' and 'Java, Linux, C, Data Structure, Database'. The image placeholders are 'zaid-png.png' and 'zaid.jpg'.

```
<body>
  <div class="profile-section">
    <table class="table">
      <tr>
        <th>Name</th>
        <th>Projects</th>
        <th>Skills</th>
        <th>Hobbies</th>
      </tr>
      <tr>
        <td>Mouath Masalmah</td>
        <td>Java, Linux, C, Data Structure, Database</td>
        <td>back-end</td>
        <td>dancing, programming</td>
      </tr>
    </table>
  </div>
  <div class="profile-section">
    <div class="profile-images">
      
      
    </div>
    <table class="table">
      <tr>
        <th>Name</th>
        <th>Projects</th>
        <th>Skills</th>
        <th>Hobbies</th>
      </tr>
      <tr>
        <td>Zaid Mousa</td>
        <td>Java, Linux, C, Data Structure, Database</td>
        <td>full stack</td>
        <td>running, programming</td>
      </tr>
    </table>
  </div>
</body>
```

Figure 50:png and jpg 3

The screenshot shows the Visual Studio Code interface with the 'main_en.html' file open in the editor. The code has been modified to reflect the changes made in Figure 50. The table now contains data for 'Zaid Mousa' with skills like 'full stack' and hobbies like 'running, programming'. The image placeholders remain the same.

```
<body>
  <div class="profile-section">
    <table class="table">
      <tr>
        <th>Name</th>
        <th>Projects</th>
        <th>Skills</th>
        <th>Hobbies</th>
      </tr>
      <tr>
        <td>Zaid Mousa</td>
        <td>Java, Linux, C, Data Structure, Database</td>
        <td>full stack</td>
        <td>running, programming</td>
      </tr>
    </table>
  </div>
</body>
```

Figure 51:png and jpg 4

The screenshot shows a code editor interface with the following details:

- File Explorer:** Shows a "NETWORK PROJECT" folder containing files: baha-png.png, baha.jpg, #error.css, error.html, main_ar.html, main_en.html, mousath-png.png, mousath.jpg, mySiteSTID.html, and server.py.
- Editor:** The main editor window displays the content of style.css. The code includes styles for body, h1, h2, and a table.
- Bottom Bar:** Includes tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS. It also shows system icons like battery level, signal strength, and volume, along with the date and time (8/13/2024, 8:21 PM).

Figure 52:css png and jpg 1

The screenshot shows a code editor interface with the following details:

- File Explorer:** Shows a "NETWORK PROJECT" folder containing files: baha-png.png, baha.jpg, #error.css, error.html, main_ar.html, main_en.html, mousath-png.png, mousath.jpg, mySiteSTID.html, and server.py.
- Editor:** The main editor window displays the content of style.css. The code includes styles for .table, .table th, .table td, .table tr:nth-child(even), .table tr:hover, and .profile-section.
- Bottom Bar:** Includes tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS. It also shows system icons like battery level, signal strength, and volume, along with the date and time (8/13/2024, 8:21 PM).

Figure 53:css png and jpg 2

The screenshot shows a code editor interface with the following details:

- File Explorer:** Shows a "NETWORK PROJECT" folder containing files: baha-png.png, baha.jpg, #error.css, error.html, main_ar.html, main_en.html, mousath-png.png, mousath.jpg, mySiteSTID.html, server.py, and #style.css.
- Editor:** The main editor window displays the content of the #style.css file. The visible code includes:

```
# style.css > body
    .table tr:hover {
        cursor: pointer;
    }
    .profile-section {
        display: flex;
        justify-content: space-between;
        align-items: center;
        width: 80%;
        background-color: #ffffff;
        border-radius: 8px;
        padding: 20px;
        margin-bottom: 20px;
        box-shadow: 0px 0px 15px rgba(0, 0, 0, 0.1);
    }
    .profile-section:not(:last-child) {
        margin-bottom: 40px;
    }
    .profile-images {
        display: flex;
        flex-direction: column;
        align-items: center;
        justify-content: center;
        margin-left: 20px;
    }
    .circle-image {
        width: 100px;
        height: 100px;
        border-radius: 50%;
        margin-bottom: 15px;
        transition: transform 0.3s ease;
    }
    .circle-image:hover {
        transform: scale(1.05);
    }
    .plain-image {
        width: 100px;
        height: 100px;
        margin-bottom: 15px;
        transition: transform 0.3s ease;
    }
    .plain-image:hover {
        transform: scale(1.05);
    }
```

The bottom status bar shows: Ln 8, Col 32 | Spaces: 4 | UTF-8 | CRLF | CSS | Filter (e.g. text, /exclude, \escape).

Figure 54:css png and jpg 3

The screenshot shows a code editor interface with the following details:

- File Explorer:** Shows a "NETWORK PROJECT" folder containing files: baha-png.png, baha.jpg, #error.css, error.html, main_ar.html, main_en.html, mousath-png.png, mousath.jpg, mySiteSTID.html, server.py, and #style.css.
- Editor:** The main editor window displays the content of the #style.css file. The visible code includes:

```
# style.css > body
    .profile-images {
        justify-content: center;
        margin-left: 20px;
    }
    .circle-image {
        width: 100px;
        height: 100px;
        border-radius: 50%;
        margin-bottom: 15px;
        box-shadow: 0px 0px 10px rgba(0, 0, 0, 0.1);
        transition: transform 0.3s ease;
    }
    .circle-image:hover {
        transform: scale(1.05);
    }
    .plain-image {
        width: 100px;
        height: 100px;
        margin-bottom: 15px;
        transition: transform 0.3s ease;
    }
    .plain-image:hover {
        transform: scale(1.05);
    }
```

The bottom status bar shows: Ln 8, Col 32 | Spaces: 4 | UTF-8 | CRLF | CSS | Filter (e.g. text, /exclude, \escape).

Figure 55:css png and jpg 4

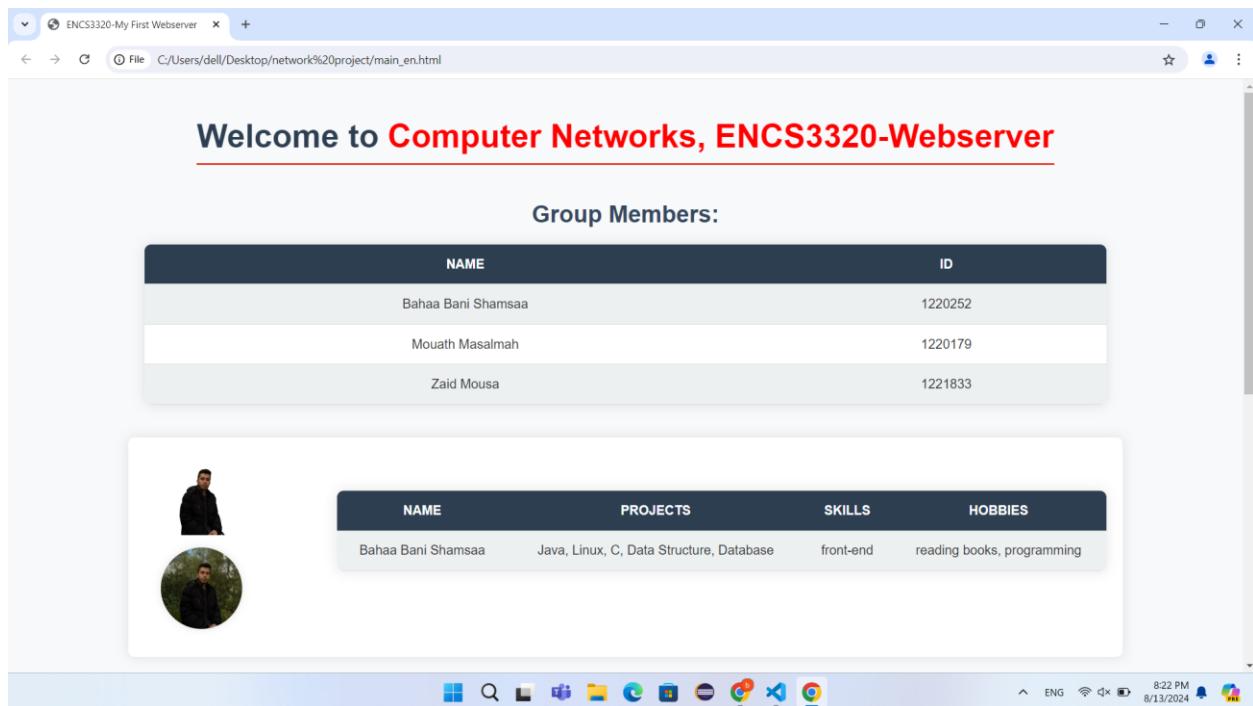


Figure 56:browser for png and jpg 1

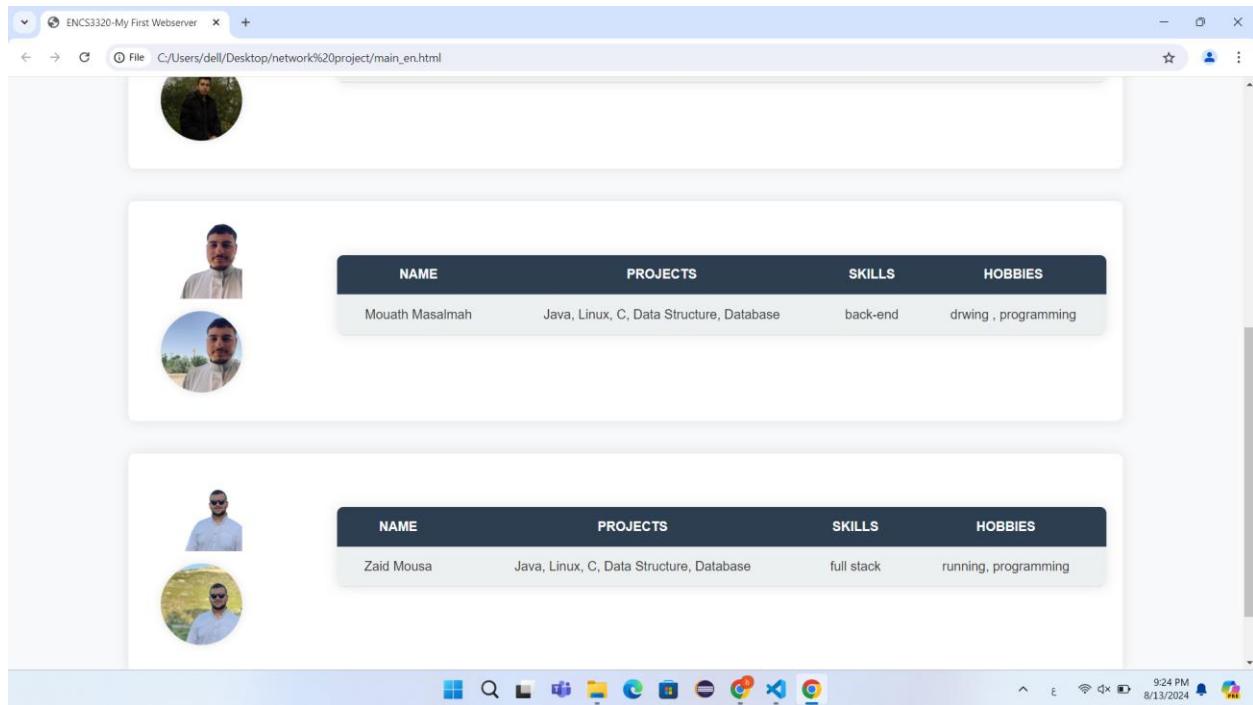
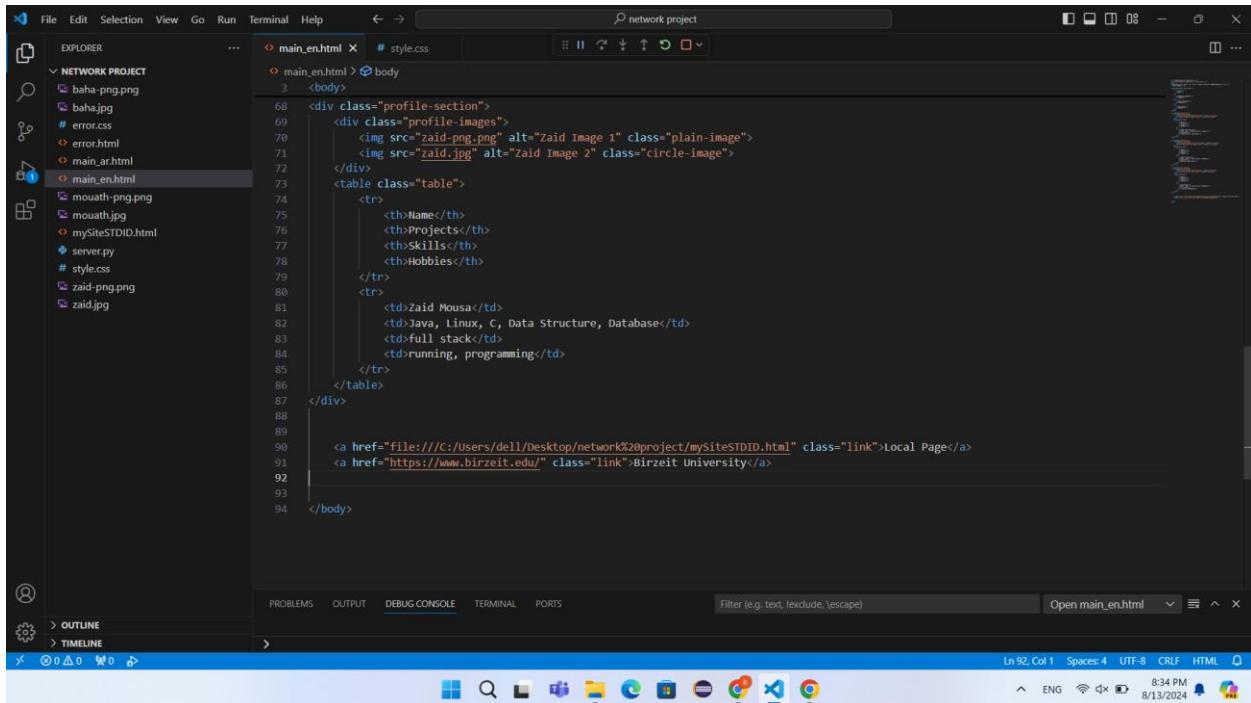


Figure 57:browser for png and jpg 2

As we see here, we put 6 images 3 .jpg and 3 .png, each member we put his images at his table , we use border-radius to make the img circle

i and j:



```
<div class="profile-section">
  <div class="profile-images">
    
    
  </div>
  <table class="table">
    <tr>
      <th>Name</th>
      <th>Projects</th>
      <th>Skills</th>
      <th>Hobbies</th>
    </tr>
    <tr>
      <td>Zaid Mousa</td>
      <td>Java, Linux, C, Data Structure, Database</td>
      <td>full stack</td>
      <td>running, programming</td>
    </tr>
  </table>
</div>

<a href="file:///C:/Users/dell/Desktop/network%20project/mySiteSTDID.html" class="link">Local Page</a>
<a href="https://www.birzeit.edu/" class="link">Birzeit University</a>
```

Figure 58: link to local html and Birzeit web

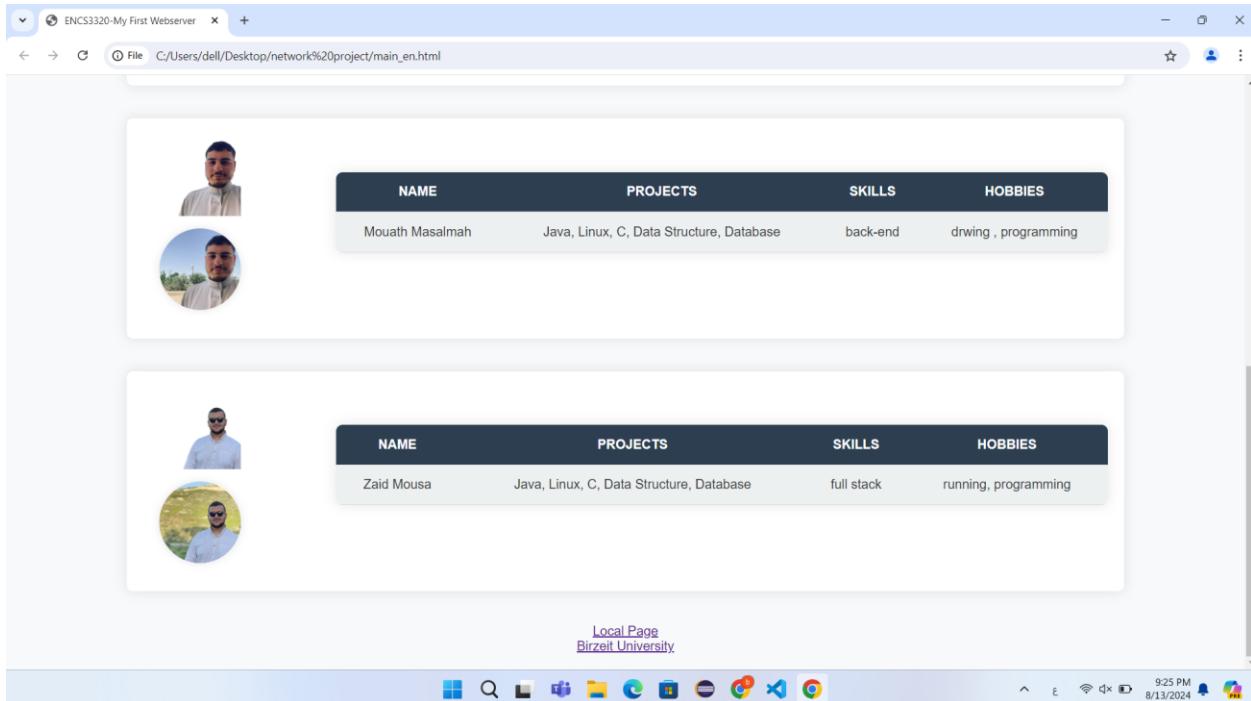


Figure 59: browser for link to local html and Birzeit web

Here we put 2 links one for Birzeit and another one for a local page that it is mySiteSTDID.html

2.

الاسم	الرقم الجامعي
بهاء بنى شمسه	1220252
معاذ مسالمة	1220179
زيد موسى	1221833

الاسم	المشروع	المهارات	الهوايات
بهاء بنى شمسه	جيادا ، لينوكس ، سي ، بنية المعلومات ، قواعد البيانات	الواجهة الامامية	قراءة الكتب ، الترجمة

Profile picture of بهاء بنى شمسه is shown on the left.

Figure 60: /ar request

الاسم	الرقم الجامعي
بهاء بنى شمسه	1220252
معاذ مسالمة	1220179
زيد موسى	1221833

الاسم	المشروع	المهارات	الهوايات
بهاء بنى شمسه	جيادا ، لينوكس ، سي ، بنية المعلومات ، قواعد البيانات	الواجهة الامامية	قراءة الكتب ، الترجمة

Profile picture of بهاء بنى شمسه is shown on the left.

Figure 61: main_ar.html request

As we see here when I search using this link <http://localhost:1833/ar> it shows the Arabic version of main_en.html which is main_ar.html

3.

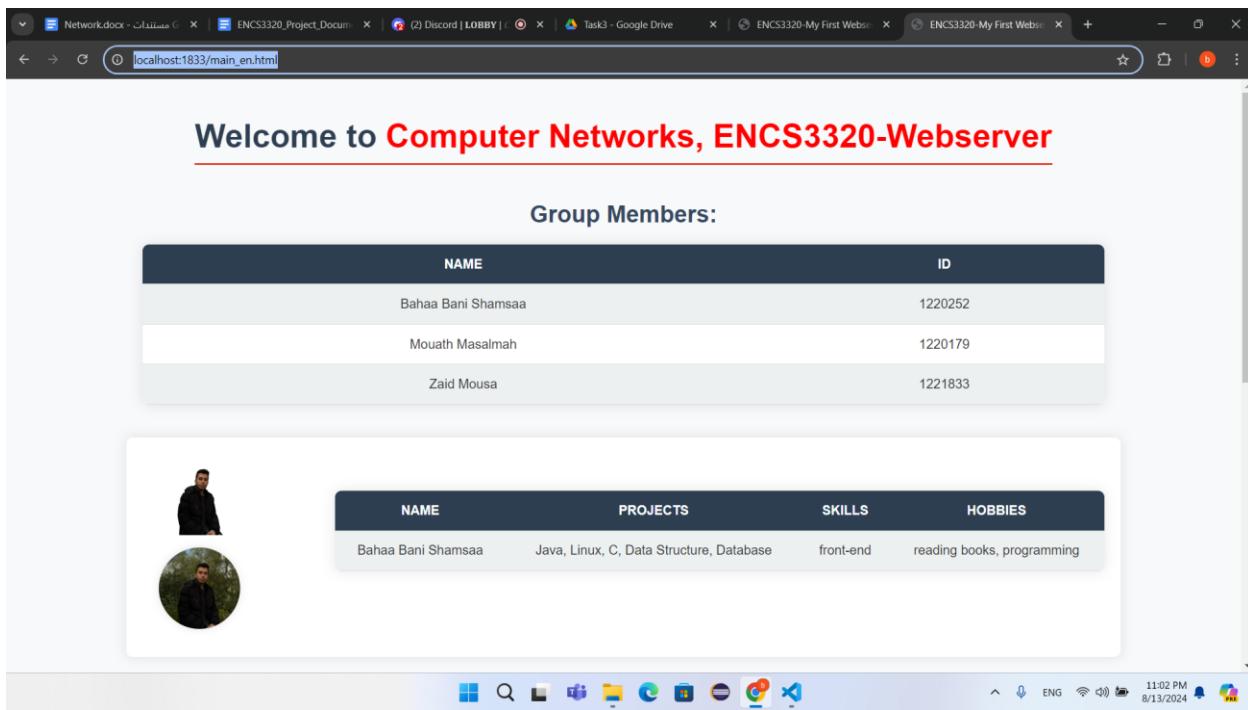
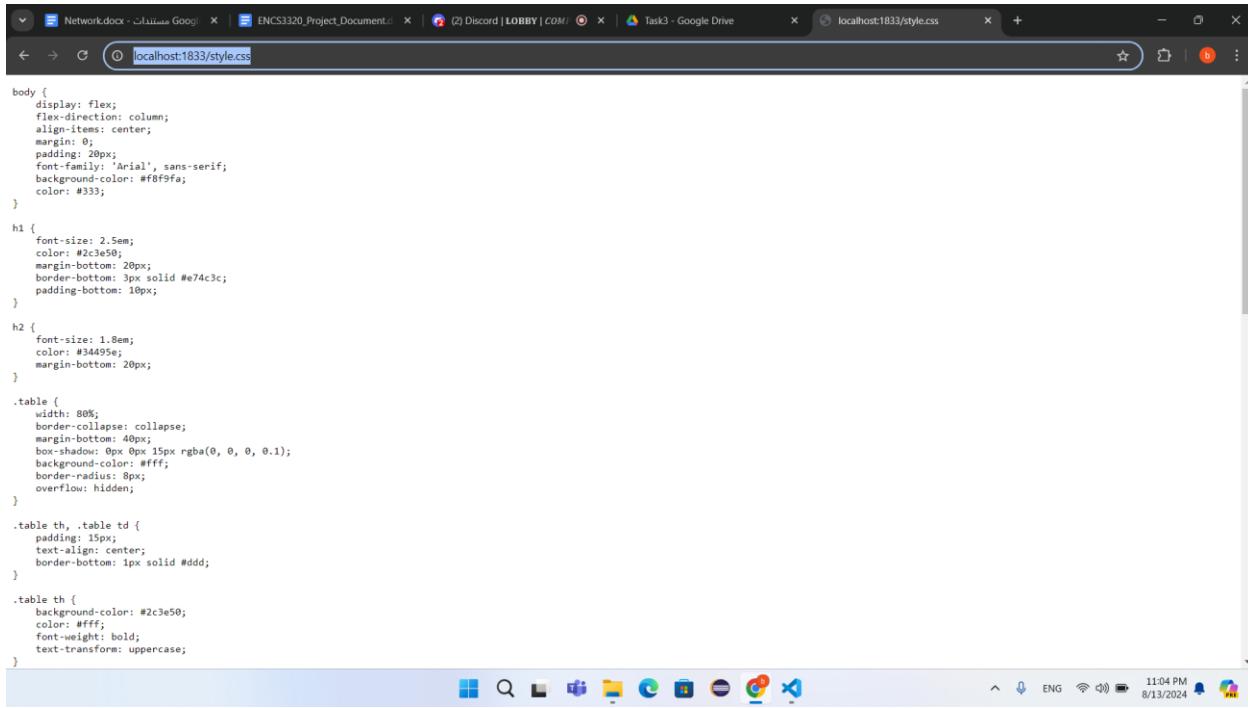


Figure 62: main_en.html request

Here we tried to search for an .html file and we search for main_en.html using the link http://localhost:1833/main_en.html and it shows the contents of main_en.html file

4.



A screenshot of a Microsoft Edge browser window. The address bar shows the URL localhost:1833/style.css. The main content area displays the CSS code for the file. The code includes styles for the body, h1, h2, and .table elements, setting various properties like font-family, background-color, and border-radius. The browser's taskbar at the bottom shows several pinned icons and the system tray on the right indicating the date and time as 8/13/2024 at 11:04 PM.

```
body {
    display: flex;
    flex-direction: column;
    align-items: center;
    margin: 0;
    padding: 20px;
    font-family: 'Arial', sans-serif;
    background-color: #f0f5fa;
    color: #333;
}

h1 {
    font-size: 2.5em;
    color: #2c3e50;
    margin-bottom: 20px;
    border-bottom: 3px solid #e74c3c;
    padding-bottom: 10px;
}

h2 {
    font-size: 1.8em;
    color: #34495e;
    margin-bottom: 20px;
}

.table {
    width: 80%;
    border-collapse: collapse;
    margin-bottom: 40px;
    box-shadow: 0px 0px 15px rgba(0, 0, 0, 0.1);
    background-color: #fff;
    border-radius: 8px;
    overflow: hidden;
}

.table th, .table td {
    padding: 15px;
    text-align: center;
    border-bottom: 1px solid #ddd;
}

.table th {
    background-color: #2c3e50;
    color: #fff;
    font-weight: bold;
    text-transform: uppercase;
}
```

Figure 63: style.css request

Here we tried to search for and .css file and we search for style.css using the link <http://localhost:1833/style.css> and it shows the content of style.css file

5.

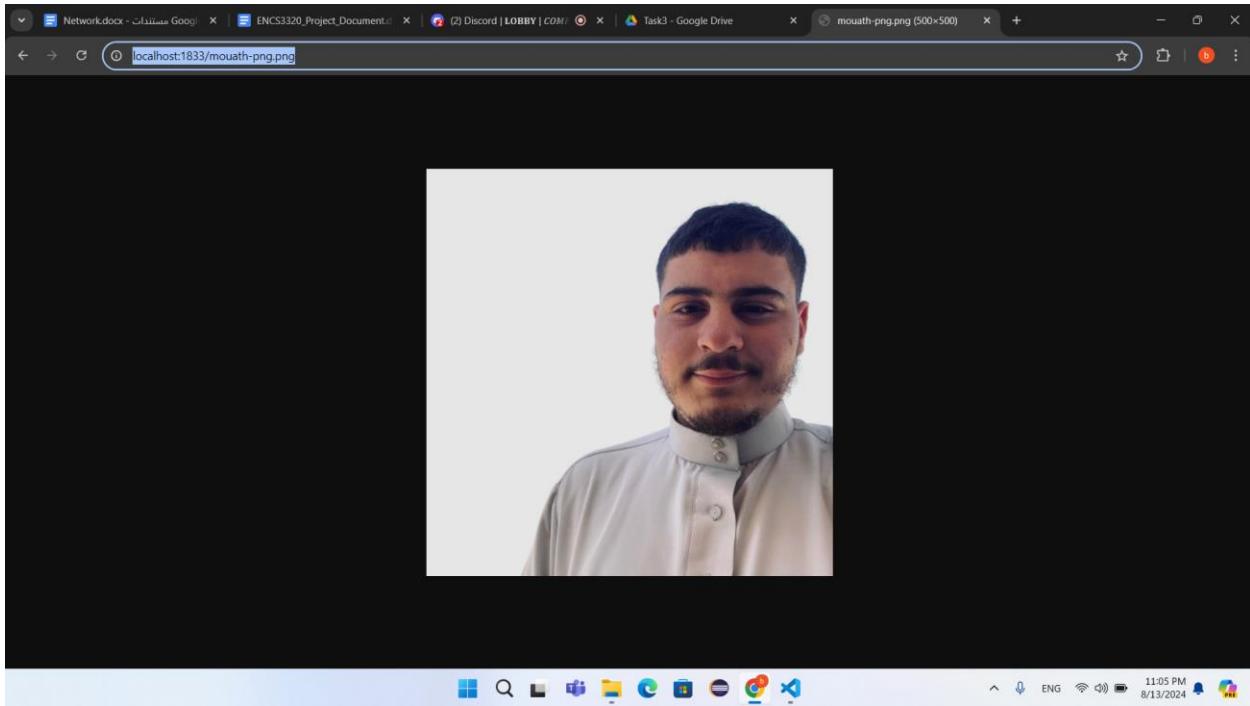


Figure 64: mouauth.png request

Here we tried to search for .png img and we search for mouauth-png.png using the link <http://localhost:1833/mouauth-png.png> and it shows the content of mouauth-png.png

6.

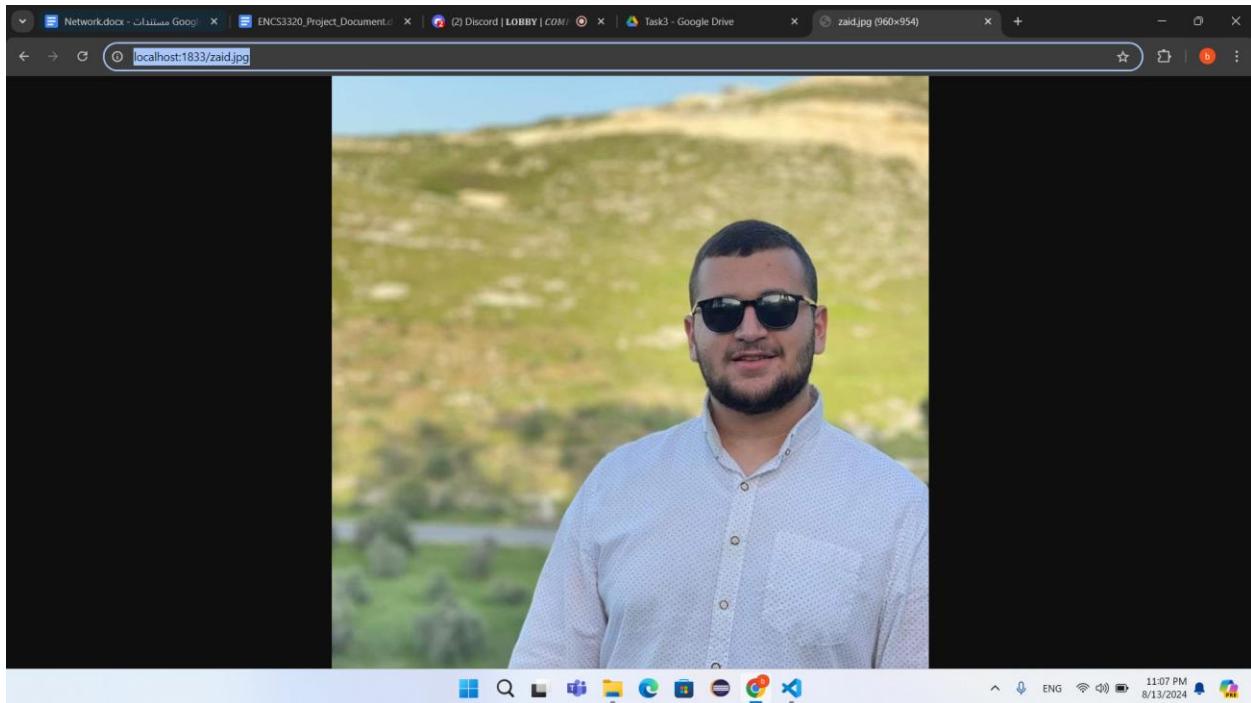


Figure 65: zaid.jpg request

Here we tried to search for .jpg img and we search for zaid.jpg using the link <http://localhost:1833/zaid.jpg> and it shows the content of zaid.jpg

7 and 8:

```

File Edit Selection View Go Run Terminal Help < > network project
EXPLORER NETWORK PROJECT
baha.png.png
baha.jpg
# error.css
error.html
main_ar.html
main_en.html
mouauth.png.png
mouauth.jpg
mySiteSTDID.html
server.py
style.css
zaid.png.png
zaid.jpg
server.py > ...
1  from socket import *
2
3  def open_file(file_name):
4      try:
5          with open(file_name, 'rb') as f:
6              return f.read()
7      except FileNotFoundError:
8          return None
9
10 serverPort = 1833
11 serverSocket = socket(AF_INET, SOCK_STREAM)
12 serverSocket.bind(('', serverPort))
13 serverSocket.listen(1)
14 print('The server is ready to receive')
15
16 while True:
17     connectionSocket, addr = serverSocket.accept()
18     ip = addr[0]
19     port = addr[1]
20     try:
21         sentence = connectionSocket.recv(1024).decode()
22         print('Received: ', sentence)
23         split = sentence.split(" ")
24         s = split[1].split("/")
25         file = s[1]
26
27         if (sentence.startswith("GET / ") or file == "en" or file == "main_en.html" or file == "index.html"):
28             content = open_file('main_en.html')
29             if content:
30                 connectionSocket.send(b'HTTP/1.1 200 OK\nContent-Type: text/html\n\n')
31                 connectionSocket.send(content)
32             else:
33                 raise FileNotFoundError
34
35         elif file == "ar" or file == "main_ar.html":
36             content = open_file('main_ar.html')
37             if content:
38                 connectionSocket.send(b'HTTP/1.1 200 OK\nContent-Type: text/html\n\n')
39                 connectionSocket.send(content)
40             else:
41                 raise FileNotFoundError
42
43         elif file.endswith(".css"):
44             content = open_file(file)
45             if content:
46                 connectionSocket.send(b'HTTP/1.1 200 OK\nContent-Type: text/css\n\n')
47                 connectionSocket.send(content)
48             else:
49                 raise FileNotFoundError
50
51         elif file.endswith(".html"):
52             content = open_file(file)
53             if content:
54                 connectionSocket.send(b'HTTP/1.1 200 OK\nContent-Type: text/html\n\n')
55                 connectionSocket.send(content)
56             else:
57                 raise FileNotFoundError
58
59         elif "get_image?image-in" in file:
60             name = sentence.split(" ")[1].split(" ")[0]
61             content = open_file(name)
62             if content:
63
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Ln 105, Col 33  Spaces: 4  UTF-8  CR/LF  Python 3.9.13 64-bit
+ ... x
Python Deb... Python Deb...
11:52 AM 8/15/2024 ENG WiFi

```

Figure 66: server 1

```

File Edit Selection View Go Run Terminal Help < > network project
EXPLORER NETWORK PROJECT
baha.png.png
baha.jpg
# error.css
error.html
main_ar.html
main_en.html
mouauth.png.png
mouauth.jpg
mySiteSTDID.html
server.py
style.css
zaid.png.png
zaid.jpg
server.py > ...
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
connectionSocket.send(b'HTTP/1.1 200 OK\nContent-Type: text/html\n\n')
connectionSocket.send(content)
else:
    raise FileNotFoundError

elif file == "ar" or file == "main_ar.html":
    content = open_file('main_ar.html')
    if content:
        connectionSocket.send(b'HTTP/1.1 200 OK\nContent-Type: text/html\n\n')
        connectionSocket.send(content)
    else:
        raise FileNotFoundError

elif file.endswith(".css"):
    content = open_file(file)
    if content:
        connectionSocket.send(b'HTTP/1.1 200 OK\nContent-Type: text/css\n\n')
        connectionSocket.send(content)
    else:
        raise FileNotFoundError

elif file.endswith(".html"):
    content = open_file(file)
    if content:
        connectionSocket.send(b'HTTP/1.1 200 OK\nContent-Type: text/html\n\n')
        connectionSocket.send(content)
    else:
        raise FileNotFoundError

elif "get_image?image-in" in file:
    name = sentence.split(" ")[1].split(" ")[0]
    content = open_file(name)
    if content:
        with open(name, 'wb') as f:
            f.write(content)
        connectionSocket.send(b'HTTP/1.1 200 OK\nContent-Type: image/jpeg\n\n')
        connectionSocket.send(content)
    else:
        raise FileNotFoundError

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Ln 105, Col 33  Spaces: 4  UTF-8  CR/LF  Python 3.9.13 64-bit
+ ... x
Python Deb... Python Deb...
11:53 AM 8/15/2024 ENG WiFi

```

Figure 67: server 2

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows a "NETWORK PROJECT" folder containing files: baha-png.png, baha.jpg, # error.css, error.html, main_ar.html, main_en.html, mouauth-png.png, mouauth.jpg, mySiteSTDID.html, server.py, # style.css, zaid-png.png, and zaid.jpg.
- Code Editor:** The active file is "server.py". The code handles file requests based on their extensions (.jpg, .png, .html) and specific file names like "so", "itc", and "ite". It uses Python's socket module to send HTTP responses back to the client.
- Terminal:** Shows the command "network project".
- Status Bar:** Displays "Ln 105, Col 33" and "Python 3.9.13 64-bit".

```

56         else:
57             raise FileNotFoundError
58
59     elif "get_image?image-in" in file:
60         name = sentence.split("=")[1].split(" ")[0]
61         content = open_file(name)
62
63         if content:
64             connectionSocket.send(b'HTTP/1.1 200 OK\r\nContent-Type: image/jpg\r\n\r\n')
65         elif name.endswith('.png'):
66             connectionSocket.send(b'HTTP/1.1 200 OK\r\nContent-Type: image/png\r\n\r\n')
67         else:
68             raise FileNotFoundError
69
70     elif file.endswith(".jpg"):
71         content = open_file(file)
72
73         if content:
74             connectionSocket.send(b'HTTP/1.1 200 OK\r\nContent-Type: image/jpg\r\n\r\n')
75         else:
76             raise FileNotFoundError
77
78     elif file.endswith(".png"):
79         content = open_file(file)
80
81         if content:
82             connectionSocket.send(b'HTTP/1.1 200 OK\r\nContent-Type: image/png\r\n\r\n')
83         else:
84             raise FileNotFoundError
85
86     elif file == "so":
87         connectionSocket.send(b'HTTP/1.1 307 Temporary Redirect\r\nContent-Type: text/html; charset=utf-8\r\nLocation: https://stacko')
88
89
90     elif file == "itc":
91         connectionSocket.send(b'HTTP/1.1 307 Temporary Redirect\r\nContent-Type: text/html; charset=utf-8\r\nLocation: https://itc.bi')
92
93     else:
94         raise FileNotFoundError
95
96
97     except Exception as e:
98         print(f"Error: {e}")
99         connectionSocket.send(b'HTTP/1.1 404 Not Found\r\nContent-Type: text/html\r\n\r\n')
100        error_content = open_file('error.html')
101        if error_content:
102            connectionSocket.send(error_content + f"<p> ip: {ip} port: {port}</p>".encode())
103        else:
104            connectionSocket.send(b'<html><body><h1>404 Not Found</h1><p>File not found.</p></body></html>')
105
106 finally:
107     connectionSocket.close()

```

Figure 68: server 3

The screenshot shows the Visual Studio Code interface with the following details:

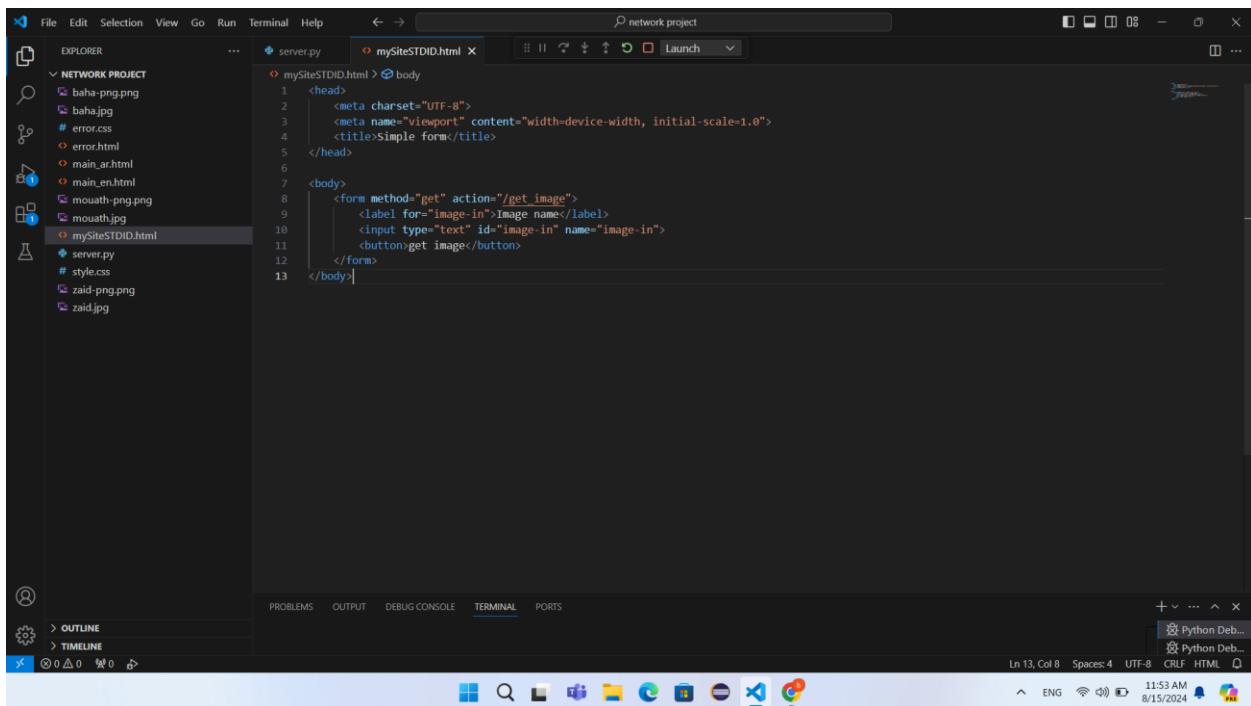
- File Explorer:** Shows a "NETWORK PROJECT" folder containing files: baha-png.png, baha.jpg, # error.css, error.html, main_ar.html, main_en.html, mouauth-png.png, mouauth.jpg, mySiteSTDID.html, server.py, # style.css, zaid-png.png, and zaid.jpg.
- Code Editor:** The active file is "server.py". This version includes exception handling for errors and provides more detailed error messages to the client.
- Terminal:** Shows the command "network project".
- Status Bar:** Displays "Ln 105, Col 33" and "Python 3.9.13 64-bit".

```

80         content = open_file(file)
81
82         if content:
83             connectionSocket.send(b'HTTP/1.1 200 OK\r\nContent-Type: image/png\r\n\r\n')
84         else:
85             raise FileNotFoundError
86
87     elif file == "so":
88         connectionSocket.send(b'HTTP/1.1 307 Temporary Redirect\r\nContent-Type: text/html; charset=utf-8\r\nLocation: https://stacko')
89
90     elif file == "itc":
91         connectionSocket.send(b'HTTP/1.1 307 Temporary Redirect\r\nContent-Type: text/html; charset=utf-8\r\nLocation: https://itc.bi')
92
93     else:
94         raise FileNotFoundError
95
96
97     except Exception as e:
98         print(f"Error: {e}")
99         connectionSocket.send(b'HTTP/1.1 404 Not Found\r\nContent-Type: text/html\r\n\r\n')
100        error_content = open_file('error.html')
101        if error_content:
102            connectionSocket.send(error_content + f"<p> ip: {ip} port: {port}</p>".encode())
103        else:
104            connectionSocket.send(b'<html><body><h1>404 Not Found</h1><p>File not found.</p></body></html>')
105
106 finally:
107     connectionSocket.close()

```

Figure 69: server 4



```
mySiteSTID.html > body
1  <head>
2    <meta charset="UTF-8">
3    <meta name="viewport" content="width=device-width, initial-scale=1.0">
4    <title>simple form</title>
5  </head>
6
7  <body>
8    <form method="get" action="/get_image">
9      <label for="image-in">Image name</label>
10     <input type="text" id="image-in" name="image-in">
11     <button>get image</button>
12   </form>
13 </body>
```

Figure 70: mySiteSTID.html

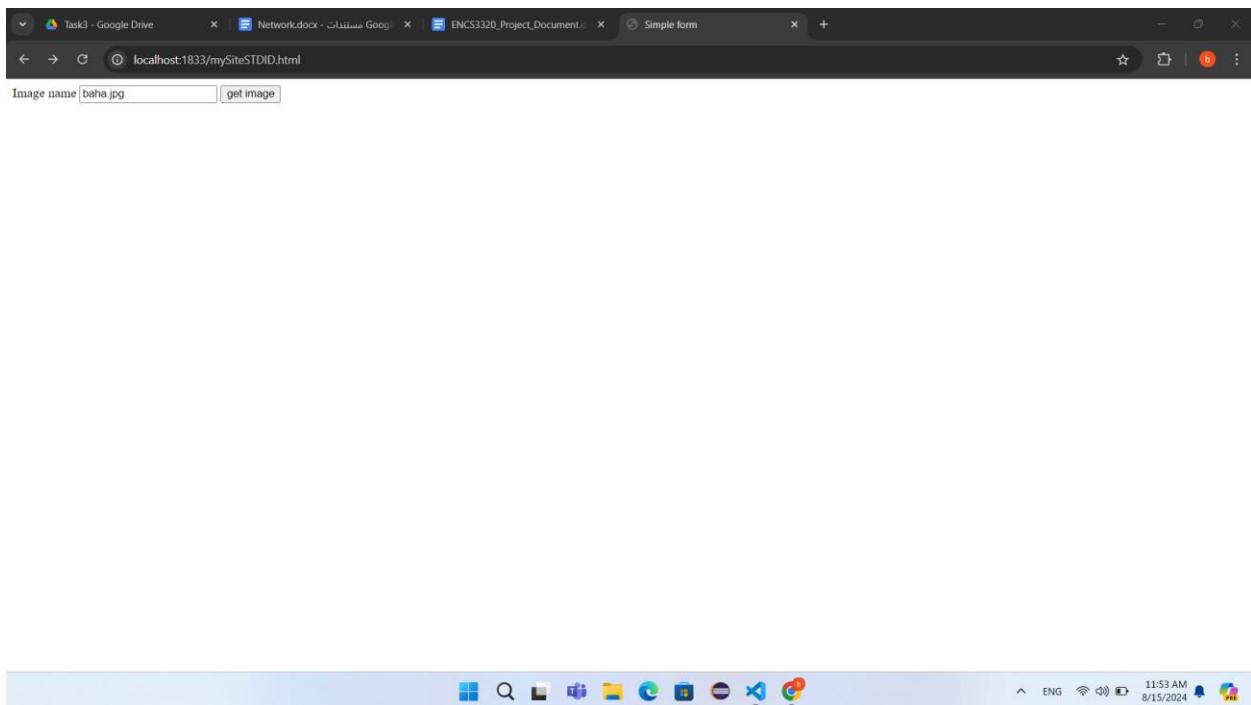


Figure 71: browsr for mySiteSTID.html

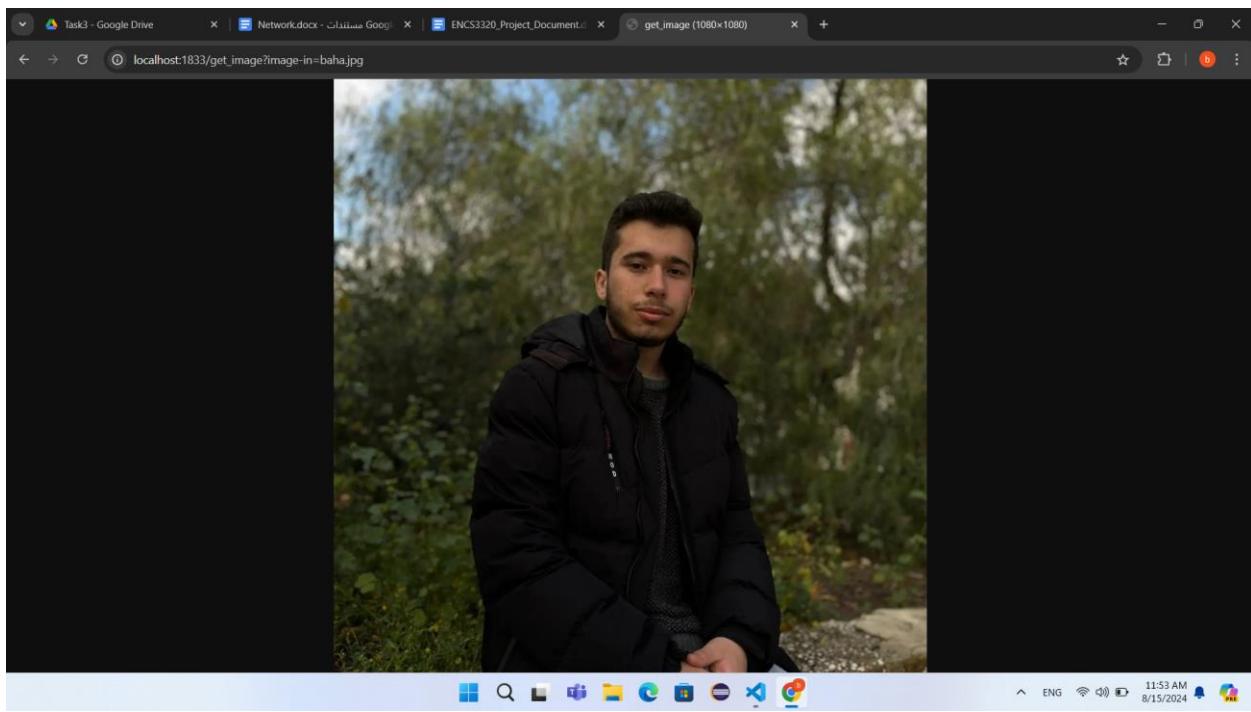


Figure 72: image browser for mySiteSTDID.html

As we see in images we use zaid ID which is 1221833 and we use last 4 numbers 1833 as a port number then we create socket then we set it up , then we see if the sentence we have start with / or en or main_en.html or index.html then open file main_en.html, else if sentence start with ar or main_ar.html so open main_ar.html file , else if sentence end with .css so it open the file requested .css , else if sentence end with .html it open the file requested .html , else if sentence contains get_image?image_in this asking for an specific image , else if file ends with .jpg search for that file , else if file end with .png search for that file

9.

a.

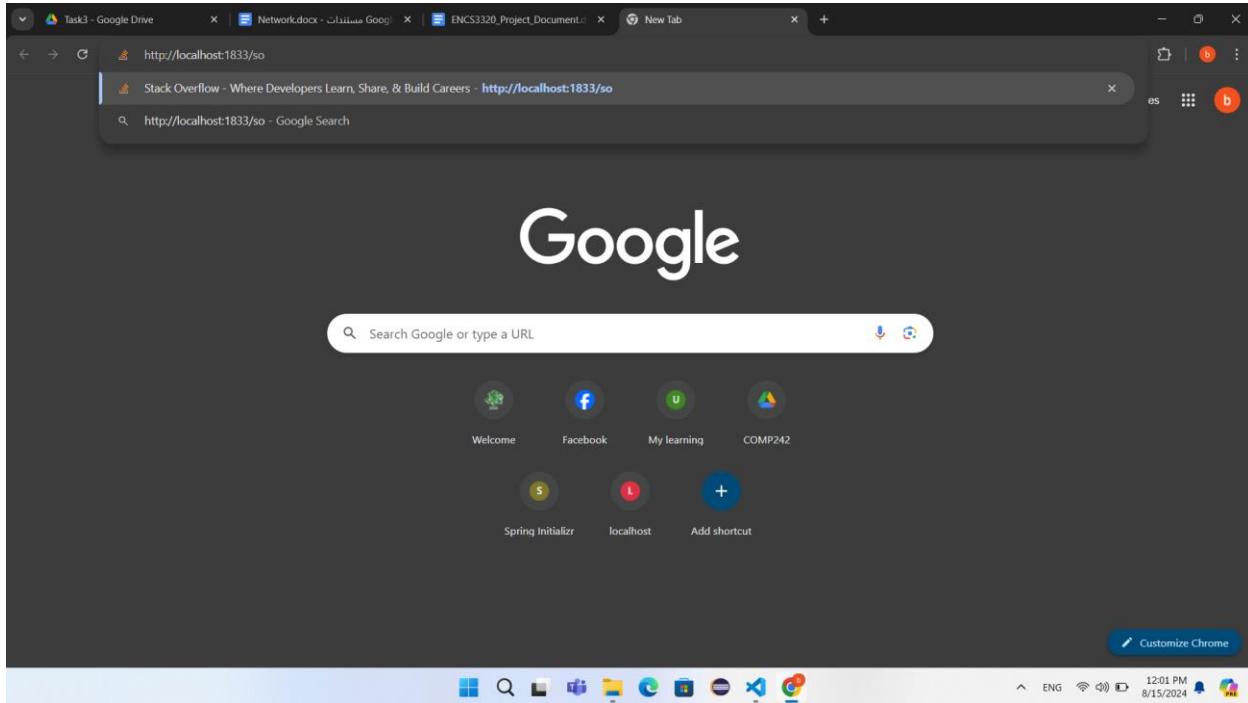


Figure 73: /so request

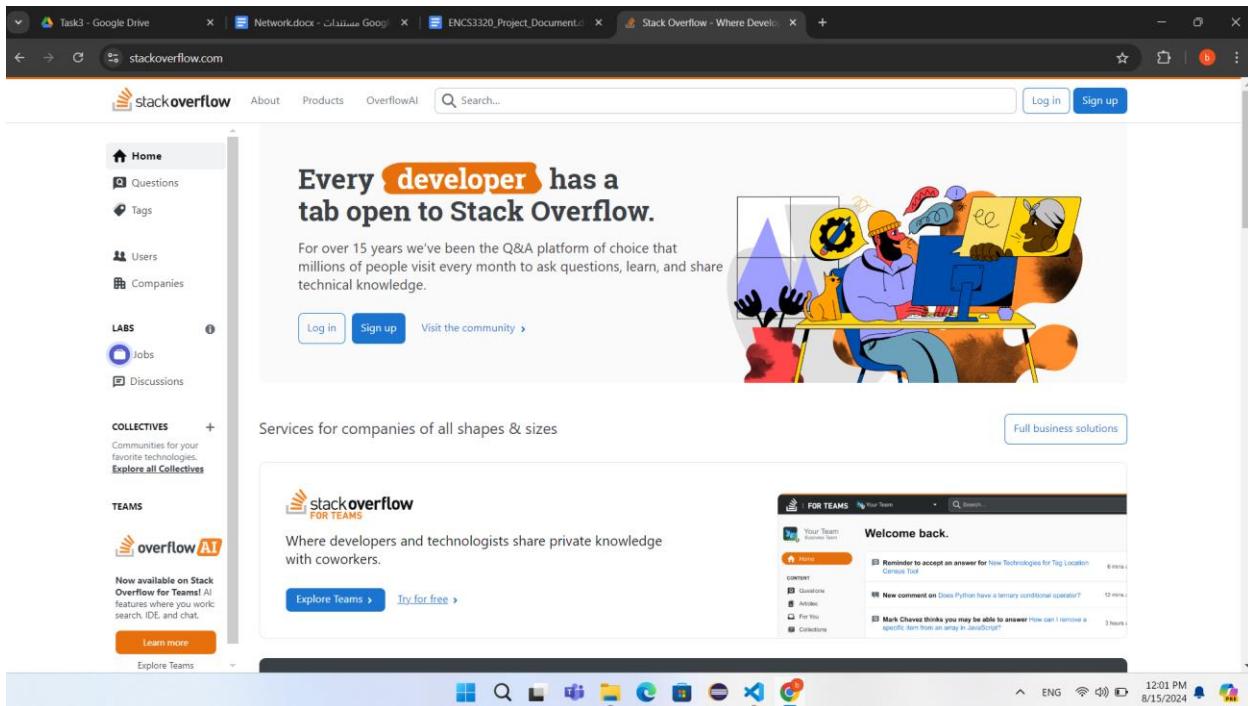


Figure 74: solution for /so

Here we search for <http://localhost:1833/so> and it goes to stackoverflow.com

b.

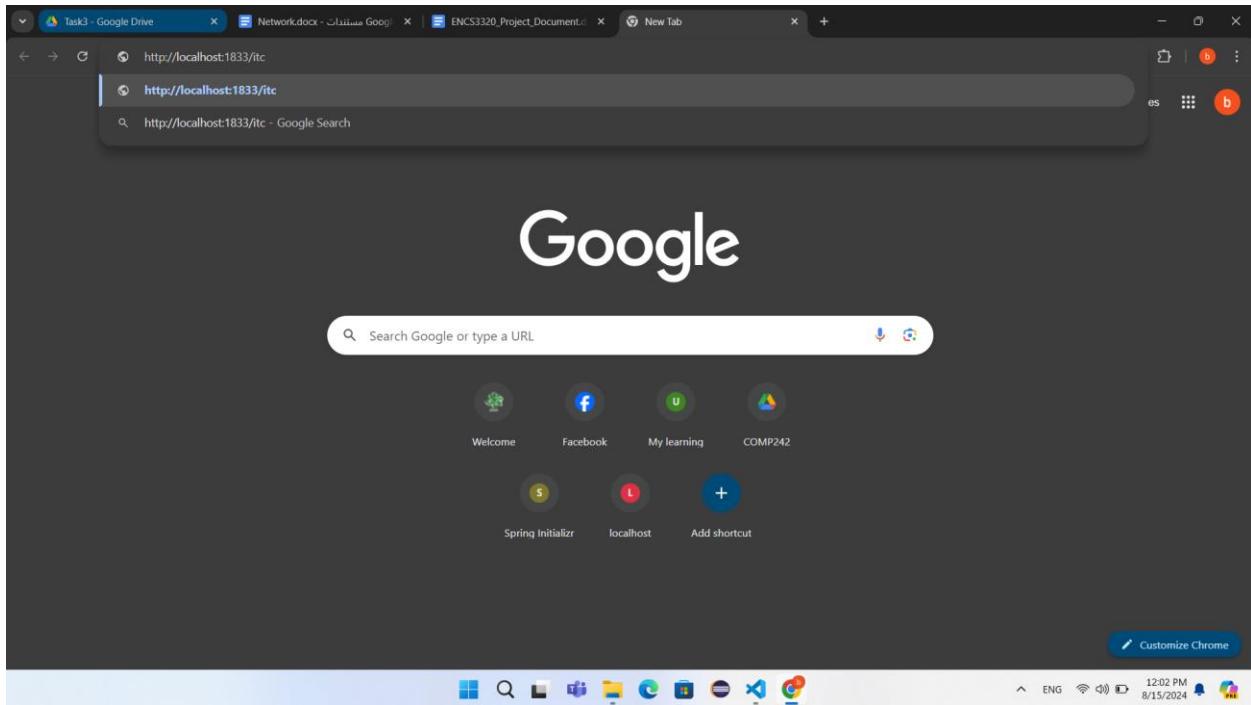


Figure 75: /itc request

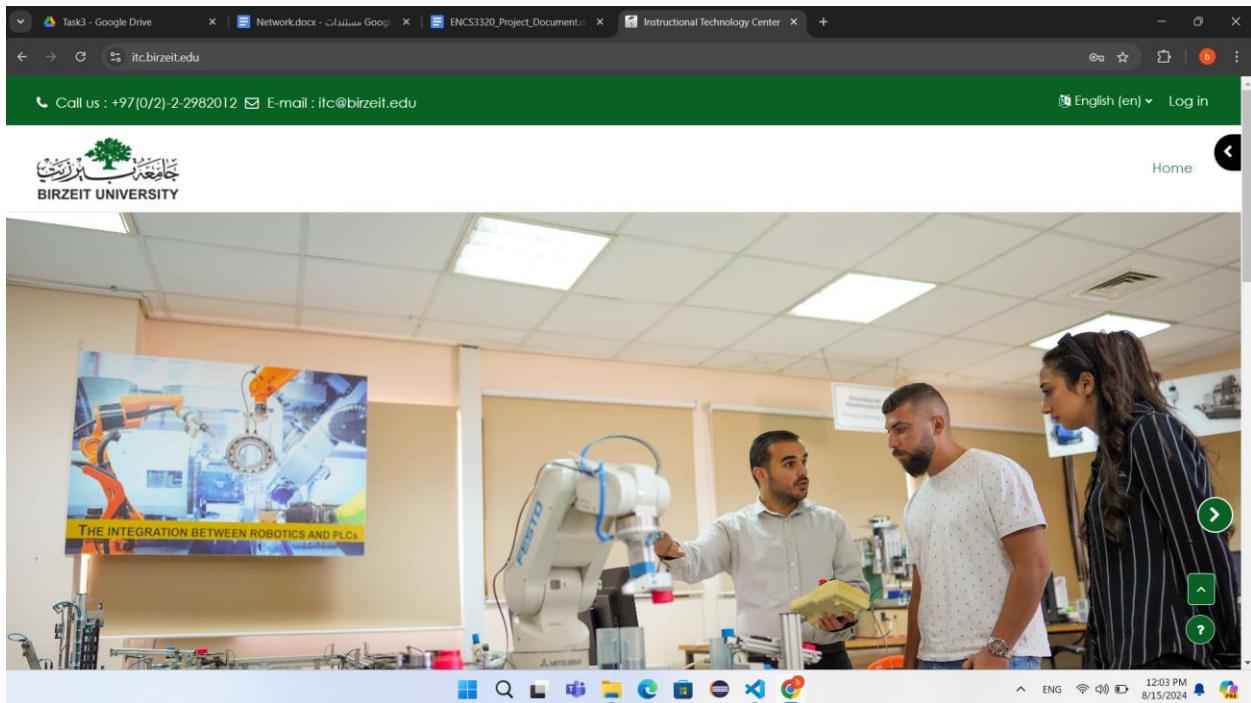


Figure 76: solution for /itc

Here we search for <http://localhost:1833/itc> and it goes to itc.birzeit.edu

10.

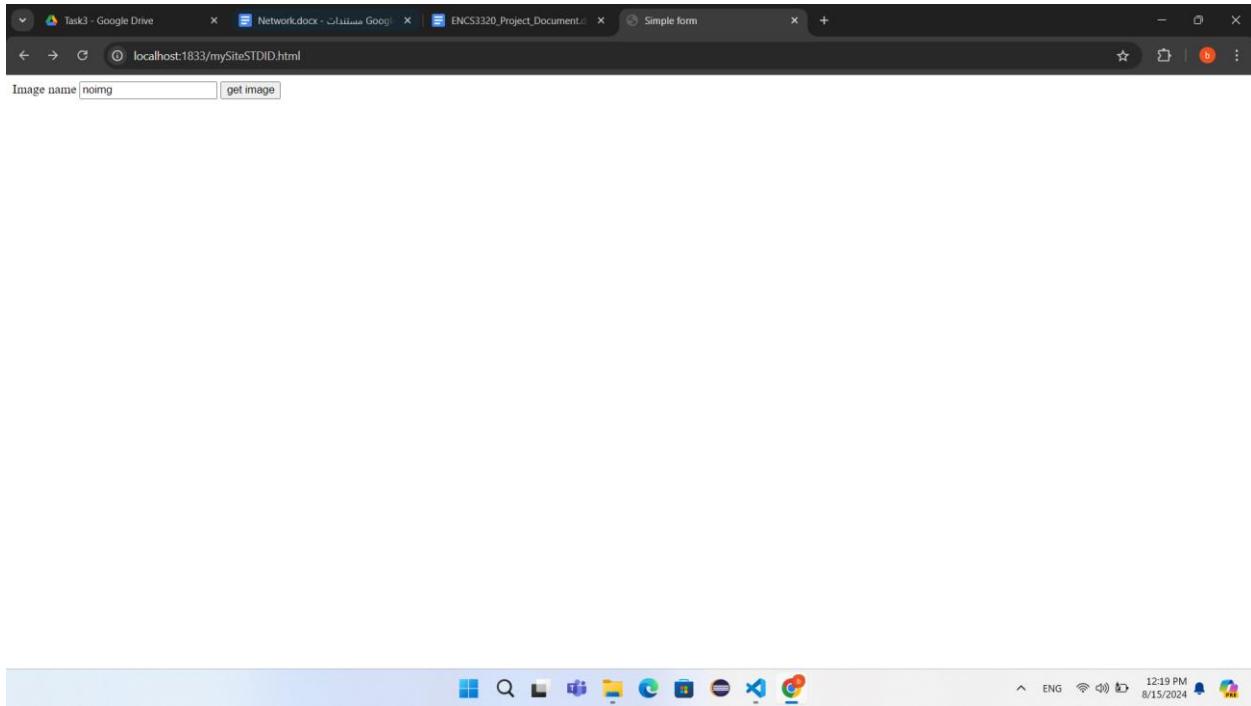


Figure 77: wrong image

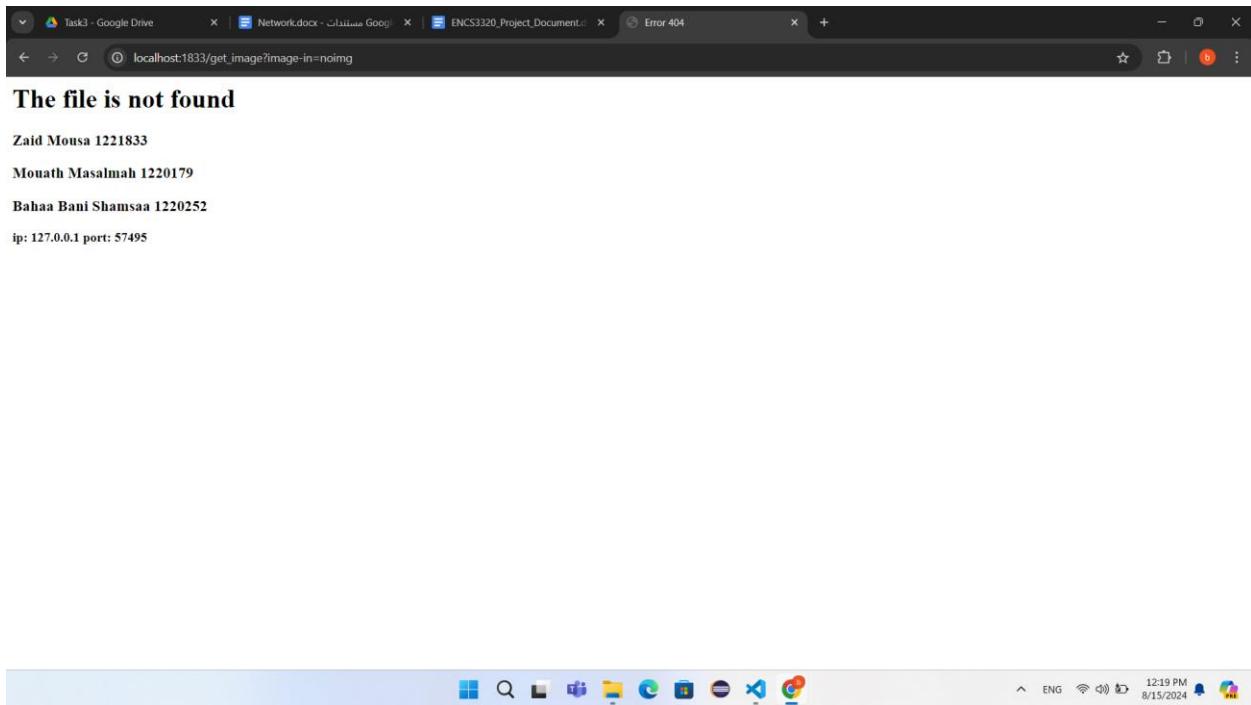


Figure 78: error.html browser

Here when we search for an image does not exist the error.html page shows

11.

The screenshot shows a terminal window within a code editor interface. The terminal tab is active, displaying the following command and its output:

```
Received: GET /get_image?image-in-mouauth.jpg HTTP/1.1
Host: localhost:1833
Connection: keep-alive
sec-ch-ua: "Not A Brand";v="99", "Google Chrome";v="127", "Chromium";v="127"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Windows"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/127.0.0.0 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: http://localhost:1833/mySiteSTID.html
Accept-Encoding: gzip, deflate, br, zstd
Accept-Language: en-US,en;q=0.9
Cookie: Idea-2a3d4e=d2af89cd-da14-4c9c-96ac-2e645a6ab28b
```

Below the main output, there is a detailed stack trace:

```
Received:
Error: list index out of range
Traceback (most recent call last):
  File "c:\Users\dell\Desktop\Network Project\server.py", line 101, in <module>
    s = split[1].split("/")
IndexError: list index out of range
```

At the bottom of the terminal window, the operating system's taskbar is visible, showing various icons and system status.

Figure 79: http request on terminal

Here first it is asking the server for a file named mouauth.jpg then the request sent to the server then it tells the server what browser and operating system are used

Problems and challenges:

We had a problem with the arabic page when we search for it , it shows like that:

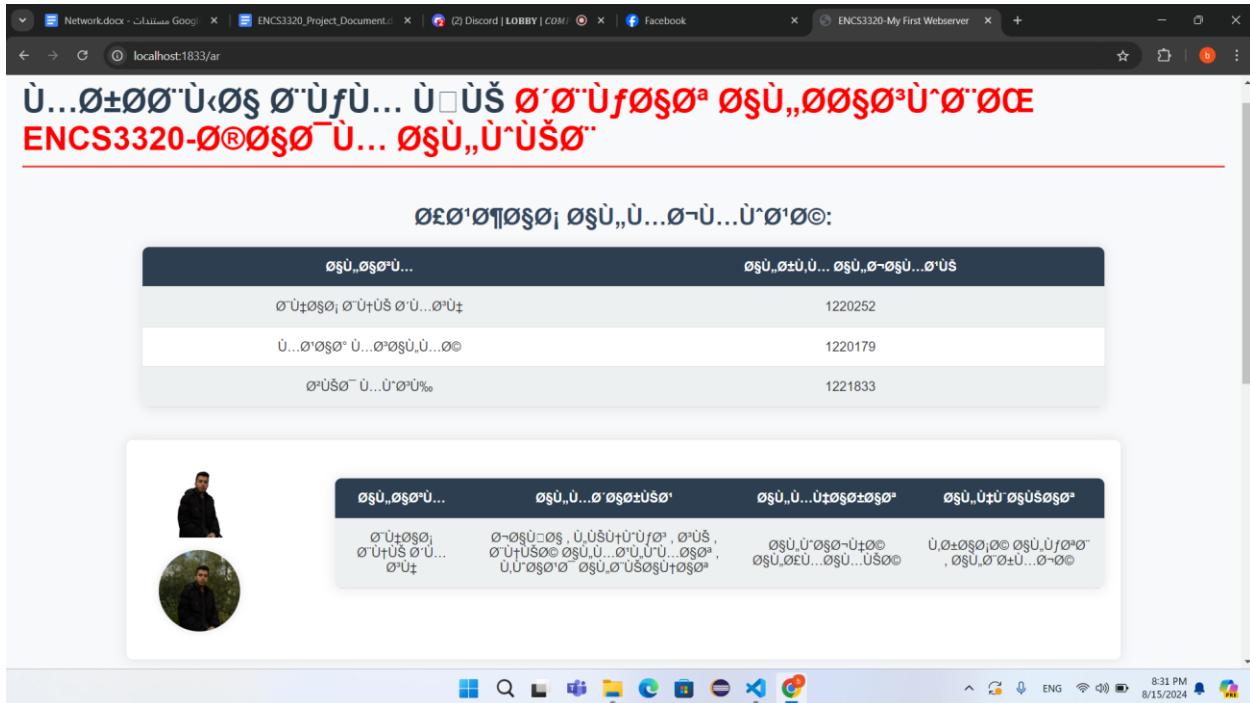
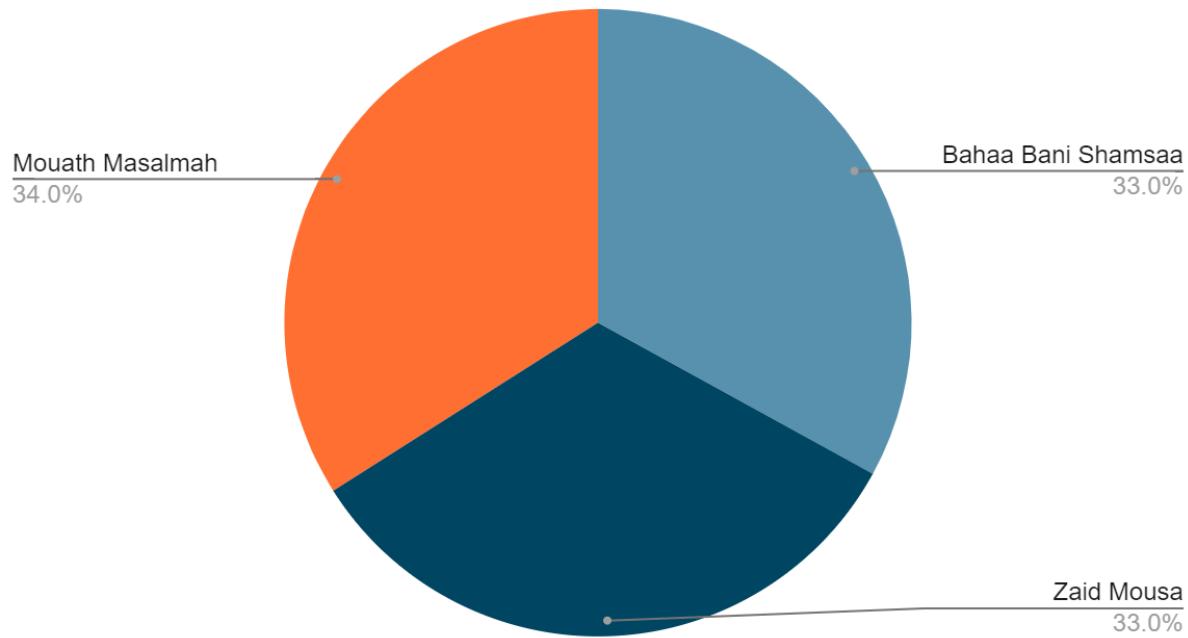


Figure 79: problem on /ar

and we solve it by adding `<meta charset="UTF-8">`

Work Done:

Work Done



Code and report:

Task1:

- 1.Mouath
- 2.A.Bahaa
- B.Bahaa
- C.Bahaa
- D.Mouath
- E.Zaid
- 3.Mouath
- 4.Zaid

Task 2:

1. Zaid , Mouath and Bahaa
- 2.Zaid and Mouath

Task 3:

1.Bahaa and Zaid

A.Bahaa

B.Bahaa

C.Bahaa

D.Bahaa

E.Bahaa

F.Bahaa

H.Bahaa

I.Bahaa

J.Bahaa

2.Mouath

3.Mouath

4.Zaid

5.Zaid

6.Bahaa

7.Bahaa

8.Mouath

9.A.Zaid

B.Zaid

10.Mouath

11.Mouath

References

DNS : <https://www.cloudflare.com/learning/dns/what-is-dns/>

Round-trip time (RTT): <https://www.cloudflare.com/learning/cdn/glossary/round-trip-time-rtt/>

Hop (networking): [https://en.wikipedia.org/wiki/Hop_\(networking\)](https://en.wikipedia.org/wiki/Hop_(networking))

"Bad Request" error: <https://www.semrush.com/blog/400-bad-request/>

vowels: <https://www.grammarly.com/blog/vowels/>

port: <https://www.cloudflare.com/learning/network-layer/what-is-a-computer-port/>

html, css & python: <https://www.w3schools.com/>

wireshark:<https://www.comptia.org/content/articles/what-is-wireshark-and-how-to-use-it#:~:text=Wireshark%20is%20a%20network%20protocol,packet%20sniffer%20in%20the%20world.>