



**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

<b>Theory:</b> .....	5
<b>Task 1: Commands &amp; Wireshark.</b> .....	5
1. Ping:.....	5
2. Tracert:.....	5
3. Nslookup:.....	5
4. Telnet:.....	5
<b>Task 2: Socket Programming (TCP and UDP).</b> .....	5
11. TCP Client-Server Application:.....	5
12. UDP Client-Server Application:.....	5
<b>Task 3: Web Server.</b> .....	6
15. Web Server Application:.....	6
<b>Procedure:</b> .....	7
<b>Task 1: Commands &amp; Wireshark.</b> .....	7
5. Ping a device in the same network:.....	7
6. Ping www.ox.ac.uk:.....	7
7. Tracert www.ox.ac.uk:.....	7
8. Nslookup www.ox.ac.uk:.....	7
9. Telnet www.ox.ac.uk:.....	7
10. Wireshark DNS Capture:.....	7
<b>Task 2: Socket Programming (TCP and UDP).</b> .....	7
13. TCP Client-Server:.....	7
14. UDP Client-Server:.....	8
<b>Task 3: Web Server.</b> .....	8
16. Web Server Implementation:.....	8
17. HTML and CSS:.....	8
18. Testing and Documentation:.....	8
<b>Result&amp;Discussion.</b> .....	9
<b>Task1: Commands &amp; Wireshark.</b> .....	9
Part 1: In your words, what are ping, tracert, nslookup, and telnet?.....	9
Part 2: Make sure that your computer is connected to the internet and then run the following commands.....	10
Part 3: Give some details about autonomous system (AS) number, number of IPs, prefixes, peers, name of Tier1-ISP of www.ox.ac.uk.....	16
<b>Task 2:.</b> .....	21
Part 1:.....	21
Part 2:.....	27
<b>Task 3:.</b> .....	33
1-.....	33
2.....	50
3.....	51

4.....	52
5.....	53
6.....	54
7 and 8:.....	55
9.....	60
10.....	62
11.....	63
<b>Problems and challenges:.....</b>	<b>64</b>
Work Done:.....	66
References.....	67



# 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.

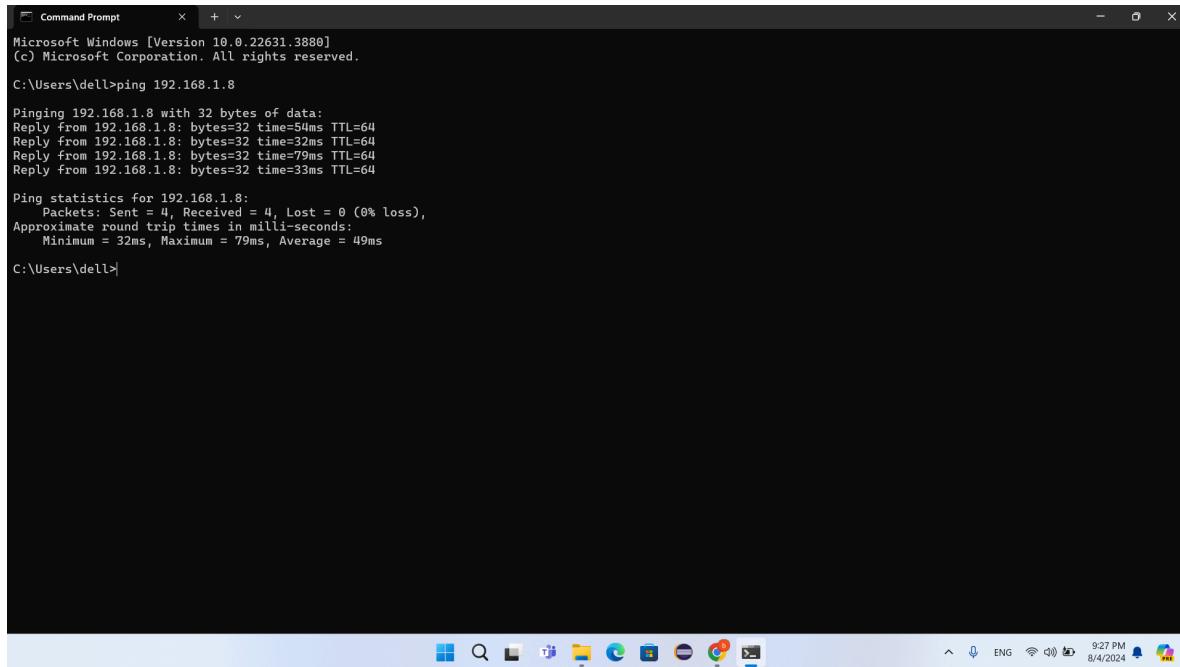
**Tracert:** It is a network tool that displays the path that data takes to reach a destination by showing all the routers it passes through. It is useful for seeing the path and checking for any network delays along the way.

**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.



A screenshot of a Windows Command Prompt window titled "Command Prompt". The window shows the following text output:

```
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

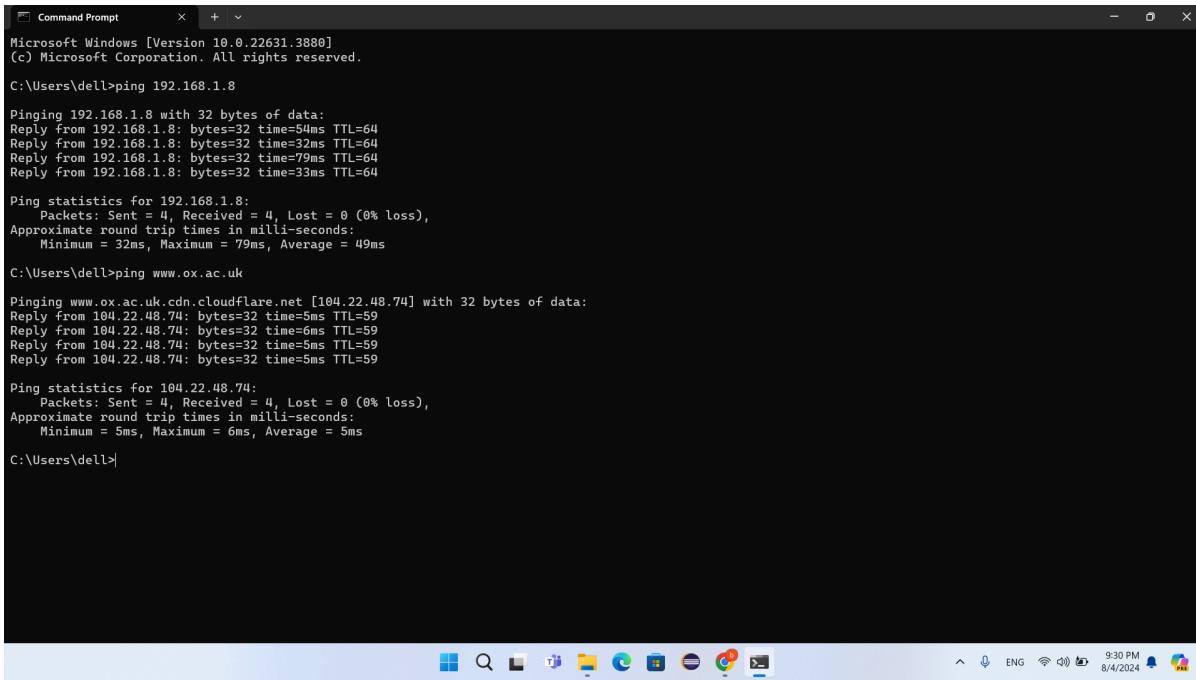
C:\Users\dell>
```

The window has a standard Windows title bar and taskbar at the bottom. The taskbar includes icons for File Explorer, Edge, and other system tools.

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



```
Microsoft Windows [Version 10.0.22631.3886]
(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](http://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.

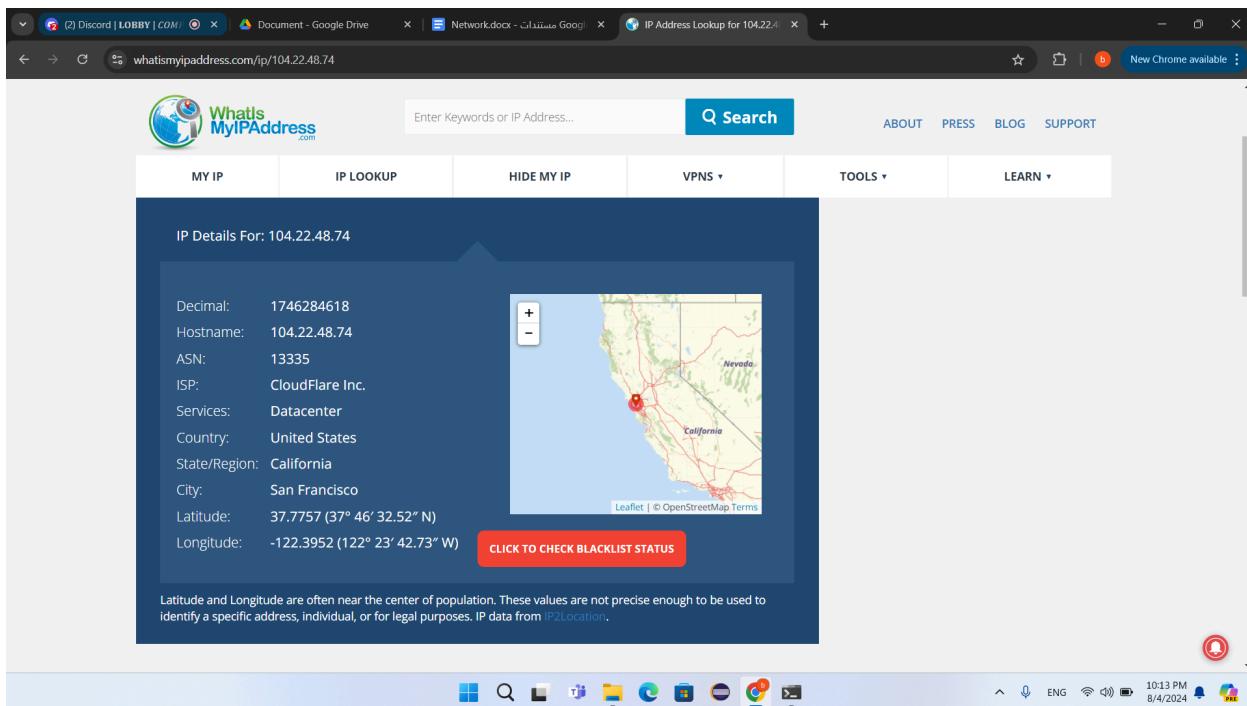
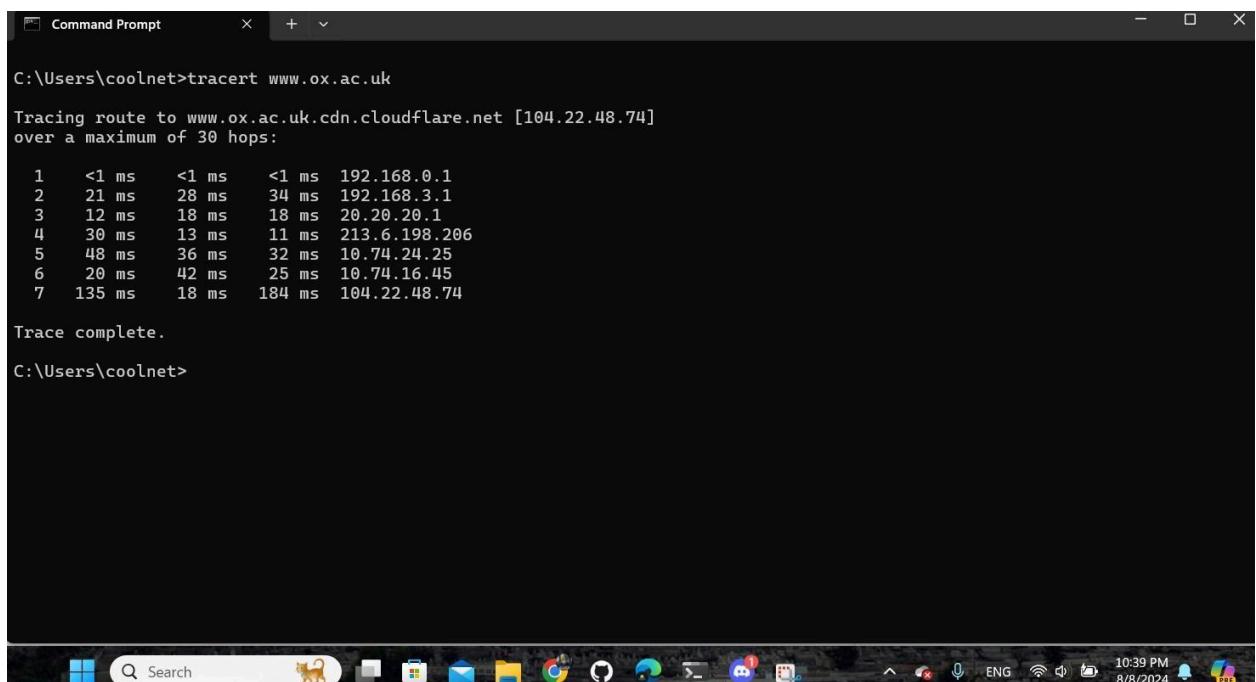


Figure 3:location server response

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 <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

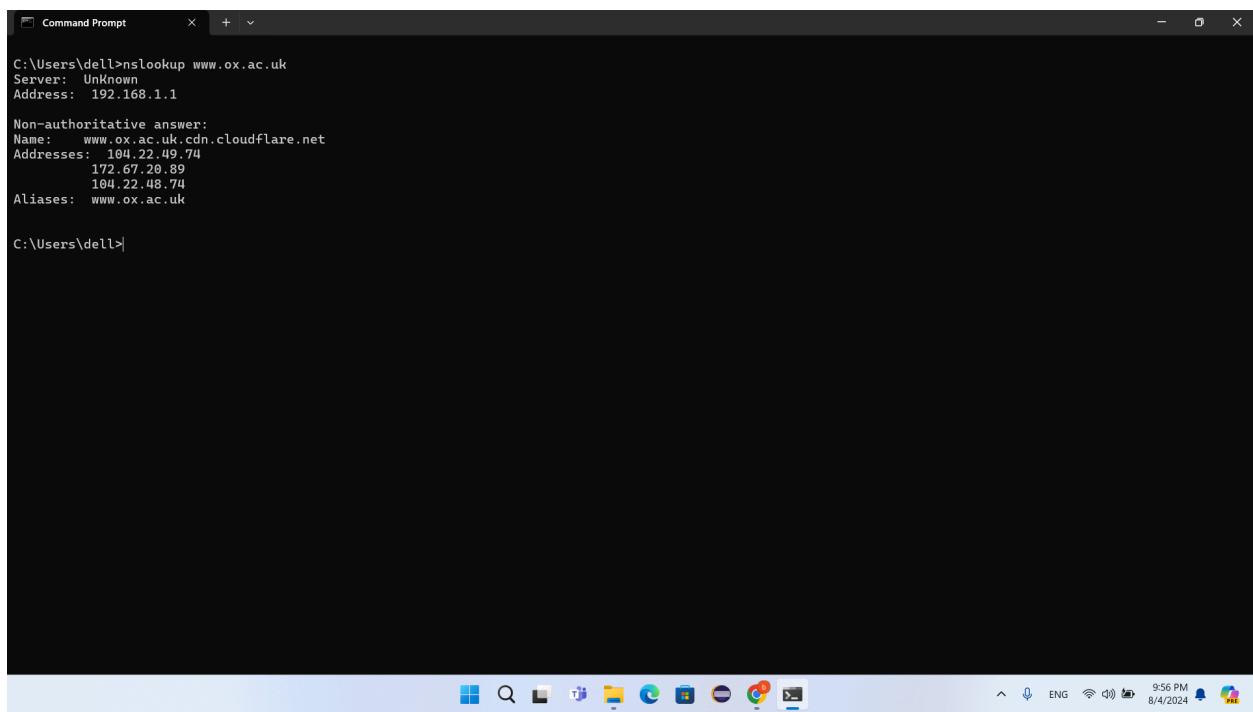
Trace complete.

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

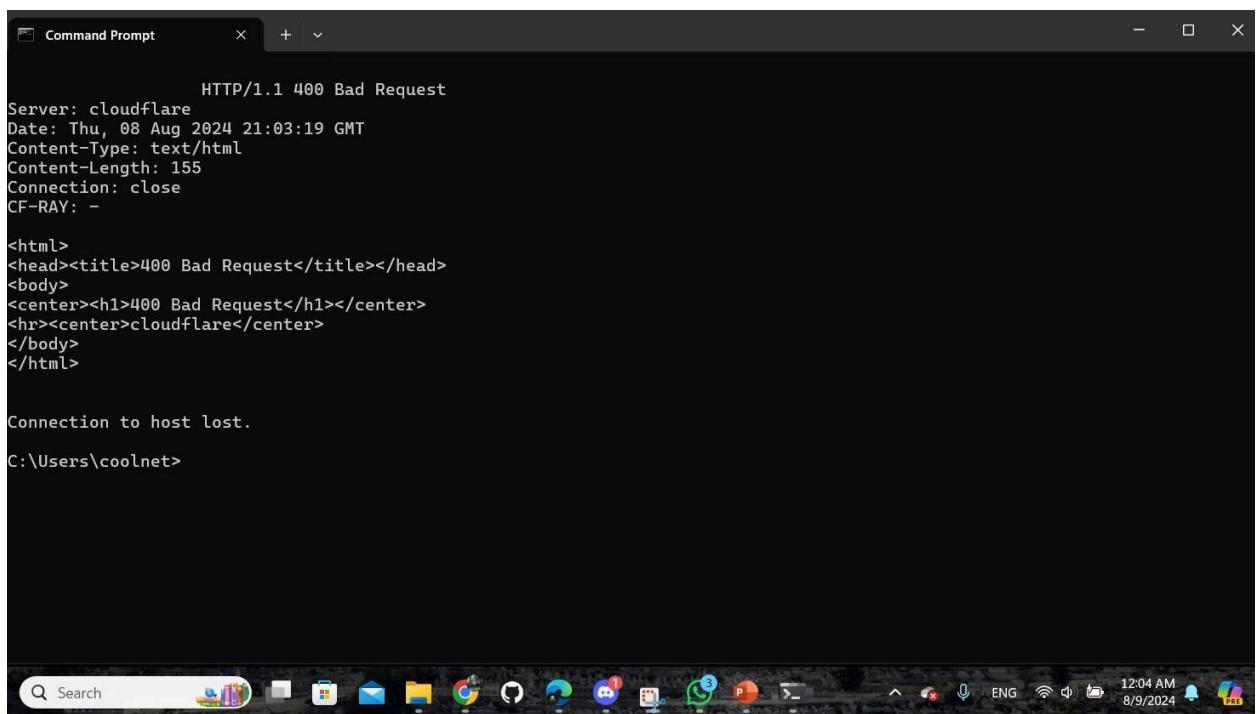
Non-authoritative answer:
Name: www.ox.ac.uk.cdn.cloudflare.net
Addresses: 104.22.49.74
          172.67.26.89
          104.22.48.74
Aliases: www.ox.ac.uk

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://www.ox.ac.uk).



A screenshot of a Windows Command Prompt window titled "Command Prompt". The window displays the following text:

```
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: -

<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>
```

The taskbar at the bottom of the screen shows various icons for applications like File Explorer, Google Chrome, and Microsoft Edge. The system tray indicates the date and time as 8/9/2024 at 12:04 AM.

Figure 6: telnet [www.ox.ac.uk](http://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](http://www.ox.ac.uk).

Unlock the Intelligence Handbook: Your Guide to Cyber Threat IntelligenceGet it for FreeX

104.22.48.74  
NO RDNS FOUND

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

**RIR Allocation Summary**

PREFIX: <a href="#">104.16.0.0/12</a>	REGIONAL REGISTRY: <a href="#">ARIN</a>
GEOIP COUNTRY:	ALLOCATION STATUS: <a href="#">Allocated</a>
IP ADDRESSES: 1,048,576	ALLOCATION DATE: 28 <sup>th</sup> March 2014



Figure 7: Details about tier1 ISP for [www.ox.ac.uk](http://www.ox.ac.uk)

Here we use this website to find all details we need about "[www.ox.ac.uk](http://www.ox.ac.uk)" that we need.

## Part 4: Wireshark

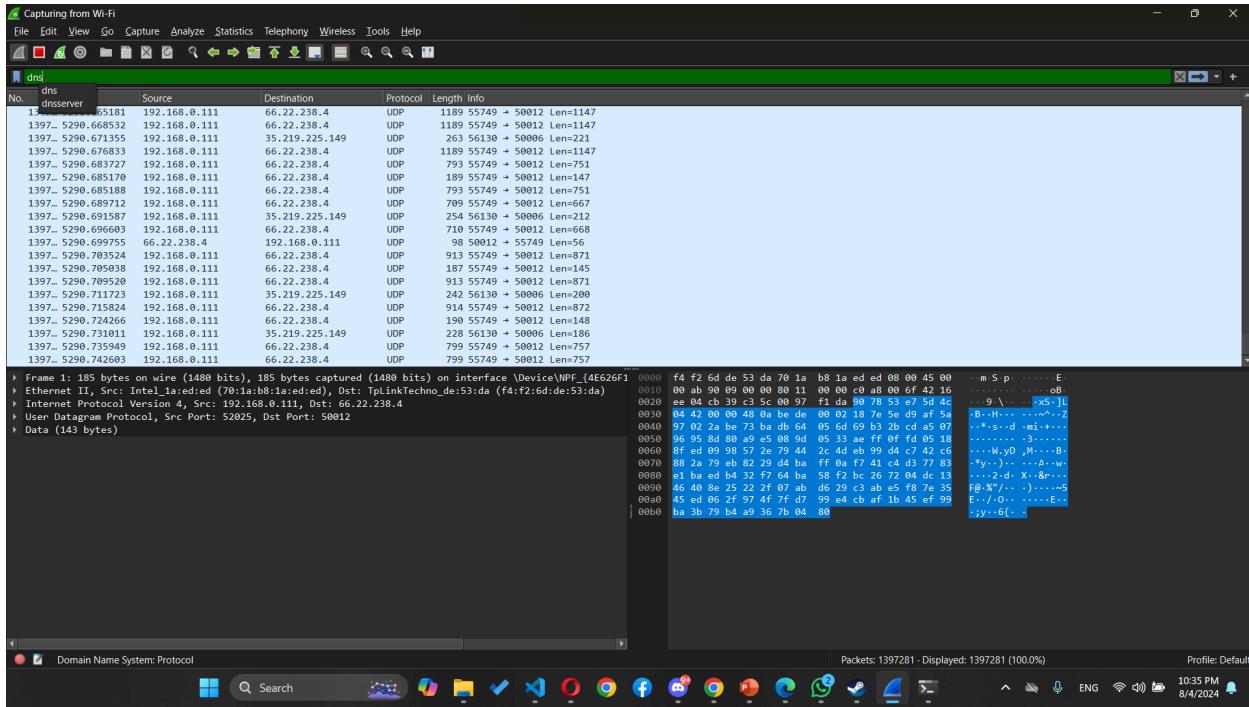


Figure 8: dns on wire shark

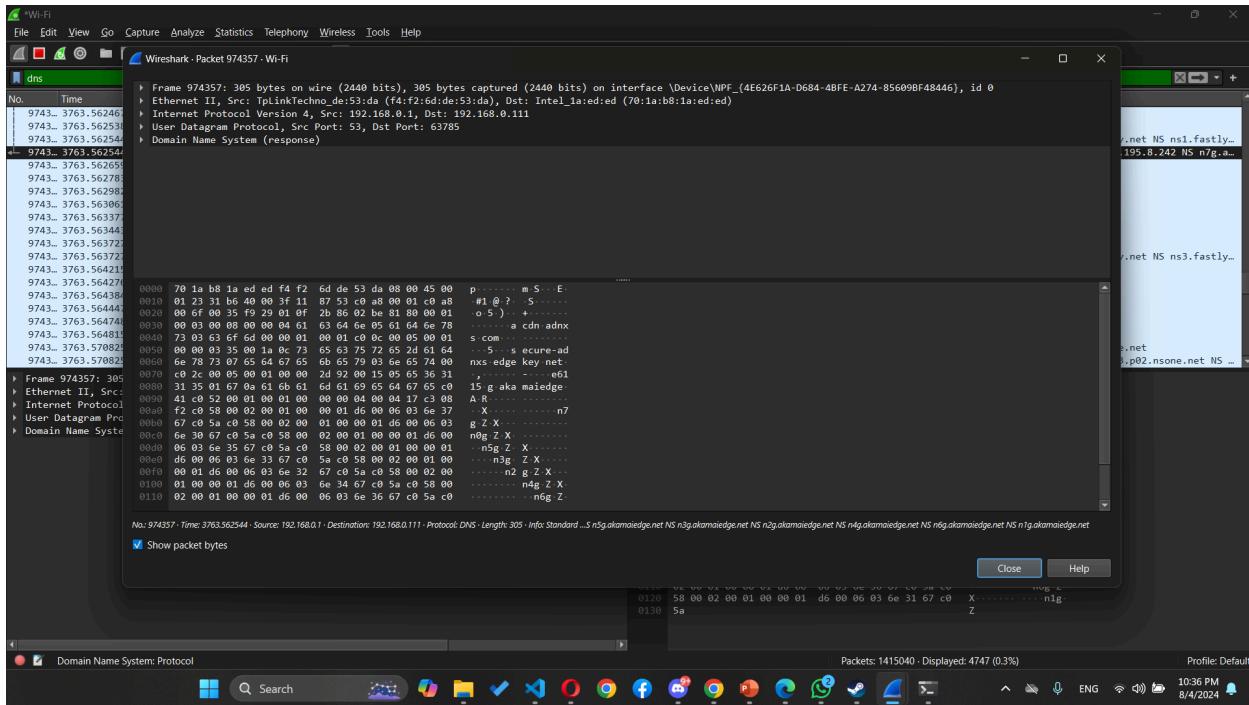


Figure 9: details for one of the packets

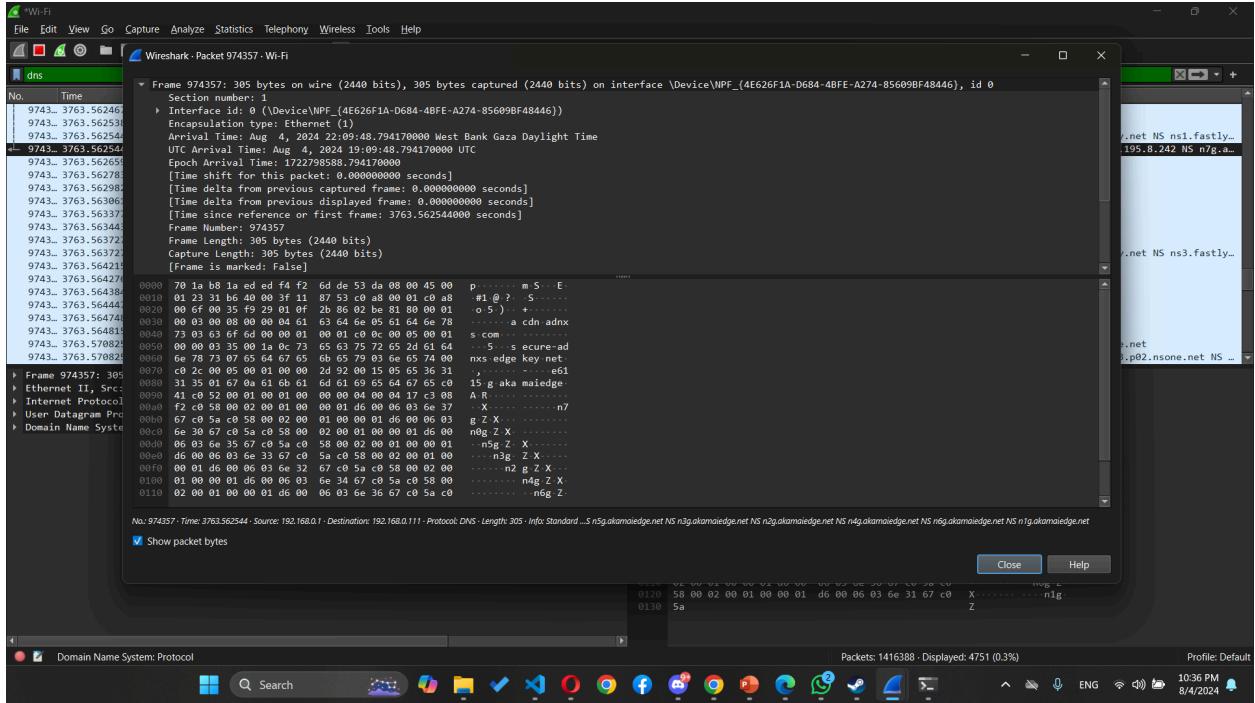


Figure 10: frame details on the packet

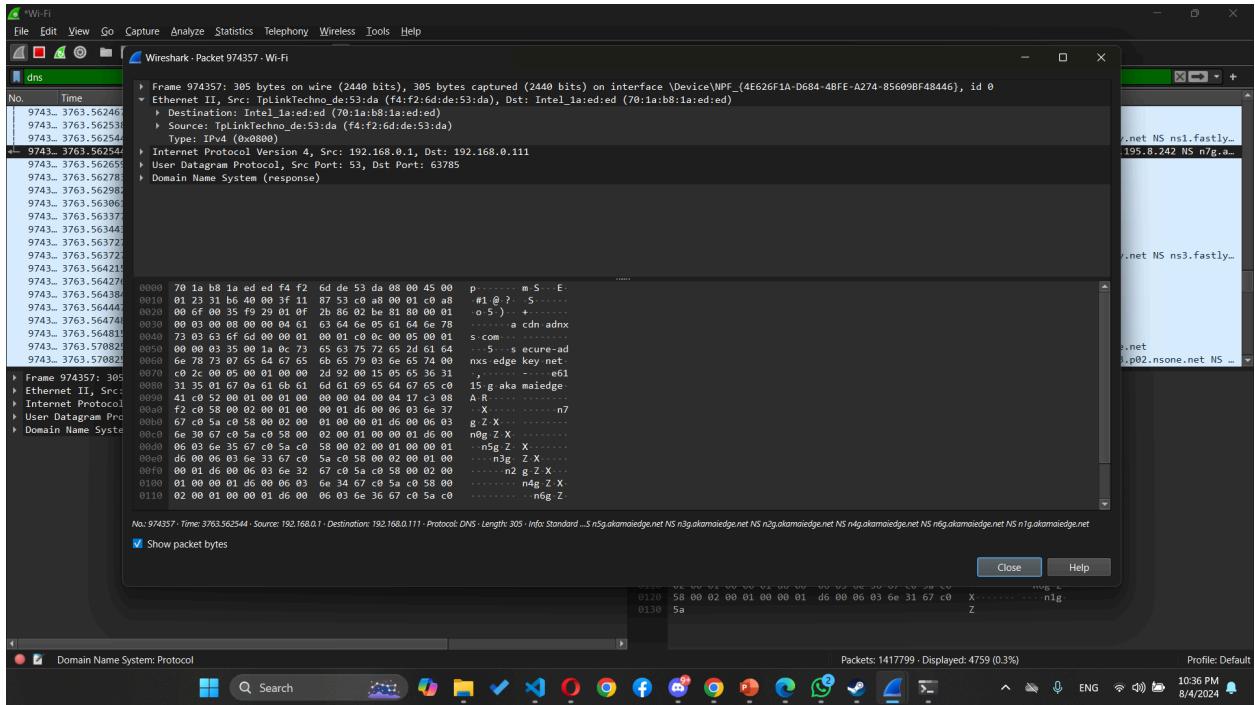
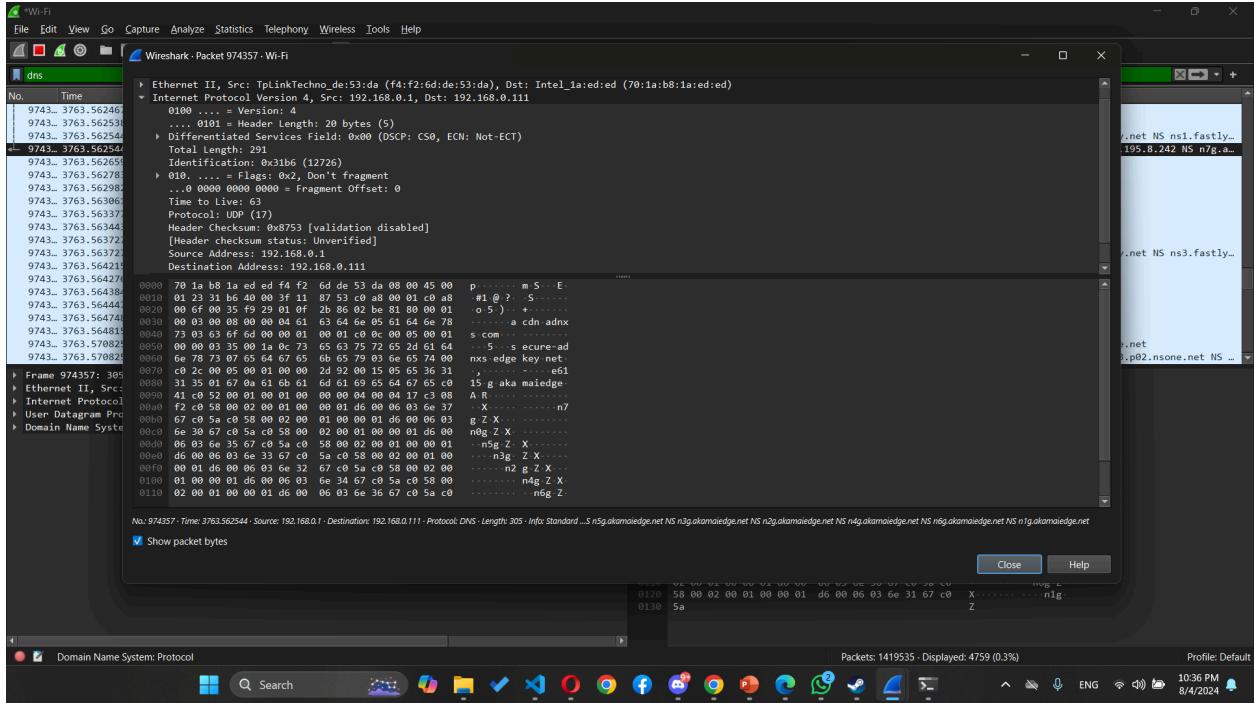
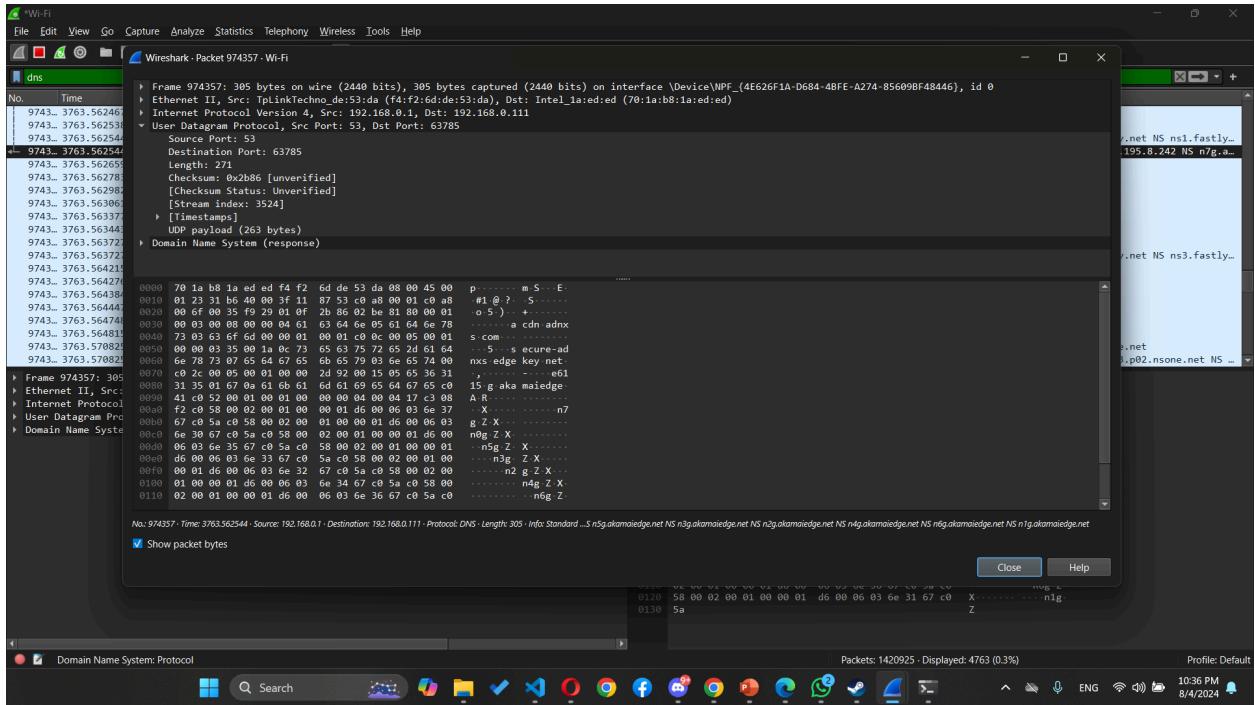


Figure 11: ethernet II details



*Figure 12: details for the source and destination for it*



*Figure 13: user datagram protocol*

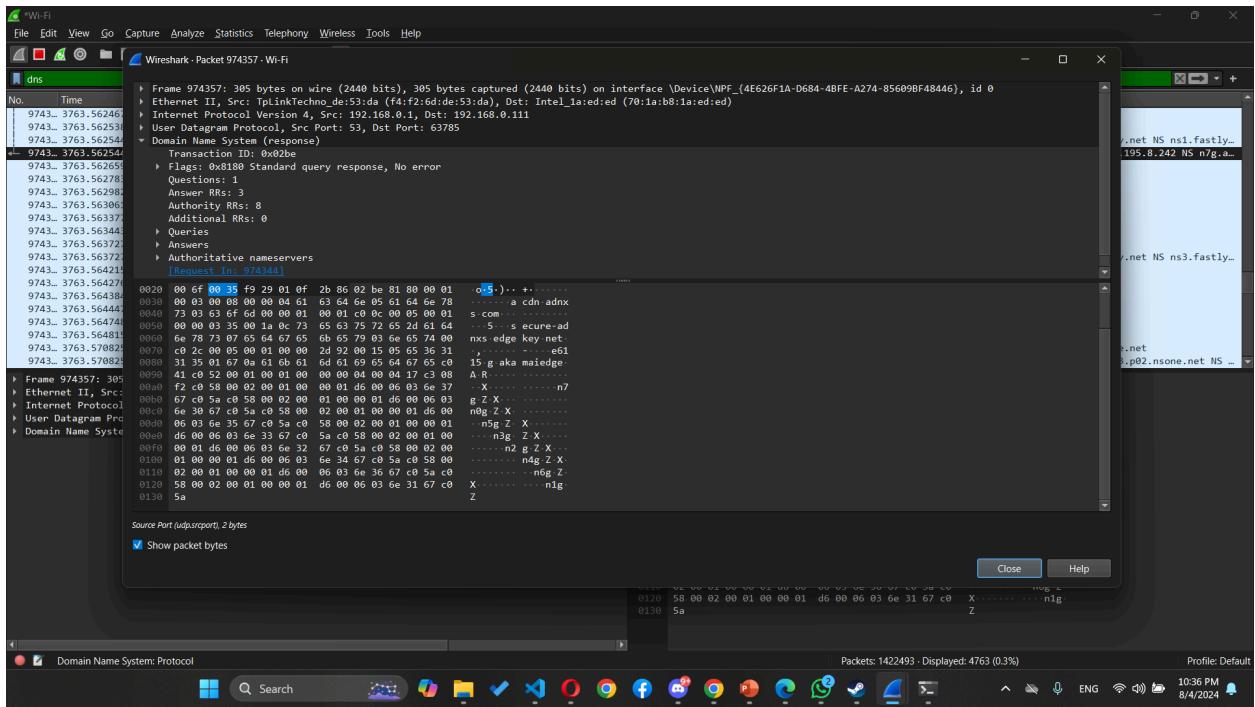


Figure 14: domain name system

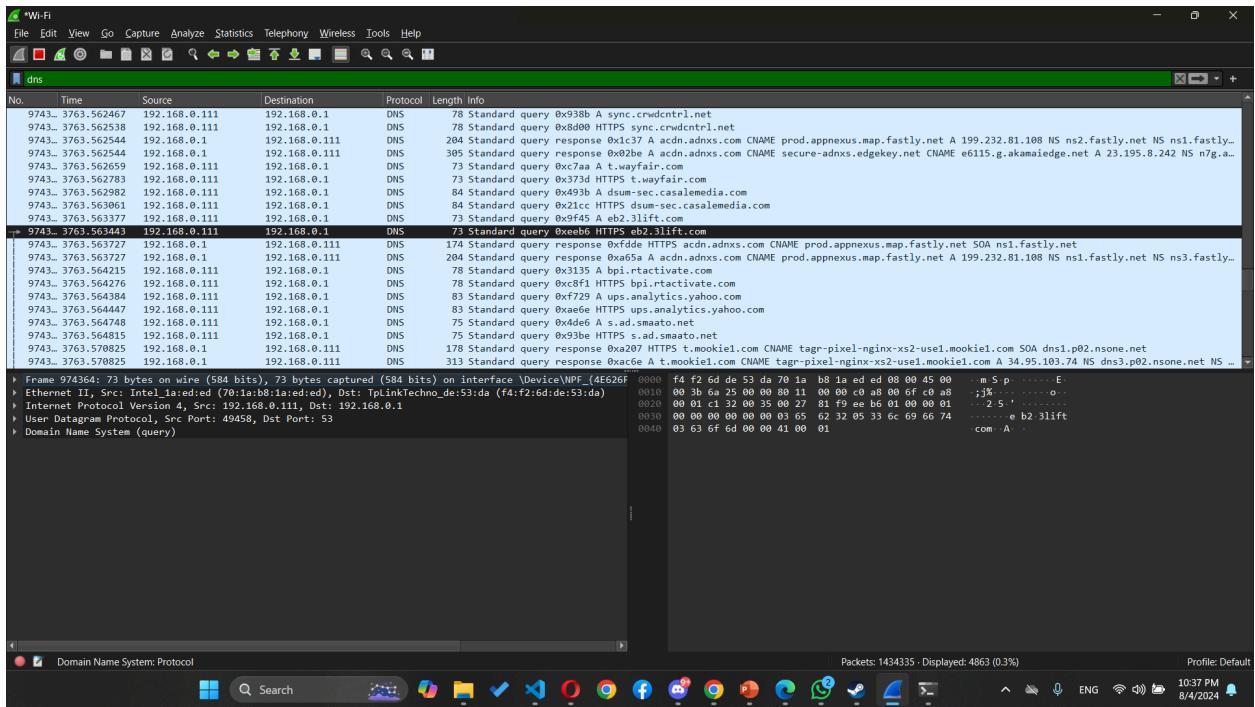


Figure 15: The packet that we show the details about it

- 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 two files: `client1.py` and `server.py`.
- Code Editor:** Displays the `server.py` file content. The code uses sockets to listen for connections and replace vowels with '#'. It includes imports for `socket`, defines a port and IP, creates a socket, binds it to the IP and port, and starts listening. It then enters a loop where it receives a message from a client, converts it to uppercase, replaces vowels with '#', and sends the modified message back to the client. Finally, it closes the connection.
- Terminal:** Shows the command line output:

```
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\bundled\libs\debugpy\adapter'...'\debugpy\launcher' '51375' '--' 'c:\Users\coolnet\Desktop\network project\task2\1\client.py'
Enter your message:
```
- Status Bar:** Shows the current file is `server.py`, line 3, column 18, spaces: 4, tab width: 8, encoding: UTF-8, CRLF, Python 3.9.16 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 pane displays the Python file `client1.py` with the following 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 sendData = input('')
13 clientSocket.send(sendData.encode())
14
15 print(clientSocket.recv(1200).decode())
16 print(clientSocket.recv(1200).decode())
17
18 clientSocket.close()
19
```

The right pane shows a terminal window with the following output:

```
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.debugpy-2024.10.0-win32-x64\bundled\libs\debugpy\adapter'..\..\debugpy\launcher' '51375' '--' 'c:\Users\coolnet\Desktop\network project\task2\1\client1.py'
Enter your message:
```

The status bar at the bottom indicates the terminal is in Python mode.

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:

The screenshot shows a Windows desktop environment with a terminal window open in Visual Studio Code. The terminal window displays Python code for a network server and its execution output. The code is as follows:

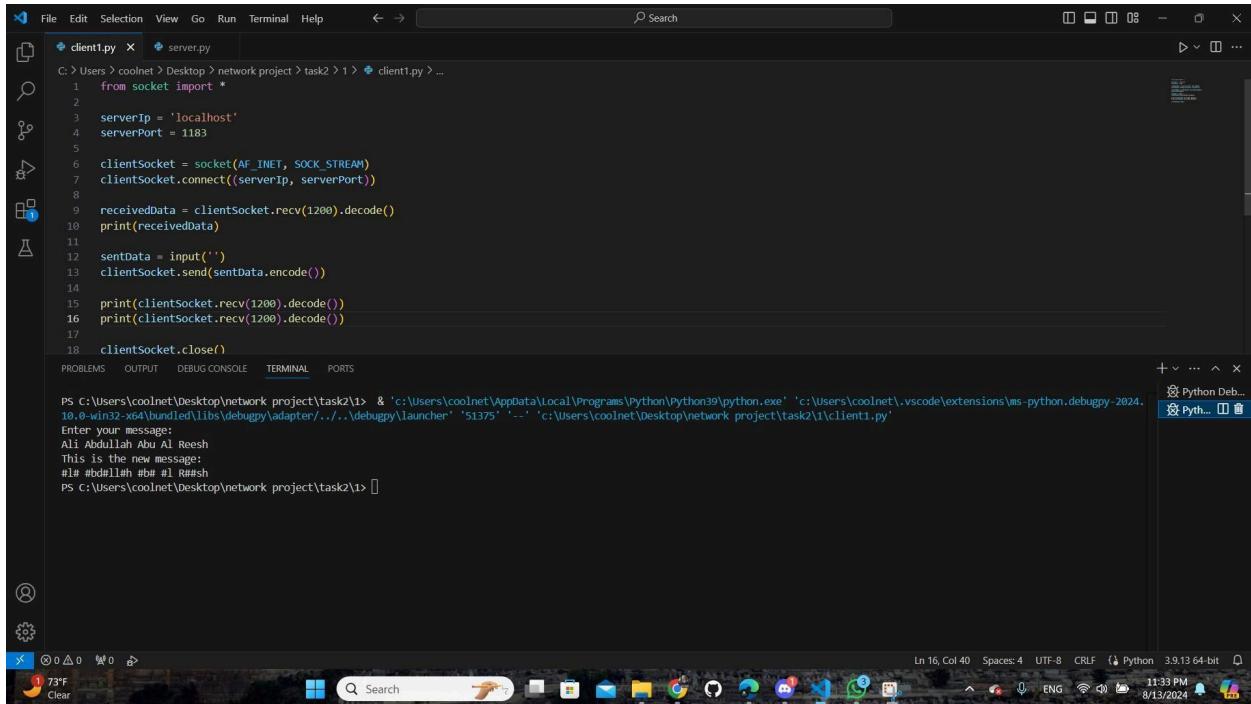
```
from socket import *
serverPort = 1183
serverIp = 'localhost'
vowels = ['a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U']
serverSocket = socket(AF_INET, SOCK_STREAM)
serverSocket.bind((serverIp, serverPort))
serverSocket.listen(1)
print("Server is on and waiting for a connection...")
while True:
    connectionClient, clientAddress = serverSocket.accept()
    print(f"Connected to {clientAddress}")
    connectionClient.send("Enter your message: ".encode())
    clientMessage = connectionClient.recv(1200).decode()
    print(clientMessage)
```

The terminal output shows the server listening on port 1183 and then connecting to a client at 127.0.0.1:51378. The client message "Server is on and waiting for a connection..." is printed, followed by "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 a Visual Studio Code interface. On the left, there's a file tree with 'client1.py' selected. The main editor area contains Python code for a client socket program. The terminal tab at the bottom shows the execution of the script and its output. The output shows the client sending a message to the server, which converts vowels to # and returns it.

```
client1.py
from socket import *
serverIp = 'localhost'
serverPort = 1183
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverIp, serverPort))
receivedData = clientSocket.recv(1200).decode()
print(receivedData)
sentData = input('')
clientSocket.send(sentData.encode())
print(clientSocket.recv(1200).decode())
print(clientSocket.recv(1200).decode())
clientSocket.close()
```

```
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.debugpy-2024.10.0-win32-x64\bundledlibs\debugpy\adapter/../.debugpy\launcher' '51375' '--' 'c:\Users\coolnet\Desktop\network project\task2\1\client1.py'
Enter your message:
Ali Abdullah Abu Al Reesh
This is the new message:
#l# #bdell# ## #1 R#h#
PS C:\Users\coolnet\Desktop\network project\task2\1>
```

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 Windows desktop environment with a terminal window open in VS Code. The terminal displays Python code for a client socket program and its execution output. The code connects to a local server at port 1183 and prints the received data. The output shows the server's response and the client's connection closure.

```
File Edit Selection View Go Run Terminal Help < > Search

client1.py x server.py
C: > Users > coolnet > Desktop > network project > task2 > 1 > client1.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  sendData = input('')
13  clientSocket.send(sendData.encode())
14
15  print(clientSocket.recv(1200).decode())
16  print(clientSocket.recv(1200).decode())
17
18  clientSocket.close()

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

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.debugpy-2024.10.0-win32-x64\bundled\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)
Message from client: Ali Abdullah Abu Al Reesh
connection closed.

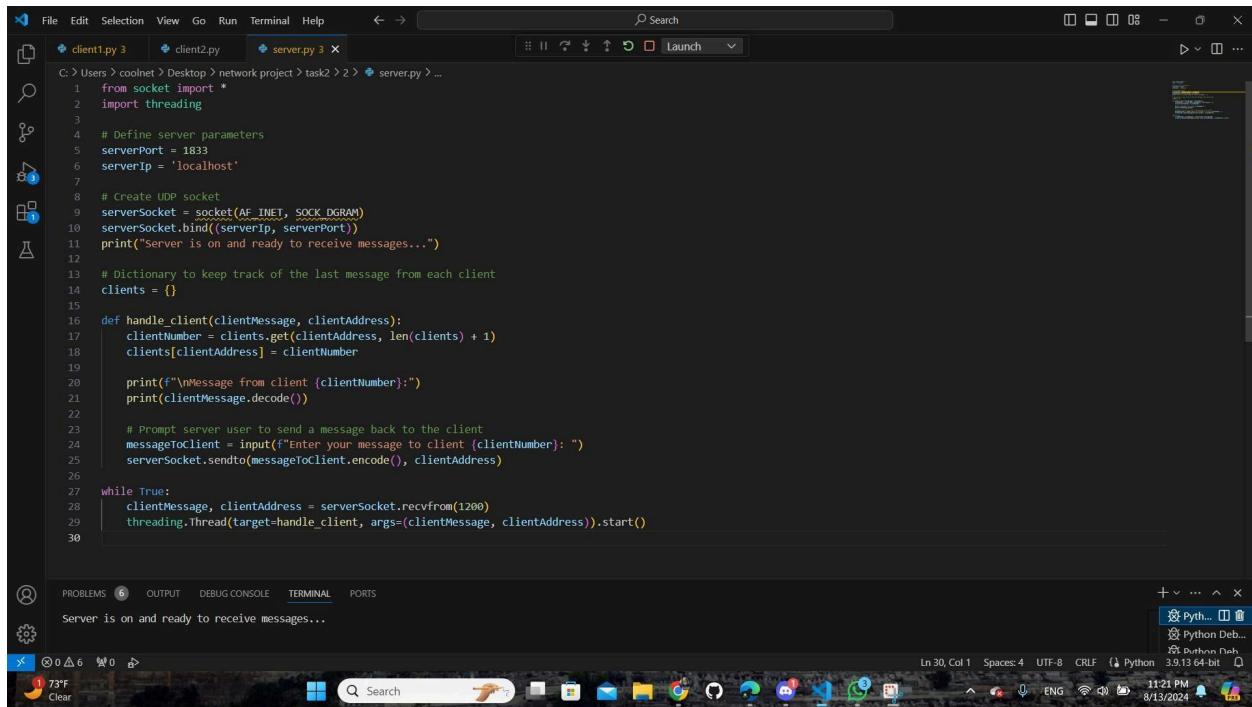
PS C:\Users\coolnet\Desktop\network project\task2\1>
```

*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:



```
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)
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(f"Enter your message to client {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
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Server is on and ready to receive messages...

Ln 30, Col 1 Spaces: 4 UTF-8 CRLF Python 3.9.13 64-bit 11:21 PM 8/13/2024

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:

```
File Edit Selection View Go Run Terminal Help ⏪ ⏴ Launch Search
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: ")
13     clientSocket.sendto(message.encode(), (serverIp, serverPort))
14
15     # Receive the server's response
16     serverMessage, serverAddress = clientSocket.recvfrom(1200)
17     print("Message from server:", serverMessage.decode())
18
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Server is on and ready to receive messages...

LN 2, COL 1 SPACES: 4 UFT-8 CRLF PYTHON 3.9.13 64-BIT 11:22 PM 8/13/2024

Figure 22:client 2 code

```
File Edit Selection View Go Run Terminal Help ⏪ ⏴ Launch Search
C:\Users\coolnet\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(1200)
17     print("Message from server:", serverMessage.decode())
18
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

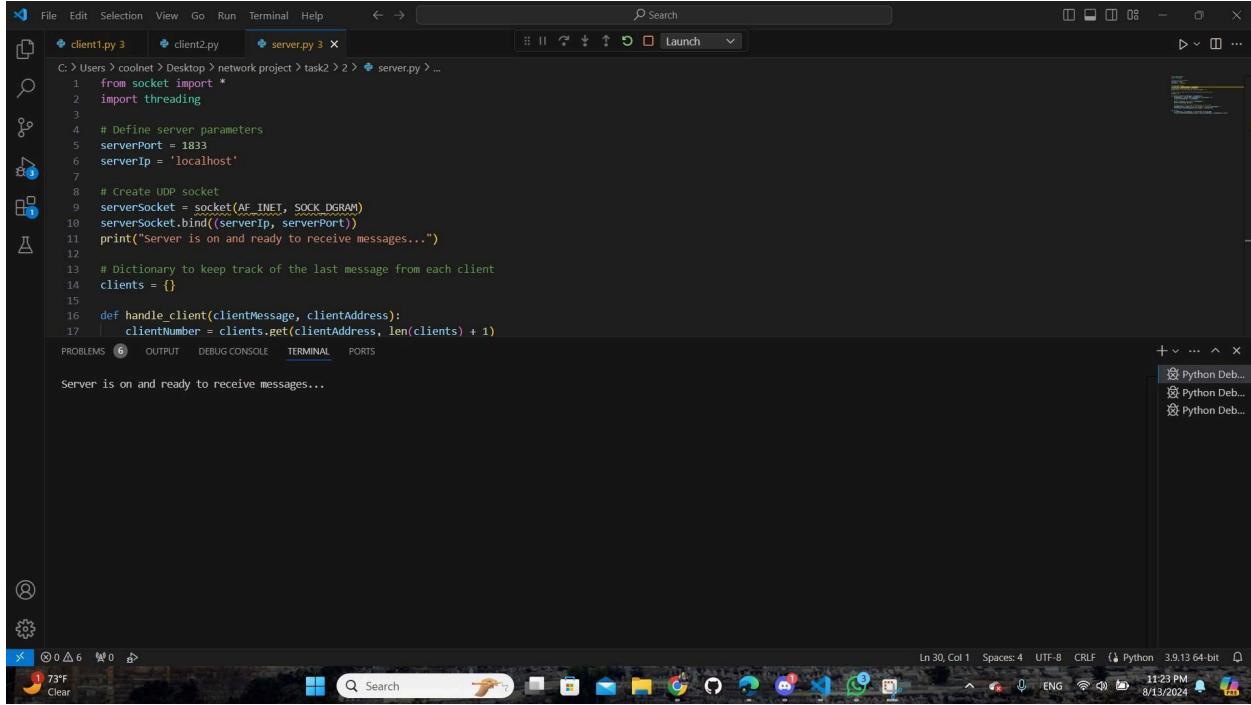
Server is on and ready to receive messages...

LN 18, COL 1 SPACES: 4 UFT-8 CRLF PYTHON 3.9.13 64-BIT 11:22 PM 8/13/2024

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 window with three tabs: 'client1.py', 'client2.py', and 'server.py'. The 'server.py' tab is active, displaying 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 editor, the terminal output shows the message: "Server is on and ready to receive messages...".

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:

A screenshot of a terminal window titled "client1.py 3". The window shows Python code for a UDP client. In the terminal pane, the user enters "Enter message to send to server: hi, my name is mouath" and receives a response "Message from server: hello, and my is baha". The terminal also displays the command "10.0-win32-x64\bundled\libs\debugpy\adapter\..\..\debugpy\launcher" and the file path "c:\Users\coolnet\Desktop\network project\task2\2\client1.py". The bottom status bar shows "Ln 2, Col 1" and "Python 3.9.13 64-bit".

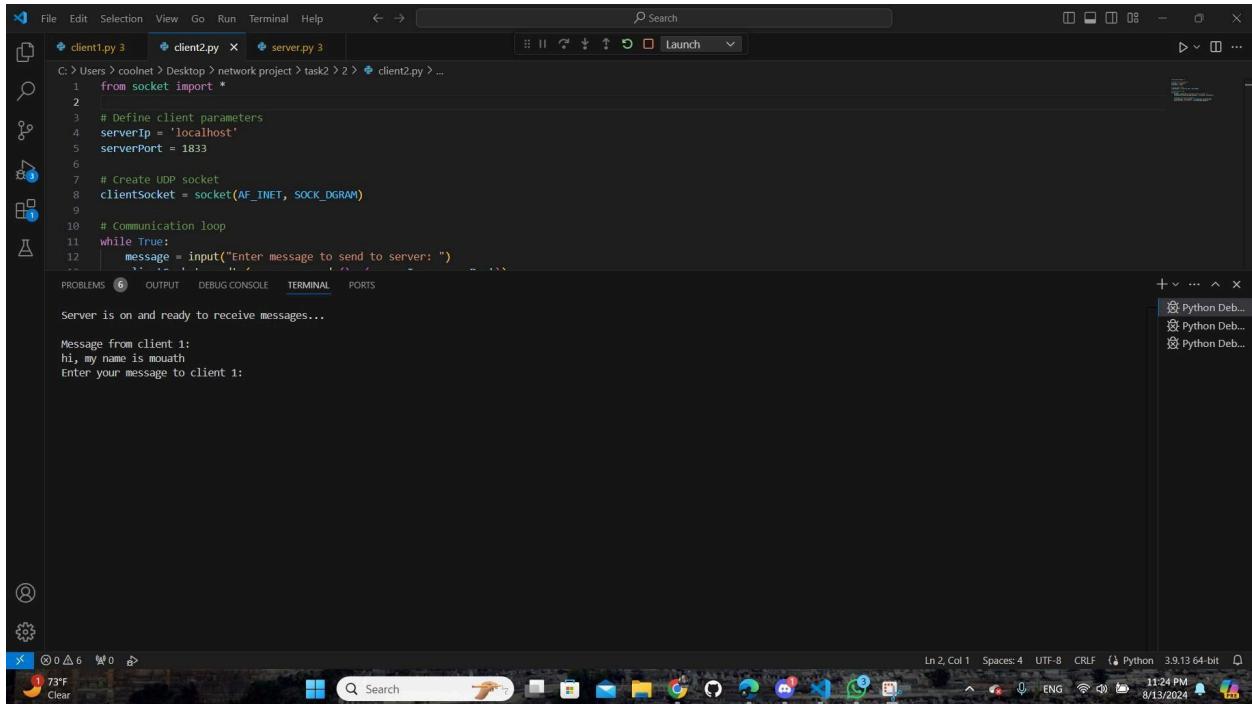
Figure 24:client 1 message

A screenshot of a terminal window titled "client2.py 3". The window shows Python code for a UDP client. The user enters "Enter message to send to server: my name is zaid mouse" and receives a response "Message from server: hello zaid, how do you do?". The terminal also displays the command "10.0-win32-x64\bundled\libs\debugpy\adapter\..\..\debugpy\launcher" and the file path "c:\Users\coolnet\Desktop\network project\task2\2\client2.py". The bottom status bar shows "Ln 2, Col 1" and "Python 3.9.13 64-bit".

Figure 25: 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 open: `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: ")

```

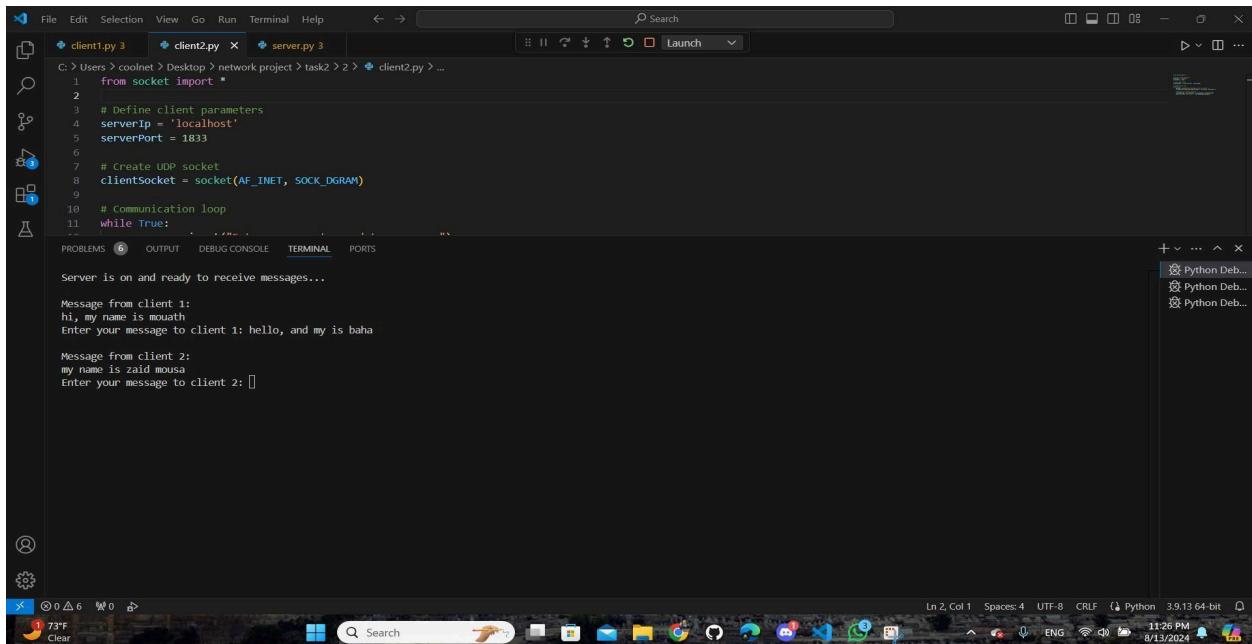
Below the code, the terminal output shows:

```

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

```

Figure 26:server receive client 1 message



The screenshot shows a code editor interface with three tabs open: `client1.py`, `client2.py`, and `server.py`. The `server.py` tab is active, displaying the same code as Figure 26.

Below the code, the terminal output shows two messages:

```

Server is on and ready to receive messages...
Message from client 1:
hi, my name is mouath
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 27: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-

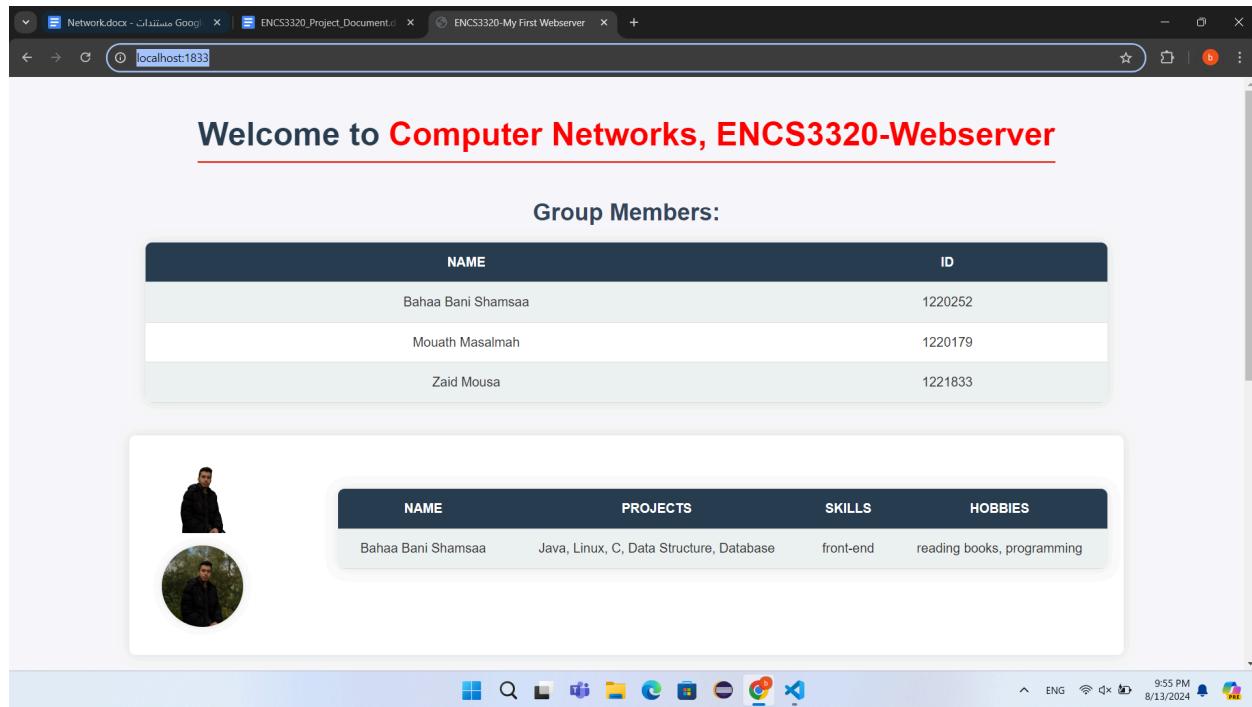


Figure 28:request /

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

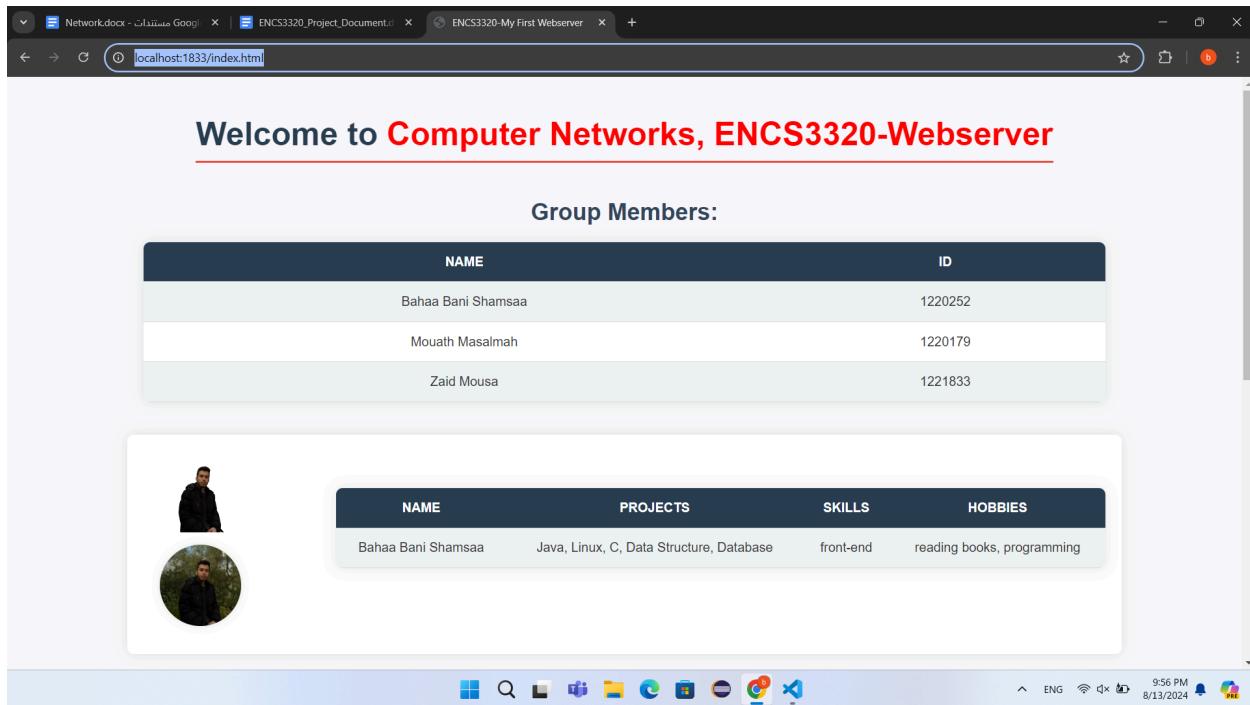


Figure 29: index.html request

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

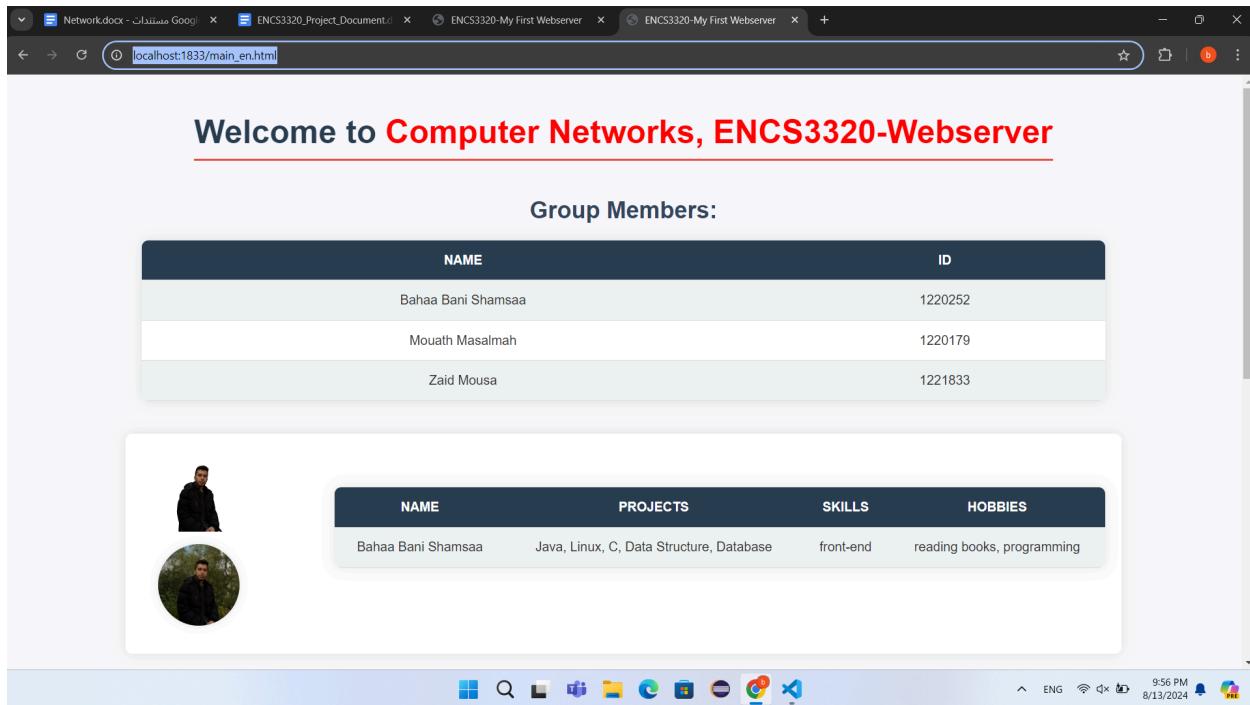


Figure 30:main\_en.html request

As we see here when I type link [http://localhost:1833/main\\_en.html](http://localhost:1833/main_en.html) it goes to the english page

a.

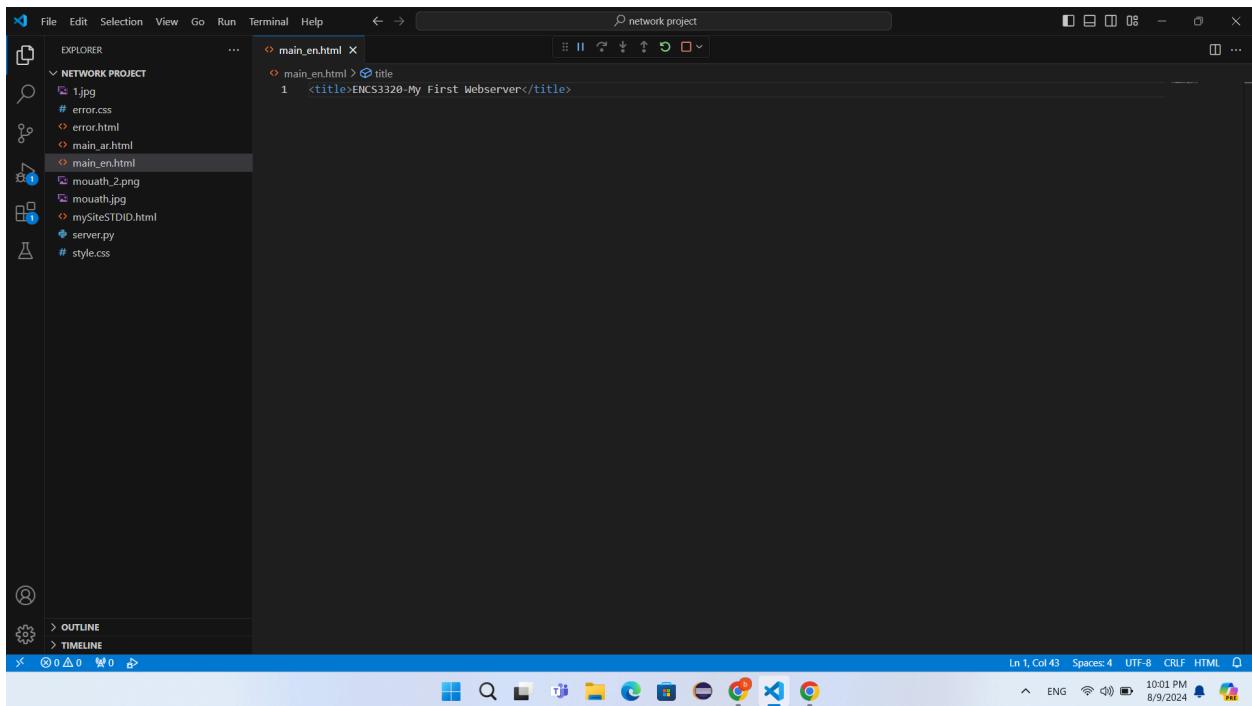


Figure 31:title of the page

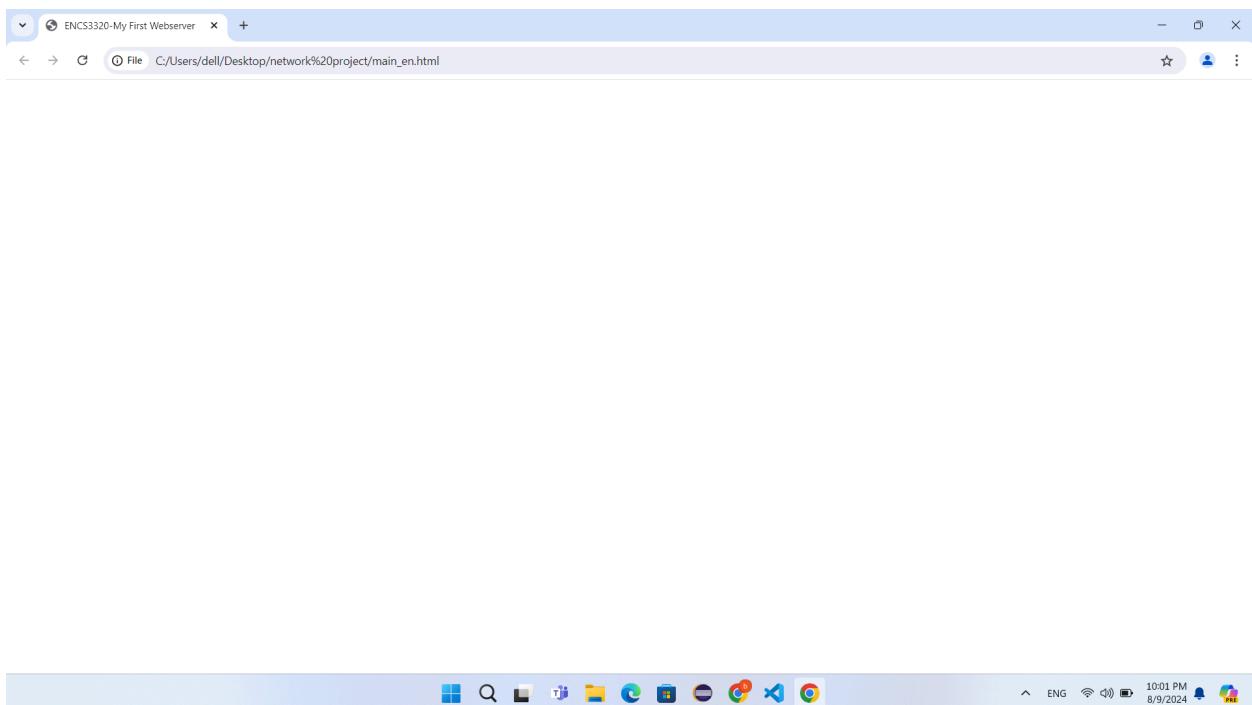
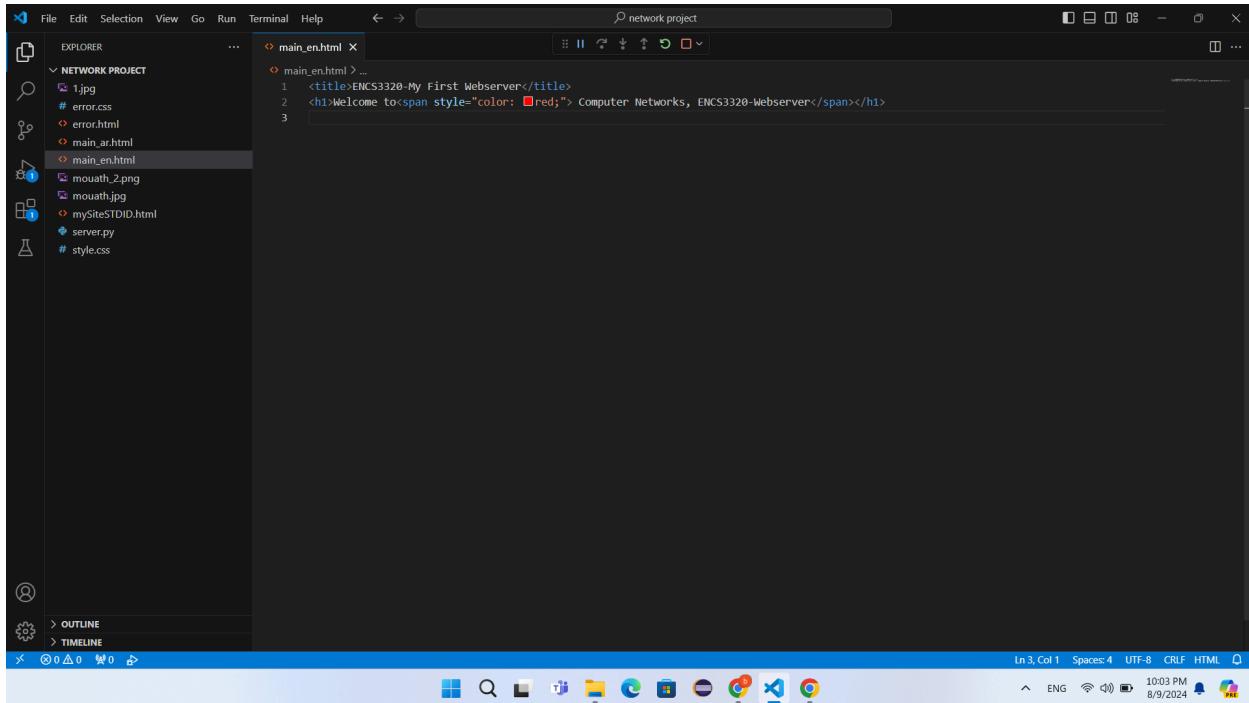


Figure 32:browser title

Here we put the title ENCS-My First Webserver

b.



The screenshot shows a code editor interface with a dark theme. The left sidebar is labeled 'EXPLORER' and lists several files and folders under 'NETWORK PROJECT'. The main pane displays the content of 'main\_en.html'. The code is as follows:

```
<title>ENCS3320-My First Webserver</title>
<h1>Welcome to<span style="color: red;"> Computer Networks, ENCS3320-Webserver</span></h1>
```

The status bar at the bottom right indicates 'Ln 3, Col 1' and '8/9/2024'.

Figure 33:welcome message

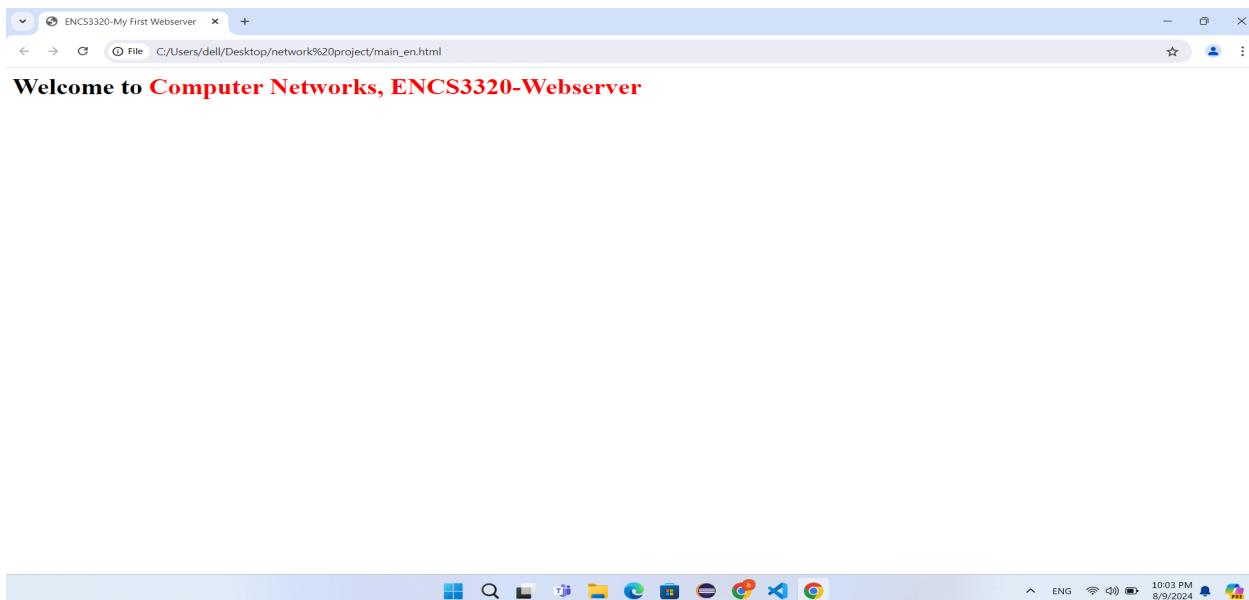


Figure 34:browser welcome

Here I use span to make the words with different colors

c.

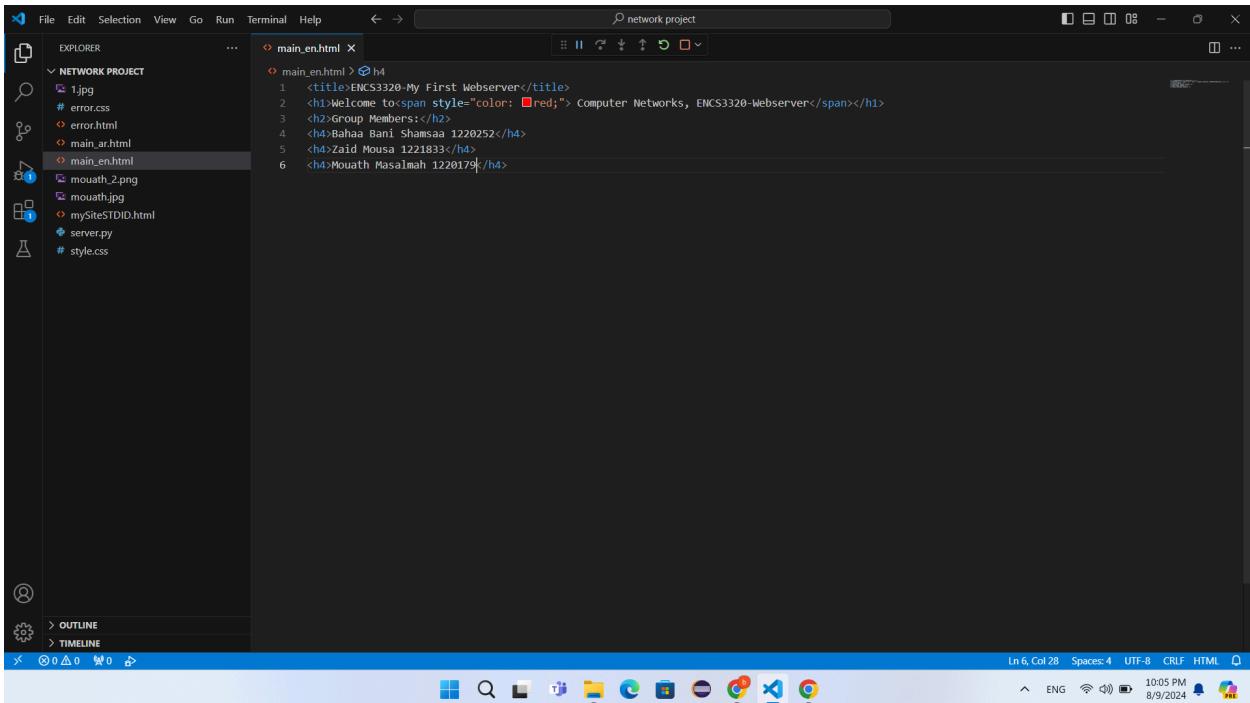


Figure 35:group members

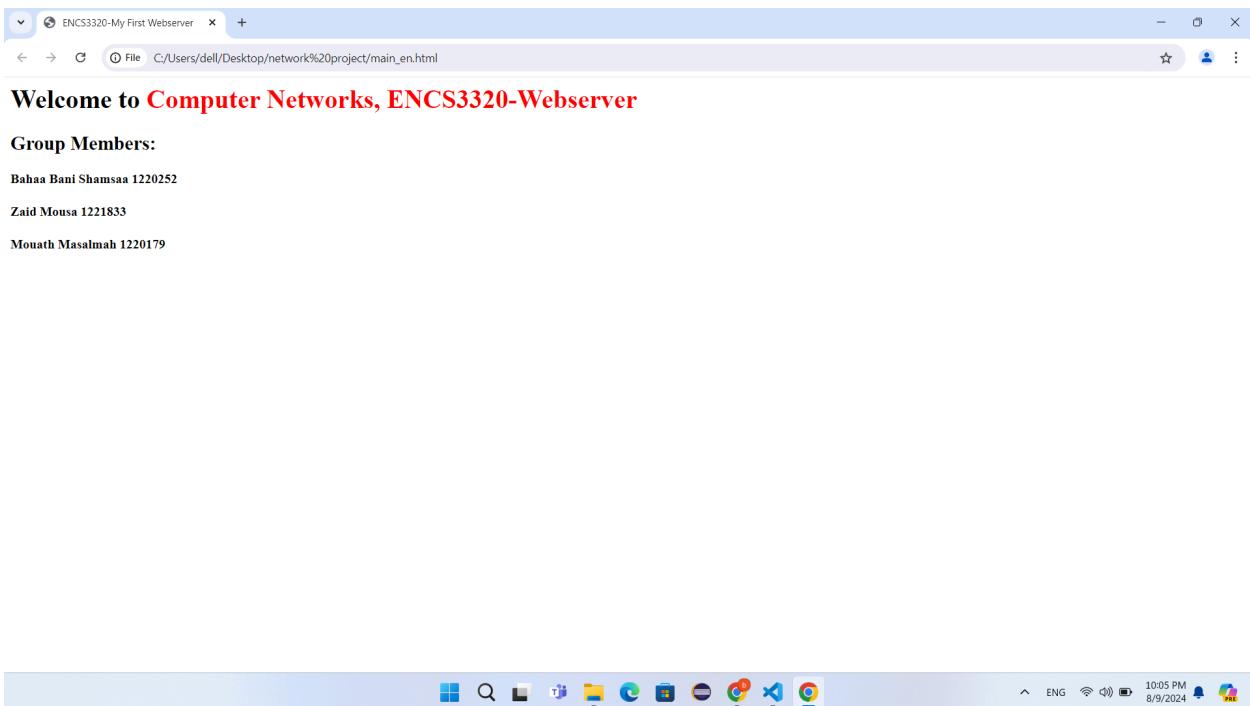
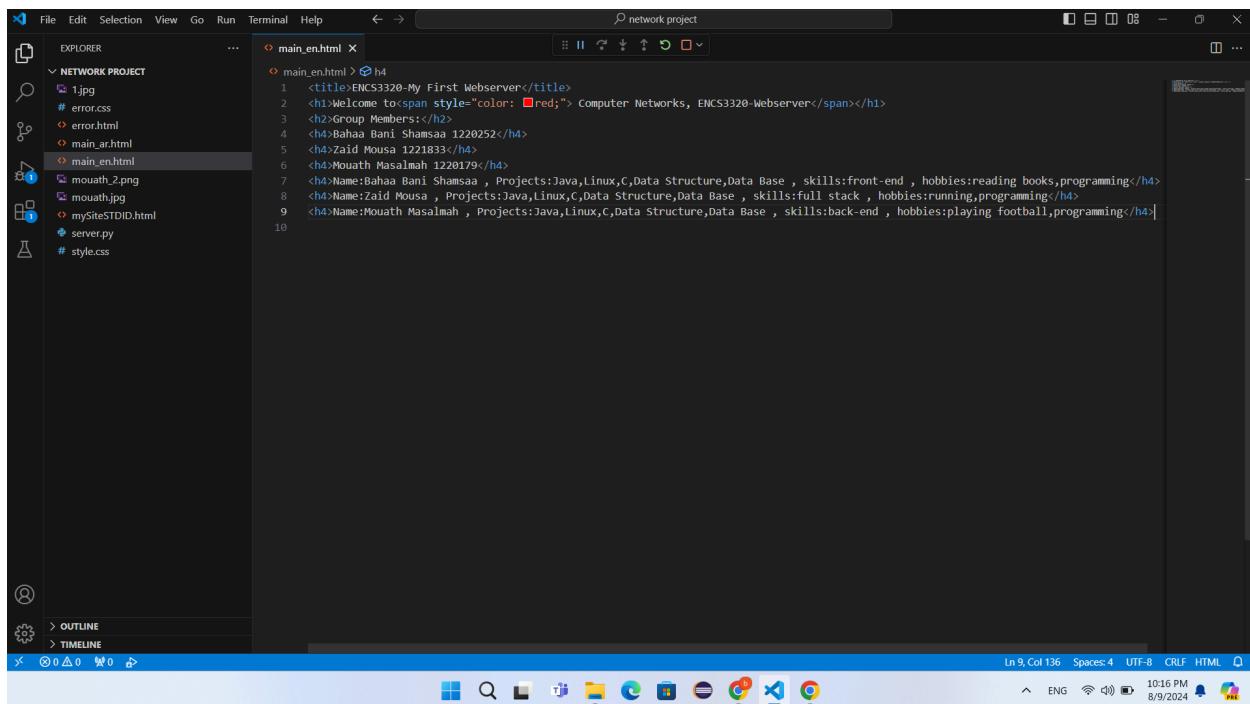


Figure 36:browser group members

Here we type the group members names with the numbers

d.



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

- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Search Bar:** network project
- Explorer:** Shows a "NETWORK PROJECT" folder containing files: 1.jpg, #error.css, main\_ar.html, main\_en.html (selected), mouauth\_2.png, mouauth.jpg, mySiteSTID.html, server.py, and #style.css.
- Code Editor:** The main\_en.html file is open, displaying the following HTML code:

```
<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>
<h4>Zaid Mousa 1221833</h4>
<h4>Mouath Masalmah 1220179</h4>
<h4>Name:Bahaa Bani Shamsaa , Projects:Java,Linux,C,Data Structure,Data Base , skills:front-end , hobbies:reading books,programming</h4>
<h4>Name:Zaid Mousa , Projects:Java,Linux,C,Data Structure,Data Base , skills:full stack , hobbies:running,programming</h4>
<h4>Name:Mouath Masalmah , Projects:Java,Linux,C,Data Structure,Data Base , skills:back-end , hobbies:playing football,programming</h4>
```
- Status Bar:** Ln 9, Col 136, Spaces: 4, UTF-8, CRLF, HTML, 10:16 PM, 8/9/2024.
- Taskbar:** Shows various application icons.

Figure 37: info about group members

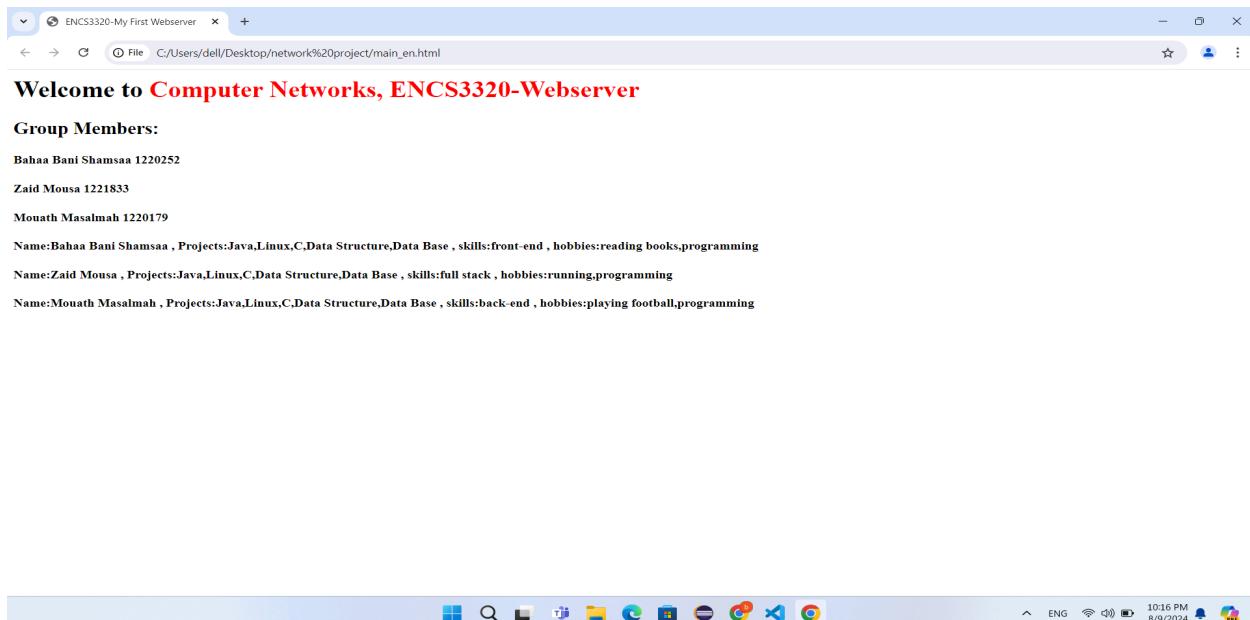


Figure 38:browser info group members

Here we type the names, projects , skills and hobbies

e and f:

```
<title>ENCS3320-My First Webserver</title>
<link rel="stylesheet" type="text/css" href="style.css">
<body>
<h1>Welcome to> Computer Networks, ENCS3320-Webserver</span></h1>
<h2>Group Members:</h2>

<table class="table">
<tr>
<th>Name</th>
<th>ID</th>
</tr>
<tr>
<td>Bahaa 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>
<table class="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>Mouath Masalmah</td>
<td>Java,Linux,C,Data Structure,Data Base</td>
<td>back-end</td>
<td>dancing,programming</td>
</tr>
</table>
```

Figure 39:table 1

```
<table class="table">
<tr>
<th>Skills</th>
<th>Hobbies</th>
</tr>
<tr>
<td>Java,Linux,C,Data Structure,Data Base</td>
<td>front-end</td>
</tr>
<tr>
<td>reading books,programming</td>
<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>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>Mouath Masalmah</td>
<td>Java,Linux,C,Data Structure,Data Base</td>
<td>back-end</td>
<td>dancing,programming</td>
</tr>
</table>
```

Figure 40: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 41:table 3

```
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 42:css for table 1

A screenshot of a code editor interface, likely Visual Studio Code, displaying CSS code for a table. The file is named style.css and is open in the editor. The code defines styles for a table, including its background color (#ffff), border radius (8px), and overflow behavior (hidden). It also styles table headers (th) with padding (15px), center text alignment, and a solid border-bottom. Even rows (tr:nth-child(even)) have a background color of #f2f2f2. Rows on hover (tr:hover) have a background color of #f9c5c5 and a cursor of pointer. Cells (td) have a color of #555. The first child of a table header (th:first-child) has a border-top-left-radius of 8px. A span element within the table has a color of #e74c3c and a font-weight of bold.

```
# style.css > span
26 .table {
27   background-color: #ffff;
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: #ffff;
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 td:first-child {
60   border-top-left-radius: 8px;
61 }
62
63 .table th:last-child,
64 .table td:last-child {
65   border-top-right-radius: 8px;
66 }
67
68 span {
69   color: #e74c3c;
70   font-weight: bold;
71 }
```

Figure 43:css for table 2

A screenshot of a code editor interface, likely Visual Studio Code, displaying CSS code for a table. The file is named style.css and is open in the editor. The code defines styles for a table, including its background color (#ffff), border radius (8px), and overflow behavior (hidden). It also styles table headers (th) with padding (15px), center text alignment, and a solid border-bottom. Even rows (tr:nth-child(even)) have a background color of #f2f2f2. Rows on hover (tr:hover) have a background color of #f9c5c5 and a cursor of pointer. Cells (td) have a color of #555. The first child of a table header (th:first-child) has a border-top-left-radius of 8px. The last child of a table header (th:last-child) has a border-top-right-radius of 8px. A span element within the table has a color of #e74c3c and a font-weight of bold.

```
# 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 td:first-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 }
```

Figure 44:css for table 3

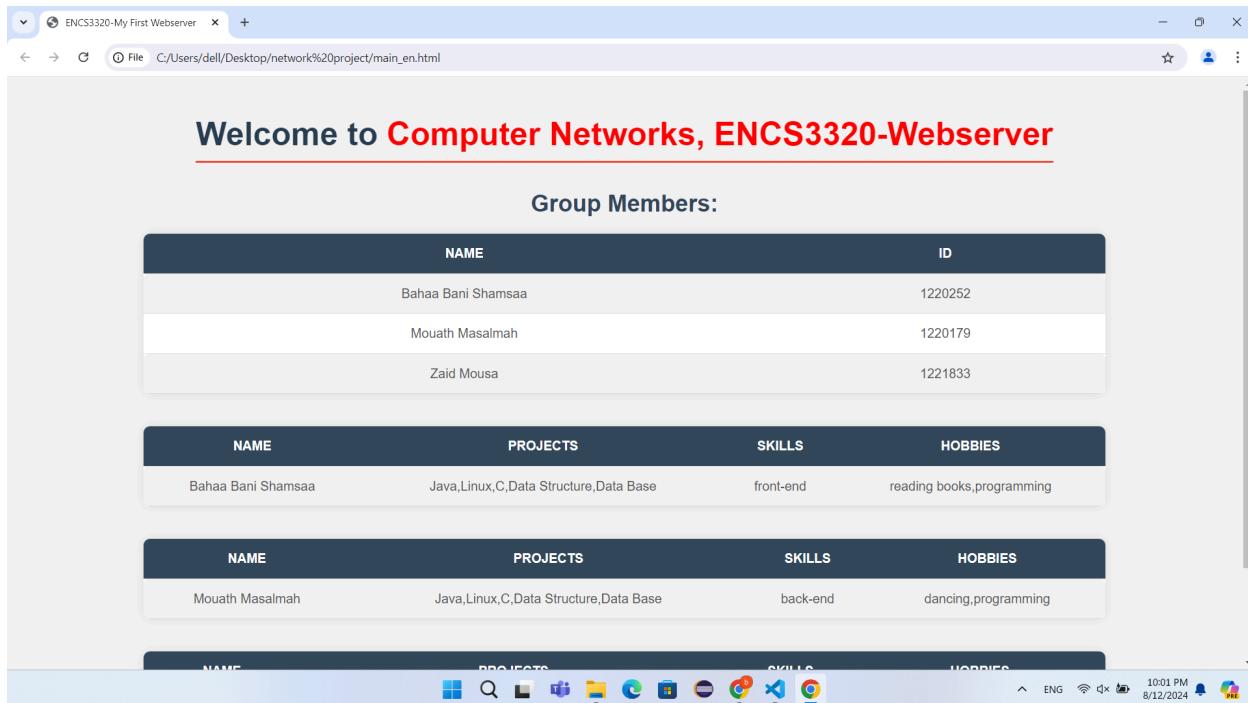


Figure 45:browser for table 1

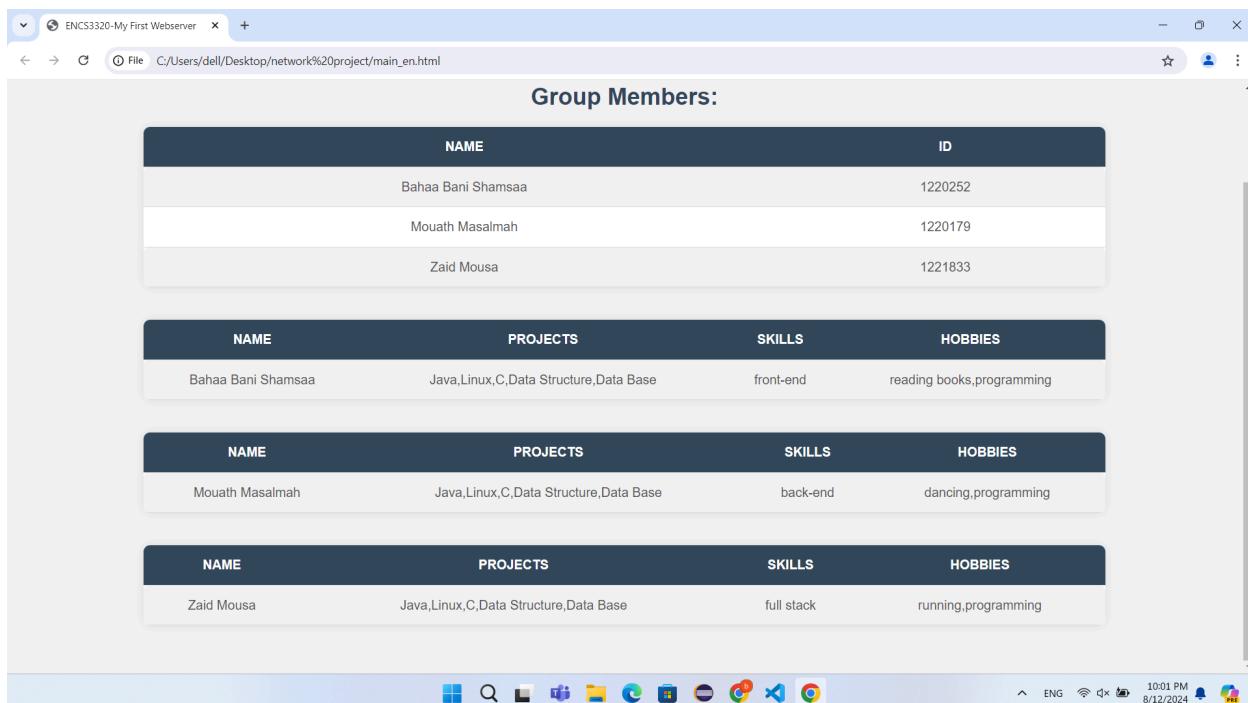
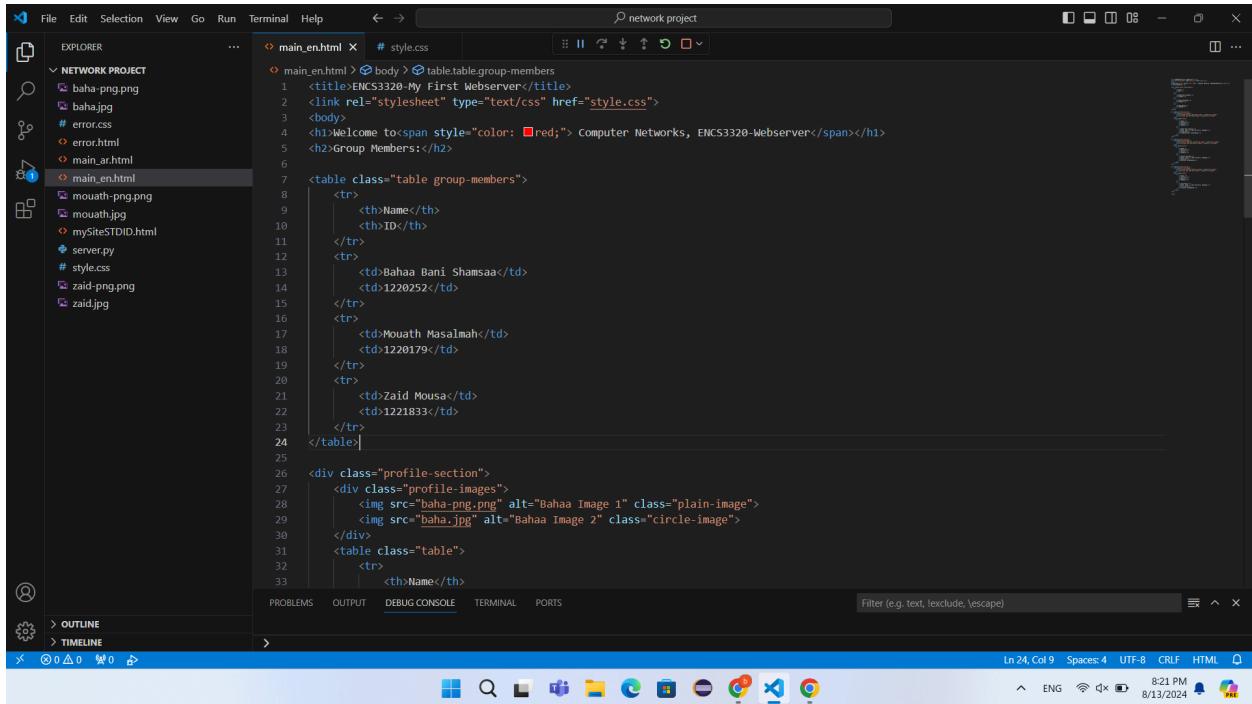


Figure 46:browser for table 2

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.**



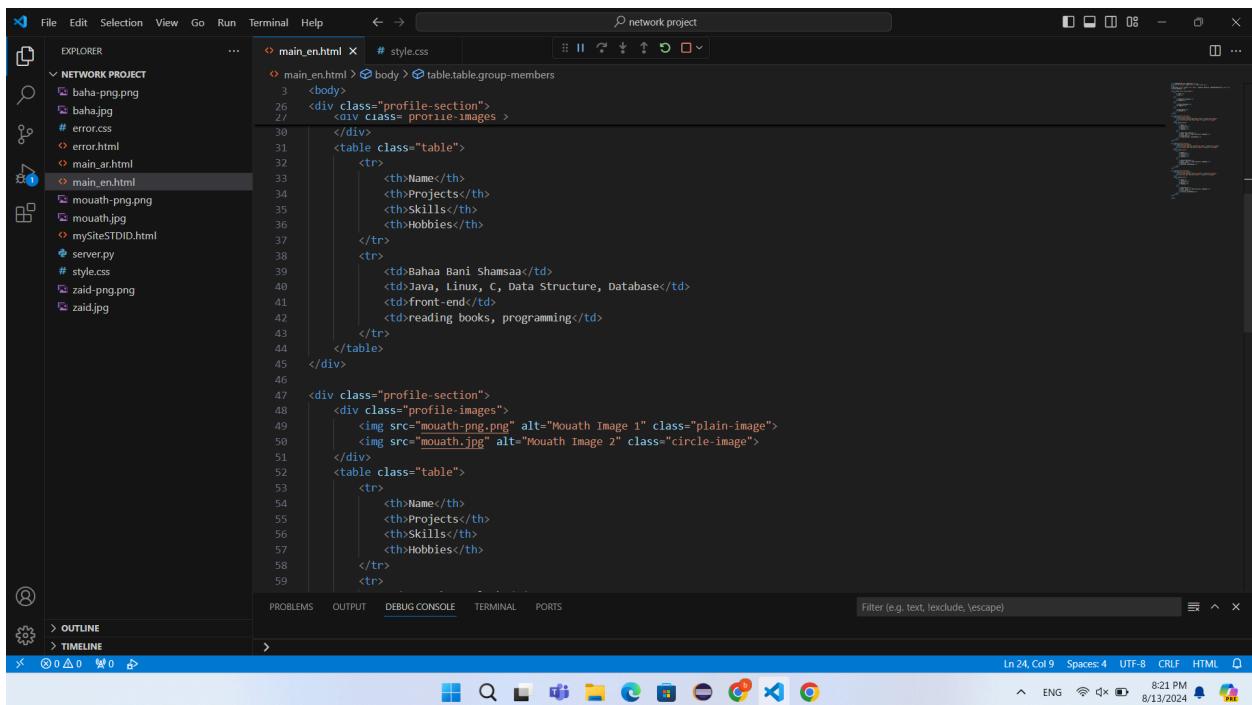
```

<main_en.html> # style.css
  <body>
    <table class="table group-members">
      <tr>
        <th>Name</th>
        <th>ID</th>
      </tr>
      <tr>
        <td>Bahaa 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>
      </table>
    </div>
  </body>
</main_en.html>

```

Figure 47:png and jpg 1



```

<main_en.html> # style.css
  <body>
    <table class="table group-members">
      <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, Database</td>
        <td>front-end</td>
        <td>reading books, programming</td>
      </tr>
    </table>

    <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>

```

Figure 48:png and jpg 2

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

```
<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>
```

The code displays two tables. The first table contains two image tags with alt attributes: "Zaid Image 1" and "Zaid Image 2". The second table has four columns: Name, Projects, Skills, and Hobbies. The first row is a header, and the second row contains data: Zaid Mousa, Java, Linux, C, Data Structure, Database, Full stack, running, programming.

Figure 49:png and jpg 3

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

```
<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>
```

The code is identical to Figure 48, except for the spelling error "full stack" instead of "Full stack" in the third column of the second table row.

Figure 50: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, mySiteSTDID.html, server.py, #style.css, zaid-png.png, and zaid.jpg.
- Code Editor:** The current file is "# style.css". The code is as follows:

```
# style.css > body
body {
    display: flex;
    flex-direction: column;
    align-items: center;
    margin: 0;
    padding: 20px;
    font-family: 'Arial', sans-serif;
    background-color: #f8f9fa;
    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;
}
```

- Bottom Bar:** Includes tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, PORTS, and a status bar showing "Ln 8, Col 32 Spaces: 4 UTF-8 CRLF CSS".

Figure 51: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, mySiteSTDID.html, server.py, #style.css, zaid-png.png, and zaid.jpg.
- Code Editor:** The current file is "# style.css". The code is as follows:

```
# style.css > body
.table {
    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;
}

.table tr:nth-child(even) {
    background-color: #ecf0f1;
}

.table tr:hover {
    background-color: #d1e7dd;
    cursor: pointer;
}

.profile-section {
    display: flex;
    .
```

- Bottom Bar:** Includes tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, PORTS, and a status bar showing "Ln 8, Col 32 Spaces: 4 UTF-8 CRLF CSS".

Figure 52:css png and jpg 2

A screenshot of the Visual Studio Code interface. The left sidebar shows a 'NETWORK PROJECT' folder containing files like 'main\_en.html', 'style.css', 'server.py', and various image files. The main editor area displays the following CSS code:

```
# 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 status bar at the bottom indicates 'Ln 8, Col 32' and '8/13/2024'.

Figure 53:css png and jpg 3

A screenshot of the Visual Studio Code interface, similar to Figure 53. The left sidebar shows a 'NETWORK PROJECT' folder. The main editor area displays the following CSS code:

```
# 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 status bar at the bottom indicates 'Ln 8, Col 32' and '8/13/2024'.

Figure 54:css png and jpg 4

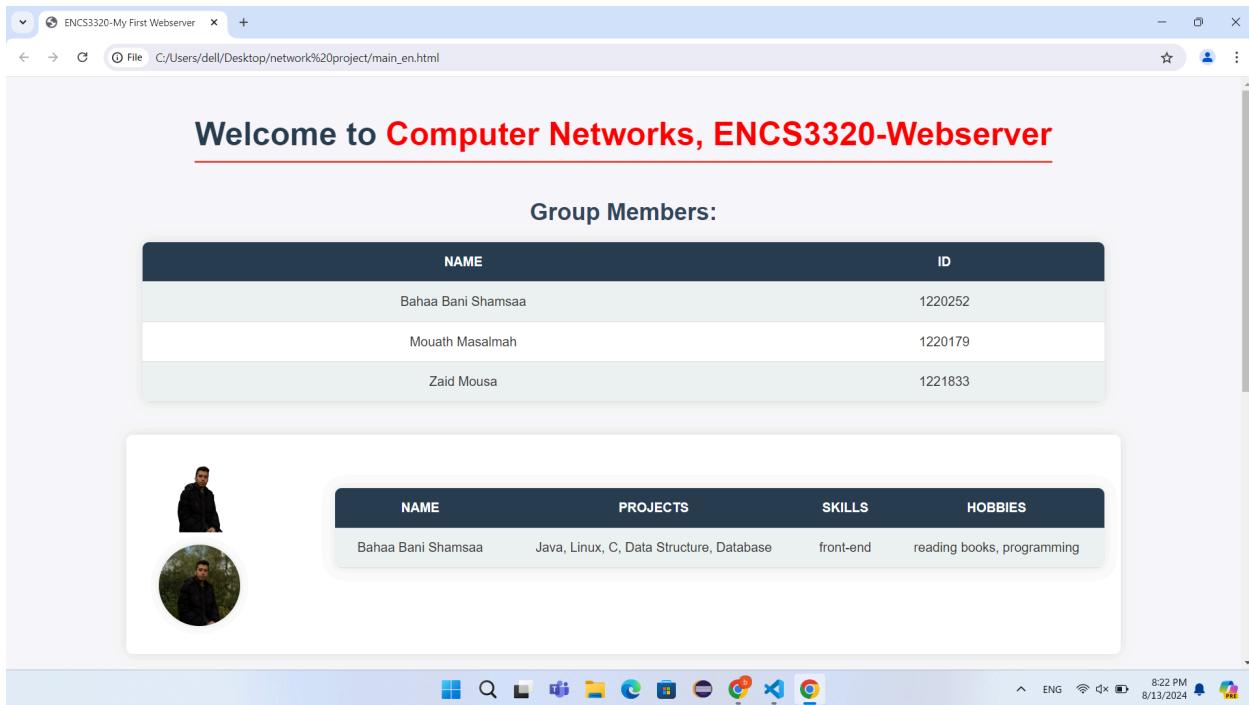


Figure 55:browser for png and jpg 1

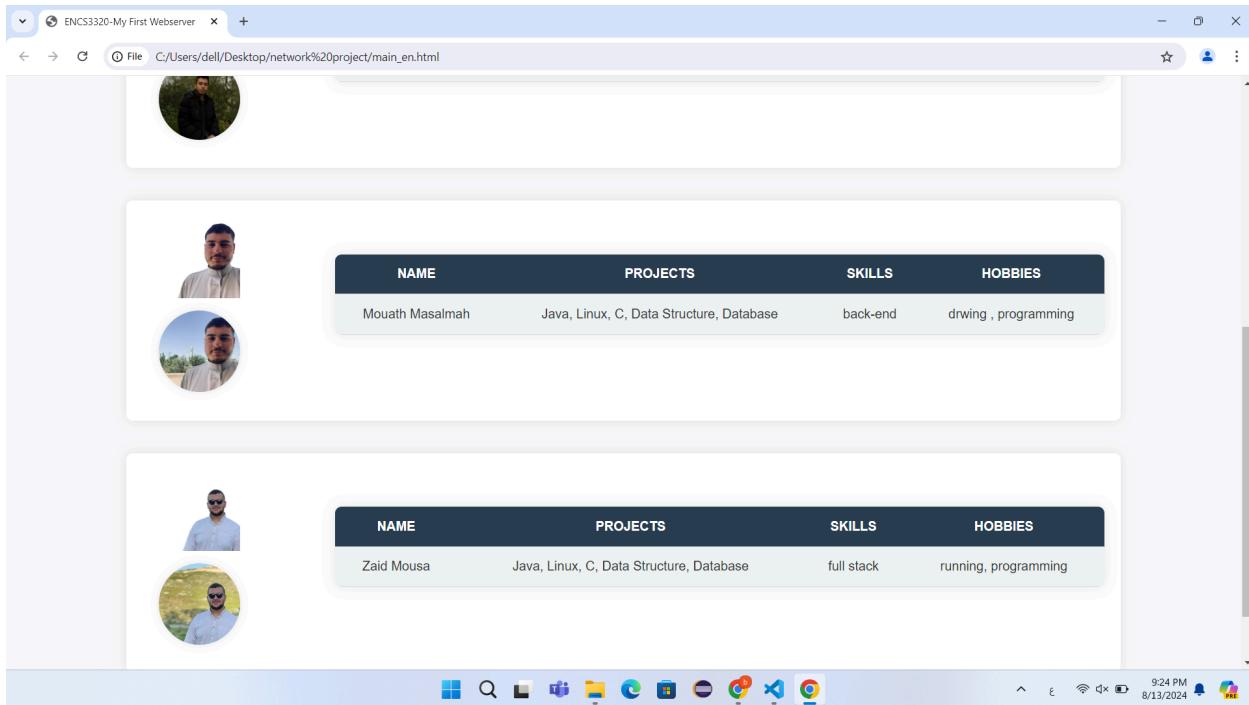
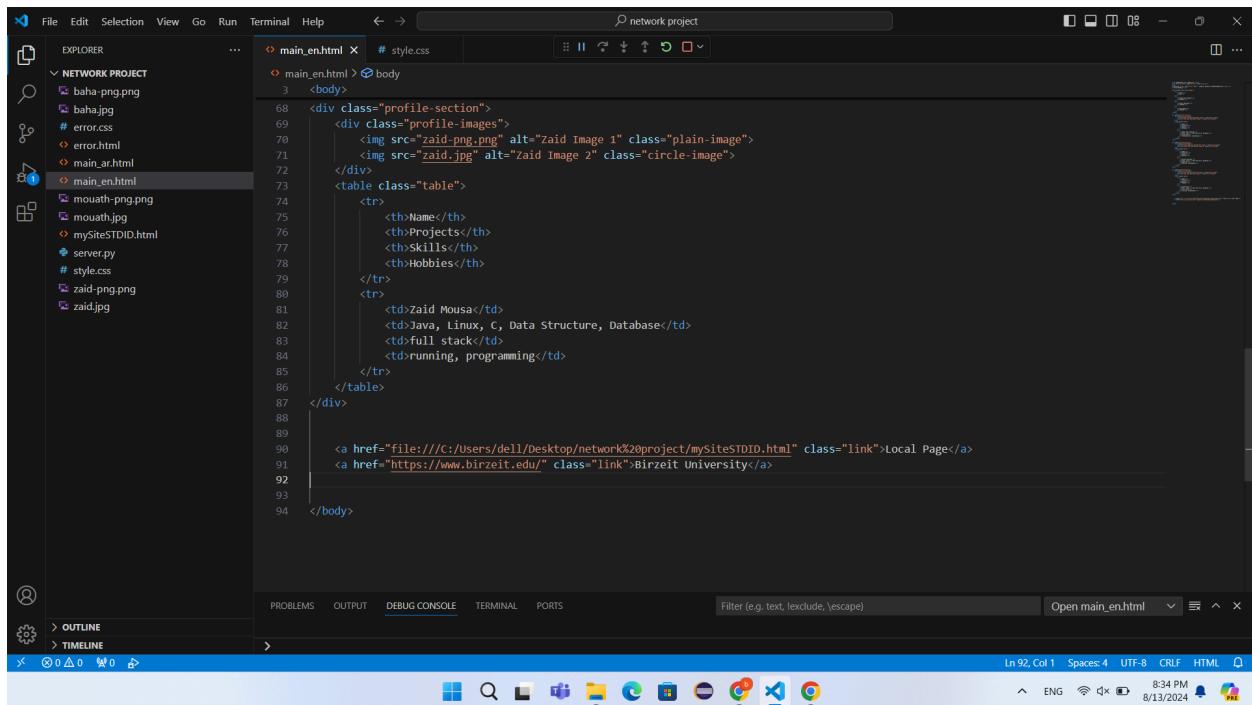


Figure 56:browser for png and jpg 2

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:



The screenshot shows the Visual Studio Code interface with the main\_en.html file open in the editor. The code displays two profiles: Mouath Masalmah and Zaid Mousa, each with a plain image and a circular image. It includes a table with columns for Name, Projects, Skills, and Hobbies. There are links to a local page and Birzeit University.

```
<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/mySiteSTOID.html" class="link">Local Page</a>
<a href="https://www.birzeit.edu/" class="link">Birzeit University</a>
```

Figure 57:link to local html and Birzeit web

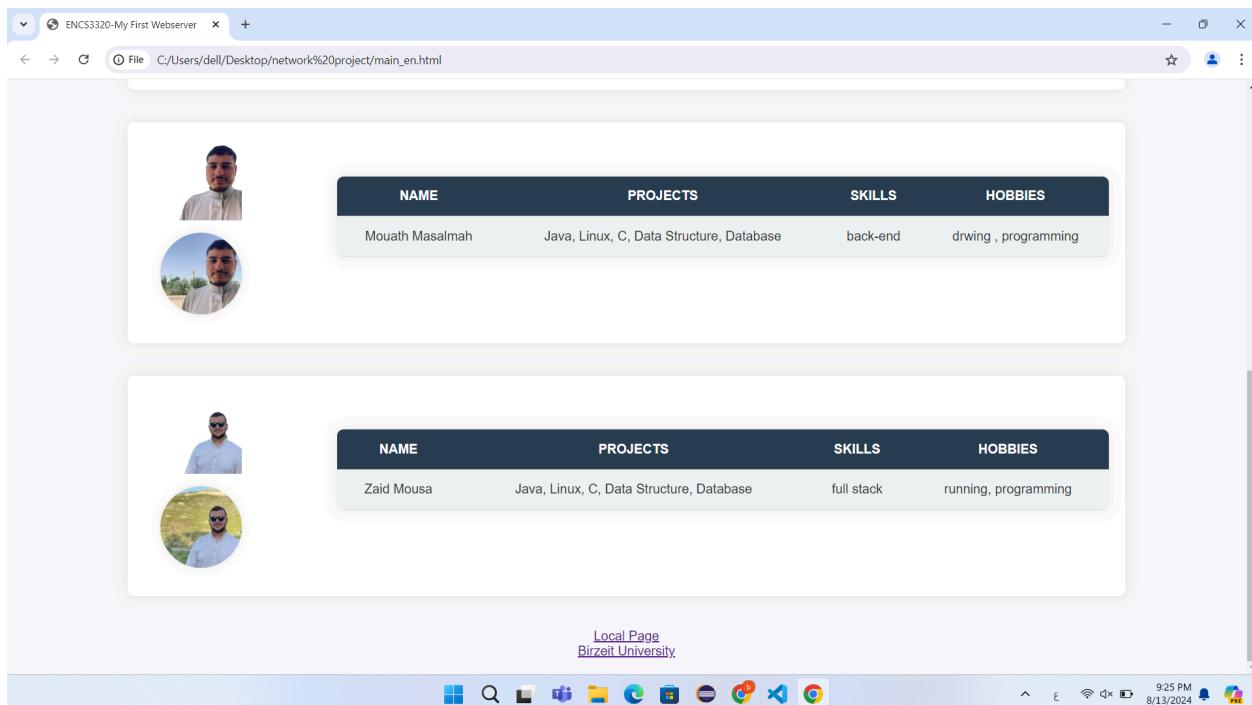


Figure 58:browser for link to local html and Birzeit web

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

2.

The screenshot shows a Microsoft Edge browser window with multiple tabs open. The active tab displays a local webpage at [localhost:1833/ar](http://localhost:1833/ar). The page title is "خادم الويب-ENCS3320، مرحبًا بكم في شبكات الحاسوب". Below the title, there is a section titled "أعضاء المجموعة:" followed by a table showing three members:

الاسم	الرقم الجامعي
بهاء بنى شمسه	1220252
معاذ سالمية	1220179
زيد مؤمني	1221833

Below this table is a larger card containing two profile pictures (one large, one small) and some descriptive text:

الاسم	المشاريع	المهارات	الهوايات
بهاء بنى شمسه	جلا، لينوكس، سبي، بنية المعلومات، قواعد البيانات	واجهة الاعمالية	قراءة الكتب، البرمجة

The browser's taskbar at the bottom shows several pinned icons: Network.docx - مستندات Google, ENCS3320\_Project\_Document.d, (2) Discord | LOBBY | COM, Task3 - Google Drive, ENCS3320-My First Webserver, and a pinned icon for mySiteSTDID.html.

Figure 59: /ar request

The screenshot shows a Microsoft Edge browser window with multiple tabs open. The active tab is titled "localhost:1833/main\_ar.html". The page content is in Arabic. At the top, it says "خادم الويب-ENCS3320، مرحباً بكم في شبكات الحاسوب". Below this is a section titled "أعضاء المجموعة:" (Group Members) with a table:

الاسم	الرقم الجامعي
بهاء بنى شمسه	1220252
معلا مسالمه	1220179
زيد موسى	1221833

Below the table is another table for a specific member:

الاسم	المشروع	المهارات	الهobbies
بهاء بنى شمسه	جاتا ، لينوكس ، سي ، بنية المعلومات ، فوائد البيانات	واجهة الامامية	قراءة الكتب ، البرمجة

On the left side of the main content area, there are two circular profile pictures.

Figure 60: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.

The screenshot shows a Microsoft Edge browser window with multiple tabs open. The active tab is titled "localhost:1833/main\_en.html". The page content is in English. At the top, it says "Welcome to Computer Networks, ENCS3320-Webserver". Below this is a section titled "Group Members:" with a table:

NAME	ID
Bahaa Bani Shamsaa	1220252
Mouath Masalmah	1220179
Zaid Mousa	1221833

Below the table is another table for a specific member:

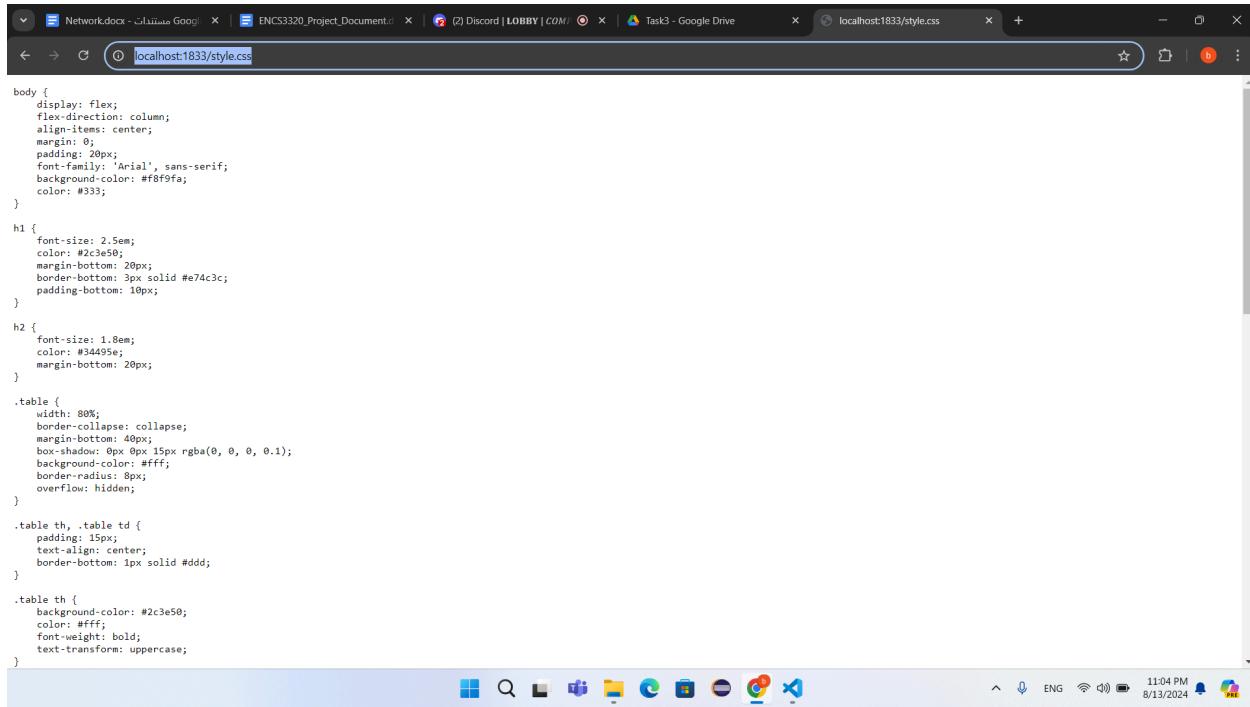
NAME	PROJECTS	SKILLS	HOBBIES
Bahaa Bani Shamsaa	Java, Linux, C, Data Structure, Database	front-end	reading books, programming

On the left side of the main content area, there are two circular profile pictures.

Figure 61: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](http://localhost:1833/main_en.html) and it shows the contents of main\_en.html file

#### 4.



```
body {
    display: flex;
    flex-direction: column;
    align-items: center;
    margin: 0;
    padding: 20px;
    font-family: 'Arial', sans-serif;
    background-color: #f8f9fa;
    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;
}

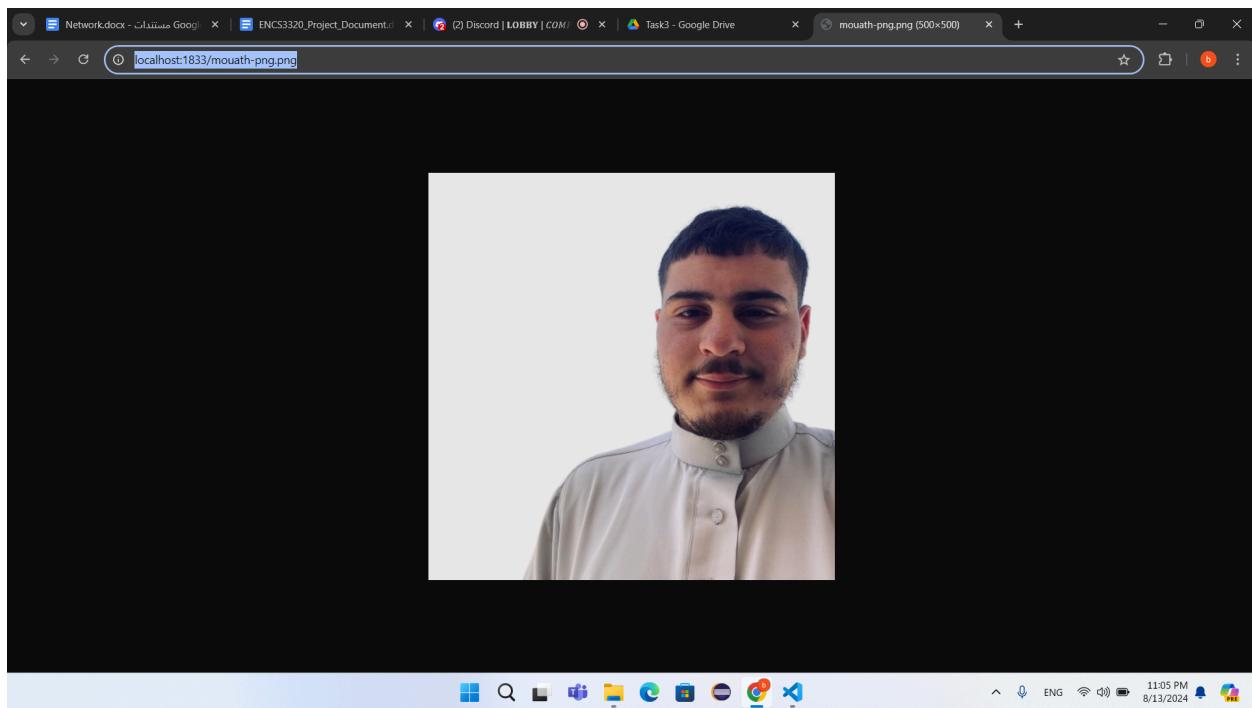
.table th, .table td {
    padding: 15px;
    text-align: center;
    border-bottom: 1px solid #ddd;
}

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

Figure 62: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 63:mouath.png request*

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

## 6.



*Figure 64: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:

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, mouth-png.png, mouth.jpg, mySiteSTDID.html, server.py, #style.css, zaid-png.png, and zaid.jpg.
- Code Editor:** The active file is `server.py`, which contains Python code for a socket-based web server. It uses `socket` to handle connections and send HTTP responses. It includes logic to serve files like `index.html`, `main_en.html`, and `main_ar.html`.
- Terminal:** Shows the command `network project`.
- Status Bar:** Displays "Ln 105, Col 33" and "Python 3.9.13 64-bit".

```

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

```

Figure 65:server 1

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, mouth-png.png, mouth.jpg, mySiteSTDID.html, server.py, #style.css, zaid-png.png, and zaid.jpg.
- Code Editor:** The active file is `server.py`, which contains Python code for a socket-based web server. It includes logic to handle image requests using the `get_image?` query parameter.
- Terminal:** Shows the command `network project`.
- Status Bar:** Displays "Ln 105, Col 33" and "Python 3.9.13 64-bit".

```

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

```

Figure 66:server 2

The screenshot shows the Microsoft 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, mouth-png.png, mouth.jpg, mySiteSTDID.html, server.py, style.css, zaid-png.png, and zaid.jpg.
- Code Editor:** Displays the content of `server.py`. The code handles file requests based on their extensions (.jpg, .png, .html, .css) and sends appropriate HTTP responses. It also handles a "so" request by performing a temporary redirect to `https://stacko`.
- Terminal:** Shows the command `Ln 105, Col 33 Spaces:4 UTF-8 CRLF`.
- Status Bar:** Includes icons for Python Debug, Python Deb..., and Python Deb... along with system status like ENG, 11:53 AM, and 8/15/2024.

```

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             if name.endswith(".jpg"):
64                 connectionSocket.send(b'HTTP/1.1 200 OK\nContent-Type: image/jpg\n\n')
65             elif name.endswith(".png"):
66                 connectionSocket.send(b'HTTP/1.1 200 OK\nContent-Type: image/png\n\n')
67             connectionSocket.send(content)
68         else:
69             |     raise FileNotFoundError
70
71     elif file.endswith(".jpg"):
72         content = open_file(file)
73         if content:
74             connectionSocket.send(b'HTTP/1.1 200 OK\nContent-Type: image/jpg\n\n')
75             connectionSocket.send(content)
76         else:
77             |     raise FileNotFoundError
78
79     elif file.endswith(".png"):
80         content = open_file(file)
81         if content:
82             connectionSocket.send(b'HTTP/1.1 200 OK\nContent-Type: image/png\n\n')
83             connectionSocket.send(content)
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

```

Figure 67:server 3

This screenshot shows the same Visual Studio Code environment as Figure 67, but with additional code visible in the editor. The code now includes exception handling for errors and a detailed response for 404 Not Found errors.

```

80         content = open_file(file)
81         if content:
82             connectionSocket.send(b'HTTP/1.1 200 OK\nContent-Type: image/png\n\n')
83             connectionSocket.send(content)
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     except Exception as e:
97         print(f"Error: {e}")
98         connectionSocket.send(b'HTTP/1.1 404 Not Found\nContent-Type: text/html\n\n')
99         error_content = open_file('error.html')
100         if error_content:
101             connectionSocket.send(error_content + f"<p> ip: {ip} port: {port}</p>".encode())
102         else:
103             connectionSocket.send(b'<html><body><h1>404 Not Found</h1><p>File not found.</p></body></html>')
104
105 finally:
106     connectionSocket.close()

```

Figure 68:server 4

The screenshot shows a code editor interface with a dark theme. On the left is the Explorer sidebar showing a 'NETWORK PROJECT' folder containing files like 'baha.png', 'baha.jpg', 'error.css', 'error.html', 'main\_ar.html', 'main\_en.html', 'mouauth.png', 'mouauth.jpg', 'server.py', '# style.css', 'zaid.png', and 'zaid.jpg'. The main editor area displays the content of 'mySiteSTDID.html':

```
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>
```

Below the editor are tabs for 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL', and 'PORTS'. The 'TERMINAL' tab is selected. At the bottom, there's a toolbar with icons for file operations and a status bar showing 'Ln 13, Col 8' and 'Python Deb...'. The system tray at the very bottom shows icons for battery, signal, and date/time.

Figure 69:mySiteSTDID.html

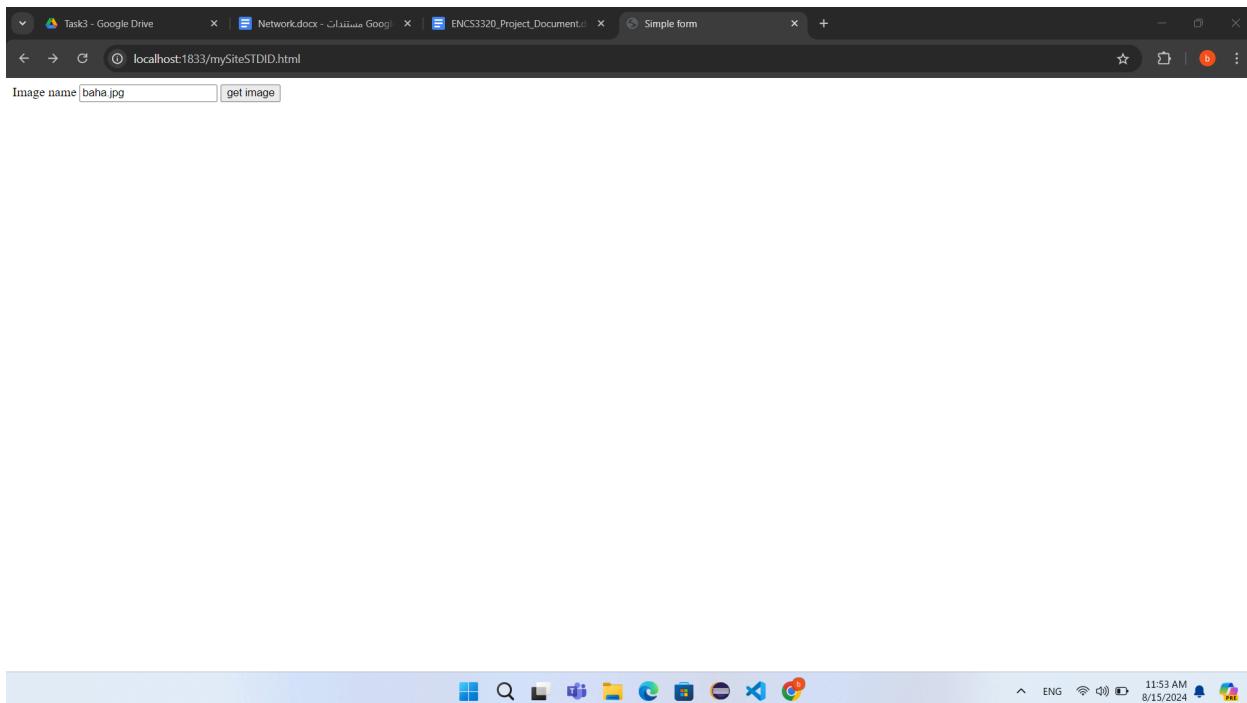


Figure 70:browesr for mySiteSTDID.html

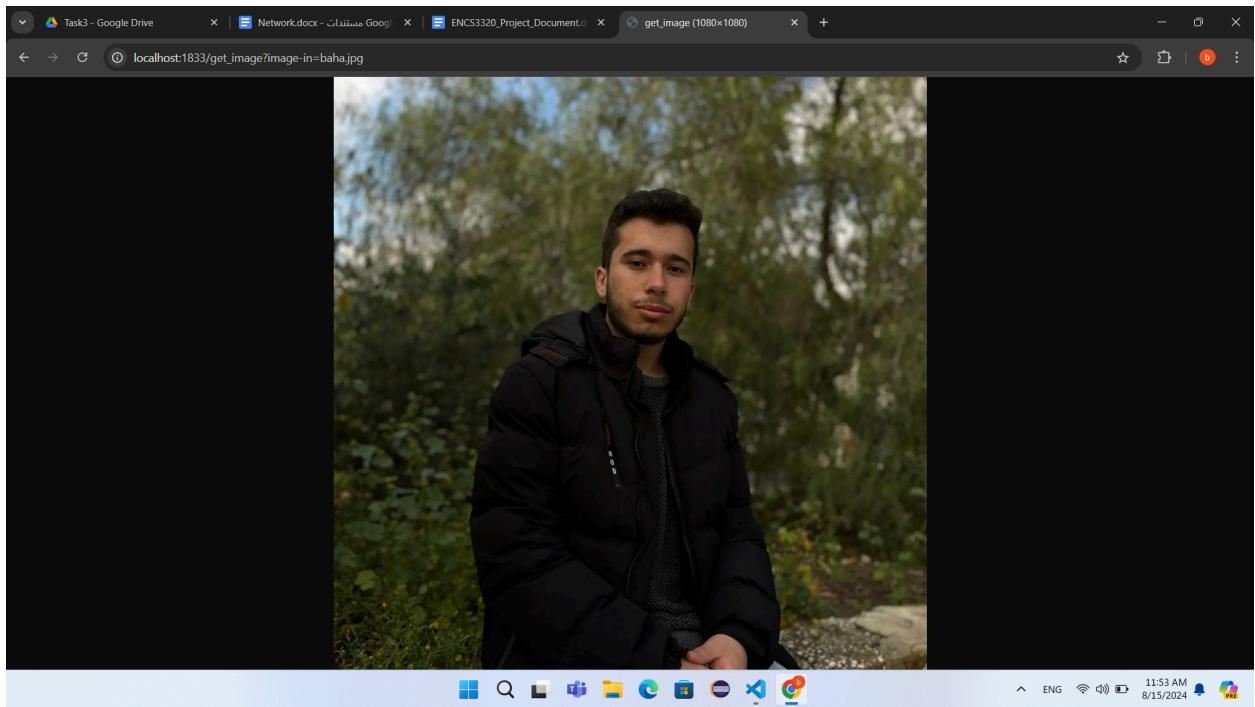


Figure 71:image browser for mySiteSTDID.html

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.

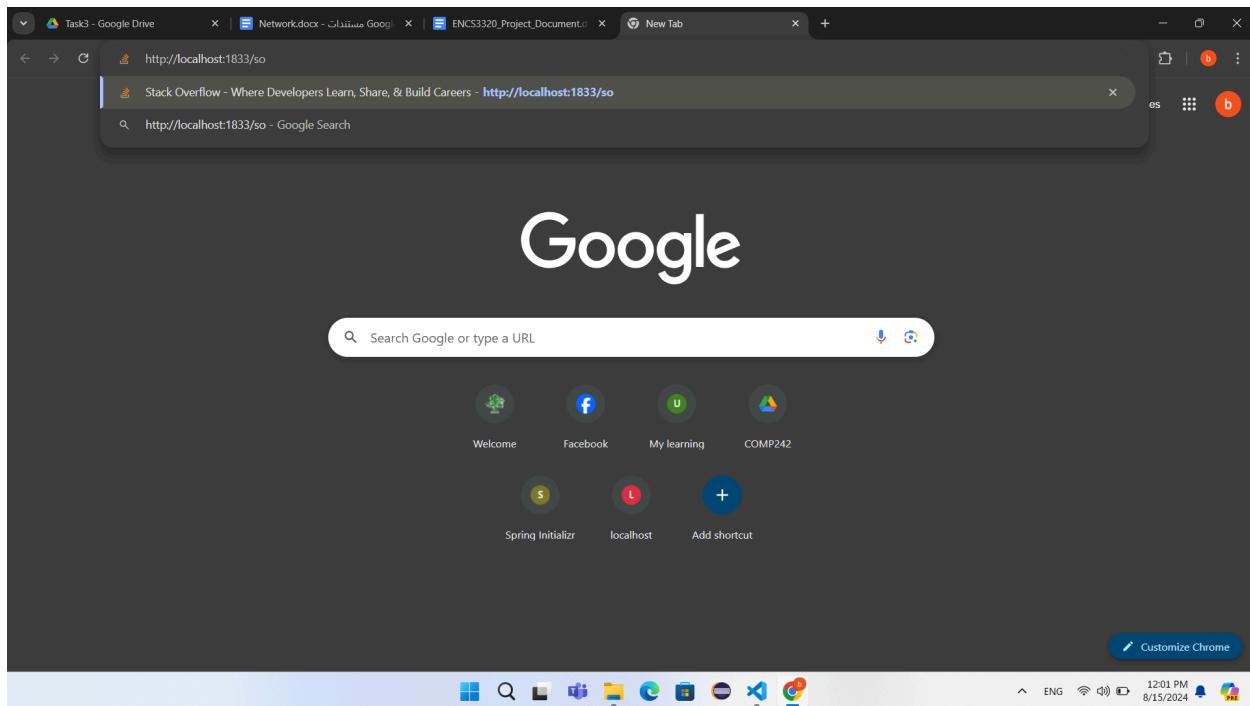


Figure 72: /so request

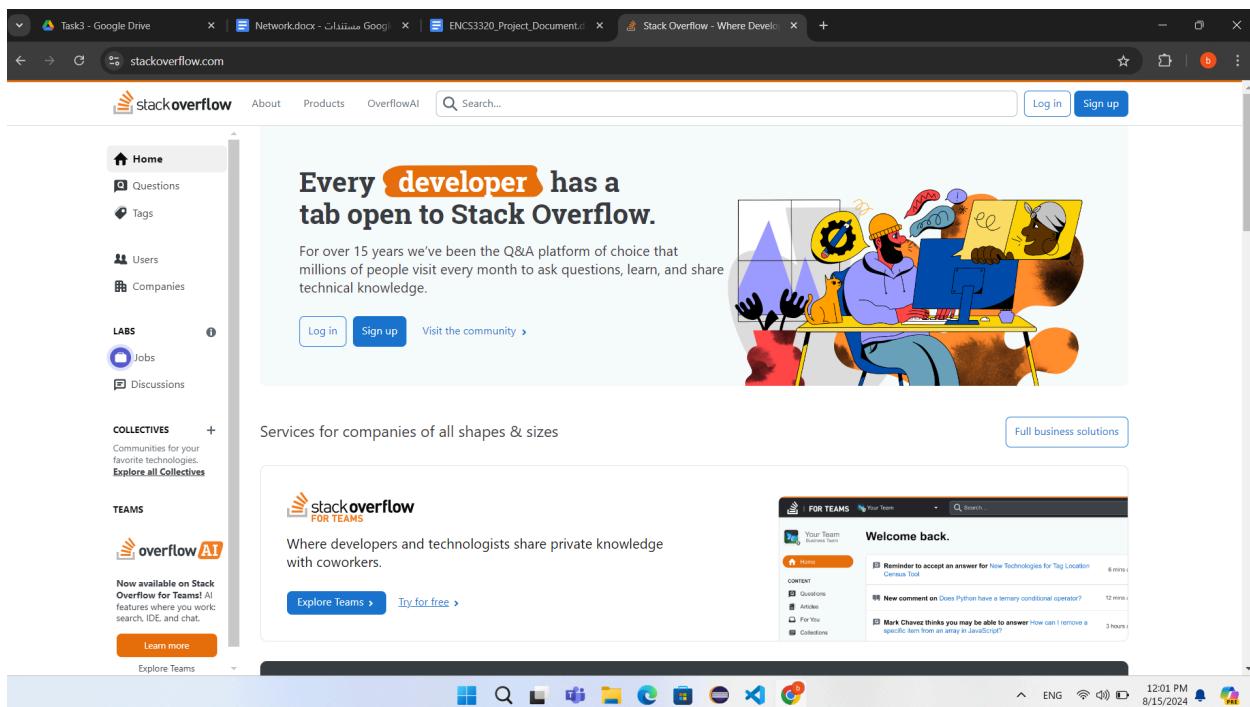


Figure 73:solution for /so

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

b.

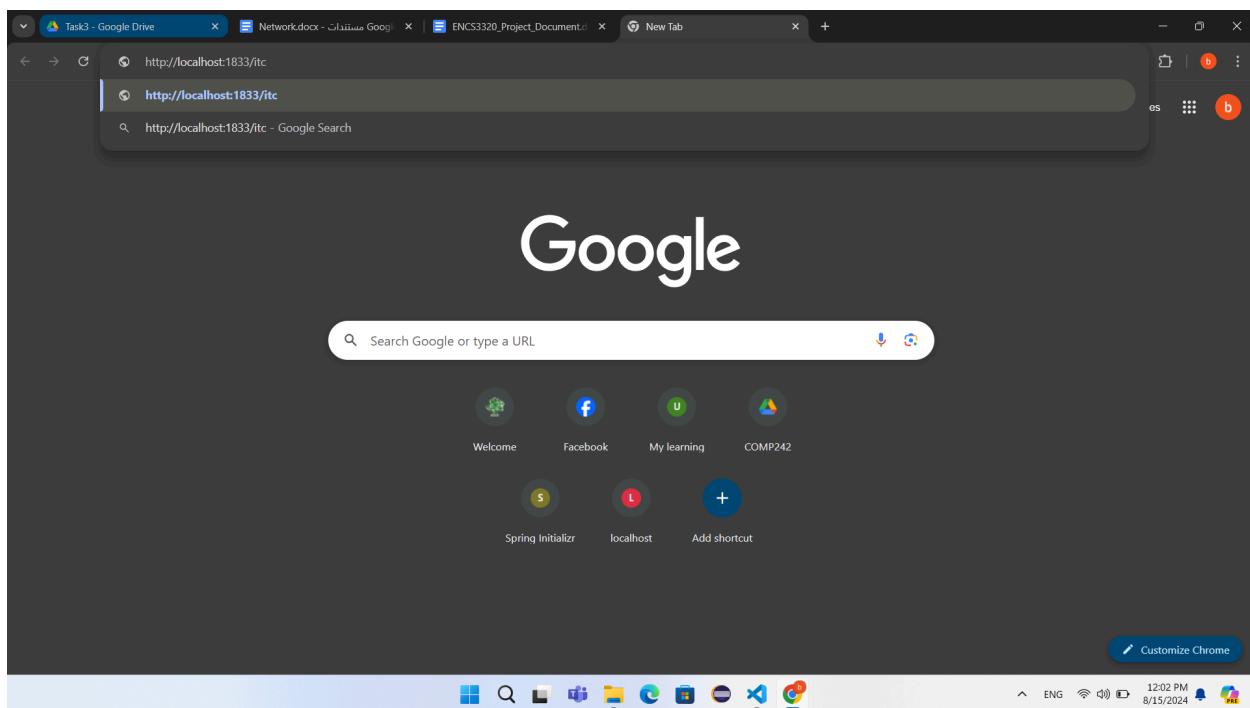
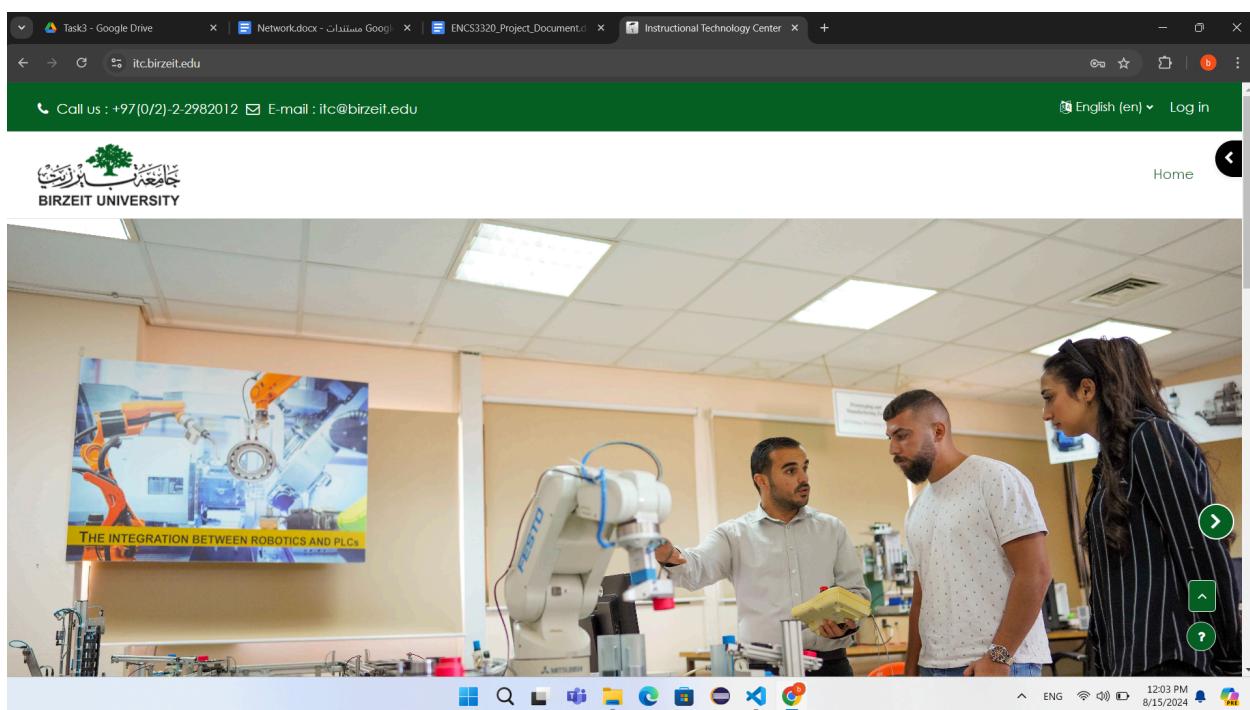


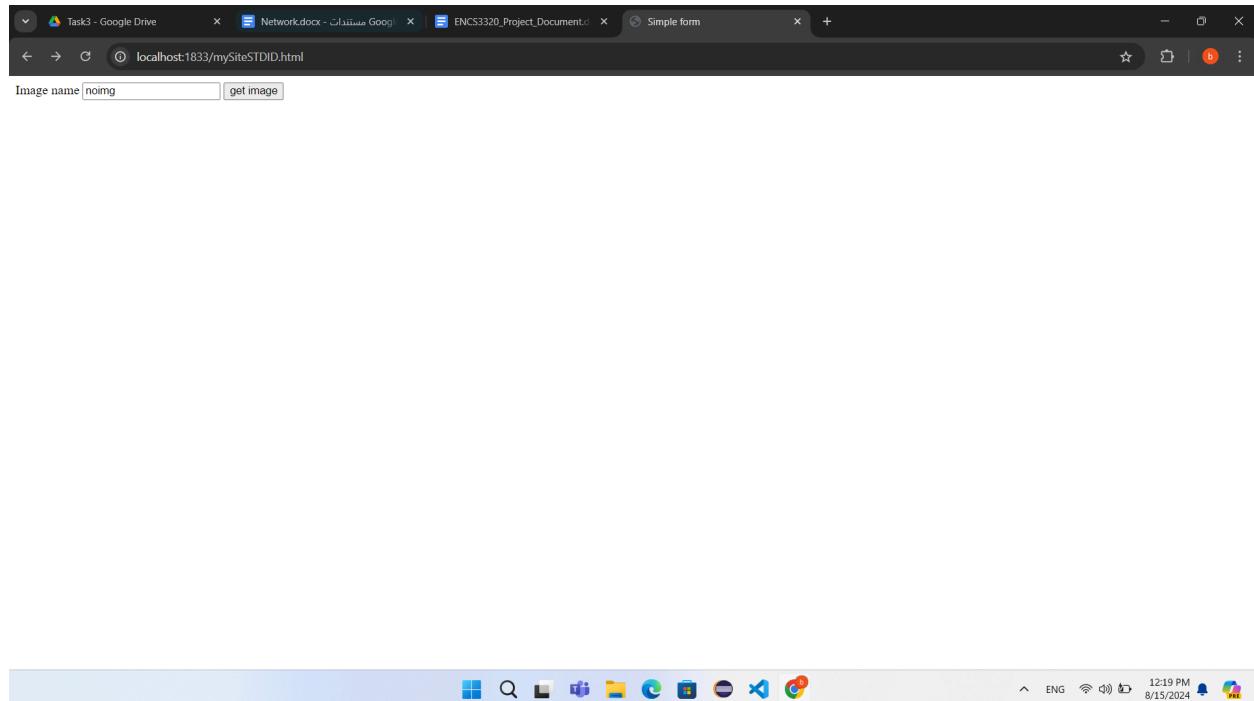
Figure 74: /itc request



*Figure 75: solution for /itc*

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

**10.**



*Figure 76:wrong image*

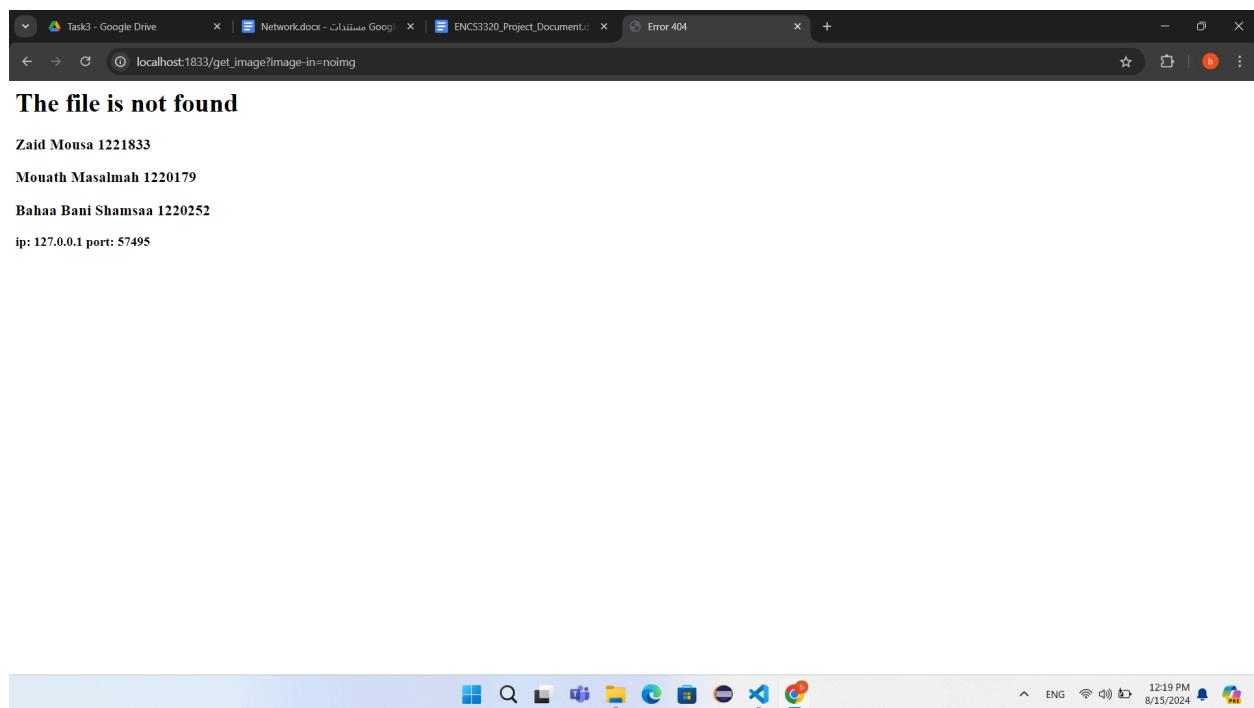


Figure 77: error.html browser

Here when we search for an image does not exist the error.html page shows  
11.

```

File "c:\Users\dell\Desktop\project\server.py", line 24, in <module>
    s = split[1].split("/")
IndexError: list index out of range

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "c:\Users\dell\Desktop\project\server.py", line 101, in <module>
    connectionSocket.send(error_content + f"<p> ip: {ip} port: {port} </p>".encode())
connectionaborted: [WinError 10053] An established connection was aborted by the software in your host machine
PS C:\Users\dell\Desktop\project>

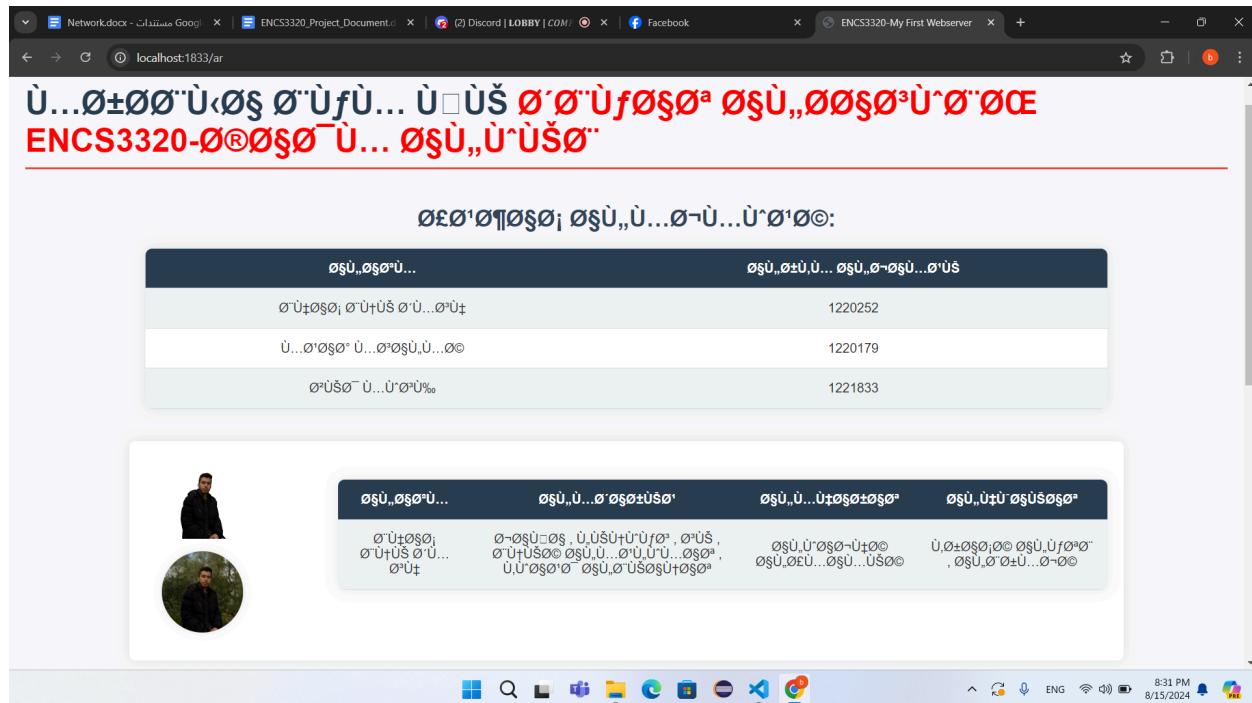
```

Figure 78: 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:

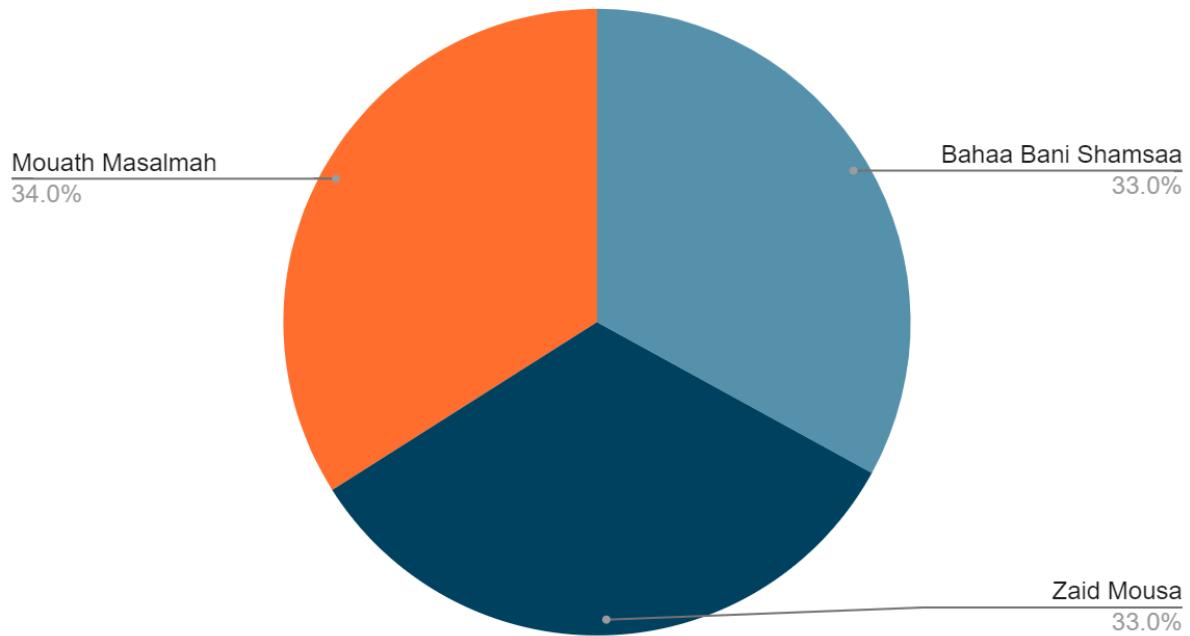


*Figure 79: problem on /ar/*

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.>