



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

Computer Networks ~ ENCS2110

Project 1 (Socket Programming ~ REPORT)

Instructor: Dr. Mohammad Jubran

Prepared by:

Roaa Hamoudah – 1210738 (Section: 5)

Ameer Yasen – 1210187 (Section: 5)

Ata Musleh – 1211245 (Section: 5)

Date: 01/12/2024

Contents

Table of figures:	Error! Bookmark not defined.
1 Part 1 Results	6
Task 1 – Network Commands and Wireshark:	6
(i) ipconfig.....	6
(ii) Ping:.....	6
(iii) Tracert:	6
(V) nslookup:	6
b) Ensure your computer is connected to the internet, then perform the following actions:.....	7
1) IP address:.....	8
2) Subnet Mask:	8
3) Default Gateway:.....	9
4) Domain Name System (DNS) server addresses:	9
Abstract:	19
2.1 Some screenshots of the web server running:.....	20
2.1.1 main_en.html page.....	20
2.1.2 main_ar.html page	23
2.1.3 supporting_material_en.html Page.....	27
2.1.3.1 Request an existing image	27
2.1.3.2 Request a non-existing image	28
HTTP status code 307 Temporary Redirect indicates that the resource being requested has been temporarily moved to a different URL. The client should resubmit the original request to the new URL provided in the Location header of the response.	29
2.1.3.3 Request an existing video	30
2.1.3.4 Request a non-existing video	31
2.1.4 supporting_material_ar.html Page.....	32

2.1.5 HTTP Requests printed one the server terminal	33
2.1.9 Links to external resources:	36
3 Part3 UDP Client-Server Trivia Game	42
3.1 Theory and Procedure	42
3.2 Server Implementation:	43
3.2.1 Server.py	45
3.2.2 client.py	54
3.2.3 live game experience	55
4 Alternative Solutions, Issues, and Limitations	60
4.1 Alternative Solutions	60
4.2 Issues and Challenges	60
4.2.1 Web Server	60
4.2.2 Trivia Game	60
4.3 Limitations	61

Table of figures:

Figure 1 IPConfig	6
Figure 2 IP address.....	8
Figure 3 Subnet Mask.....	8
Figure 4 Default Gateway	9
Figure 5) Domain Name System (DNS)	9
Figure 6 local network device Ping	10
Figure 7 Ping discover.engineering.utoronto.ca.....	11
Figure 8 tracert on discover.engineering.utoronto.ca	13
Figure 9 nslookup to retrieve the DNS	14
Figure 10 connect with telnet to discover.engineering.utoronto.ca ...	15
Figure 11 Use the Wireshark packet analyzer to capture a DNS query	17
Figure 12 main_en.html page.....	20
Figure 13 main_en.html page network topic.....	21
Figure 14 UDP and TCP Applications.....	22
Figure 15 main_en.html page extra resources links	23
Figure 16 main_er.html Page	23
Figure 17 supporting_material_en.html Page.....	27
Figure 18 Request an existing image	27
Figure 19 Request a non-existing image.....	28
Figure 20 HTTP status code 307 Temporary Redirect.....	29
Figure 21 Request an existing video	30
Figure 22 Request a non-existing video.....	31
Figure 23 supporting_material_ar.html Page	32
Figure 24 HTTP Requests printed one the server terminal.....	33
Figure 25 2.1.6.....	35
Figure 26 Error 404 HTML page	37
Figure 27 The status line "HTTP/1.1 404 Not Found"	38
Figure 28 view from external device.....	39
Figure 29 Server.py source code.....	45
Figure 30 server.py source code 2	49
Figure 31 server.py source code 3	53

Figure 32 Client.py source code	54
Figure 33 server terminal game output	57
Figure 34 Client terminal output	57

1 Part 1 Results

Task 1 – Network Commands and Wireshark

a): In your own words, provide a brief (two-sentence maximum) explanation of each of the following commands: (i) ipconfig, (ii) ping, (iii) tracert, (iv) telnet, and (v) nslookup.

(i) ipconfig: (Internet Protocol Configuration) A command line utility that is used to display and manage the IP address assigned to the machine. In Windows, typing ipconfig without any parameters displays the computer's currently assigned IP, subnet mask, and default gateway addresses.

(ii) Ping: a utility used diagnostically to ensure that a host computer the user is trying to reach is operating and measures the round-trip time (RTT) between devices by sending packets and recording the time taken for their return. Typically, it sends multiple packets to ensure accurate measurements.

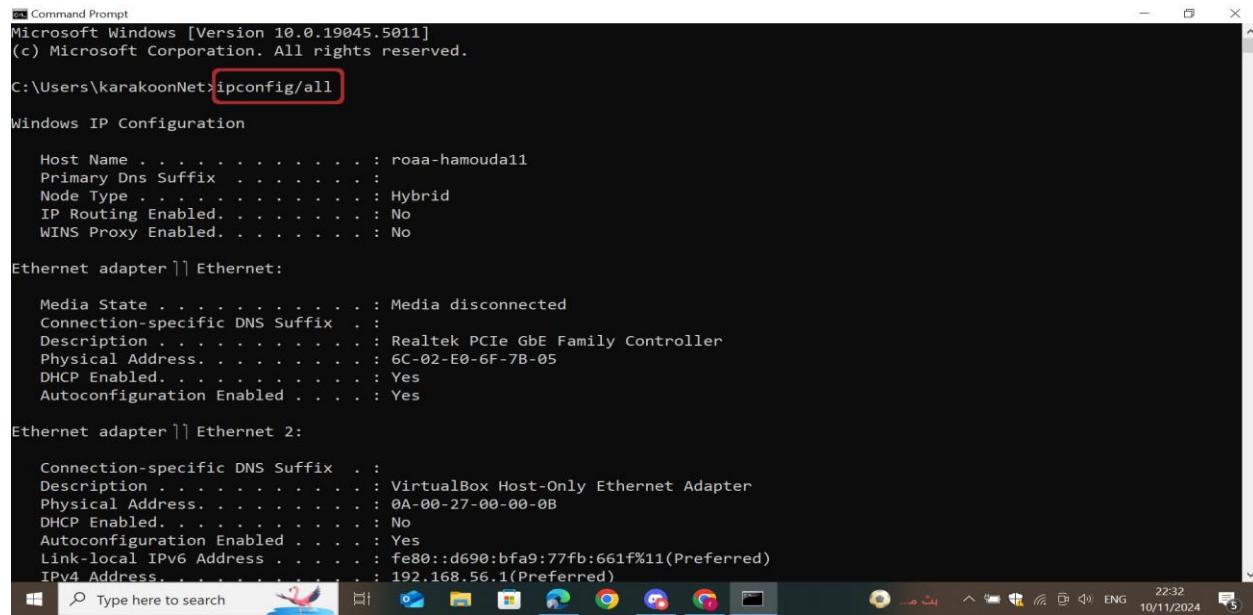
(iii) Tracert: also known as traceroute, traces and maps the journey of data packets from a source to a destination, revealing each intermediate hop and the time taken for the packets to reach each stop along the way by incrementing the TTL (time to live) from source to destination. It also enables to locate where the data was unable to be sent along, known as points of failure.

(iv) Telnet: a client/server application protocol that provides access to virtual terminals of remote systems on local area networks or the Internet.

(V) nslookup: short for nameserver lookup, is a tool used to find the corresponding IP address or domain name system (DNS) record for a given hostname. This command also supports reverse DNS lookups by inputting the IP addresses of the domains to be looked up. (Essentially, Nslookup facilitates the translation between hostnames and IP addresses in a network environment.

b) Ensure your computer is connected to the internet, then perform the following actions:

1.b.1 Run the ipconfig /all command on your computer and identify the IP address, subnet mask, default gateway, and Domain Name System (DNS) server addresses for your primary network interface.



```
Microsoft Windows [Version 10.0.19045.5011]
(c) Microsoft Corporation. All rights reserved.

C:\Users\karakoonNet>ipconfig/all

Windows IP Configuration

Host Name . . . . . : roaa-hamouda11
Primary Dns Suffix . . . . . :
Node Type . . . . . : Hybrid
IP Routing Enabled. . . . . : No
WINS Proxy Enabled. . . . . : No

Ethernet adapter ]| Ethernet:

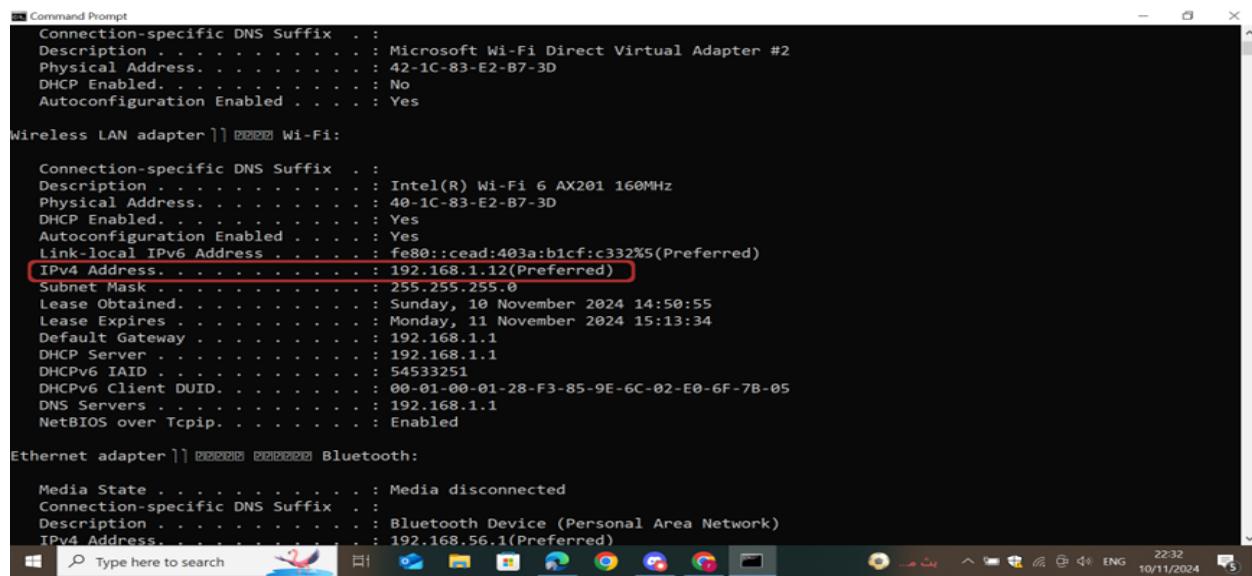
Media State . . . . . : Media disconnected
Connection-specific DNS Suffix . . . . . :
Description . . . . . : Realtek PCIe GbE Family Controller
Physical Address. . . . . : 6C-02-E0-6F-7B-05
DHCP Enabled. . . . . : Yes
Autoconfiguration Enabled . . . . . : Yes

Ethernet adapter ]| Ethernet 2:

Connection-specific DNS Suffix . . . . . :
Description . . . . . : VirtualBox Host-Only Ethernet Adapter
Physical Address. . . . . : 0A-00-27-00-00-0B
DHCP Enabled. . . . . : No
Autoconfiguration Enabled . . . . . : Yes
Link-local IPv6 Address . . . . . : fe80::d690:bfa9:77fb:661f%11(Preferred)
IPv4 Address. . . . . : 192.168.56.1(Preferred)
```

Figure 1 ipconfig

1) IP address:



```
Command Prompt
Connection-specific DNS Suffix . .
Description . . . . . : Microsoft Wi-Fi Direct Virtual Adapter #2
Physical Address. . . . . : 42-1C-83-E2-B7-3D
DHCP Enabled. . . . . : No
Autoconfiguration Enabled . . . . . : Yes

Wireless LAN adapter Wi-Fi:

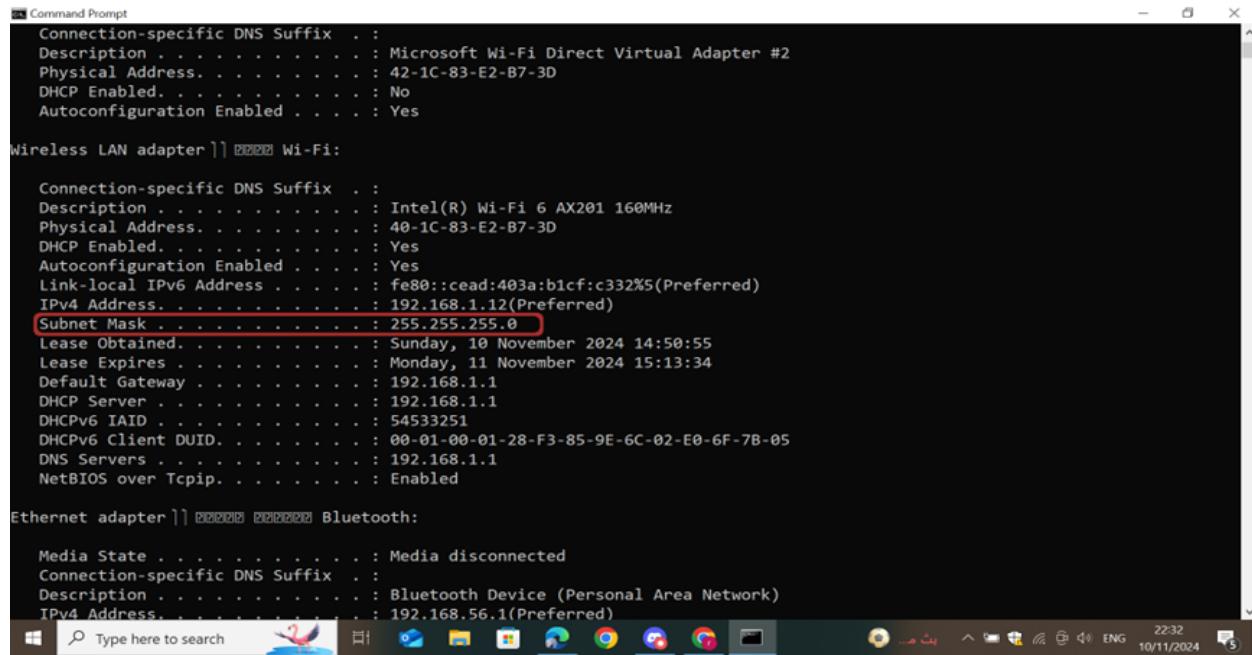
Connection-specific DNS Suffix . .
Description . . . . . : Intel(R) Wi-Fi 6 AX201 160MHz
Physical Address. . . . . : 40-1C-83-E2-B7-3D
DHCP Enabled. . . . . : Yes
Autoconfiguration Enabled . . . . . : Yes
Link-local IPv6 Address . . . . . : fe80::cead:403a:b1cf:c332%5(PREFERRED)
IPv4 Address. . . . . : 192.168.1.12(PREFERRED)
Subnet Mask . . . . . : 255.255.255.0
Lease Obtained. . . . . : Sunday, 10 November 2024 14:50:55
Lease Expires . . . . . : Monday, 11 November 2024 15:13:34
Default Gateway . . . . . : 192.168.1.1
DHCP Server . . . . . : 192.168.1.1
DHCPv6 IAID . . . . . : 54533251
DHCPv6 Client DUID. . . . . : 00-01-00-01-28-F3-85-9E-6C-02-E0-6F-7B-05
DNS Servers . . . . . : 192.168.1.1
NetBIOS over Tcpip. . . . . : Enabled

Ethernet adapter Bluetooth:

Media State . . . . . : Media disconnected
Connection-specific DNS Suffix . .
Description . . . . . : Bluetooth Device (Personal Area Network)
IPv4 Address. . . . . : 192.168.56.1(PREFERRED)
```

Figure 2 IP address

2) Subnet Mask:



```
Command Prompt
Connection-specific DNS Suffix . .
Description . . . . . : Microsoft Wi-Fi Direct Virtual Adapter #2
Physical Address. . . . . : 42-1C-83-E2-B7-3D
DHCP Enabled. . . . . : No
Autoconfiguration Enabled . . . . . : Yes

Wireless LAN adapter Wi-Fi:

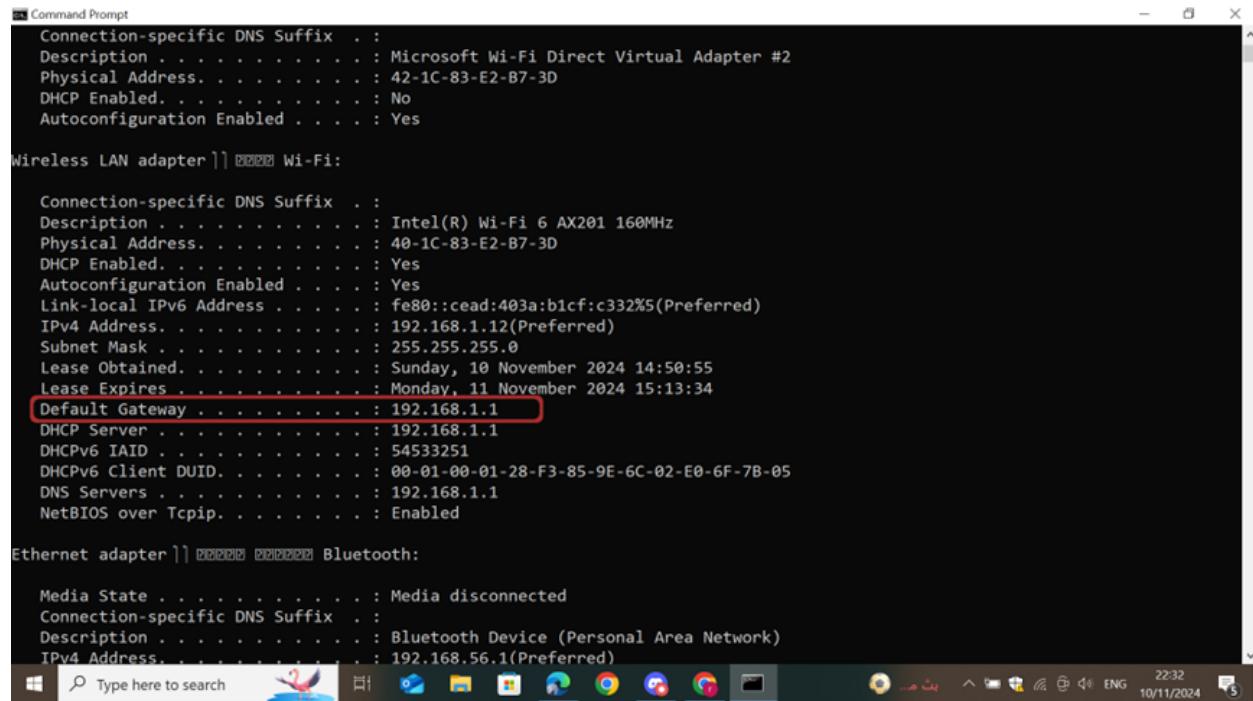
Connection-specific DNS Suffix . .
Description . . . . . : Intel(R) Wi-Fi 6 AX201 160MHz
Physical Address. . . . . : 40-1C-83-E2-B7-3D
DHCP Enabled. . . . . : Yes
Autoconfiguration Enabled . . . . . : Yes
Link-local IPv6 Address . . . . . : fe80::cead:403a:b1cf:c332%5(PREFERRED)
IPv4 Address. . . . . : 192.168.1.12(PREFERRED)
Subnet Mask . . . . . : 255.255.255.0
Lease Obtained. . . . . : Sunday, 10 November 2024 14:50:55
Lease Expires . . . . . : Monday, 11 November 2024 15:13:34
Default Gateway . . . . . : 192.168.1.1
DHCP Server . . . . . : 192.168.1.1
DHCPv6 IAID . . . . . : 54533251
DHCPv6 Client DUID. . . . . : 00-01-00-01-28-F3-85-9E-6C-02-E0-6F-7B-05
DNS Servers . . . . . : 192.168.1.1
NetBIOS over Tcpip. . . . . : Enabled

Ethernet adapter Bluetooth:

Media State . . . . . : Media disconnected
Connection-specific DNS Suffix . .
Description . . . . . : Bluetooth Device (Personal Area Network)
IPv4 Address. . . . . : 192.168.56.1(PREFERRED)
```

Figure 3 Subnet Mask

3) Default Gateway:



```
Command Prompt
Connection-specific DNS Suffix . :
Description . . . . . : Microsoft Wi-Fi Direct Virtual Adapter #2
Physical Address. . . . . : 42-1C-83-E2-B7-3D
DHCP Enabled. . . . . : No
Autoconfiguration Enabled . . . . . : Yes

Wireless LAN adapter Wi-Fi:

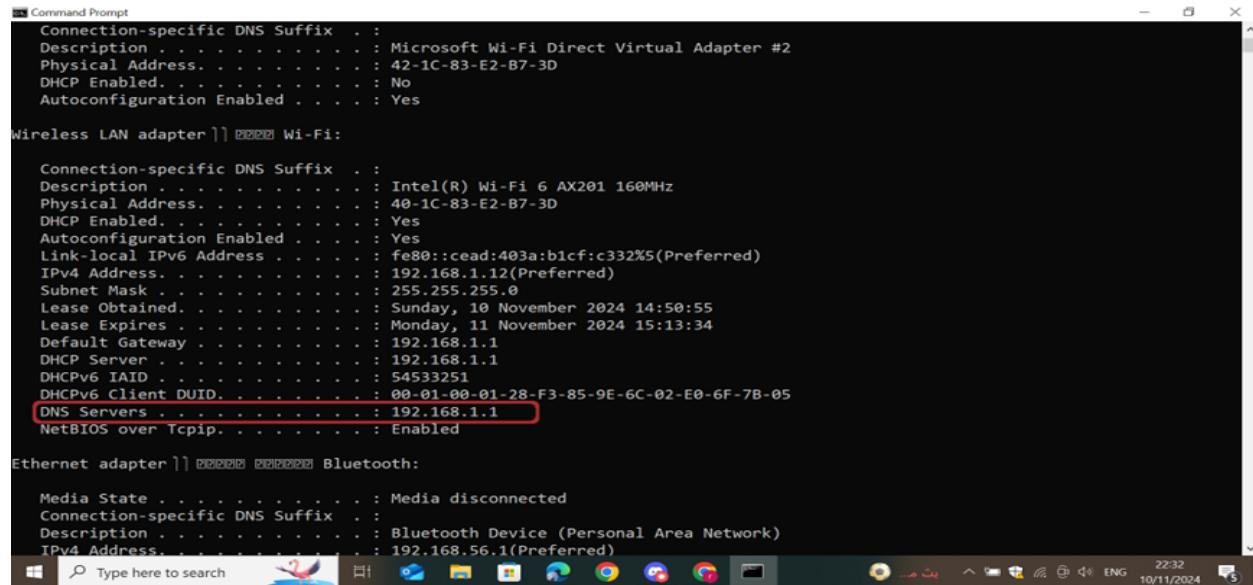
Connection-specific DNS Suffix . :
Description . . . . . : Intel(R) Wi-Fi 6 AX201 160MHz
Physical Address. . . . . : 40-1C-83-E2-B7-3D
DHCP Enabled. . . . . : Yes
Autoconfiguration Enabled . . . . . : Yes
Link-local IPv6 Address . . . . . : fe80::cead:403a:b1cf:c332%5(Preferred)
IPv4 Address. . . . . : 192.168.1.12(Preferred)
Subnet Mask . . . . . : 255.255.255.0
Lease Obtained. . . . . : Sunday, 10 November 2024 14:50:55
Lease Expires . . . . . : Monday, 11 November 2024 15:13:34
Default Gateway . . . . . : 192.168.1.1
DHCP Server . . . . . : 192.168.1.1
DHCPv6 IAID . . . . . : 54533251
DHCPv6 Client DUID. . . . . : 00-01-00-01-28-F3-85-9E-6C-02-E0-6F-7B-05
DNS Servers . . . . . : 192.168.1.1
NetBIOS over Tcpip. . . . . : Enabled

Ethernet adapter Bluetooth:

Media State . . . . . : Media disconnected
Connection-specific DNS Suffix . :
Description . . . . . : Bluetooth Device (Personal Area Network)
IPv4 Address. . . . . : 192.168.56.1(Preferred)
```

Figure 4 Default Gateway

4) Domain Name System (DNS) server addresses:



```
Command Prompt
Connection-specific DNS Suffix . :
Description . . . . . : Microsoft Wi-Fi Direct Virtual Adapter #2
Physical Address. . . . . : 42-1C-83-E2-B7-3D
DHCP Enabled. . . . . : No
Autoconfiguration Enabled . . . . . : Yes

Wireless LAN adapter Wi-Fi:

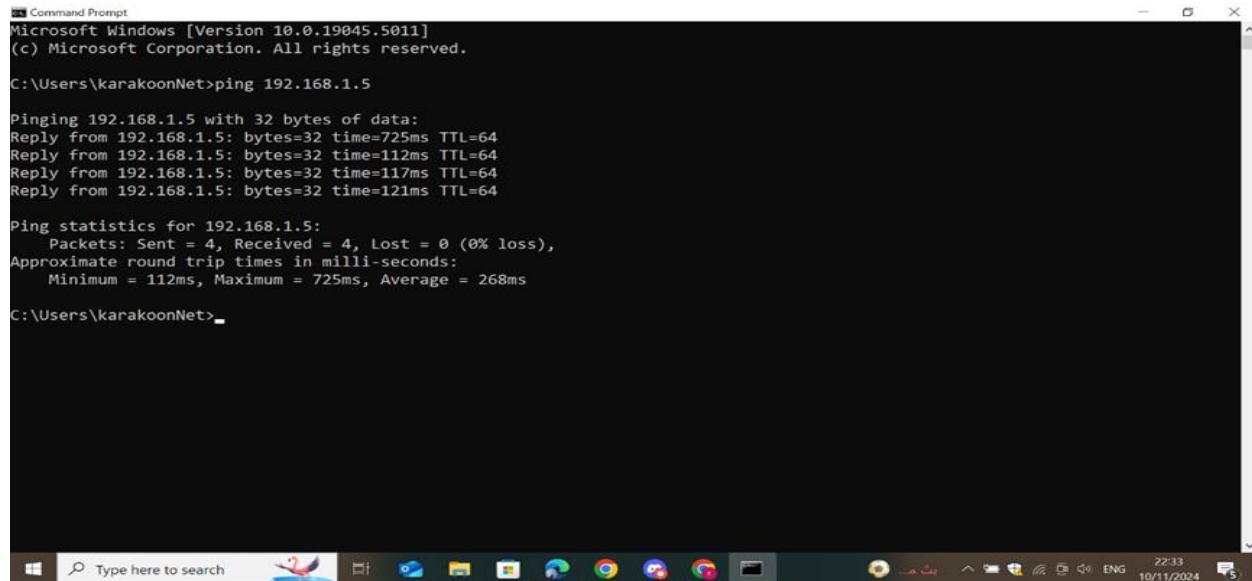
Connection-specific DNS Suffix . :
Description . . . . . : Intel(R) Wi-Fi 6 AX201 160MHz
Physical Address. . . . . : 40-1C-83-E2-B7-3D
DHCP Enabled. . . . . : Yes
Autoconfiguration Enabled . . . . . : Yes
Link-local IPv6 Address . . . . . : fe80::cead:403a:b1cf:c332%5(Preferred)
IPv4 Address. . . . . : 192.168.1.12(Preferred)
Subnet Mask . . . . . : 255.255.255.0
Lease Obtained. . . . . : Sunday, 10 November 2024 14:50:55
Lease Expires . . . . . : Monday, 11 November 2024 15:13:34
Default Gateway . . . . . : 192.168.1.1
DHCP Server . . . . . : 192.168.1.1
DHCPv6 IAID . . . . . : 54533251
DHCPv6 Client DUID. . . . . : 00-01-00-01-28-F3-85-9E-6C-02-E0-6F-7B-05
DNS Servers . . . . . : 192.168.1.1
NetBIOS over Tcpip. . . . . : Enabled

Ethernet adapter Bluetooth:

Media State . . . . . : Media disconnected
Connection-specific DNS Suffix . :
Description . . . . . : Bluetooth Device (Personal Area Network)
IPv4 Address. . . . . : 192.168.56.1(Preferred)
```

Figure 5) Domain Name System (DNS)

1.b.2 Ping a device within your local network (e.g., from your laptop to a smartphone on the same Wi-Fi network)



```
Command Prompt
Microsoft Windows [Version 10.0.19045.5011]
(c) Microsoft Corporation. All rights reserved.

C:\Users\karakoonNet>ping 192.168.1.5

Pinging 192.168.1.5 with 32 bytes of data:
Reply from 192.168.1.5: bytes=32 time=725ms TTL=64
Reply from 192.168.1.5: bytes=32 time=112ms TTL=64
Reply from 192.168.1.5: bytes=32 time=117ms TTL=64
Reply from 192.168.1.5: bytes=32 time=121ms TTL=64

Ping statistics for 192.168.1.5:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 112ms, Maximum = 725ms, Average = 268ms

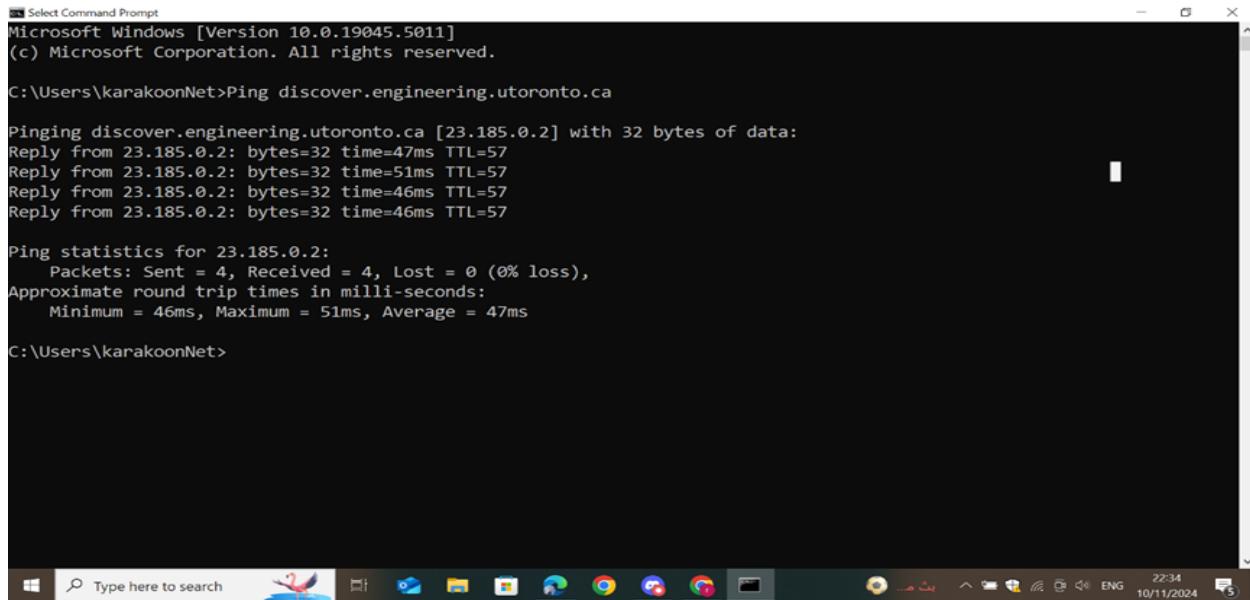
C:\Users\karakoonNet>
```

Figure 6 local network device Ping

The packet is specified with a size of 32 bytes and each request will contain 32 bytes of data payload. The line reply indicates that the destination device has received the request packet successfully, and a reply has been received from the specified IP address.

Number of packets sent and received is 4. Statics indicate that all four packets were successfully received from the device. Each line shows the details of every packet including the round trip time and TTL and no packet loss (percentage is 0%). Which means our connection was stable and responsive.

1.b.3 Ping discover.engineering.utoronto.ca. Based on the results, briefly explain whether you believe the response originates from Canada



The screenshot shows a Microsoft Windows Command Prompt window titled "Select Command Prompt". The window displays the following command and its output:

```
Microsoft Windows [Version 10.0.19045.5011]
(c) Microsoft Corporation. All rights reserved.

C:\Users\karakoonNet>Ping discover.engineering.utoronto.ca

Pinging discover.engineering.utoronto.ca [23.185.0.2] with 32 bytes of data:
Reply from 23.185.0.2: bytes=32 time=47ms TTL=57
Reply from 23.185.0.2: bytes=32 time=51ms TTL=57
Reply from 23.185.0.2: bytes=32 time=46ms TTL=57
Reply from 23.185.0.2: bytes=32 time=46ms TTL=57

Ping statistics for 23.185.0.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 46ms, Maximum = 51ms, Average = 47ms

C:\Users\karakoonNet>
```

The taskbar at the bottom of the screen shows various icons for common applications like File Explorer, Edge, and Google Chrome.

Figure 7 Ping discover.engineering.utoronto.ca.

The packet is specified with a size of 32 bytes and each request will contain 32 bytes of data payload. The line reply indicates that the destination device has received the request packet successfully, and a reply has been received from the specified IP address. TTL has an initial value of 57.

The output indicates that there was no loss, meaning that each packet request has had a corresponding packet response (This suggests a responsive and stable network connection to the specified domain). Number of packets sent and received is 4.

Based on the results, briefly explain whether you believe the response originates from Canada:

The response looks like it is from Canada.

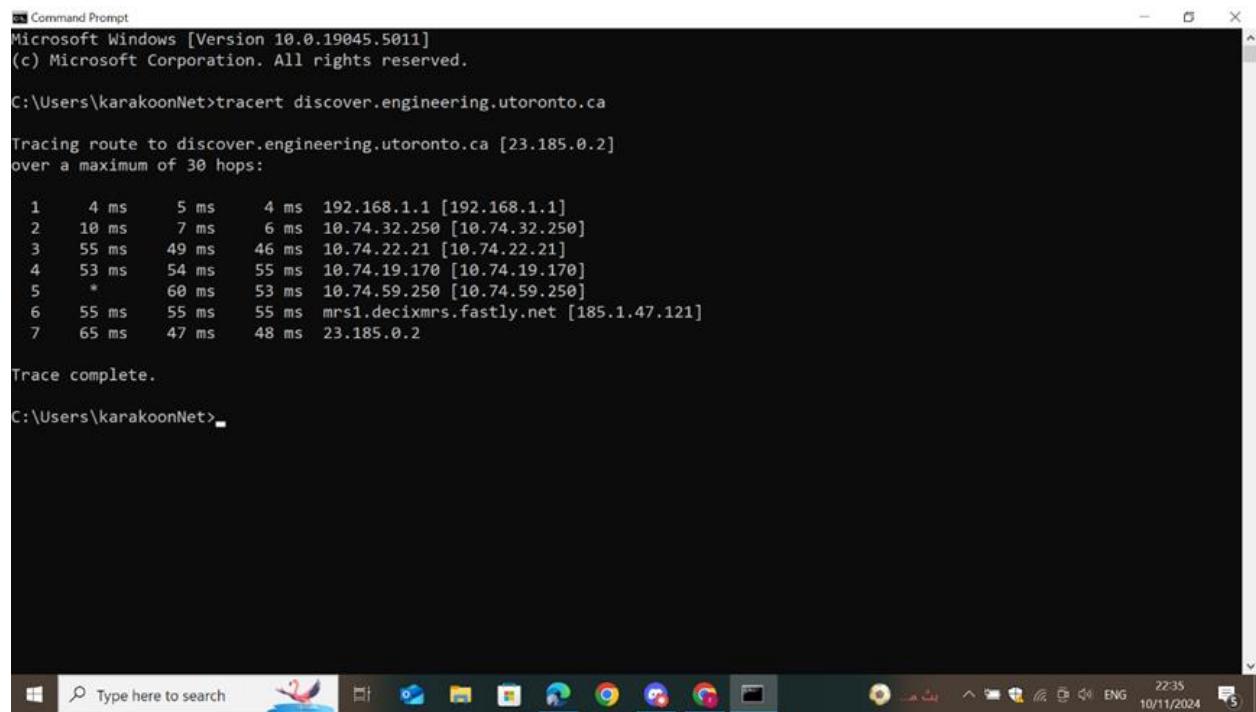
The hostname `discover.engineering.utoronto.ca` strongly suggests a connection to the University of Toronto's Faculty of Applied Science and Engineering, which is located in Toronto, Canada. The `.ca` top-level domain is specifically designated for Canada.

Low Latency/TTL: While not definitive proof, the relatively low latency and initial TTL of 57 are consistent with a server located relatively nearby. If the server were located on a different continent, you'd typically expect higher latency and a lower TTL due to the greater number of hops the packets would have to traverse.

But when we used the IP address online tool we found that the server is located in San Francisco California

Even though the hostname `discover.engineering.utoronto.ca` strongly suggests a Canadian location, the University of Toronto likely uses a Content Delivery Network (CDN) or cloud hosting provider with servers located around the world. CDNs improve website performance and availability by serving content from servers closer to users. In this case, the request was routed to a server in San Francisco, likely for better performance.

1.b.4 Run tracert on discover.engineering.utoronto.ca



```
Command Prompt
Microsoft Windows [Version 10.0.19045.5011]
(c) Microsoft Corporation. All rights reserved.

C:\Users\karakoonNet>tracert discover.engineering.utoronto.ca

Tracing route to discover.engineering.utoronto.ca [23.185.0.2]
over a maximum of 30 hops:

 1   4 ms    5 ms    4 ms  192.168.1.1 [192.168.1.1]
 2   10 ms   7 ms    6 ms  10.74.32.250 [10.74.32.250]
 3   55 ms   49 ms   46 ms  10.74.22.21 [10.74.22.21]
 4   53 ms   54 ms   55 ms  10.74.19.170 [10.74.19.170]
 5   *       60 ms   53 ms  10.74.59.250 [10.74.59.250]
 6   55 ms   55 ms   55 ms  mrs1.decixmrs.fastly.net [185.1.47.121]
 7   65 ms   47 ms   48 ms  23.185.0.2

Trace complete.

C:\Users\karakoonNet>
```

Figure 8 tracert on discover.engineering.utoronto.ca

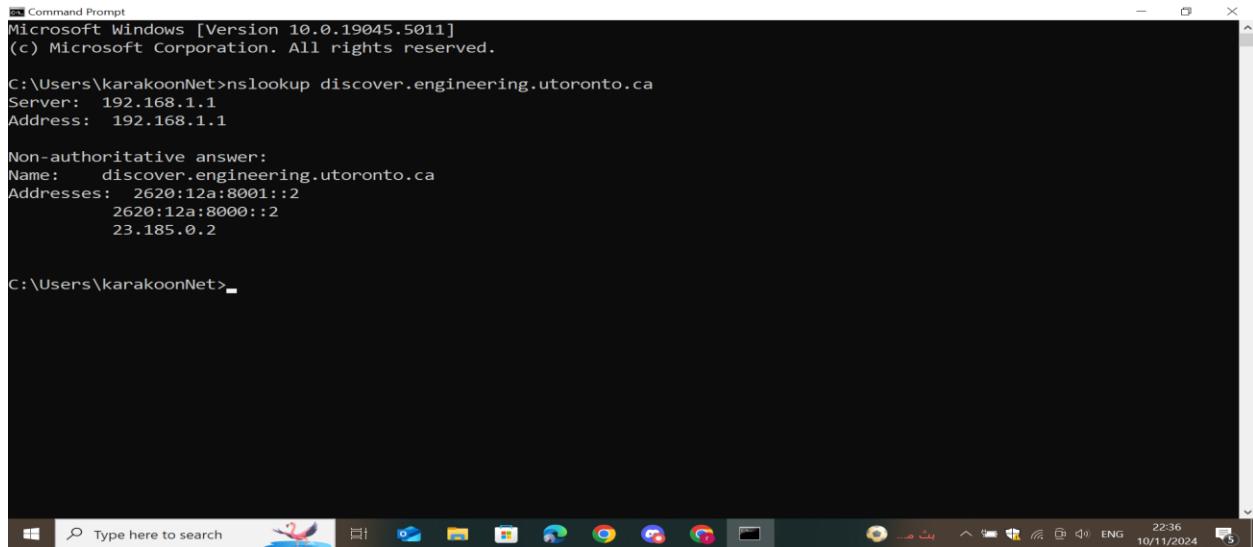
A maximum number of hops or network devices the command will attempt to reach before stopping the trace is 30 hops. Each hop in the output shows the hop number, the round-trip time in milliseconds (ms) for packets to reach that hop and return, and the IP address of the router at that hop.

The first IP Address is my network IP address (192.168.1.1), and the last IP address is usually associated with the destination IP Address if the trace is successfully completed. The first hop is the local router at 192.168.1.1

Asterisks (*) means traceroute couldn't access that router or unreachable due to the router's configuration if it was set to reject ICMP packets.

“Trace completed” message indicates that the command has finished tracing the route, and the output provides information about the routers traversed from computer to reach discover.engineering.utoronto.ca

1.b.5 Use nslookup to retrieve the DNS information for discover.engineering.utoronto.ca



```
Microsoft Windows [Version 10.0.19045.5011]
(c) Microsoft Corporation. All rights reserved.

C:\Users\karakoonNet>nslookup discover.engineering.utoronto.ca
Server: 192.168.1.1
Address: 192.168.1.1

Non-authoritative answer:
Name: discover.engineering.utoronto.ca
Addresses: 2620:12a:8001::2
           2620:12a:8000::2
           23.185.0.2

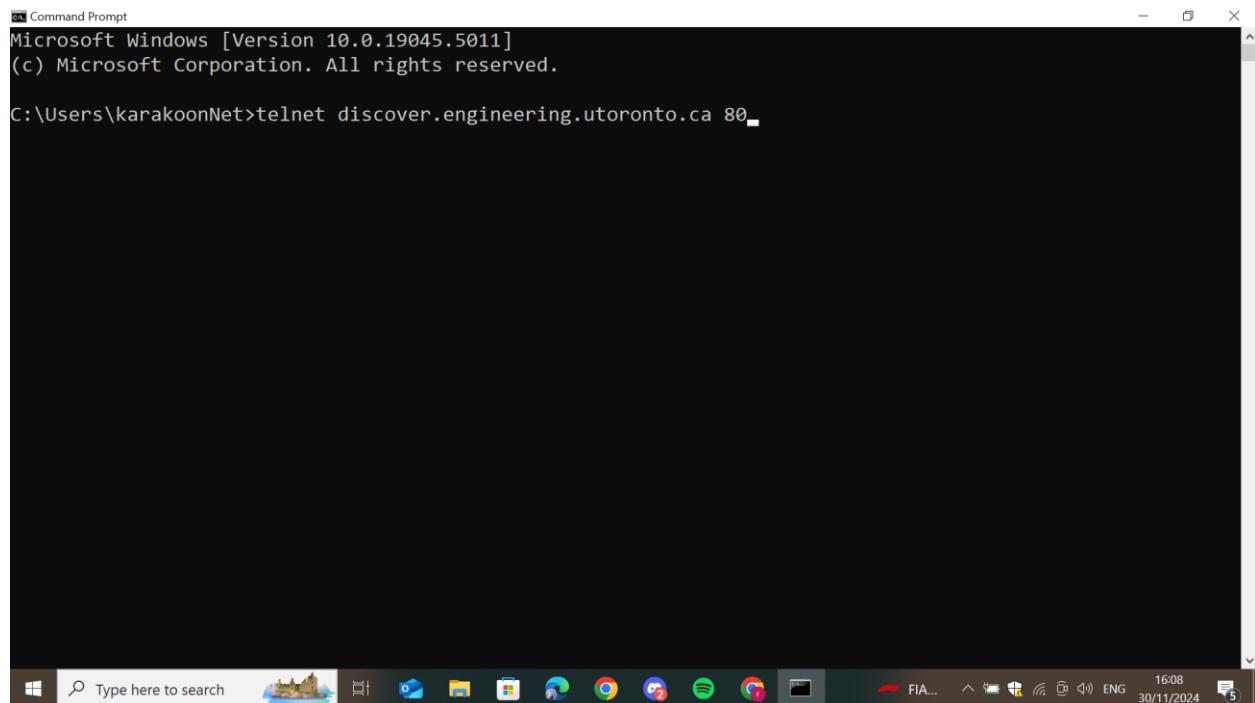
C:\Users\karakoonNet>
```

Figure 9 nslookup to retrieve the DNS

The output displays that the DNS server used for the query is at IP address 192.168.1.1. It also provides the IP addresses associated with the domain name "discover.engineering.utoronto.ca" which are "2620:12a:8001::2" and "2620:12a:8000::2" and "23.185.0.2"

The information is non-authoritative, meaning it's not directly from the authoritative DNS server for the domain but is obtained from a caching or intermediary DNS server.

1.b.6 Attempt to connect with telnet to discover.engineering.utoronto.ca



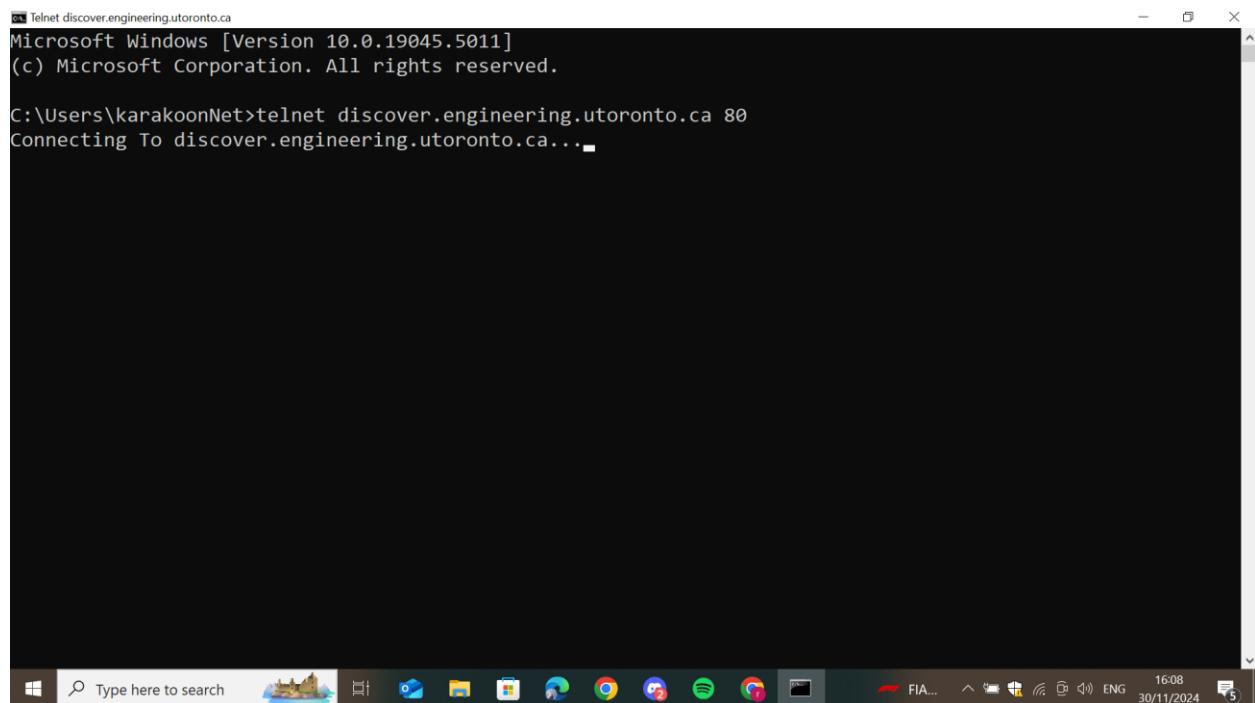
A screenshot of a Microsoft Windows Command Prompt window. The title bar says "Command Prompt". The window content shows:

```
Microsoft Windows [Version 10.0.19045.5011]
(c) Microsoft Corporation. All rights reserved.

C:\Users\karakoonNet>telnet discover.engineering.utoronto.ca 80
```

The taskbar at the bottom shows various pinned icons and the date/time: 30/11/2024 16:08.

Figure 10 connect with telnet to discover.engineering.utoronto.ca



A screenshot of a Microsoft Windows Telnet window. The title bar says "Telnet discover.engineering.utoronto.ca". The window content shows:

```
Microsoft Windows [Version 10.0.19045.5011]
(c) Microsoft Corporation. All rights reserved.

C:\Users\karakoonNet>telnet discover.engineering.utoronto.ca 80
Connecting To discover.engineering.utoronto.ca...
```

The taskbar at the bottom shows various pinned icons and the date/time: 30/11/2024 16:08.

```
C:\>telnet discover.engineering.utoronto.ca
```

```
C:\>Command Prompt
HTTP/1.1 400 Bad Request
Connection: close
Content-Length: 11
content-type: text/plain; charset=utf-8
X-served-by: cache-mrs10543

Bad Request

Connection to host lost.

C:\Users\karakoonNet>
```

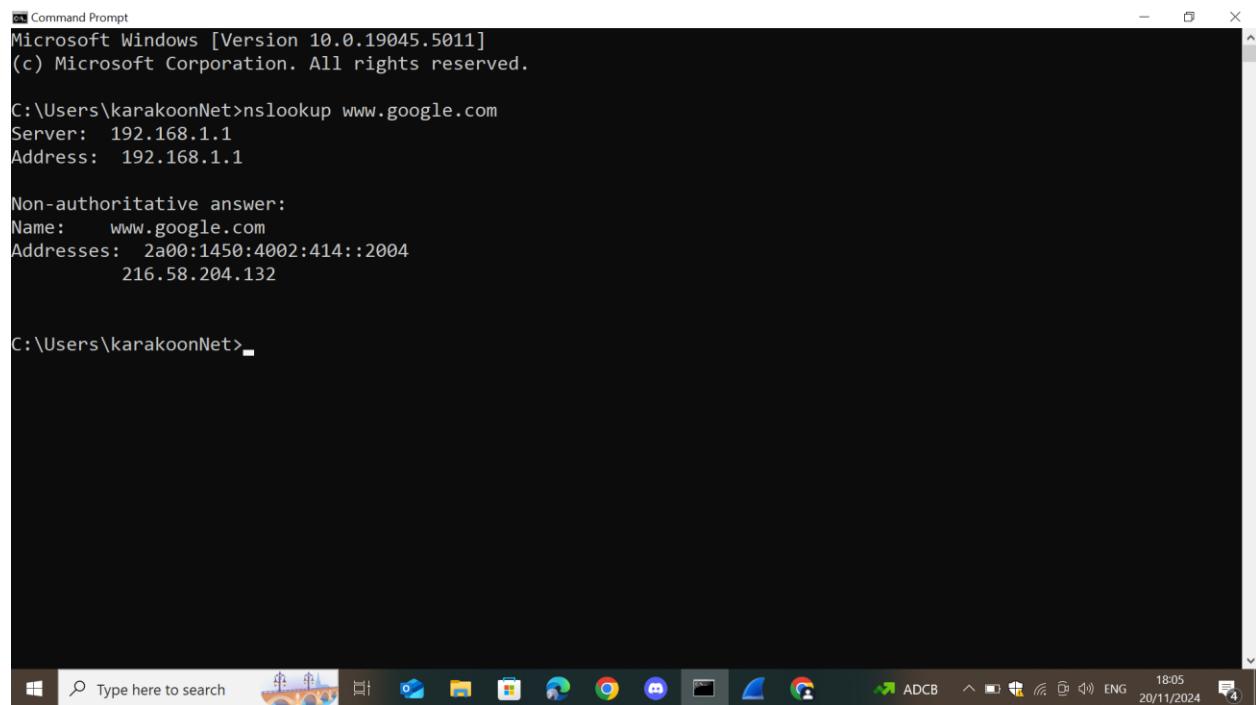
We used the Telnet command to establish a connection to the server at discover.engineering.utoronto.ca on port 80, which is the standard port for HTTP communication.

After connecting, we sent a basic HTTP GET request:

The server responded with:

HTTP/1.1 400 Bad Request

1.b.7 Use the Wireshark packet analyzer to capture a DNS query and reply for any hostname of your choice.



```
Command Prompt
Microsoft Windows [Version 10.0.19045.5011]
(c) Microsoft Corporation. All rights reserved.

C:\Users\karakoonNet>nslookup www.google.com
Server: 192.168.1.1
Address: 192.168.1.1

Non-authoritative answer:
Name: www.google.com
Addresses: 2a00:1450:4002:414::2004
          216.58.204.132

C:\Users\karakoonNet>
```

Figure 11 Use the Wireshark packet analyzer to capture a DNS query and reply for any hostname

The screenshots show the Wireshark interface capturing DNS traffic. The top screenshot displays a standard query from 192.168.1.20 to www.google.com (216.58.204.132) and a corresponding response. The bottom screenshot provides a detailed view of the same transaction, including the DNS message structure and a hex dump of the captured bytes.

We used nslookup to send packets and then WireShark to capture the DNS messages.

2 Part2 Web Server Results

Abstract:

This part presents the implementation of a simple HTTP server Python sockets. The server listens for connections and handles HTTP requests from clients. The server's goal is to serve requested files and redirect specific URLs.

Built with the socket module in Python, the server follows a loop-based structure, continuously listening for new connections. Upon accepting a connection, it parses the client's HTTP request to extract the requested URL path. Based on the path, the server determines the file to serve and sets the appropriate content type.

The server supports various file types like HTML, CSS, PNG, and JPG. It handles requests for different language versions of the main page and provides a 404 Not Found response for invalid paths or requests using a predefined "NotFound.html" file.

The implementation showcases error handling for file not found and I/O errors. Detailed status messages are printed during execution for debugging and verification.

2.1 Some screenshots of the web server running:

2.1.1 main_en.html page

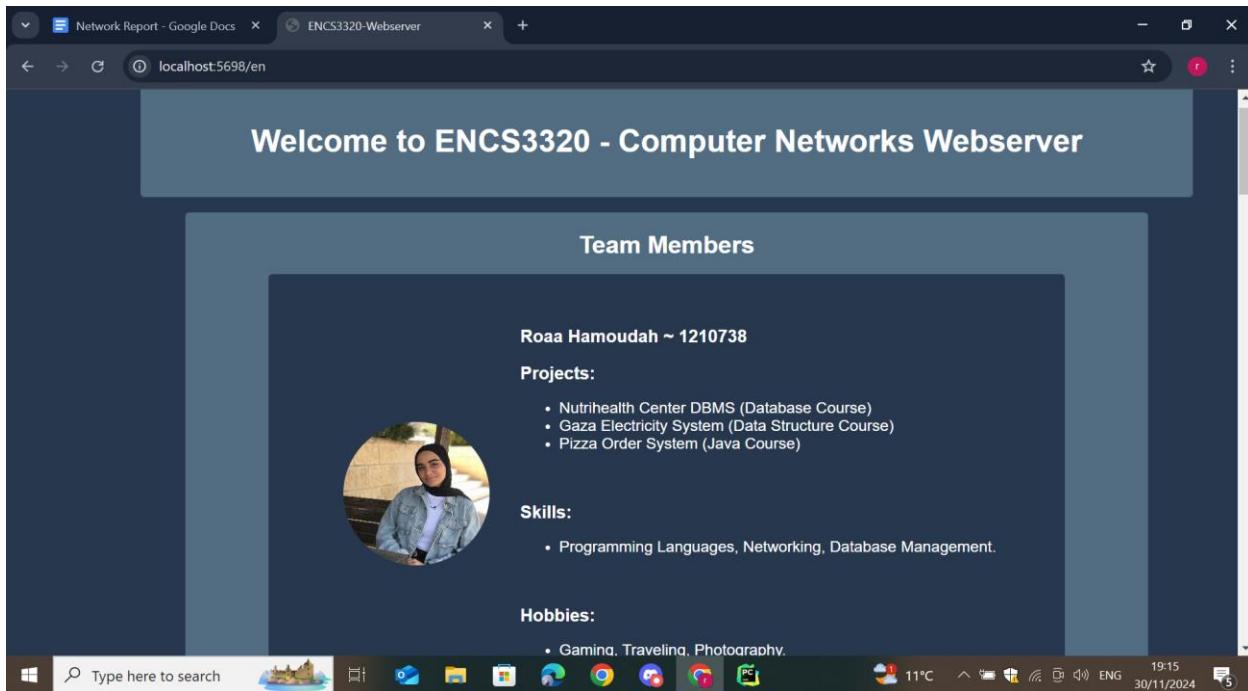


Figure 12 main_en.html page



A screenshot of a web browser window titled "ENCS3320-Webserver". The URL bar shows "localhost:5698/en". The main content area displays a profile for "Ameer Yasen ~ 1210187". It includes a circular profile picture of a man in a red t-shirt standing in front of a large stack of shipping containers. Below the picture, there is a summary section with the following details:

Projects:

- Pac-Man Game (Java 2 Course)
- Car Rental Web App (Web Course)
- Dijkstra's Algorithm Visualization (Algorithms Course)

Skills:

- Mobile Development, Machine Learning/AI, Cloud Development.

Hobbies:

- Swimming, Hiking, Puzzle Solving.

The browser interface at the bottom shows the Windows taskbar with various pinned icons and system status indicators like battery level, temperature (12°C), and date/time (30/11/2024).

A screenshot of a web browser window titled "ENCS3320-Webserver". The URL bar shows "localhost:5698/en". The main content area displays a page titled "Understanding TCP and UDP". The page has the following sections:

Overview:

TCP (Transmission Control Protocol) and UDP (User Datagram Protocol) are two fundamental communication protocols used for transmitting data over networks. While they serve similar purposes, their approaches differ significantly, catering to various needs of network communication.

What is TCP?

TCP is a **connection-oriented protocol** designed for reliable communication. It establishes a connection before transferring data and ensures that all packets are delivered and in the correct order.

What is UDP?

UDP is a **connectionless protocol** known for its speed. Unlike TCP, it doesn't ensure the delivery or order of packets, making it ideal for real-time communication.

The browser interface at the bottom shows the Windows taskbar with various pinned icons and system status indicators like battery level, temperature (12°C), and date/time (30/11/2024).

Figure 13 main_en.html page network topic

Here are the main differences between TCP and UDP:

Factor	TCP	UDP
Connection type	Requires an established connection before transmitting data	No connection is needed to start and end a data transfer
Data sequence	Can sequence data (send in a specific order)	Cannot sequence or arrange data
Data retransmission	Can retransmit data if packets fail to arrive	No data retransmitting. Lost data can't be retrieved
Delivery	Delivery is guaranteed	Delivery is not guaranteed
Check for errors	Thorough error-checking guarantees data arrives in its intended state	Minimal error-checking covers the basics but may not prevent all errors
Broadcasting	Not supported	Supported
Speed	Slow, but complete data delivery	Fast, but at risk of incomplete data delivery

Applications of TCP and UDP:

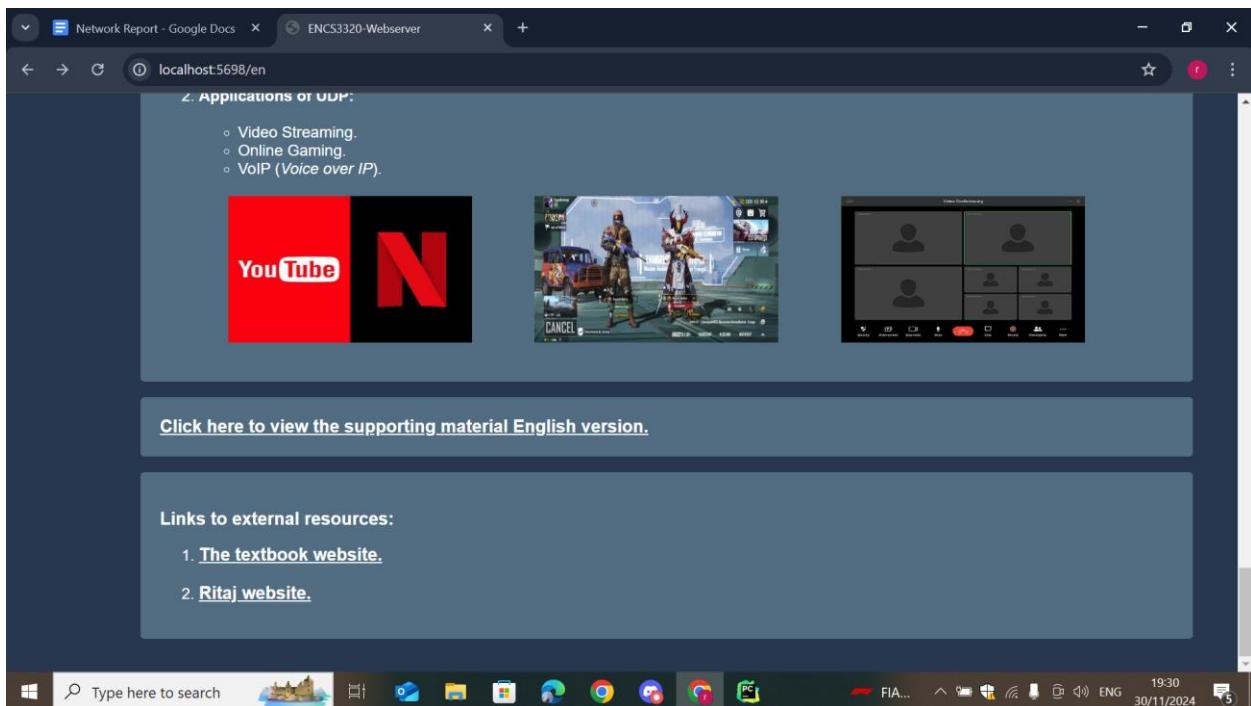
1. Applications of TCP:

- Email Communication (HTTP/HTTPS).
- File Transfers.
- Web Browsing.

2. Applications of UDP:

- Video Streaming.
- Online Gaming.
- VoIP (Voice over IP).

Figure 14 UDP and TCP Applications



[Click here to view the supporting material English version.](#)

Links to external resources:

1. [The textbook website](#),
2. [Ritaj website](#).

Figure 15 main_en.html page extra resources links

2.1.2 main_ar.html page



Figure 16 main_ar.html Page

Network Report - Google Docs ENCS3320 خادم الويب ~ 1211245

localhost:5698/ar

عطـا مصلـح ~ 1211245

المشاريع:

- لعبة باك من (دورـة جـافـا 2)
- تطـبـيق تـأـيـيـر السـيـارـات (دورـة الوـيب)
- تصـوـير خـواـرـزمـيـة دـيـكـسـتـرا (دورـة الـخـواـرـزمـيـات)

الـمـهـارـات:

- تطـبـيق الـلـعـبـات، أـمـنـ الشـبـكـاتـ، تـحلـيلـ الـبـيـانـاتـ.

الـهـوـىـات:

- الـذـهـابـ إـلـىـ الصـالـةـ الـرـياـضـيـةـ، الـعـزـفـ عـلـىـ الـجـيـتـارـ، الـقـرـاءـةـ.

أمير ياسين ~ 1210187

Type here to search Start... ENG 30/11/2024

Network Report - Google Docs ENCS3320 خادم الويب ~ 1210187

localhost:5698/ar

أمير ياسين ~ 1210187

المشاريع:

- لعبة باك من (دورـة جـافـا 2)
- تطـبـيق تـأـيـيـر السـيـارـات (دورـة الوـيب)
- تصـوـير خـواـرـزمـيـة دـيـكـسـتـرا (دورـة الـخـواـرـزمـيـات)

الـمـهـارـات:

- تطـبـيقـ الـطـبـيـقـاتـ الـمـحمـولـةـ، الـلـطـمـ الـآـلـيـ/الـنـكـاءـ الـاصـطـنـاعـيـ، تـطـبـيقـ الـسـحـابـةـ.

الـهـوـىـات:

- الـمـيـاجـةـ، الـمـشـيـ فـيـ الطـبـيـعـةـ، حلـ الـأـلـغازـ.

Type here to search Start... ENG 30/11/2024

فهم بروتوكول UDP و TCP

نظرة عامة:

بروتوكول التحكم في الإرسال (TCP) و بروتوكول بث البيانات المستخدم (UDP) هما بروتوكولان أساسيان للتواصل يستخدمان لنقل البيانات عبر الشبكات. على الرغم من أنها تخدمان أغراضًا مشابهة، إلا أن طرقهما يختلف بشكل كبير، مما يلبي احتياجات مختلفة في التواصل الشبكي.

ما هو بروتوكول TCP؟

بروتوكول TCP هو بروتوكول موجه نحو الاتصال مصمم للتواصل الموثوق. يقوم بإنشاء اتصال قبل نقل البيانات ويضمن أن جميع الحزم يتم تسليمها وبالترتيب الصحيح.

ما هو بروتوكول UDP؟

بروتوكول UDP هو بروتوكول غير موجه نحو الاتصال معروف بسرعةه. على عكس بروتوكول TCP، فإنه لا يضمن تسليم الحزم أو ترتيبها، مما يجعله مثالياً للتواصل في الوقت الحقيقي.

الإختلافات الرئيسية بين بروتوكول UDP و TCP

الإختلافات الرئيسية بين بروتوكول UDP و TCP

عامل	TCP	UDP
نوع الاتصال	يطلب اتصالاً مؤسساً قبل نقل البيانات	لا حاجة لاتصال لبدء وإنتهاء نقل البيانات
ترتيب البيانات	يمكنه ترتيب البيانات (إرسالها بترتيب محدد)	لا يمكنه ترتيب أو تنظيم البيانات
إعادة إرسال البيانات	يمكن إعادة إرسال البيانات إذا ث除了 الحزم في الوصول	لا توجد إعادة إرسال البيانات. البيانات المفقودة لا يمكن استردادها
التسليم	التسليم مضبوط	التسليم غير مضبوط
التحقق من الأخطاء	التحقق الشامل من الأخطاء يضمن وصول البيانات في حالتها المقصودة	التحقق الأندي من الأخطاء يعطي الأساليب ولكن قد لا يمنع جميع الأخطاء
البث	غير مدعم	مدعم
السرعة	بطيء، ولكن تسليم بيانات كاملة	سرير، ولكن مع خطر تسليم بيانات غير كاملة

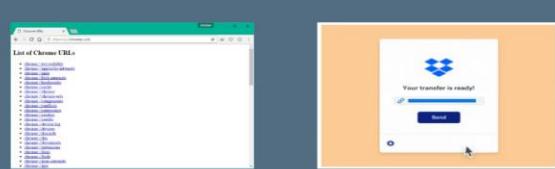
تطبيقات بروتوكول UDP و TCP:

- تطبيقات بروتوكول TCP:

تطبيقات بروتوكول TCP و UDP:

1. تطبيقات بروتوكول TCP:

- التواصل عبر البريد الإلكتروني (HTTP/HTTPS).
- نقل الملفات.
- تصفح الويب.



2. تطبيقات بروتوكول UDP:

- بث الفيديو.
- الألعاب عبر الإنترنت.
- الصوت عبر بروتوكول الإنترنت (VoIP).



Windows taskbar at the bottom showing various application icons and system status.

2. تطبيقات بروتوكول UDP:

- بث الفيديو.
- الألعاب عبر الإنترنت.
- الصوت عبر بروتوكول الإنترنت (VoIP).



Call to action: [القى هنا لعرض الموارد الداعمة باللغة العربية](#).

روابط لمصادر خارجية:

1. [موقع الكتاب](#).
2. [موقع ينتاج](#).

Windows taskbar at the bottom showing various application icons and system status.

2.1.3 supporting_material_en.html Page

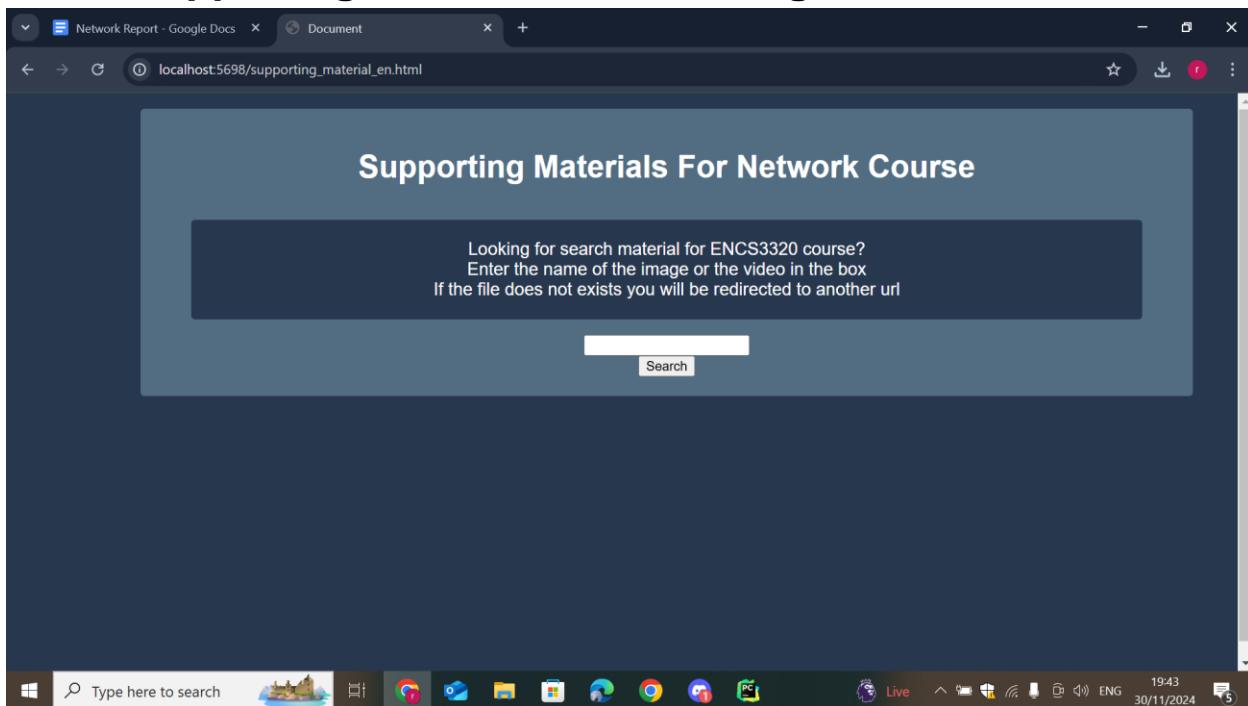


Figure 17 supporting_material_en.html Page

2.1.3.1 Request an existing image

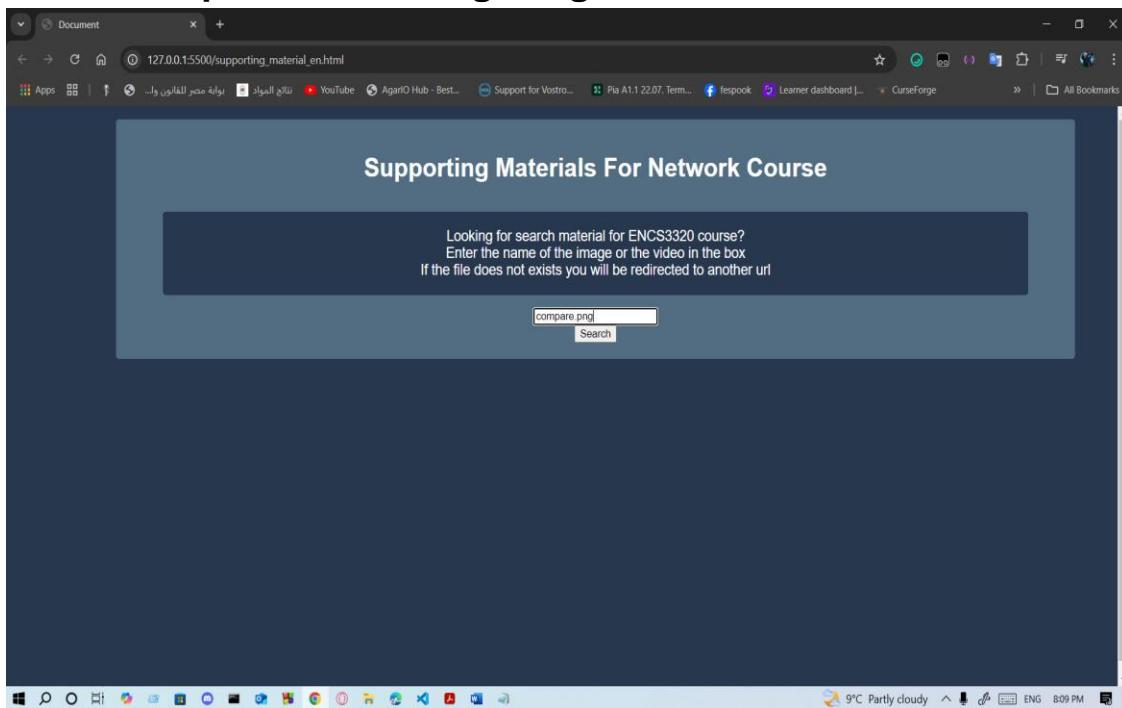


Figure 18 Request an existing image

TCP	UDP
Secure	Unsecure
Connection-Oriented	Connectionless
Slow	Fast
Guaranteed Transmission	No Guarantee
Used by Critical Applications	Used by Real-Time Applications
Packet Reorder Mechanism	No Reorder Mechanism
Flow Control	No Flow Control
Advanced Error Checking	Basic Error Checking (Checksum)
20 Bytes Header	8 Bytes Header
Acknowledgement Mechanism	No Acknowledgement
Three-Way Handshake	No Handshake Mechanism
DNS, HTTPS, FTP, SMTP etc.	DNS, DHCP, TFTP, SNMP etc.

2.1.3.2 Request a non-existing image

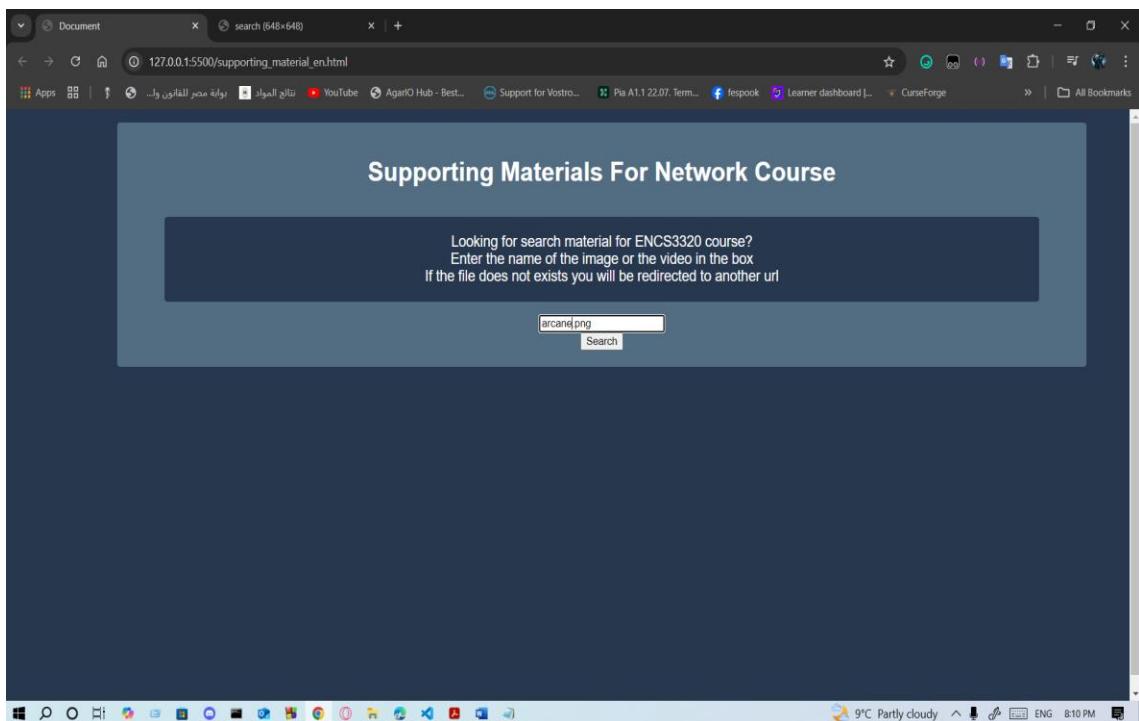
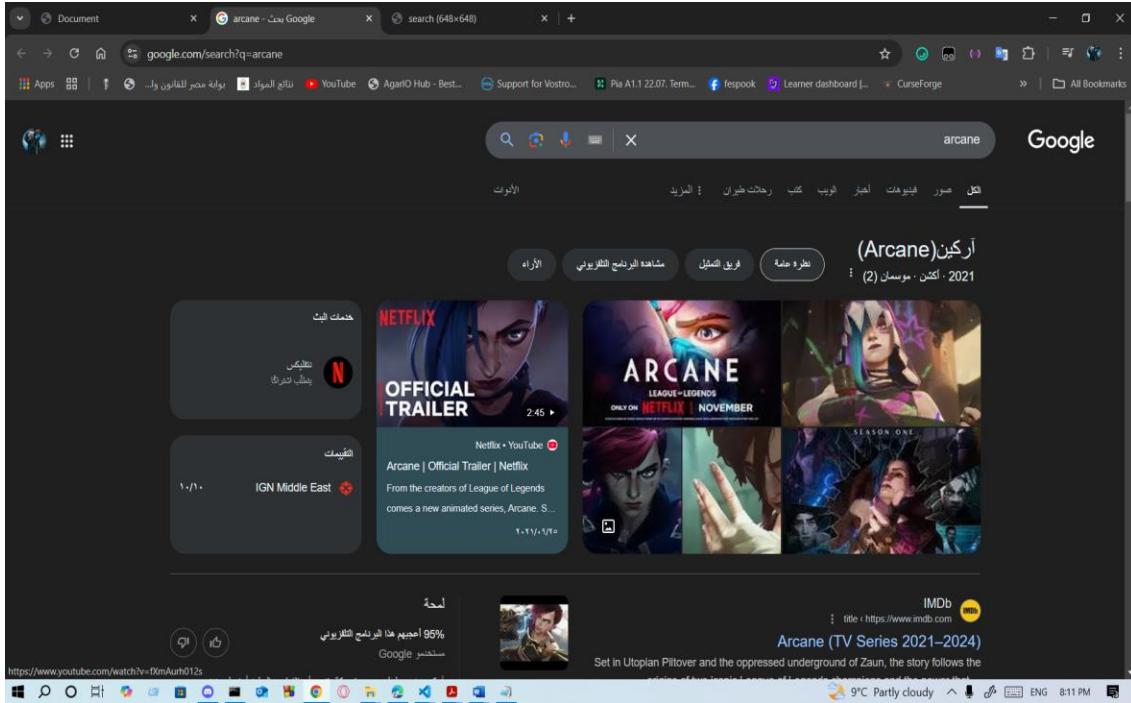


Figure 19 Request a non-existing image



HTTP status code 307 Temporary Redirect indicates that the resource being requested has been temporarily moved to a different URL. The client should resubmit the original request to the new URL provided in the **Location** header of the response.

In the last image case it should look like this:

```
HTTP/1.1 307 Temporary Redirect
Date: Sun, 1 Dec 2024 4:21:00 GMT
Server: Apache/2.4.52 (Ubuntu) (Example server)
Location: https://www.google.com/search?q=arcane
Content-Type: text/html; charset=utf-8
Content-Length: 0
Connection: close
```

Figure 20 HTTP status code 307 Temporary Redirect

That the server response header and the new location redirect to the client browser know that the resource location changed so it sends

another request to the new location that is in the response header. The original request must not be changed when resubmitting the request to the new location. This preserves the intended action of the original request. 307 redirects are often used when a server wants to temporarily move a resource without changing the semantics of the client's interaction with it.

2.1.3.3 Request an existing video

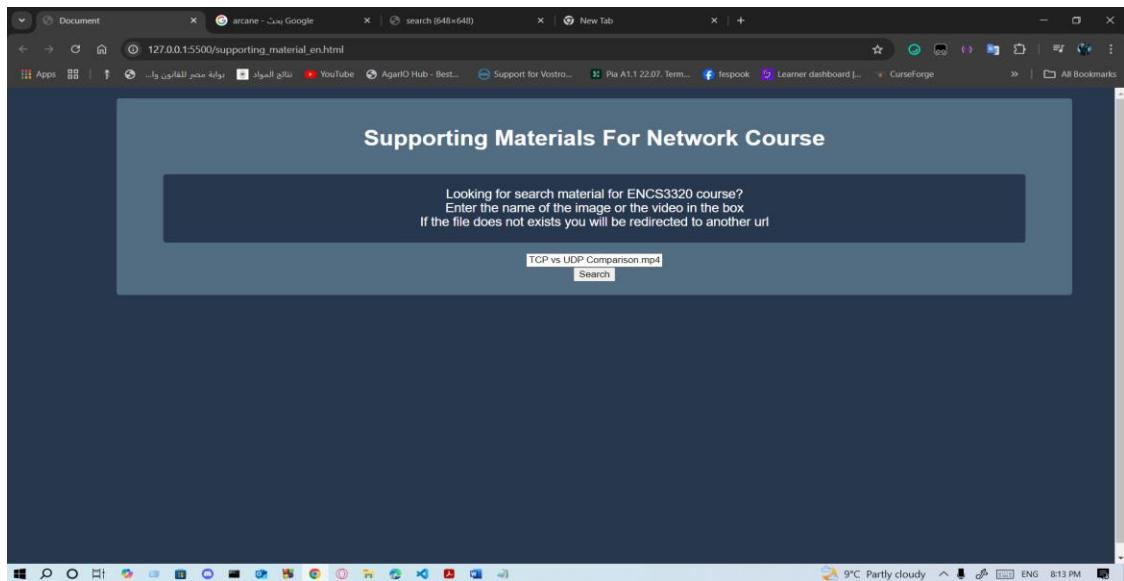
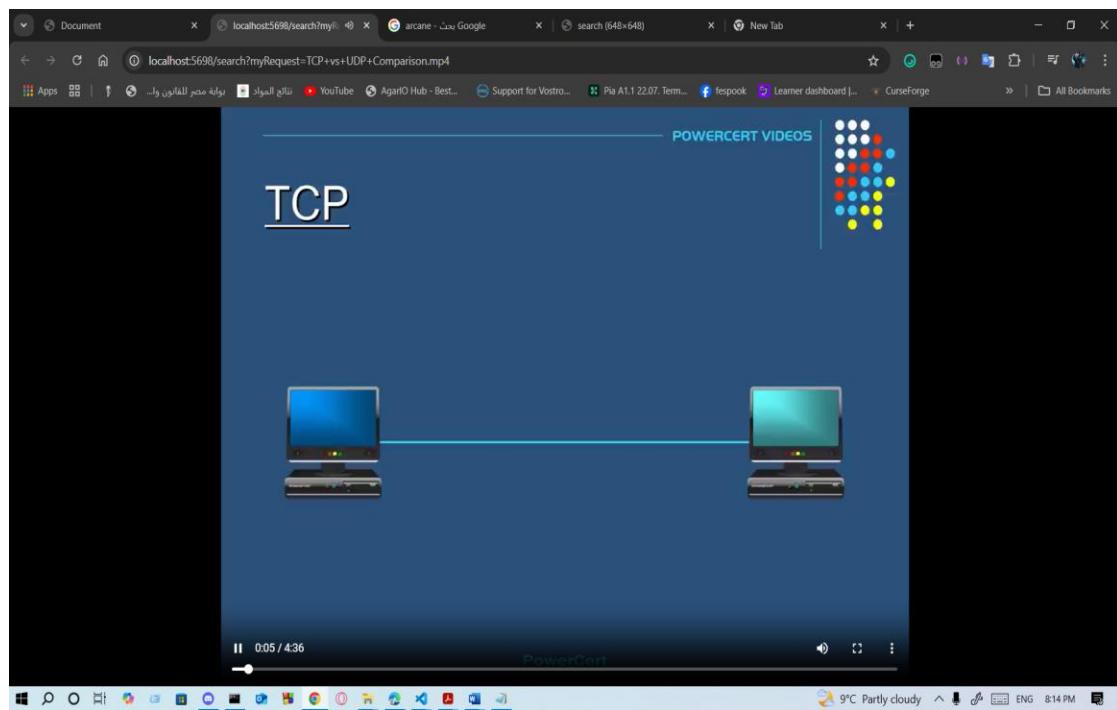


Figure 21 Request an existing video



2.1.3.4 Request a non-existing video

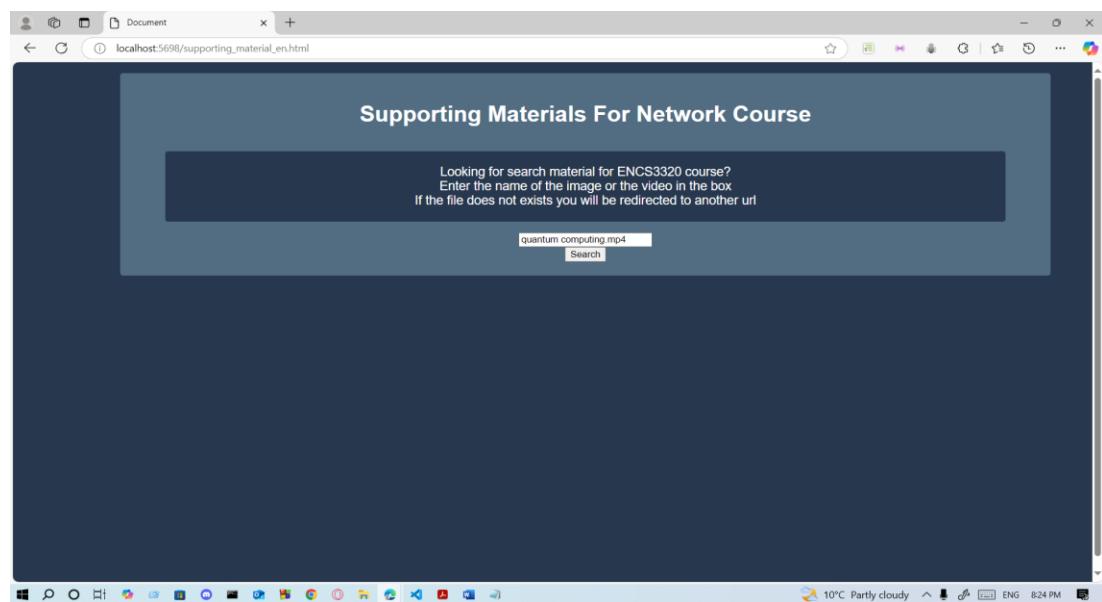
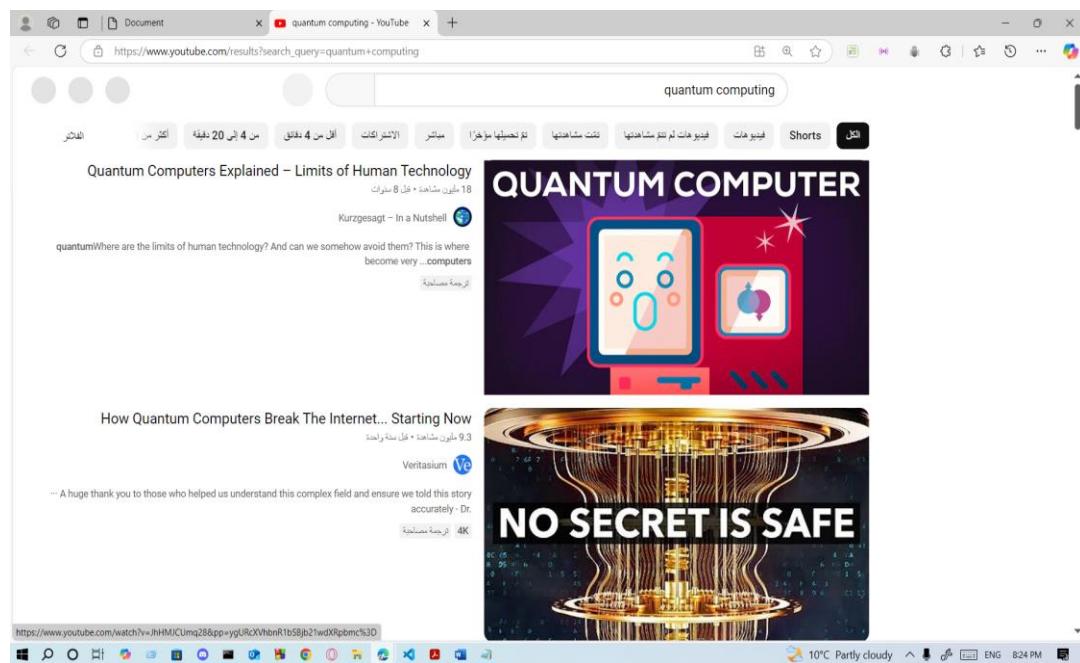


Figure 22 Request a non-existing video



Redirect to a YouTube search URL that filters results to videos based on the user's input.

2.1.4 supporting_material_ar.html Page

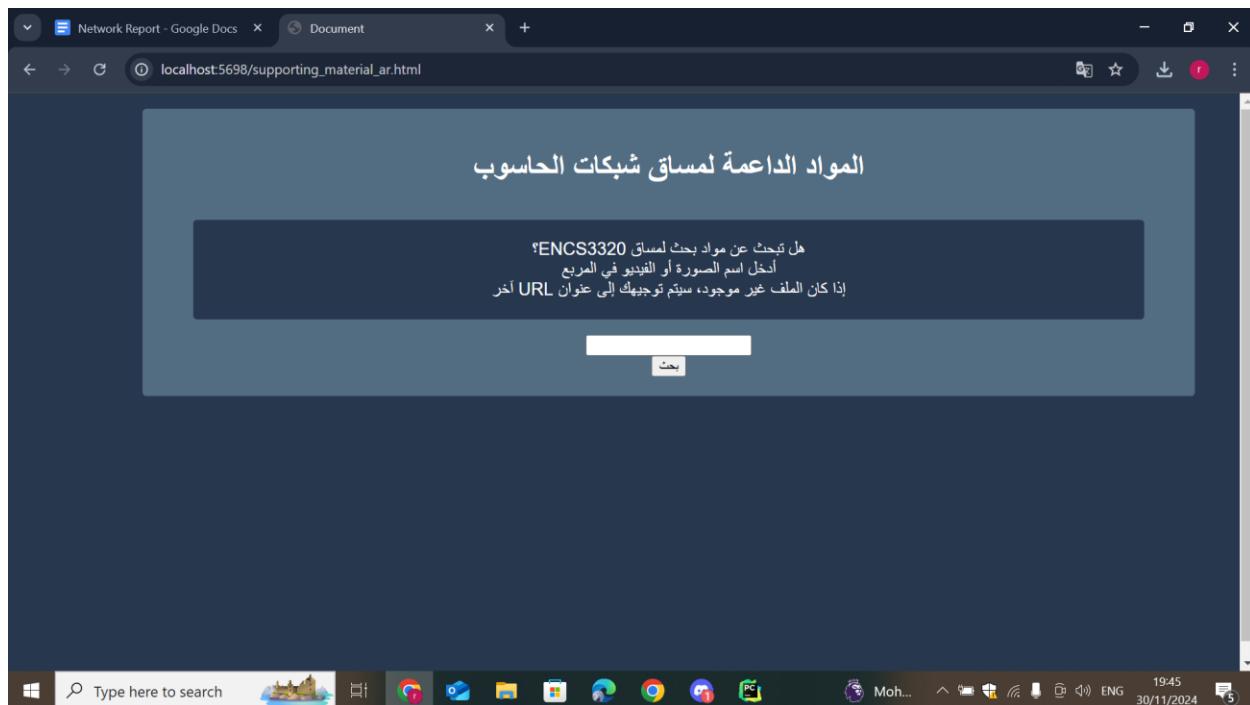
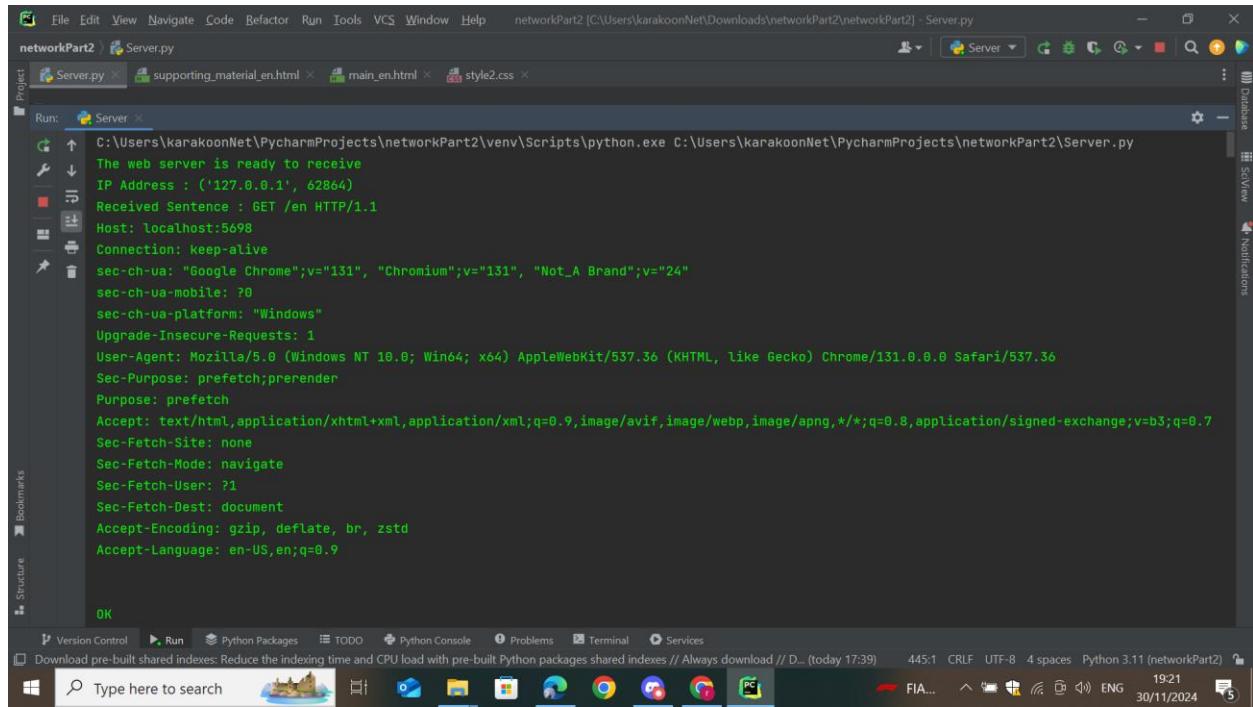


Figure 23 supporting_material_ar.html Page

2.1.5 HTTP Requests printed one the server terminal

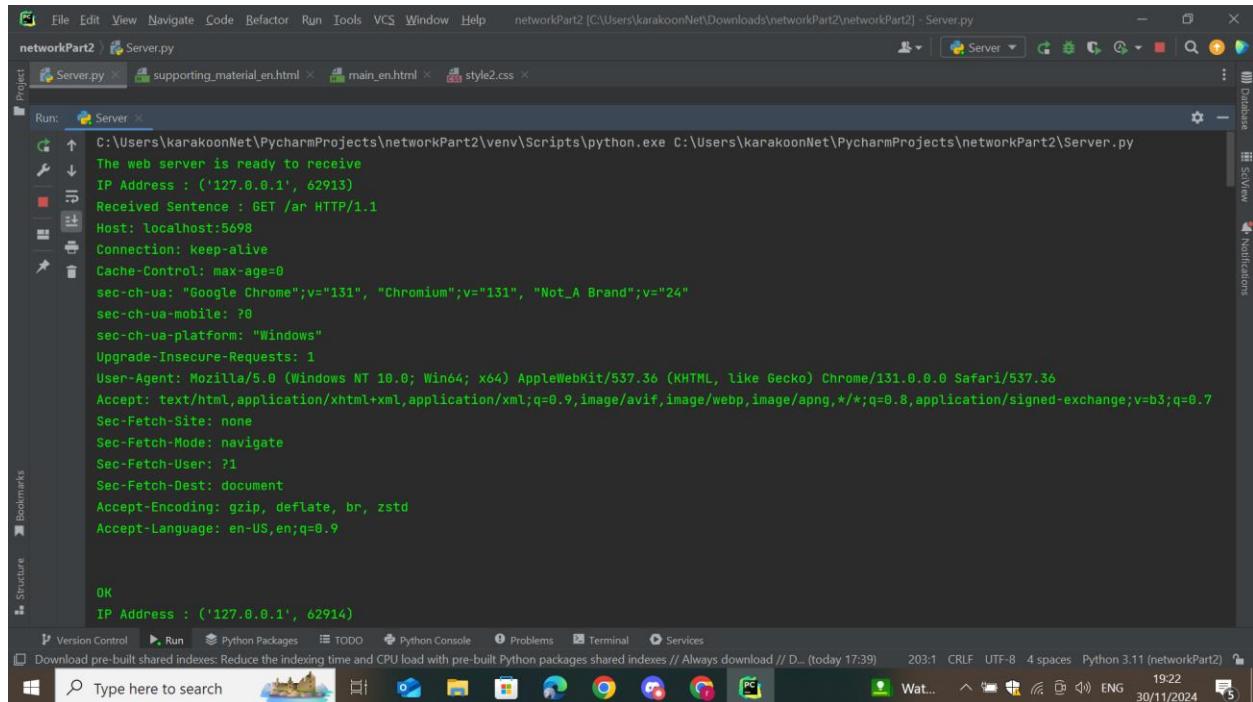


The screenshot shows the PyCharm IDE interface with the 'Server' run configuration selected. The terminal window displays the following log output:

```
C:\Users\karakoonNet\PycharmProjects\networkPart2\venv\Scripts\python.exe C:\Users\karakoonNet\PycharmProjects\networkPart2\Server.py
The web server is ready to receive
IP Address : ('127.0.0.1', 62864)
Received Sentence : GET /en HTTP/1.1
Host: localhost:5698
Connection: keep-alive
sec-ch-ua: "Google Chrome";v="131", "Chromium";v="131", "Not_A_Brand";v="24"
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/131.0.0.0 Safari/537.36
Sec-Purpose: prefetch;prefetch
Purpose: prefetch
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: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate, br, zstd
Accept-Language: en-US,en;q=0.9

OK
```

Figure 24 HTTP Requests printed one the server terminal



The screenshot shows the PyCharm IDE interface with the 'Server' run configuration selected. The terminal window displays the following log output:

```
C:\Users\karakoonNet\PycharmProjects\networkPart2\venv\Scripts\python.exe C:\Users\karakoonNet\PycharmProjects\networkPart2\Server.py
The web server is ready to receive
IP Address : ('127.0.0.1', 62913)
Received Sentence : GET /ar HTTP/1.1
Host: localhost:5698
Connection: keep-alive
Cache-Control: max-age=0
sec-ch-ua: "Google Chrome";v="131", "Chromium";v="131", "Not_A_Brand";v="24"
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/131.0.0.0 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: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate, br, zstd
Accept-Language: en-US,en;q=0.9

OK
IP Address : ('127.0.0.1', 62914)
```

The screenshot shows the PyCharm IDE interface with the 'Server' tool window open. The 'Run' tab is selected, displaying captured network traffic. The first entry is an 'OK' response with the IP Address '(127.0.0.1, 62928)'. The 'Received Sentence' is 'GET /Resources/WebBrowsing.jpg HTTP/1.1'. The 'Host' header is 'localhost:5698'. The 'Connection' header is 'keep-alive'. The 'sec-ch-ua-platform' header is 'Windows'. The 'User-Agent' header is 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.0.0 Safari/537.36'. The 'sec-ch-ua' header is 'Google Chrome";v="131", "Chromium";v="131", "Not_A_Brand";v="24". The 'sec-ch-ua-mobile' header is '0'. The 'Accept' header is 'image/avif,image/webp,image/apng,image/svg+xml,image/*,*/*;q=0.8'. The 'Sec-Fetch-Site' header is 'same-origin'. The 'Sec-Fetch-Mode' header is 'no-cors'. The 'Sec-Fetch-Dest' header is 'image'. The 'Referer' header is 'http://localhost:5698/ar'. The 'Accept-Encoding' header is 'gzip, deflate, br, zstd'. The 'Accept-Language' header is 'en-US,en;q=0.9'. The second entry is another 'OK' response with the IP Address '(127.0.0.1, 62929)'. The 'Received Sentence' is 'GET /ar HTTP/1.1'. The 'Host' header is 'localhost:5698'. The 'Connection' header is 'keep-alive'. The 'sec-ch-ua-platform' header is 'Windows'. The 'User-Agent' header is 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.0.0 Safari/537.36'. The 'sec-ch-ua' header is 'Google Chrome";v="131", "Chromium";v="131", "Not_A_Brand";v="24". The 'sec-ch-ua-mobile' header is '0'. The 'Accept' header is 'text/css,*/*;q=0.1'. The 'Sec-Fetch-Site' header is 'same-origin'. The 'Sec-Fetch-Mode' header is 'no-cors'. The 'Sec-Fetch-Dest' header is 'style'. The 'Referer' header is 'http://localhost:5698/ar'. The 'Accept-Encoding' header is 'gzip, deflate, br, zstd'. The 'Accept-Language' header is 'en-US,en;q=0.9'. The bottom status bar shows the Python version as 3.11 (networkPart2).

The screenshot shows the PyCharm IDE interface with the 'Server' tool window open. The 'Run' tab is selected, displaying captured network traffic. The first entry is an 'OK' response with the IP Address '(127.0.0.1, 62914)'. The 'Received Sentence' is 'GET /styles.css HTTP/1.1'. The 'Host' header is 'localhost:5698'. The 'Connection' header is 'keep-alive'. The 'sec-ch-ua-platform' header is 'Windows'. The 'User-Agent' header is 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.0.0 Safari/537.36'. The 'sec-ch-ua' header is 'Google Chrome";v="131", "Chromium";v="131", "Not_A_Brand";v="24". The 'sec-ch-ua-mobile' header is '0'. The 'Accept' header is 'text/css,*/*;q=0.1'. The 'Sec-Fetch-Site' header is 'same-origin'. The 'Sec-Fetch-Mode' header is 'no-cors'. The 'Sec-Fetch-Dest' header is 'style'. The 'Referer' header is 'http://localhost:5698/ar'. The 'Accept-Encoding' header is 'gzip, deflate, br, zstd'. The 'Accept-Language' header is 'en-US,en;q=0.9'. The second entry is another 'OK' response with the IP Address '(127.0.0.1, 62919)'. The 'Received Sentence' is 'GET /ar HTTP/1.1'. The 'Host' header is 'localhost:5698'. The 'Connection' header is 'keep-alive'. The 'sec-ch-ua-platform' header is 'Windows'. The 'User-Agent' header is 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.0.0 Safari/537.36'. The 'sec-ch-ua' header is 'Google Chrome";v="131", "Chromium";v="131", "Not_A_Brand";v="24". The 'sec-ch-ua-mobile' header is '0'. The 'Accept' header is 'text/css,*/*;q=0.1'. The 'Sec-Fetch-Site' header is 'same-origin'. The 'Sec-Fetch-Mode' header is 'no-cors'. The 'Sec-Fetch-Dest' header is 'style'. The 'Referer' header is 'http://localhost:5698/ar'. The 'Accept-Encoding' header is 'gzip, deflate, br, zstd'. The 'Accept-Language' header is 'en-US,en;q=0.9'. The bottom status bar shows the Python version as 3.11 (networkPart2).

2.1.6 http://localhost:5698/Resources/Roaa.png request

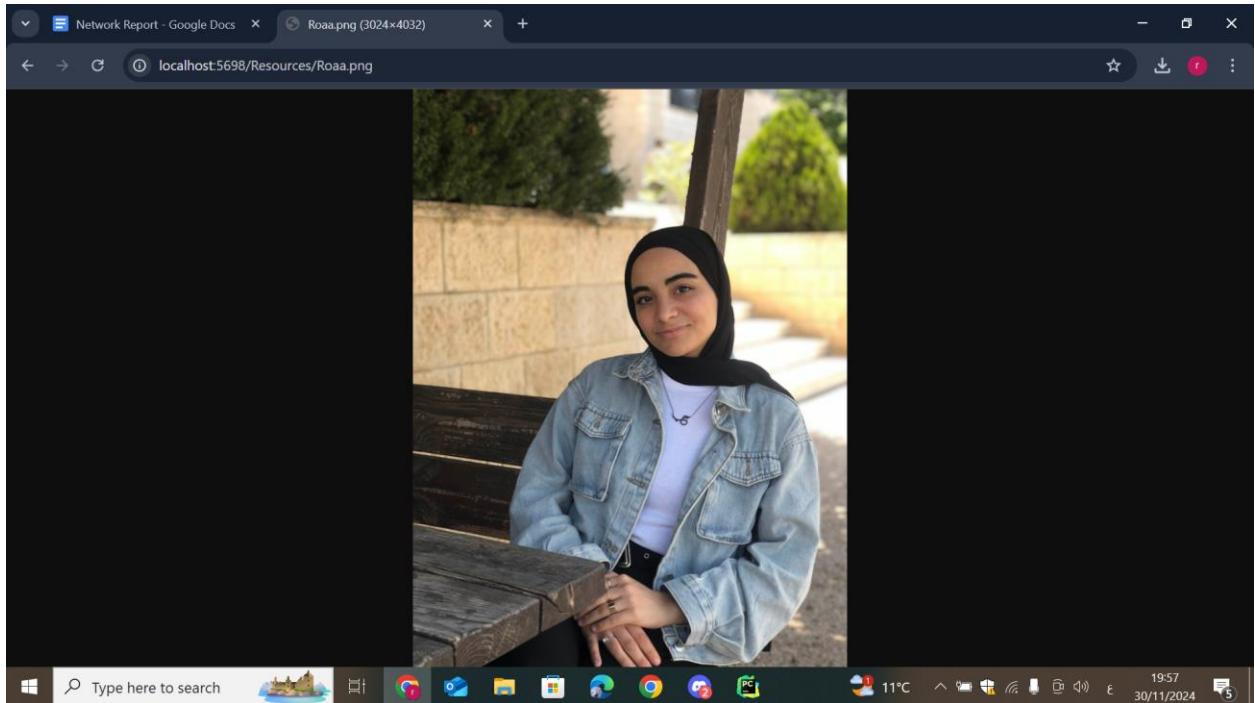
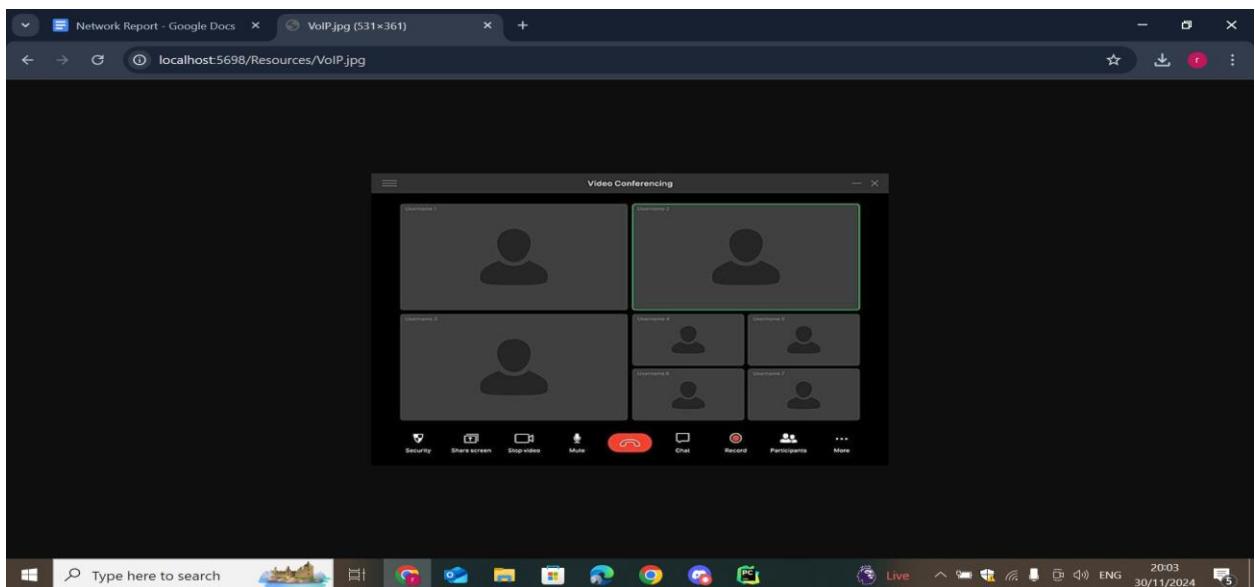
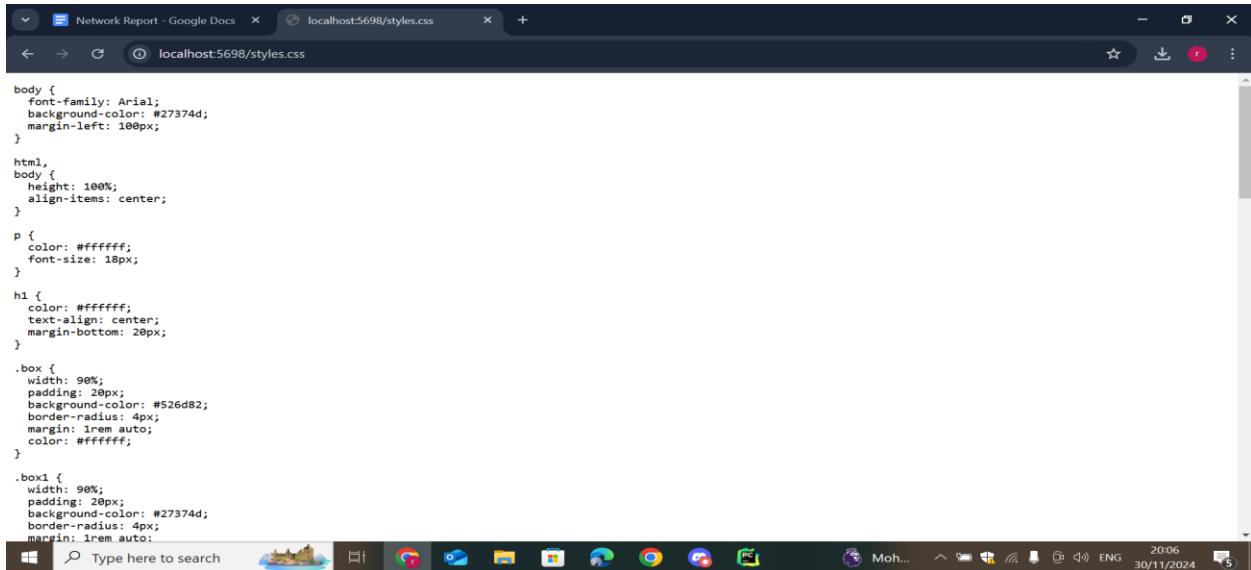


Figure 25 2.1.6

2.1.7 http://localhost:5698/Resources/VoIP.jpg request



2.1.8 http://localhost:5698/styles.css request



```
body {
    font-family: Arial;
    background-color: #27374d;
    margin-left: 100px;
}

html,
body {
    height: 100%;
    align-items: center;
}

p {
    color: #ffffff;
    font-size: 18px;
}

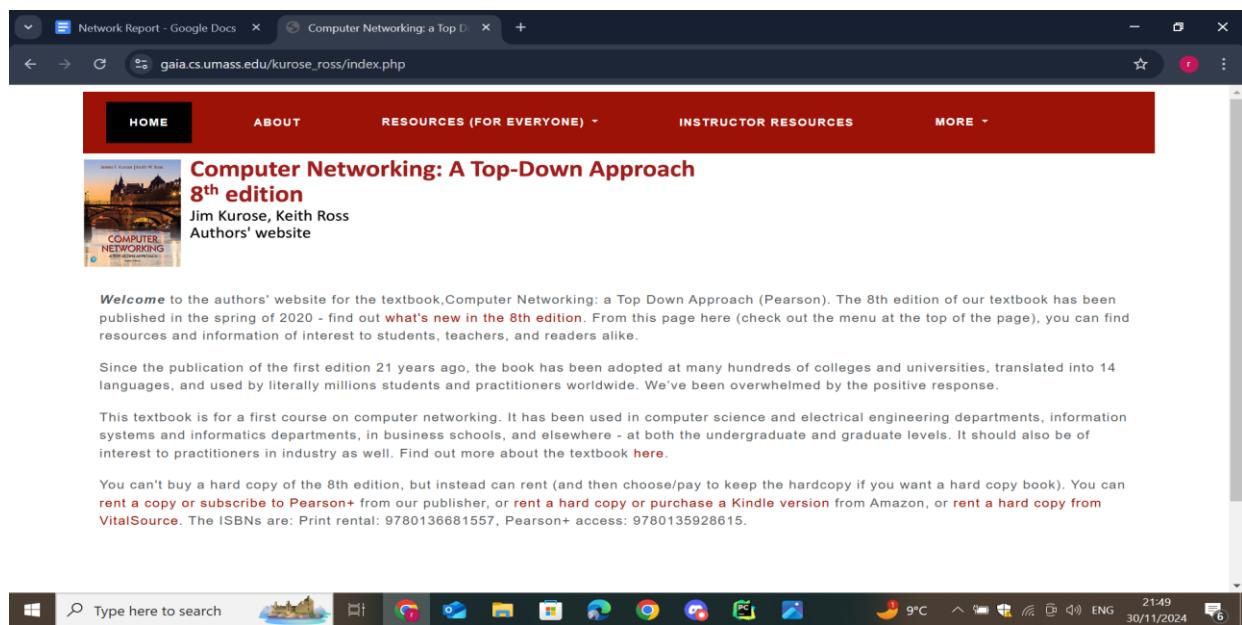
h1 {
    color: #ffffff;
    text-align: center;
    margin-bottom: 20px;
}

.box {
    width: 90%;
    padding: 20px;
    background-color: #526d82;
    border-radius: 4px;
    margin: 1rem auto;
    color: #ffffff;
}

.box1 {
    width: 90%;
    padding: 20px;
    background-color: #27374d;
    border-radius: 4px;
    margin: 1rem auto;
}
```

2.1.9 Links to external resources:

2.1.9.1 The textbook website



Welcome to the authors' website for the textbook, Computer Networking: a Top Down Approach (Pearson). The 8th edition of our textbook has been published in the spring of 2020 - find out what's new in the 8th edition. From this page here (check out the menu at the top of the page), you can find resources and information of interest to students, teachers, and readers alike.

Since the publication of the first edition 21 years ago, the book has been adopted at many hundreds of colleges and universities, translated into 14 languages, and used by literally millions of students and practitioners worldwide. We've been overwhelmed by the positive response.

This textbook is for a first course on computer networking. It has been used in computer science and electrical engineering departments, information systems and informatics departments, in business schools, and elsewhere - at both the undergraduate and graduate levels. It should also be of interest to practitioners in industry as well. Find out more about the textbook [here](#).

You can't buy a hard copy of the 8th edition, but instead can rent (and then choose/pay to keep the hardcopy if you want a hard copy book). You can [rent a copy or subscribe to Pearson+](#) from our publisher, or [rent a hard copy or purchase a Kindle version](#) from Amazon, or [rent a hard copy from VitalSource](#). The ISBNs are: Print rental: 9780136681557, Pearson+ access: 9780135928615.

2.1.9.2 Ritaj website

The screenshot shows the Ritaj - Birzeit University Academic and Administrative Portal. The main content area displays a login form with fields for 'Username' and 'Password', and a 'Forgot your password?' link. To the right is a photograph of a female student in a lab coat and mask looking through a microscope. The sidebar on the left contains sections for 'معلومات' (Information) and 'مدونات' (Blogs), listing various academic and administrative links. The right sidebar includes sections for 'روابط' (Links) and 'خدمات' (Services), such as the university's official account on ito and its electronic mail system.

2.1.10 Error 404 HTML page

The screenshot shows an 'HTTP/1.1 404 Not Found' error page. The title bar indicates the URL is 'localhost:5698/edksskkr'. The page content reads 'The file is not found !' and 'IP Address and Port: 127.0.0.1:63855'. The taskbar at the bottom shows the Windows Start button, a search bar, and various pinned application icons.

Figure 26 Error 404 HTML page

2.1.10.1 The status line "HTTP/1.1 404 Not Found"

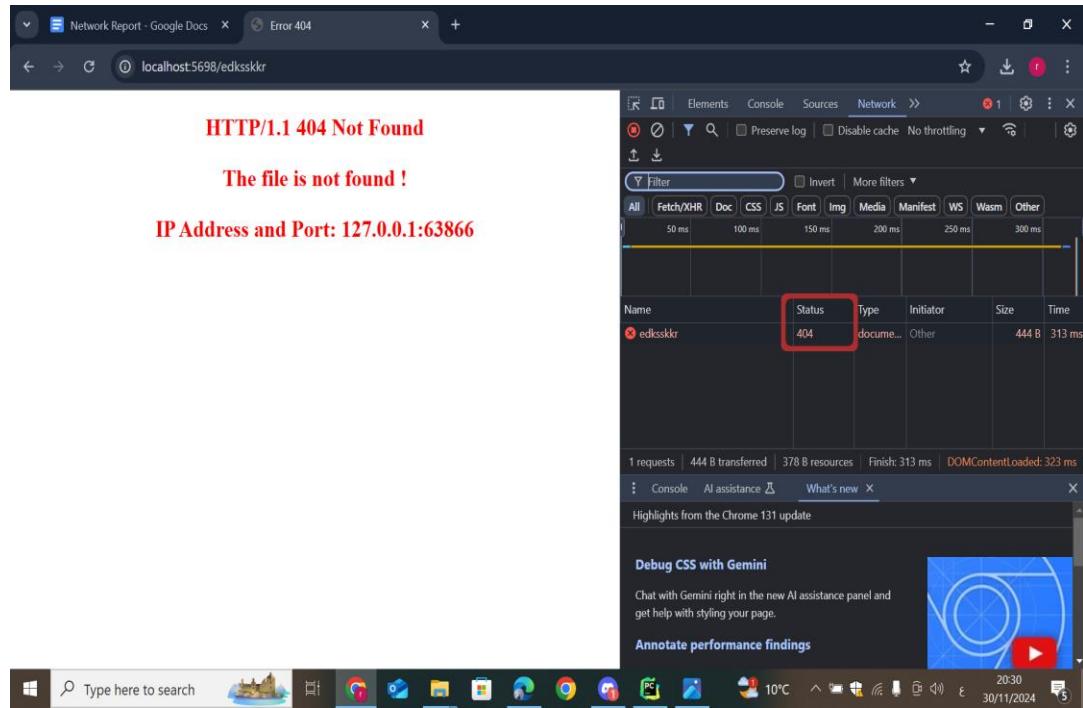


Figure 27 The status line "HTTP/1.1 404 Not Found"

2.1.11 Test the project from an external device

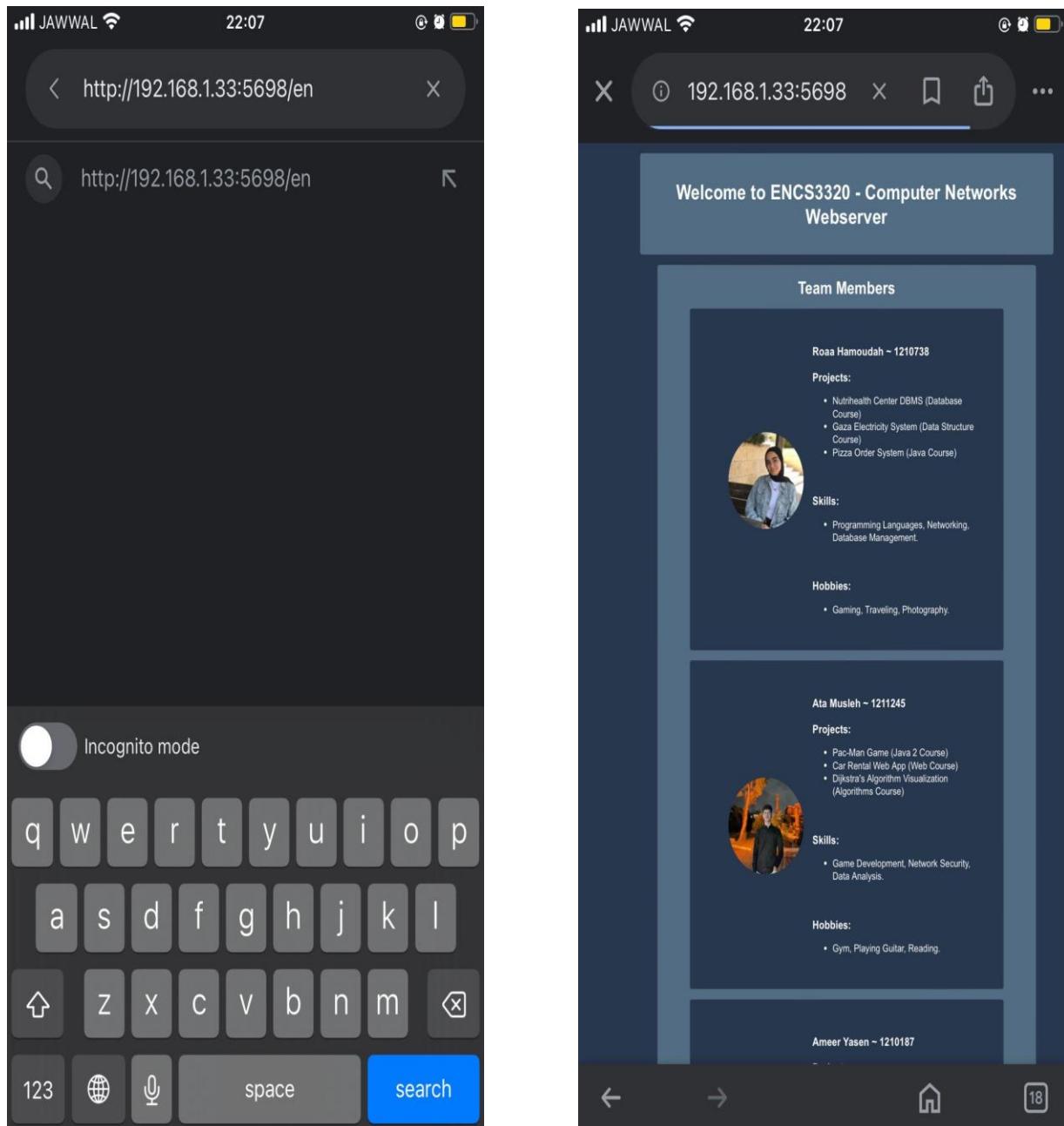


Figure 28 view from external device

The image consists of two side-by-side screenshots from a mobile device displaying a web application interface.

Screenshot 1 (Left):

- Header: JAWWAL 22:04
- Address bar: http://192.168.1.33:5698/ar
- Content area:
 - Profile card for student 1210738 (Roudy Hmoudeh):
 - ال المشاريع:
 - نظام إدارة قاعدة بيانات مركز التغذية (دوره قادمة البيانات)
 - نظام الكورسات في نزرة (دوره مهاكل البيانات)
 - نظام طلب الميزا (دوره جاما)
 - ال المهارات:
 - إلغاء الرياحنة، الشبكات، إدارة قواعد البيانات.
 - الأداء، السلر، التصوير.
 - Photo of Roudy Hmoudeh.
 - Profile card for student 1211245 (Husna Mousa):
 - ال المشاريع:
 - لغة باتل مان (دوره جاما 2)
 - تطبيق تثبيت السيارات (دوره الويب)
 - تصوير خوارزمية ديكسترا (دوره الخوارزميات)
 - ال المهارات:
 - تطوير الألعاب، أمن الشبكات، تحليل البيانات.
 - الدخول إلى المسألة الرياضية، المرف، على الجبار، القراءة.
 - Photo of Husna Mousa.
 - Profile card for student 1210187 (Ahmed Yousif):
 - ال المشاريع:
 - الدخول إلى المسألة الرياضية، المرف، على الجبار، القراءة.
 - Photo of Ahmed Yousif.
- Bottom navigation: back, forward, home, refresh, search, and a page number indicator (18).

Screenshot 2 (Right):

- Header: JAWWAL 22:04
- Address bar: 192.168.1.33:5698
- Content area:

مرحباً بكم في ENCS3320 - خادم الويب لشبكات الكمبيوتر

أعضاء الفريق

 - 1210738 - رويد حمودة
 - 1211245 - هنسا موسى
 - 1210187 - احمد يلسين

2.1.11.1 Include a Screenshot of the HTTP Request Output

The screenshot shows the PyCharm IDE interface with the code editor displaying a Python script named `Server.py`. The terminal window below it shows the output of the script's execution. The output is a series of HTTP requests and responses. A specific line in the request section is highlighted with a red box: "IP Address : ('192.168.1.40', 56676)". This indicates the server is listening on port 56676. The rest of the output shows standard HTTP headers and responses.

```
C:\Users\karakoonNet\PycharmProjects\networkPart2\venv\Scripts\python.exe C:\Users\karakoonNet\PycharmProjects\networkPart2\Server.py
The web server is ready to receive
IP Address : ('192.168.1.40', 56676)
Received Sentence : GET /en HTTP/1.1
Host: 192.168.1.33:5698
Upgrade-Insecure-Requests: 1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
User-Agent: Mozilla/5.0 (iPhone; CPU iPhone OS 16_7 like Mac OS X) AppleWebKit/605.1.15 (KHTML, like Gecko) GSA/343.0.695551749 Mobile/15E148 Safari
Accept-Language: en-US,en;q=0.9
Accept-Encoding: gzip, deflate
Connection: keep-alive

OK
IP Address : ('192.168.1.40', 56677)
Received Sentence : GET /styles.css HTTP/1.1
Host: 192.168.1.33:5698
Connection: keep-alive
Accept: text/css,*/*;q=0.1
User-Agent: Mozilla/5.0 (iPhone; CPU iPhone OS 16_7 like Mac OS X) AppleWebKit/605.1.15 (KHTML, like Gecko) GSA/343.0.695551749 Mobile/15E148 Safari
Accept-Language: en-US,en;q=0.9
Referer: http://192.168.1.33:5698/en
Accept-Encoding: gzip, deflate
```

This screenshot is nearly identical to the one above, showing the same PyCharm interface and terminal output. The highlighted line "IP Address : ('192.168.1.40', 56687)" is again present, indicating the server is listening on port 56687. The output shows two separate HTTP requests, one for the root path and one for the styles.css file.

```
C:\Users\karakoonNet\PycharmProjects\networkPart2\venv\Scripts\python.exe C:\Users\karakoonNet\PycharmProjects\networkPart2\Server.py
The web server is ready to receive
IP Address : ('192.168.1.40', 56687)
Received Sentence : GET /ar HTTP/1.1
Host: 192.168.1.33:5698
Upgrade-Insecure-Requests: 1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
User-Agent: Mozilla/5.0 (iPhone; CPU iPhone OS 16_7 like Mac OS X) AppleWebKit/605.1.15 (KHTML, like Gecko) GSA/343.0.695551749 Mobile/15E148 Safari
Accept-Language: en-US,en;q=0.9
Accept-Encoding: gzip, deflate
Connection: keep-alive

OK
IP Address : ('192.168.1.40', 56688)
Received Sentence : GET /styles.css HTTP/1.1
Host: 192.168.1.33:5698
Connection: keep-alive
Accept: text/css,*/*;q=0.1
User-Agent: Mozilla/5.0 (iPhone; CPU iPhone OS 16_7 like Mac OS X) AppleWebKit/605.1.15 (KHTML, like Gecko) GSA/343.0.695551749 Mobile/15E148 Safari
Accept-Language: en-US,en;q=0.9
Referer: http://192.168.1.33:5698/ar
Accept-Encoding: gzip, deflate
```

3 Part3 UDP Client-Server Trivia Game

This section shows the implementation and results of the UDP Trivia Game, outlining the design choices, observed behavior, and encountered challenges. The game is designed to allow multiple clients to connect to a main server and compete by answering random questions in a limited time range.

The trivia game utilizes UDP sockets protocol for communication between the client and server. UDP's connectionless nature allows for simpler implementation and is suitable for this application so many players can join the game and leave without creating multiple connections that always stay connected and take a lot of resources.

.

3.1 Theory and Procedure

The trivia game utilizes UDP sockets protocol for communication between the client and server. UDP's connectionless nature simplifies the implementation and suits this application where some packet loss is acceptable. Unlike TCP, which requires dedicated connections for each client, UDP's lightweight approach reduces server overhead, allowing for better scalability with a larger number of players joining and leaving dynamically without the overhead of maintaining persistent connections.

3.2 Server Implementation:

The server manages client connections, broadcasts questions, receives and evaluates answers, tracks scores, and declares the winner. Key components include:

Client Management: A dictionary (`active_clients`) stores active client information, mapping their (IP, port) to their chosen usernames. This dictionary is dynamically updated as players join and leave.

Game Logic: The `handle_game_round()` function orchestrates the game flow. It selects random questions from a predefined list of questions and then broadcasts them to the players, receives answers within a 15-second timeframe, evaluates the answers, and updates scores based on the scoring logic.

Scoring System: The game implements a dynamic scoring mechanism. Faster correct answers get higher points: 10 points for answers within 5 seconds, 5 points between 5 and 10 seconds, and 2 points for answers after 10 seconds but within the 15-second round window. This adds a time-pressure element to the game.

Broadcasting: The `broadcast_message()` function efficiently disseminates messages (questions, scores, game status updates) to all connected clients.

Disconnection Handling: The server checks client connections using a short timeout on the `sendto()` function. This mechanism finds unresponsive clients and removes them from the `active_clients` list and

the scores dictionary, ensuring the game progresses smoothly even with intermittent client disconnections.

Client Implementation:

The client connects to the server by the ip address for the server and port number, sends a username, receives and displays questions, submits answers within the allocated time, and displays the score updates. It continuously listens for messages from the server, ensuring responsiveness to game events. Upon connection, the client sends its desired username to the server, which then adds it to the active_clients list. If a username is already taken, the client needs to enter a different name.

Description of server.py (found in .zip file):

The server and client source code in Python:

3.2.1 Server.py

```
client.py  server.py x
C:\Users\HP\Desktop\networkProgramming>task3>Task3_Ata> server.py > ...
1 import socket
2 import random
3 import time
4 import threading
5
6 # Game| questions
7 questions = [
8     ("What is the capital of Palestine?", "Jerusalem"),
9     ("What is the best football club in the world?", "Real Madrid"),
10    ("How many minutes are in a full week?", "10800"),
11    ("How many dots are on one six-sided die?", "21"),
12    ("What is the nickname of Walter White in Breaking Bad?", "Heisenberg"),
13    ("In which year did World War I begin?", "1914"),
14    ("What are diamonds made of?", "Carbon")
15 ]
16
17 server_ip = '0.0.0.0'
18 server_port = 5698
19 buffer_size = 2048
20 active_clients = {}
21 scores = {}
22 game_active = False
23
24 server_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
25 server_socket.bind((server_ip, server_port))
26
27 print(f"Trivia Game Server is up and listening on IP {socket.gethostbyname(socket.gethostname())}, port {server_port}")
28
29 def broadcast_message(message):
30     """Send a message to all active clients."""
31     for client in active_clients:
32         server_socket.sendto(message.encode(), client)
33
34 def remove_disconnected_clients():
35     """Remove clients that are no longer responding."""
36     for client in list(active_clients):
37         try:
38             server_socket.settimeout(1)
39             server_socket.sendto("Connection check".encode(), client)
40         except Exception:
41             print(f"Client {active_clients[client]} has disconnected.")
42             del active_clients[client]
43             del scores[client]
```

Figure 29 Server.py source code

- First import the needed libraries for the game like the socket library for client-server communication and threading for a smoother game experience and making the game logic spirit from the communication

logic then use the time library for time calculation and the random for far game rules and question distribution

- A predefined list of random general knowledge questions with their answers stored in a dictionary (key-value pairs)
- **Declared variables :**

`server_ip = '0.0.0.0'` means that the server will listen for incoming connections on all available network interfaces on the machine.

`server_port = 5698`: The port number the server will listen on.

`buffer_size = 2048`: The maximum size of data that can be received at once.

`active_clients = {}`: A dictionary to store connected clients (address: name) pairs.

`scores = {}`: A dictionary to keep track of player scores.

`game_active = False`: A flag to indicate whether a game is currently running.

- Socket Creation and Binding:

```
24  server_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
25  server_socket.bind((server_ip, server_port))
```

Creates a UDP socket. `socket.AF_INET` specifies the **IPv4** address family and `socket.SOCK_DGRAM` indicates a **UDP** socket.

Binds the socket to the specified IP address and port, making the server start listening for incoming connections on that address and port.

- Broadcast message function:

This function sends a message to all currently connected clients.

`def broadcast_message(message):` Defines the function name and takes a message (a string) as input.

`for client in active_clients::` This loop iterates through the keys (client addresses) of the active_clients dictionary. Active_clients is a dictionary where keys are client addresses (IP, port) and values are client names.

`server_socket.sendto(message.encode(), client):` encode the message to binary and send it to the client using UDP socket server_socket, encoding is UTF-8 by default.

- Remove disconnected clients function:

The remove_disconnected_clients() function identifies and removes unresponsive clients from the game. It loops through a [copy](#) of connected clients (copy of active_clients list). For each client, it send "[Connection check](#)" message with a [1-second timeout](#) (`server_socket.settimeout(1)`). If the send fails the client information is removed from both the active_clients and scores dictionaries. A message is printed to the console confirming the disconnection.

```

44
45     def handle_game_round():
46         """Conduct a round of trivia, asking questions and handling scoring."""
47         global game_active
48         game_active = True
49
50         if len(active_clients) < 2:
51             print("Not enough players. Waiting for more clients...")
52             return
53
54         broadcast_message("Game starting now! Get ready!")
55         time.sleep(2)
56
57         round_scores = {client: 0 for client in active_clients}
58         used_questions = []
59         total_questions = len(questions)
60
61         for i in range(total_questions):
62             remove_disconnected_clients()
63
64             available_questions = [q for q in questions if q not in used_questions]
65             if not available_questions:
66                 break
67
68             question, answer = random.choice(available_questions)
69             used_questions.append((question, answer))
70
71             print(f"Broadcasting Question {i + 1}/{total_questions}: {question}")
72             broadcast_message(f"Question {i + 1}/{total_questions}: {question}")
73
74             start_time = time.time()
75             answered_clients = {}
76             fastest_client = None
77             fastest_time = float('inf')
78
79             while time.time() - start_time < 15:
80                 try:
81                     server_socket.settimeout(1)
82                     message, client_address = server_socket.recvfrom(buffer_size)
83                     response = message.decode().strip().lower()
84
85                     if client_address in active_clients:
86                         if client_address not in answered_clients:
87                             answered_clients[client_address] = response
88                         if response == answer.lower():
89                             time_elapsed = time.time() - start_time
90                             points = 10 if time_elapsed <= 5 else 5 if time_elapsed <= 10 else 2
91                             round_scores[client_address] += points
92                             scores[client_address] += points
93                             if time_elapsed < fastest_time:
94                                 fastest_time = time_elapsed
95                                 fastest_client = client_address
96
97                 except socket.timeout:
98                     continue
99
100            for client in active_clients:
101                if client not in answered_clients:
102                    broadcast_message(f"(active_clients[{client}]) did not answer in time.")
103
104            broadcast_message(f"Time's up! The correct answer was: {answer}")
105
106            leaderboard = "\n".join([f"{active_clients[client]}: {scores[client]} points" for client in scores])
107            broadcast_message(f"Current Scores:\n{leaderboard}")
108            print(f"Leaderboard after Question {i + 1}")
109            print(leaderboard)
110            time.sleep(5)
111
112            game_active = False
113            max_score = max(scores.values(), default=0)
114            winners = [active_clients[client] for client in scores if scores[client] == max_score]
115
116            if winners:
117                winner_names = ", ".join(winners)
118                broadcast_message(f"Game Over! The winners are: {winner_names} with {max_score} points!")
119            else:
120                broadcast_message("Game Over! No winner this time.")
121
122            final_scores = "\n".join([f"{active_clients[client]}: {scores[client]} points" for client in scores])
123            broadcast_message(f"Final Scores:\n{final_scores}")
124            print("Final Scores:")
125            print(final_scores)

```

Figure 30 server.py source code 2

- Handling the game round function:

The handle_game_round() function runs a complete trivia round. It first checks for sufficient players (2 or more) and then broadcasts a start message.

```
45 def handle_game_round():
46     """Conduct a round of trivia, asking questions and handling scoring."""
47     global game_active
48     game_active = True
49
50     if len(active_clients) < 2:
51         print("Not enough players. Waiting for more clients...")
52         return
53
54     broadcast_message("Game starting now! Get ready!")
55     time.sleep(2)
56
57     round_scores = {client: 0 for client in active_clients}
58     used_questions = []
59     total_questions = len(questions)
```

The function iterates through available questions, selecting one randomly and sending it to all clients, and checking for disconnected clients before each question. available_questions filter out questions already asked. A random question (question, answer) is selected from the remaining available questions. The chosen question is added to used_questions to prevent repetition. The question is broadcast with its number to all clients.

```
61     for i in range(total_questions):
62         remove_disconnected_clients()
63
64         available_questions = [q for q in questions if q not in used_questions]
65         if not available_questions:
66             break
67
68         question, answer = random.choice(available_questions)
69         used_questions.append((question, answer))
70
71         print(f"Broadcasting Question {i + 1}/{total_questions}: {question}")
72         broadcast_message(f"Question {i + 1}/{total_questions}: {question}")
```

A [15-second timer starts](#), during this time the server is listening for client answers. `server_socket.settimeout(1)` [sets a 1-second timeout on the socket to prevent blocking indefinitely](#).

`server_socket.recvfrom(buffer_size)` receives a client's answer and their address. If the client hasn't already answered, the client's response is recorded in `answered_clients`. If the response is correct, the time taken is calculated. Points are given based on correctness and response time, with bonuses for faster answers ([10 points for answering in less than 5 seconds](#), [5 points for answering within 10 seconds](#), and [two points for any answer after that](#)).

`except socket.timeout:` Handles timeouts if no message is received within 1 second, allowing the loop to continue listening.

```

73     start_time = time.time()
74     answered_clients = {}
75     fastest_client = None
76     fastest_time = float('inf')
77
78     while time.time() - start_time < 15:
79         try:
80             server_socket.settimeout(1)
81             message, client_address = server_socket.recvfrom(buffer_size)
82             response = message.decode().strip().lower()
83
84             if client_address in active_clients:
85                 if client_address not in answered_clients:
86                     answered_clients[client_address] = response
87                 if response == answer.lower():
88                     time_elapsed = time.time() - start_time
89                     points = 10 if time_elapsed <= 5 else 5 if time_elapsed <= 10 else 2
90                     round_scores[client_address] += points
91                     scores[client_address] += points
92                     if time_elapsed < fastest_time:
93                         fastest_time = time_elapsed
94                         fastest_client = client_address
95
96         except socket.timeout:
97             continue

```

After the 15-second answer window closes, the game server processes the results of the question. It notifies all players who failed to submit an answer within the time limit. Then, the correct answer is displayed to everyone. A leaderboard, displaying the current scores of all players, is broadcasted to all clients and printed on the server console. Finally, a 5-second pause is before the next question begins, allowing players a short break to review the results and prepare for the next round.

```

99     for client in active_clients:
100         if client not in answered_clients:
101             broadcast_message(f"{active_clients[client]} did not answer in time.")
102
103             broadcast_message(f"Time's up! The correct answer was: {answer}")
104
105             leaderboard = "\n".join([f"{active_clients[client]}: {scores[client]} points" for client in scores])
106             broadcast_message("Current Scores:\n" + leaderboard)
107             print(f"Leaderboard after Question {i + 1}")
108             print(leaderboard)
109             time.sleep(5)

```

The game round is finished. It deactivates the game flag and determines the highest score. A list of winners is displayed by naming players whose scores match the highest score. The winning score is broadcast to

all clients. Otherwise, a "no winner" message is sent. Finally, a formatted string of final scores is created and broadcast to all players, and also printed to the server console for review.

```
111     game_active = False
112     max_score = max(scores.values(), default=0)
113     winners = [active_clients[client] for client in scores if scores[client] == max_score]
114
115     if winners:
116         winner_names = ", ".join(winners)
117         broadcast_message(f"Game Over! The winners are: {winner_names} with {max_score} points!")
118     else:
119         broadcast_message("Game Over! No winner this time.")
120
121     final_scores = "\n".join([f"{active_clients[client]}: {scores[client]} points" for client in scores])
122     broadcast_message("Final Scores:\n" + final_scores)
123     print("Final Scores:")
124     print(final_scores)
```

- Main server logic and new client handling:

The code defines two main functions: handle_new_client and server_main. handle_new_client manages new client connections, storing their usernames and initializing their scores. It checks for duplicate usernames before accepting a new client, preventing conflicts. If a game is already in progress, new clients are informed they will join the next round. Otherwise, if enough players are present, a new game round is initiated in a separate thread using handle_game_round. This threading is crucial for preventing the server from blocking while a game is running, allowing it to continue accepting new connections.

The server_main function drives the server's operation. It continuously listens for incoming client connections and messages within an infinite loop. When a client connects and sends their username, server_main calls handle_new_client to process the connection. A try-except block handles keyboard interrupts, allowing for graceful server shutdown. Closing the server socket releases the port and resources.

```
126 # new Client can join but an error will occur on the server side!!
127 def handle_new_client(client_address, username):
128     active_clients[client_address] = username
129     scores[client_address] = 0
130     print(f"New client joined: {username} from {client_address}")
131     broadcast_message(f"{username} has joined the game.")
132
133     # If a game is in progress, put the new player on the wait list and add them on the next round
134     if game_active:
135         server_socket.sendto("A game is currently in progress. You'll join the next round.".encode(), client_address)
136     else:
137         if len(active_clients) >= 2:
138             # Start a new game round in a separate thread
139             game_thread = threading.Thread(target=handle_game_round)
140             game_thread.start()
141
142 def server_main():
143     global game_active
144     while True:
145         try:
146             message, client_address = server_socket.recvfrom(buffer_size)
147             username = message.decode().strip()
148             handle_new_client(client_address, username)
149         except KeyboardInterrupt:
150             print("Shutting down server...")
151             break
152
153     server_socket.close()
154
155 server_main()
```

Figure 31 server.py source code 3

3.2.2 client.py



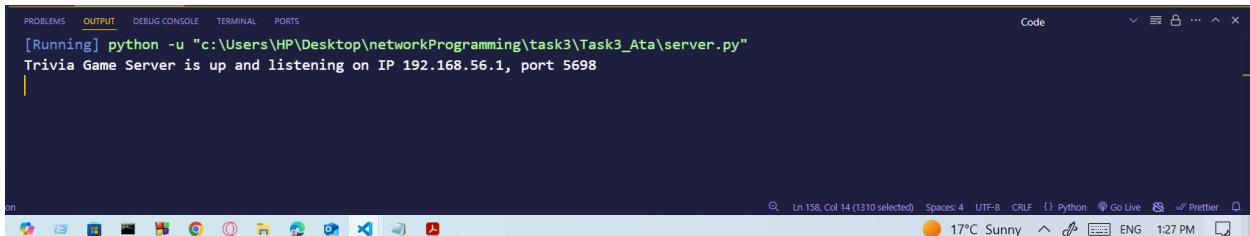
```
client.py > ...
1 import socket
2
3 buffer_size = 2048
4
5 server_ip = input("Enter the server IP address: ")
6 server_port = int(input("Enter the server port: "))
7
8 client_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
9
10 username = input("Enter your username: ")
11 client_socket.sendto(username.encode(), (server_ip, server_port))
12
13 print(f"Connected to the Trivia Game Server at {server_ip}:{server_port}")
14
15 try:
16     while True:
17         message, _ = client_socket.recvfrom(buffer_size)
18         decoded_message = message.decode()
19         print("\n" + decoded_message)
20
21         if "Question" in decoded_message:
22             answer = input("Your answer (press Enter to skip): ").strip()
23             client_socket.sendto(answer.encode(), (server_ip, server_port))
24
25 except KeyboardInterrupt:
26     print("\nExiting the game...")
27 finally:
28     client_socket.close()
```

Figure 32 Client.py source code

This code implements a basic trivia game client using the UDP sockets protocol. The client first enters the server's IP address and port number, then creates a UDP socket and sends the user's chosen username to the server. It then enters a main loop where it continuously listens for messages from the server. Upon receiving a message, it displays it to the user. If the message contains a question, the client prompts the user for their answer and sends it back to the server. The client handles keyboard

interrupts gracefully, allowing the user to exit the game with Ctrl+C. A final block ensures the socket is always closed properly.

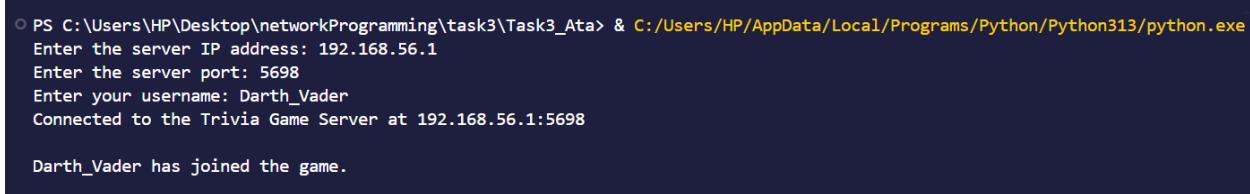
3.2.3 live game experience



A screenshot of a terminal window in Visual Studio Code. The tab bar at the top shows 'PROBLEMS', 'OUTPUT' (which is selected), 'DEBUG CONSOLE', 'TERMINAL', and 'PORTS'. The status bar at the bottom shows the date and time as '12:27 PM'. The main area of the terminal displays the output of a Python script:

```
[Running] python -u "c:\Users\HP\Desktop\networkProgramming\task3\Task3_Ata\server.py"
Trivia Game Server is up and listening on IP 192.168.56.1, port 5698
```

The game starts when running the server code, it prints the IP address for the machine and the port number and then starts listening for clients requests to join the game



A screenshot of a terminal window in Visual Studio Code. The tab bar at the top shows 'PROBLEMS', 'OUTPUT' (selected), 'DEBUG CONSOLE', 'TERMINAL' (which is selected), and 'PORTS'. The status bar at the bottom shows the date and time as '12:27 PM'. The main area of the terminal displays the output of a Python script:

```
PS C:\Users\HP\Desktop\networkProgramming\task3\Task3_Ata> & C:/Users/HP/AppData/Local/Programs/Python/Python313/python.exe
Enter the server IP address: 192.168.56.1
Enter the server port: 5698
Enter your username: Darth_Vader
Connected to the Trivia Game Server at 192.168.56.1:5698

Darth_Vader has joined the game.
```

First client joined, The client Enters the server ip address and port number and his username in the game, in our case: Darth Vader (from Starwars)

The server response with {username} has joined the game.



A screenshot of a terminal window in Visual Studio Code. The tab bar at the top shows 'PROBLEMS', 'OUTPUT' (selected), 'DEBUG CONSOLE', 'TERMINAL' (selected), and 'PORTS'. The status bar at the bottom shows the date and time as '12:27 PM'. The main area of the terminal displays the output of a Python script:

```
Enter your username: Heisenberg
Connected to the Trivia Game Server at 192.168.56.1:5698

Heisenberg has joined the game.

Game starting now! Get ready!

Connection check
```

A new client also joined the game with the user name Heisenberg (from Breaking Bad)

The server broadcasts a message that the game will begin, and then the handle round function starts checking if all players are still responding with each round

The server dictates a new thread to run the handle round function so the

game continues receiving clients' requests to join the game while the round is running.

Server terminal output:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Code ▾ ⌂ ⌂ ⌂ ⌂ ⌂ ⌂ ⌂ ⌂ ⌂ [Running] python -u "c:\Users\HP\Desktop\networkProgramming\task3\Task3_Ata\server.py"
[Running] python -u "c:\Users\HP\Desktop\networkProgramming\task3\Task3_Ata\server.py"
Trivia Game Server is up and listening on IP 192.168.56.1, port 5698
New client joined: Darth_Vader from ('192.168.56.1', 57436)
New client joined: Heisenberg from ('192.168.56.1', 57491)
```

First-round server terminal output:

```
New client joined: Heisenberg from ('192.168.56.1', 57491)
Broadcasting Question 1/7: How many dots are on one six-sided die?
Leaderboard after Question 1
Darth_Vader: 0 points
Heisenberg: 0 points
```

Client terminal (Darth Vader):

Question 1/7: How many dots are on one six-sided die?

Your answer (press Enter to skip):

Darth_Vader did not answer in time.

Heisenberg did not answer in time.

Time's up! The correct answer was: 21

Current Scores:

Darth_Vader: 0 points

Heisenberg: 0 points

The rest of rounds server terminal output (until the game finished):

```
Leaderboard after Question 1
Darth_Vader: 0 points
Heisenberg: 0 points
Broadcasting Question 2/7: What is the nickname of Walter White in Breaking Bad?
Leaderboard after Question 2
Darth_Vader: 0 points
Heisenberg: 10 points
Broadcasting Question 3/7: What are diamonds made of?
Leaderboard after Question 3
Darth_Vader: 5 points
Heisenberg: 10 points
Broadcasting Question 4/7: What is the best football club in the world?
Leaderboard after Question 4
Darth_Vader: 10 points
Heisenberg: 10 points
Broadcasting Question 5/7: What is the capital of Palestine?
Leaderboard after Question 5
Darth_Vader: 15 points
Heisenberg: 12 points
Broadcasting Question 6/7: In which year did World War I begin?
Leaderboard after Question 6
Darth_Vader: 15 points
Heisenberg: 22 points
Broadcasting Question 7/7: How many minutes are in a full week?
Leaderboard after Question 7
Darth_Vader: 20 points
Heisenberg: 24 points
Final Scores:
Darth_Vader: 20 points
Heisenberg: 24 points
```

[Done] exited with code=1 in 210.619 seconds

Figure 33 server terminal game output

For Darth_Vader:

```
Question 2/7: What is the nickname of Walter White in Breaking Bad?
Your answer (press Enter to skip): Heisenberg
Time's up! The correct answer was: Heisenberg

Current Scores:
Darth_Vader: 0 points
Heisenberg: 10 points

Connection check

Question 3/7: What are diamonds made of?
Your answer (press Enter to skip): Carbon
Heisenberg did not answer in time.

Time's up! The correct answer was: Carbon

Current Scores:
Darth_Vader: 5 points
Heisenberg: 10 points

Connection check

Question 4/7: What is the best football club in the world?
Your answer (press Enter to skip): Real Madrid
Time's up! The correct answer was: Real Madrid

Current Scores:
Darth_Vader: 10 points
Heisenberg: 10 points

Connection check

Question 5/7: What is the capital of Palestine?
Your answer (press Enter to skip): Jerusalem
Time's up! The correct answer was: Jerusalem

Current Scores:
Darth_Vader: 15 points
```

Figure 34 Client terminal output

In question 2 Heisenberg answered the question in less than 5 seconds so he got 10 points after his score was 0 from the last round

But Darth_Vader does not answer and just press ok to skip the question. In question 3 Darth_Vader answers the question after the first 5 seconds but before 10 seconds pass so he has got 5 points while Heisenberg did not answer and just skipped the question.

Same thing in question 4, Darth_Vader gets 5 points and nothing for Heisenberg

In question 5 Darth_Vader gets 5 points for answering within 10 seconds and Heisenberg gets only two because he answered after 2 seconds!

The screenshot shows a terminal window titled "Task3_Ata". The terminal output is as follows:

```
File Edit Selection View Go Run Terminal Help ← → Task3_Ata
EXPLORER PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
TASK3_ATA
client.py
server.py
Heisenberg: 12 points
Connection check
Question 6/7: In which year did World War I begin?
Your answer (press Enter to skip):
Time's up! The correct answer was: 1914
Current Scores:
Darth Vader: 15 points
Heisenberg: 22 points
Connection check
Question 7/7: How many minutes are in a full week?
Your answer (press Enter to skip): 10800
Time's up! The correct answer was: 10800
Current Scores:
Darth Vader: 20 points
Heisenberg: 24 points
Game Over! The winners are: Heisenberg with 24 points!
Question 7/7: How many minutes are in a full week?
Your answer (press Enter to skip): 10800
Time's up! The correct answer was: 10800
Current Scores:
Darth Vader: 20 points
Heisenberg: 24 points
Game Over! The winners are: Heisenberg with 24 points!
Final Scores:
Darth Vader: 20 points
Heisenberg: 24 points
```

The terminal shows a game between two players, Heisenberg and Darth Vader. The game consists of two rounds. In each round, a question is asked, and the player who answers correctly within 10 seconds gets points. The final scores are: Darth Vader: 20 points and Heisenberg: 24 points. Heisenberg wins the game.

The rest of the game status from The client Darth_Vader Terminal
Heisenberg won the game after 7 rounds

The Terminal output for Heisenberg:

The image shows two side-by-side terminal windows in Visual Studio Code, both titled "Task3_Ata".

Terminal 1 (Top):

```
Question 2/7: What is the nickname of Walter White in Breaking Bad?  
Your answer (press Enter to skip): Heisenberg  
Time's up! The correct answer was: Heisenberg  
Current Scores:  
Darth Vader: 0 points  
Heisenberg: 10 points  
Connection check  
Question 3/7: What are diamonds made of?  
Your answer (press Enter to skip):  
Heisenberg did not answer in time.  
Time's up! The correct answer was: Carbon  
Current Scores:  
Darth Vader: 5 points  
Heisenberg: 10 points  
Connection check  
Question 4/7: What is the best football club in the world?  
Your answer (press Enter to skip): Real Madrid  
Time's up! The correct answer was: Real Madrid  
Current Scores:  
Darth Vader: 10 points  
Heisenberg: 10 points  
Connection check  
Question 5/7: What is the capital of Palestine?  
Your answer (press Enter to skip): Jerusalem  
Time's up! The correct answer was: Jerusalem  
Current Scores:  
Darth Vader: 15 points
```

Terminal 2 (Bottom):

```
Question 5/7: What is the capital of Palestine?  
Your answer (press Enter to skip): Jerusalem  
Time's up! The correct answer was: Jerusalem  
Current Scores:  
Darth Vader: 15 points  
Heisenberg: 12 points  
Connection check  
Question 6/7: In which year did World War I begin?  
Your answer (press Enter to skip): 1914  
Time's up! The correct answer was: 1914  
Current Scores:  
Darth Vader: 15 points  
Heisenberg: 22 points  
Connection check  
Question 7/7: How many minutes are in a full week?  
Your answer (press Enter to skip): 10800  
Time's up! The correct answer was: 10800  
Current Scores:  
Darth Vader: 20 points  
Heisenberg: 24 points  
Game Over! The winners are: Heisenberg with 24 points!  
Final Scores:  
Darth Vader: 20 points  
Heisenberg: 24 points
```

Both terminals show the same command-line interface with tabs for EXPLORER, PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL (which is selected), and PORTS. The status bar at the bottom of each terminal window indicates "Ln 12, Col 74 (10 selected)", "Spaces: 4", "UTF-8", "CRLF", "Python 3.13.0 64-bit", "Go Live", "Prettier", and other standard VS Code icons.

4 Alternative Solutions, Issues, and Limitations

4.1 Alternative Solutions

A key challenge was handling concurrent client joins mid-game. The initial single-threaded server design blocked new connections while a round was active, as the server couldn't process new requests while managing the game. To resolve this, threading was introduced, handling new connections via `handle_new_client` concurrently with gameplay. This improved responsiveness but introduced potential race conditions with shared resources like active clients and scores.

4.2 Issues and Challenges

4.2.1 Web Server

- Ensuring cross-browser compatibility can be challenging.
- Error handling is crucial to prevent unexpected behavior.

4.2.2 Trivia Game

- Network congestion can lead to message loss or delays.

4.3 Limitations

The question database is hardcoded within the server script, restricting the game's flexibility and scalability. Adding or modifying questions requires code changes and server restarts, making content updates cumbersome. The reliance on UDP, while beneficial for responsiveness, inherently risks packet loss, which can lead to missing questions, answers, or score updates, impacting gameplay and fairness. The basic disconnection handling, using timeouts, may not be robust enough in all network environments, potentially leading to delayed removal of inactive players or incorrect score calculations. The threading model, while improving concurrency, lacks robust synchronization mechanisms, creating a vulnerability to race conditions when multiple clients interact simultaneously, particularly during scoring and connection updates. Finally, the server's capacity is limited by its single-process, threaded architecture, hindering scalability for a larger number of players or more complex game features. These limitations could lead to performance bottlenecks or unexpected behavior under heavy load.

Teamwork

Name	ID	Tasks
Roaa	1210738	<p>Task 1: created the Definitions and tested and interrupted the Commands</p> <p>Task 2 (Code): Developed HTML and CSS for main_en.html and main_ar.html and its logic in the server code.</p> <p>Task 2 (Report): Authored the first two sections of the report, providing a detailed overview of the project's objectives, literature review, and related technologies.</p> <p>Task 2 (Code): Created the error HTML page and its logic.</p>
Ameer	1210187	<p>Task 2 (Supporting Materials): Developed HTML and CSS for Arabic and English pages (Supporting_Materials_en and Supporting_Materials_ar), implemented server redirection logic (HTTP status 307), and handled resource requests.</p> <p>Task 2 (Report): Contributed to Task 2 of the report.</p> <p>Task 3 (Report): Contributed to the main report structure and tables.</p>
Ata	1211245	<p>Task 3 (Code): Implemented both server-side and client-side code.</p> <p>Contributed to Report: Helped with resource collection and management and contributed to the report.</p>