Faculty of Engineering & Technology

Electrical & Computer Engineering Department

COMPUTER NETWORKS - ENCS3320

First Project Report

"Socket Programming"

Prepared by:

Name: Lina Abufarha	**ID:** 1210241	**Section:** 2
Name: Arein Almasri	**ID:** 120596	**Section:** 1
Name: Raghed Isleem	**ID:** 1211326	**Section:** 1

Date: 2/12/2024

1.Abstract:

The main aim of this project is dealing with application-layer protocols such as HTTP and DNS. The first part of the project consists of defining and working with network diagnostic tools or network troubleshooting tools. Where each tool serves a purpose. The command line interface is used to explore these tools by the given test cases. Then Wireshark is used to capture some DNS messages created by doing a certain action on the network and observing the messages.

The second task involves building a simple web server using socket programming. The server delivers English and Arabic webpages with team details, a networking topic, and links to resources. It also supports a form for requesting images or videos, with unavailable requests redirected appropriately. The focus is on organizing files, handling HTTP requests, and ensuring proper server-client communication.

The third part This task involves creating a trivia game using UDP socket programming. The server manages the game by broadcasting questions, collecting answers from clients, and updating scores. Clients join the game, receive questions, and submit their answers within a set time. The task highlights server-client interaction and real-time data exchange.

Table of Contents:

Table of figures:

2. Task (1):

2.1 Description of the following commands are (ipconfig, ping, tracert, nslookup, telnet)

• **ipconfig**: Displays and manages a system's network configuration, including IP address, gateway, and DNS settings.

• **ping**: A command-line tool used to check if a server can be reached by measuring the round-trip time, packet time to live (TTL), and packet loss statistics.

• **tracert**: A command tool used to trace the route from a source to a destination by sending three packets to each device on the path and computing the round-trip time for each packet.

• **nslookup**: A command-line tool used to query the DNS to obtain domain name or IP address mapping or any other specific DNS record.

• **telnet**: A protocol that lets you connect to and control remote computers or systems over a network. Can access command line interface or virtual terminals. It works for both sending and receiving data in 8-bit format.

2.2 Use an Internet-connected computer to test the above commands on Windows OS

2.2.1 ipconfig a device in the same network

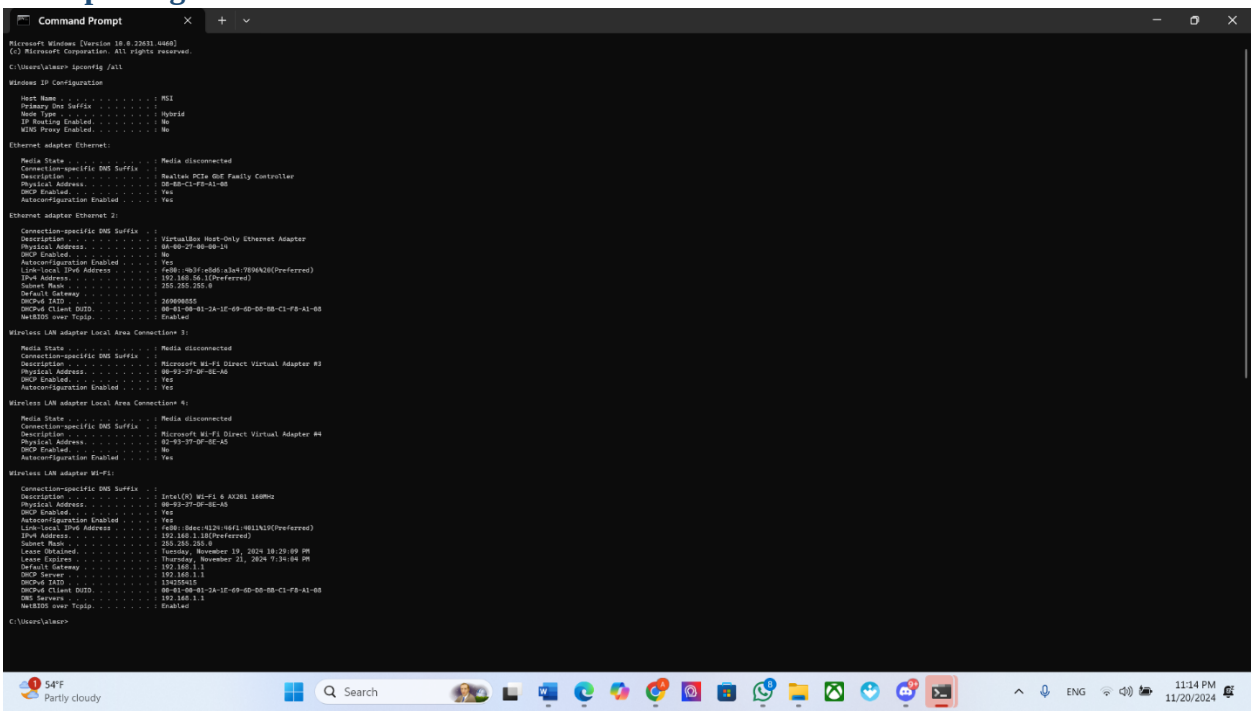


Figure 1: Results of using the command ipconfig

```
Wireless LAN adapter Wi-Fi:

   Connection-specific DNS Suffix  . :
   Description . . . . . . . . . . . : Intel(R) Wi-Fi 6 AX201 160MHz
   Physical Address. . . . . . . . . : 00-93-37-DF-8E-A5
   DHCP Enabled. . . . . . . . . . . : Yes
   Autoconfiguration Enabled . . . . : Yes
   Link-local IPv6 Address . . . . . : fe80::8dec:4124:46f1:4011%19(Preferred)
   IPv4 Address. . . . . . . . . . . : 192.168.1.18(Preferred)
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Lease Obtained. . . . . . . . . . : Tuesday, November 19, 2024 10:29:09 PM
   Lease Expires . . . . . . . . . . : Thursday, November 21, 2024 7:34:04 PM
   Default Gateway . . . . . . . . . : 192.168.1.1
   DHCP Server . . . . . . . . . . . : 192.168.1.1
   DHCPv6 IAID . . . . . . . . . . . : 134255415
   DHCPv6 Client DUID. . . . . . . . : 00-01-00-01-2A-1E-69-6D-D8-BB-C1-F8-A1-08
   DNS Servers . . . . . . . . . . . : 192.168.1.1
   NetBIOS over Tcpip. . . . . . . . : Enabled

C:\Users\almsr>
```

Figure 2: ipconfig command

The figure shows the network configuration of a Wi-Fi adapter ("Intel(R) Wi-Fi 6 AX201 160MHz") obtained via the ipconfig command. The adapter has a MAC address of "00-93-37-

DF-8E-A5" and is configured with DHCP enabled, assigning it the IPv4 address "192.168.1.18" (Preferred) with a subnet mask of "255.255.255.0." The default gateway and DNS server are both set to "192.168.1.1," indicating the router's address. The IP lease was obtained on November 19, 2024, at 10:29 PM and expires on November 21, 2024, at 7:34 PM. Additionally, NetBIOS over TCP/IP is enabled, allowing legacy network applications to function, confirming proper connectivity and configuration for network communication.

2.2.2 Ping

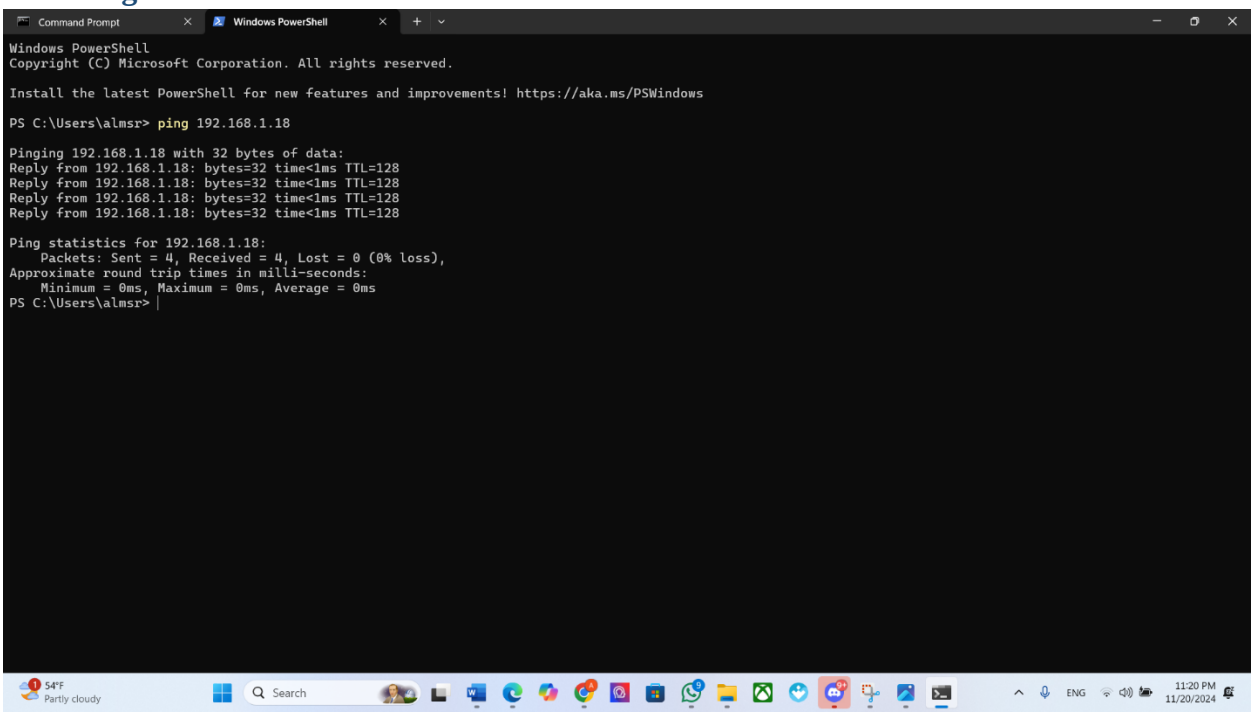


Figure 3: Pinging a smartphone from a laptop which are both connected to the same network

the computer pings the IP address 192.168.1.18, sending 4 packets, each containing 32 bytes of data. All packets are successfully received without any packet loss. The round-trip time (RTT) for these packets is reported as 0 milliseconds for minimum, maximum, and average times, indicating near-instantaneous communication with the device. The TTL (Time To Live) value of 128 implies that packets can traverse up to 128 network devices before being discarded. These results confirm that the target device at 192.168.1.18 is reachable and responds efficiently to requests from the computer.

2.2.3 Ping" discover.engineering.utoronto.ca"

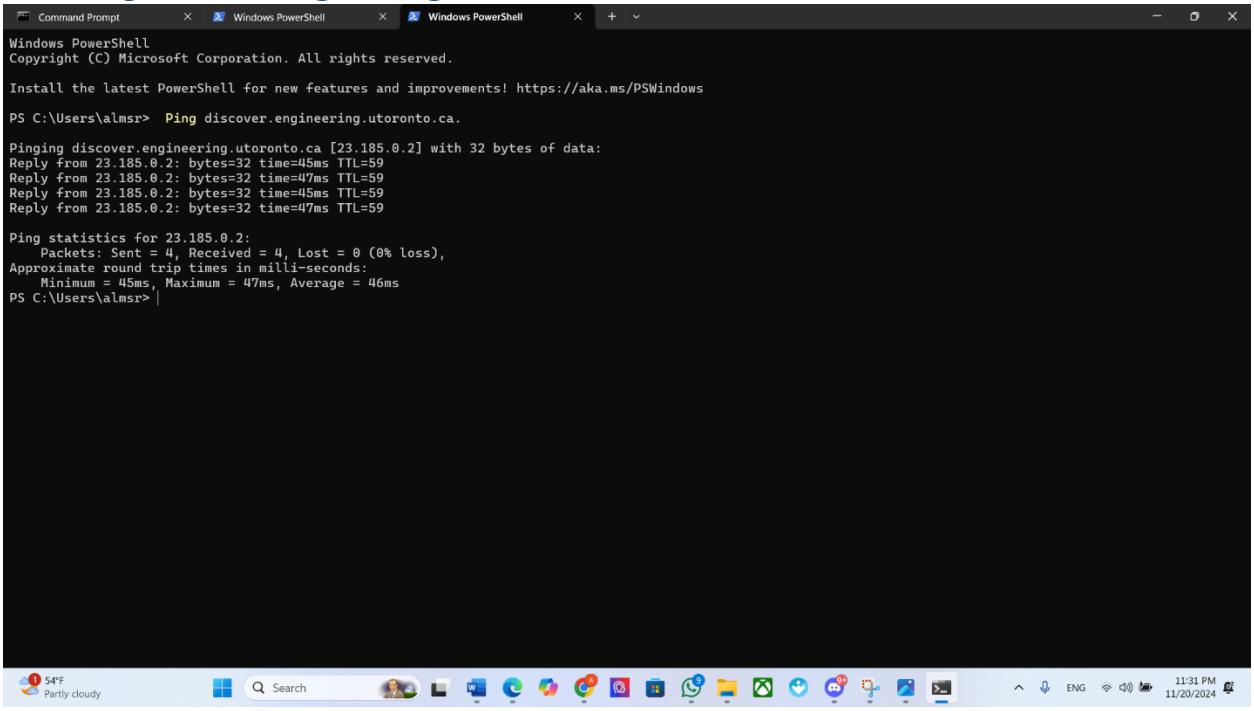


Figure 4: Ping result of "discover.engineering.utoronto.ca"

When discover.engineering.utoronto.ca was pinged, 4 packets were sent and successfully received, each with a round-trip time (RTT) averaging 46 ms and a TTL of 59. The higher RTT compared to devices within the same network suggests that the destination is located in a geographically distant network, likely in Canada. The IP address 23.185.0.2 confirms the server's remote nature. The relatively low TTL value indicates that the packets passed through several network devices or hops before reaching the destination.

2.2.4 Tracert “discover.engineering.utoronto.ca”

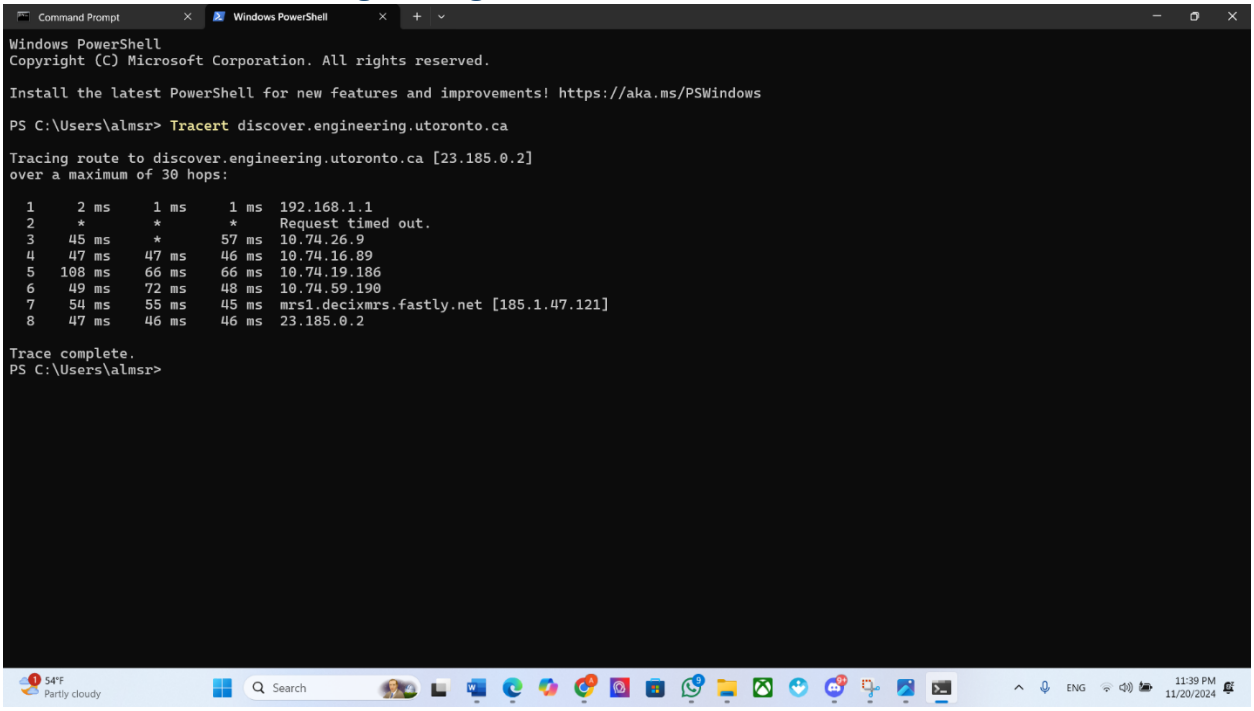


Figure 5: Results of using the command Tracert “discover.engineering.utoronto.ca”

As shown in above figure, the packets were routed through 7 hops to reach the destination, indicating that the three packets had to traverse 7 network devices or systems.

The first column represents the hop number, while the next three columns display the round-trip time (RTT) in milliseconds for each of the three packets transmitted during that hop. The last column specifies the destination IP or hostname of the network device handling the packets at that hop.

Notably, one of the hops is marked with an asterisk (*) and "Request timed out," indicating that the router at this point did not respond with a reply message within the predefined time limit. However, the traceroute successfully completed, reaching the destination host (discover.engineering.utoronto.ca).

2.2.5 nslookup “discover.engineering.utoronto.ca”

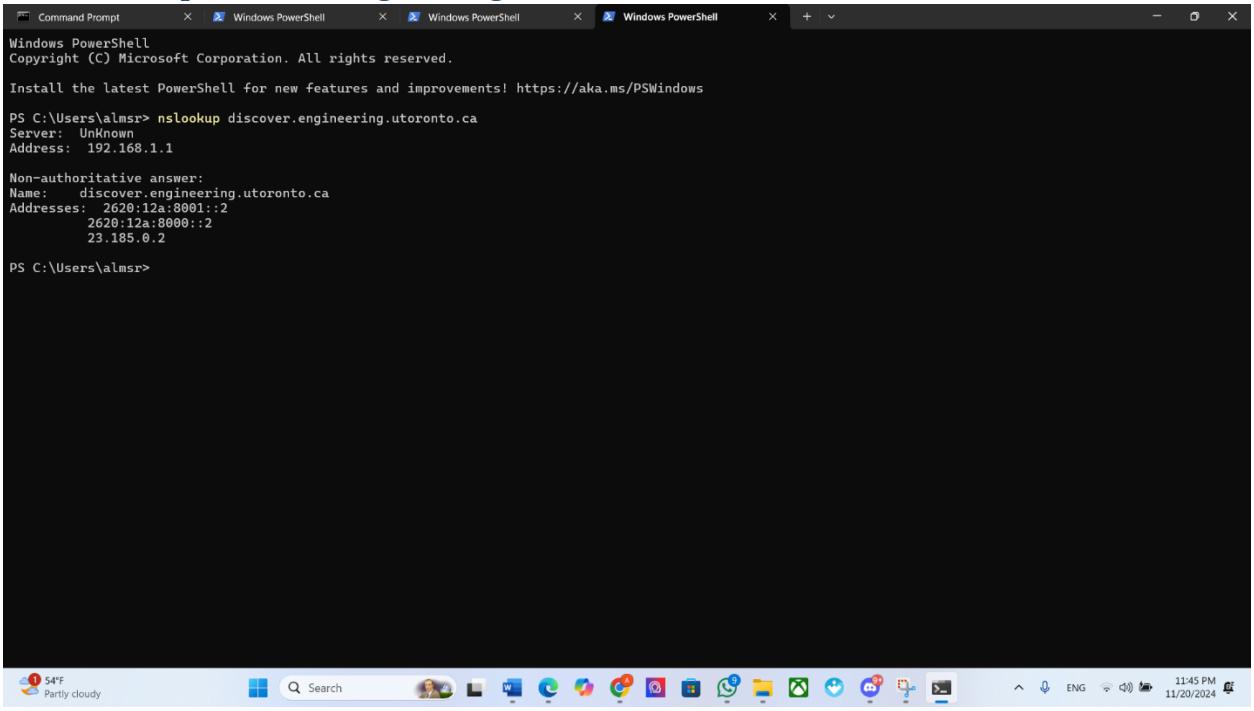


Figure 6: Results of using the command nslookup “discover.engineering.utoronto.ca”

The response includes a "non-authoritative answer," indicating that the DNS server used for this query is not the authoritative DNS server for the domain discover.engineering.utoronto.ca. The output lists the domain name and associated IP addresses: 2620:12a:8001::2 (IPv6), 2620:12a:8000:2 (IPv6), and 23.185.0.2 (IPv4).

These IP addresses can be used to access the resources linked to the domain discover.engineering.utoronto.ca. The inclusion of multiple IP addresses suggests redundancy or support for both IPv4 and IPv6 connectivity.

2.2.6 telnet “discover.engineering.utoronto.ca”

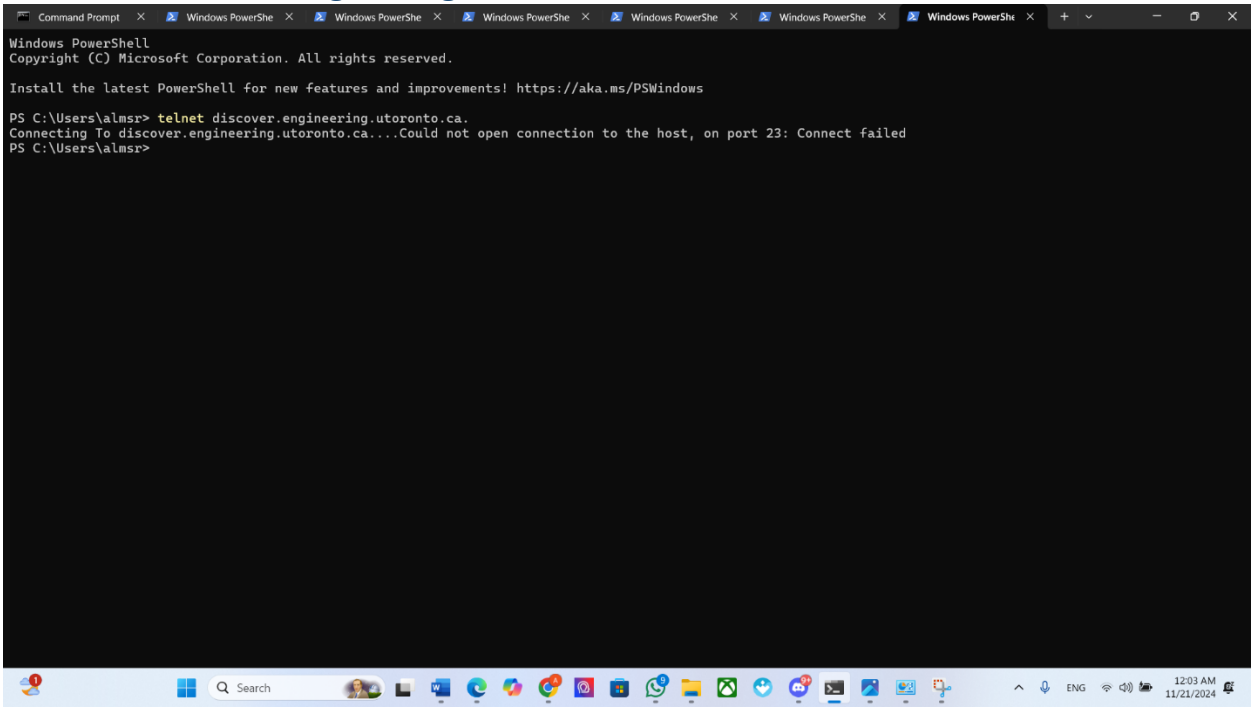


Figure 7: Results of using the command telnet “discover.engineering.utoronto.ca”

The figure shows the result of attempting to connect to discover.engineering.utoronto.ca on port 23 using the telnet command, which failed with the message: "Could not open connection to the host, on port 23: Connect failed." This indicates that the server is not running a Telnet service on port 23, or the connection is being blocked by a firewall or network restriction. Modern servers often disable Telnet due to security vulnerabilities, as it transmits data in plaintext, favoring secure alternatives like SSH.

2.3 Capture some DNS messages using “Wireshark”

Wireshark is an open- source network protocol analyzer widely used for capturing, analyzing, and troubleshooting network traffic. Developed by the Wireshark community, it provides a detailed and real-time view of data moving across a network, helping users understand the communication patterns between devices. Here we must use the wireshark to capture some DNS messages, we chose Wi-Fi, the network interface that corresponds to the network we want to monitor. To focus on DNS messages, we applied a display filter (DNS).

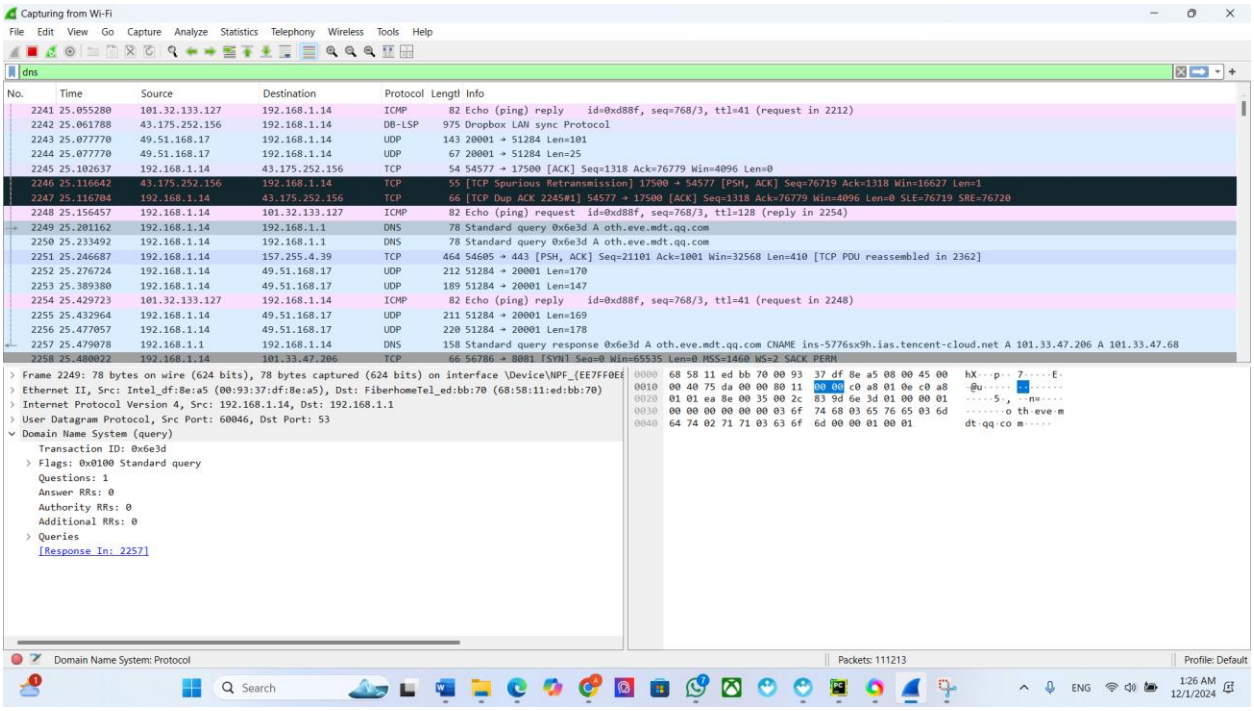


Figure 8: DNS query response message captured by Wireshark

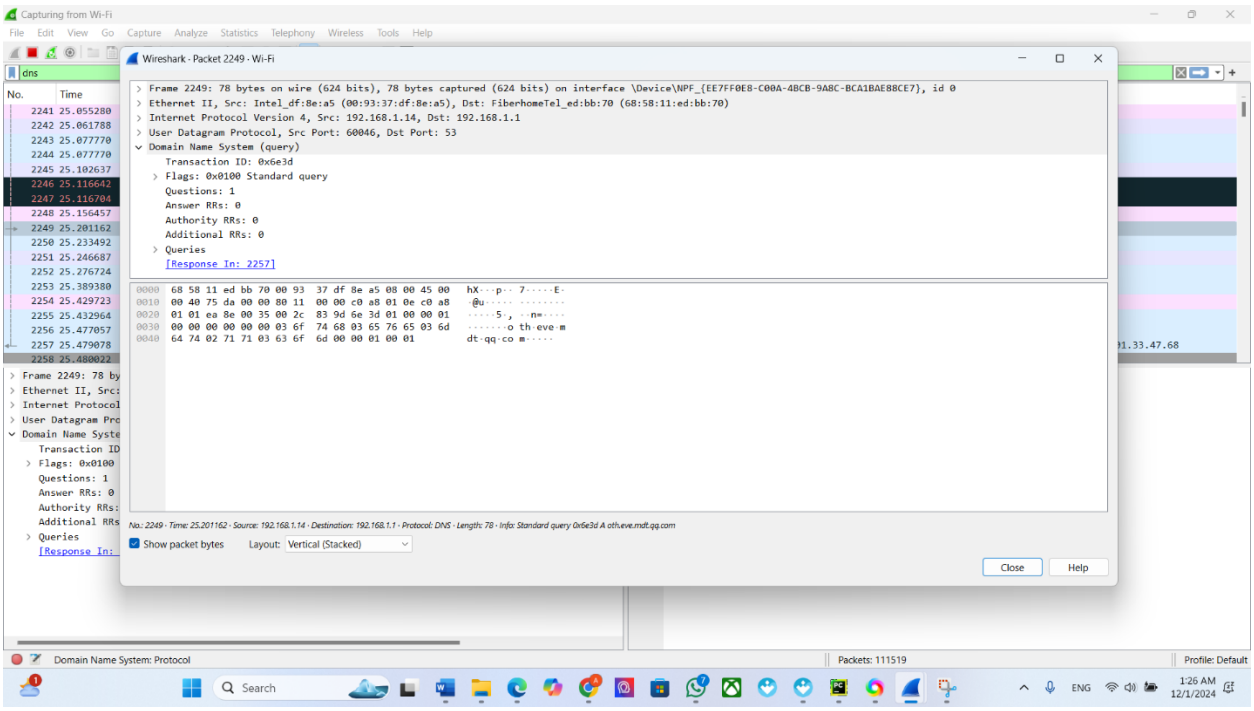


Figure 9: DNS query response message captured by Wireshark

As seen above, we have captured many DNS messages, each one of them has its own frame number that is a unique identifier for each captures packet, also there is a time that it's the time elapsed since the start of the capture, and the source and 6 destination IP addresses (or MAC addresses, depending on the layer) of the communication. The protocol Indicates the protocol used for the communication, such as TCP, UDP, or DNS, in addition the info column provides a summary of the packet content. For example, for DNS packets, you might see information about the DNS query or response.

Task 2 – TCP Theory and Procedure Socket Programming

The task utilizes the core principles of networking, specifically the HTTP protocol, socket programming, and multithreading:
HTTP Protocol: The server communicates with the client using the HTTP protocol. It sends responses such as HTTP/1.1 200 OK for successful requests, HTTP/1.1 307 Temporary Redirect for redirections, and HTTP/1.1 404 Not Found for missing resources.

Socket Programming: The server establishes a socket connection using Python's socket library. It binds to a specific IP address (127.0.0.1, or localhost) and port (5698) to listen for incoming requests.

Multithreading: The server handles multiple client connections simultaneously using the threading module. Each request is processed in a separate thread to ensure concurrent handling.

Initializing the Server, The server creates a socket, binds it to the localhost IP and port, and starts listening for incoming client connections.

```
104  def start_server():
105      if not os.path.exists(FILE_DIR):
106          os.makedirs(FILE_DIR)
107
108      server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
109      server_socket.bind((HOST, PORT))
110      server_socket.listen(5)
111      print("Server running on " + HOST + ":" + str(PORT))
112
113      while True:
114          client_socket, client_address = server_socket.accept()
115          print("Connected by: " + str(client_address))
116          threading.Thread(target=handleResponse, args=(client_socket, client_address)).start()
117
```

Figure 10: Web Server Startup Log

Handling Requests: When a client connects, the server receives the request and determines the requested file path. The request is parsed to extract the file_name parameter, and the file is retrieved from the files directory.

```
46  def handleResponse(client_socket, client_address):
47      try:
48          request = client_socket.recv(1024).decode('utf-8')
49          print("Request received from " + str(client_address) + ":\n" + request)
50
51          if "GET" in request:
52              lines = request.splitlines()
53              request_line = lines[0]
54              parts = request_line.split(" ")
55
```

Figure 11: Successful HTTP File Request

Responding to File Requests:
The server checks if the requested file exists:

Image or Video Available: If the file exists and is a valid type (e.g., .jpg, .png, .mp4), it constructs an HTML response to display the file using appropriate HTML tags (<img> or <video>).
File Not Found: If the file is unavailable, the server redirects the user to Google Images or YouTube with a 307 Temporary Redirect response.

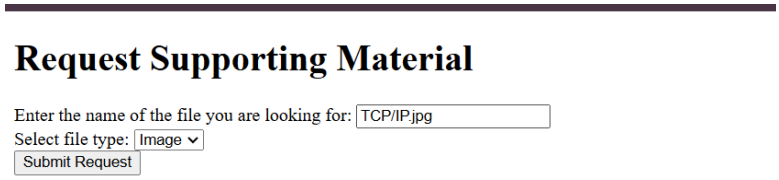


Figure 12:File Not Found Response

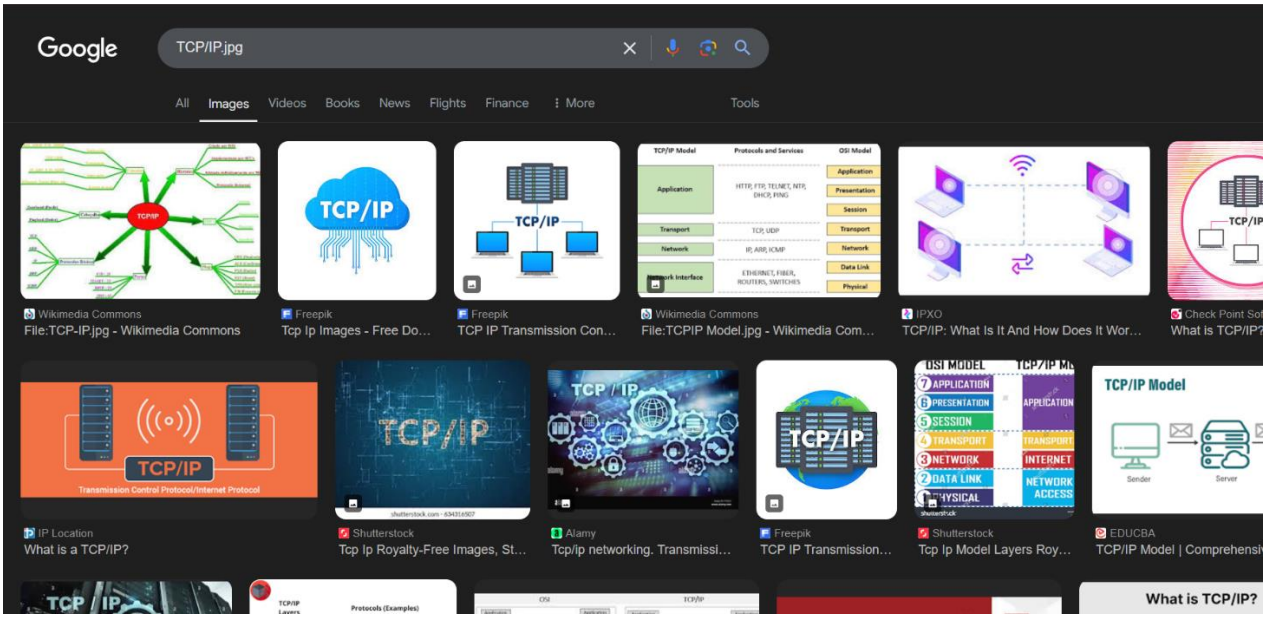


Figure 13: File Not Found Response

Request Supporting Material

Enter the name of the file you are looking for: TCP/IP.mp4
Select file type: Video
Submit Request

Figure 14: HTML Response for Video File

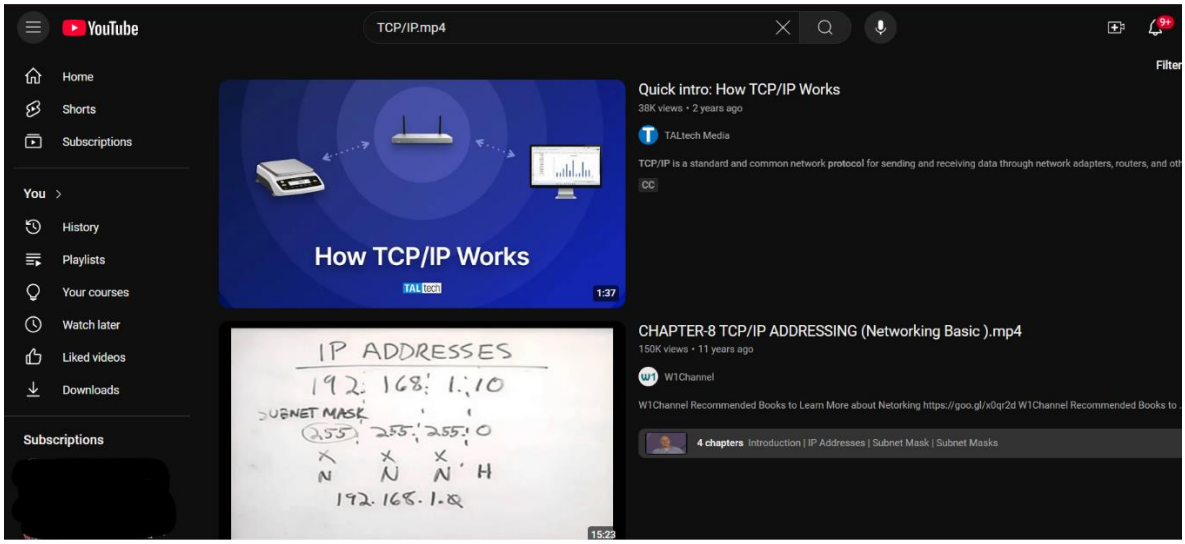


Figure 15: HTML Response for Video File

This URL contains several components,
https://www.google.com/search: This is the base URL for Google's search engine.
?q=file_name: This query parameter specifies the search keyword. The server appends the requested file's name to q=, allowing Google to search for the file.
&tbm=isch: This parameter tells Google to return image search results (tbm=isch stands for "Tableaux Mode Image Search").

The URL is dynamically constructed based on the file name provided by the user, making the server flexible in handling various search queries.

```
        if os.path.exists(file_path):
            if file_name.endswith((".jpg", ".png", ".jpeg")):
                content = (
                    "<html lang=\"ar\" dir=\"rtl\">"
                    if is_arabic
                    else "<html lang=\"en\">"
                )
```

```
    else:
        return "HTTP/1.1 400 Bad Request\r\n\r\n" + ("نوع الملف غير مدعوم!" if is_arabic else "Unsupported file type!")
else:
    if file_name.endswith((".jpg", ".png", ".jpeg")):
        return "HTTP/1.1 307 Temporary Redirect\r\nLocation: https://www.google.com/search?q=" + file_name + "&tbm=isch\r\n\r\n"
    elif file_name.endswith(".mp4"):
        return "HTTP/1.1 307 Temporary Redirect\r\nLocation: https://www.youtube.com/results?search_query=" + file_name + "\r\n\r
    else:
        return "HTTP/1.1 404 Not Found\r\n\r\n" + ("الملف المطلوب غير موجود!" if is_arabic else "File not found!")
```

The server uses two main endpoints to handle file requests:

/request_file: For English responses.
/request_ar_file: For Arabic responses.
When a user requests a file, the server checks whether the file exists in the files directory. If the file is found, it generates an HTML page to display the image or video. If the file is not found, it performs a URL redirection.

One of the main issues encountered was that even when the file existed in the local directory, the server sometimes redirected to Google instead of displaying the file. This issue arises due to errors in handling the path or parsing the file name correctly. Further debugging would be required to address this, such as ensuring proper URL decoding and file existence checks.
This is likely due to an error in the file existence check logic (os.path.exists(file_path)).

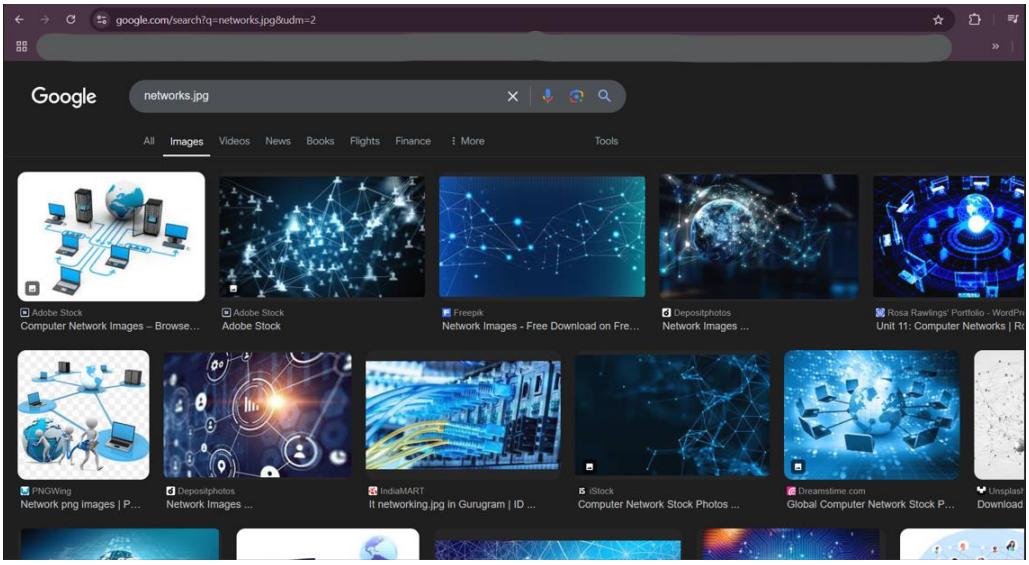
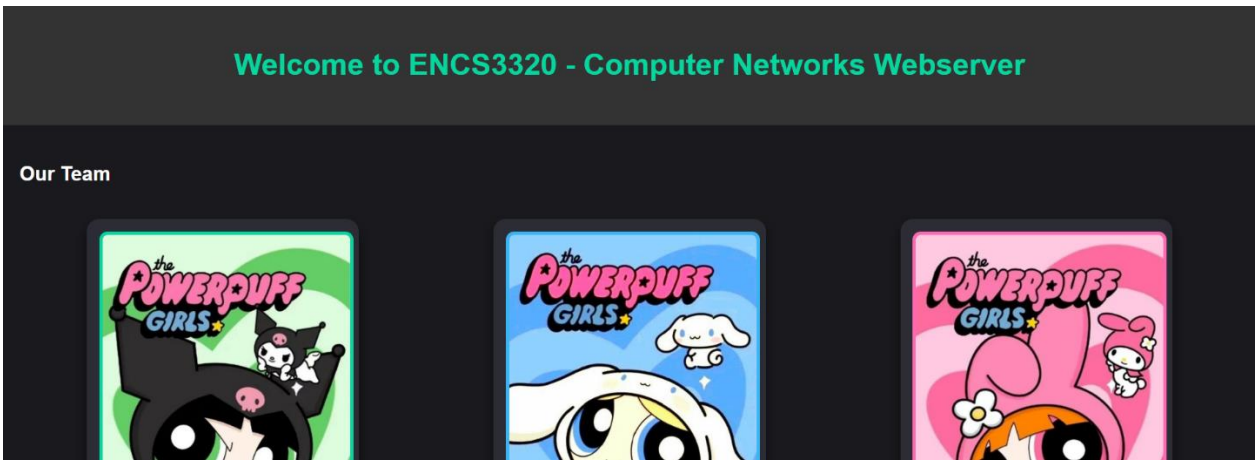
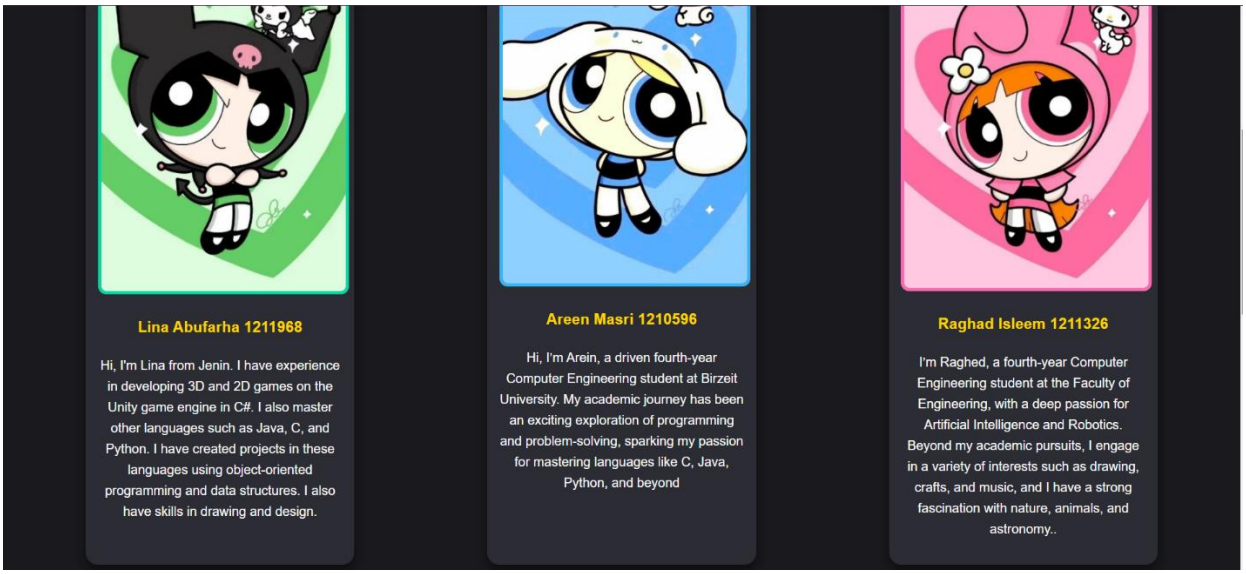


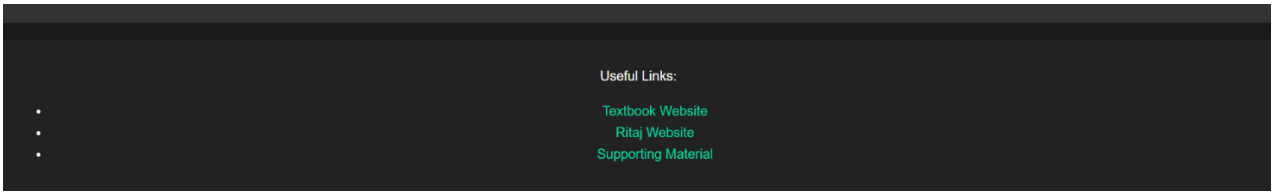
Figure 16: Generated Redirect URL

Components and Their Functions:

socket: Establishes communication between the server and clients.
threading: Allows the server to handle multiple client connections simultaneously.
os.path: Manages file path operations, checking for file existence.
unquote: Decodes URL-encoded file names.



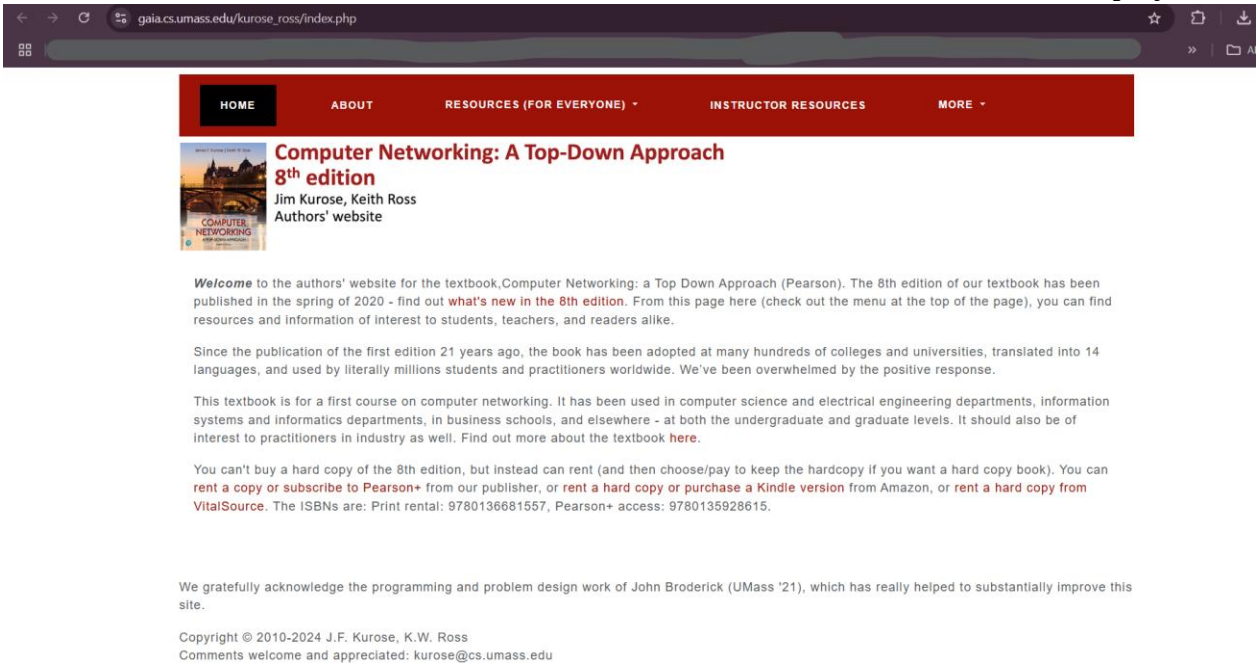




The Links:

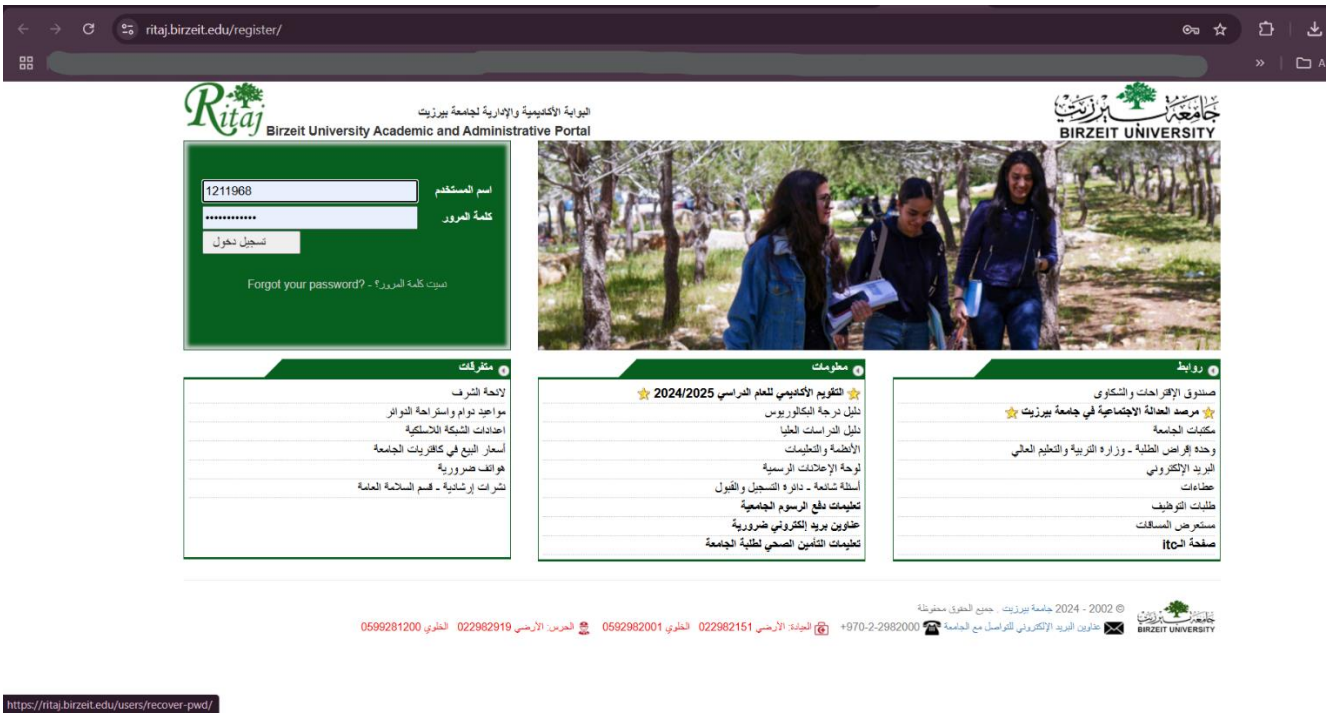
Textbook Website

This link redirects the user to the official website of the textbook used in the course or project.



Ritaj Website

This link leads to the "Ritaj" platform, used by students to access university resources, course materials, and academic information.



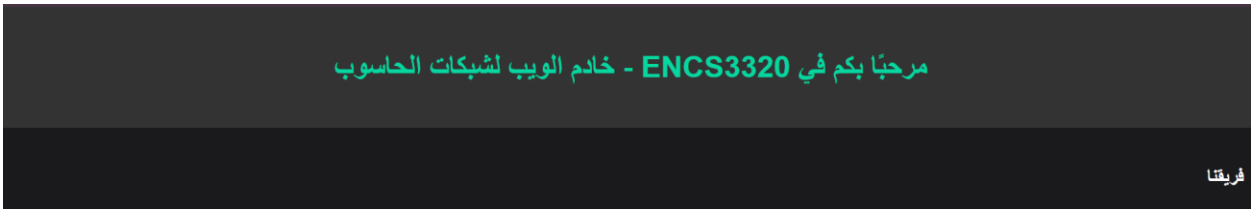
Supporting Materials (Image Search)

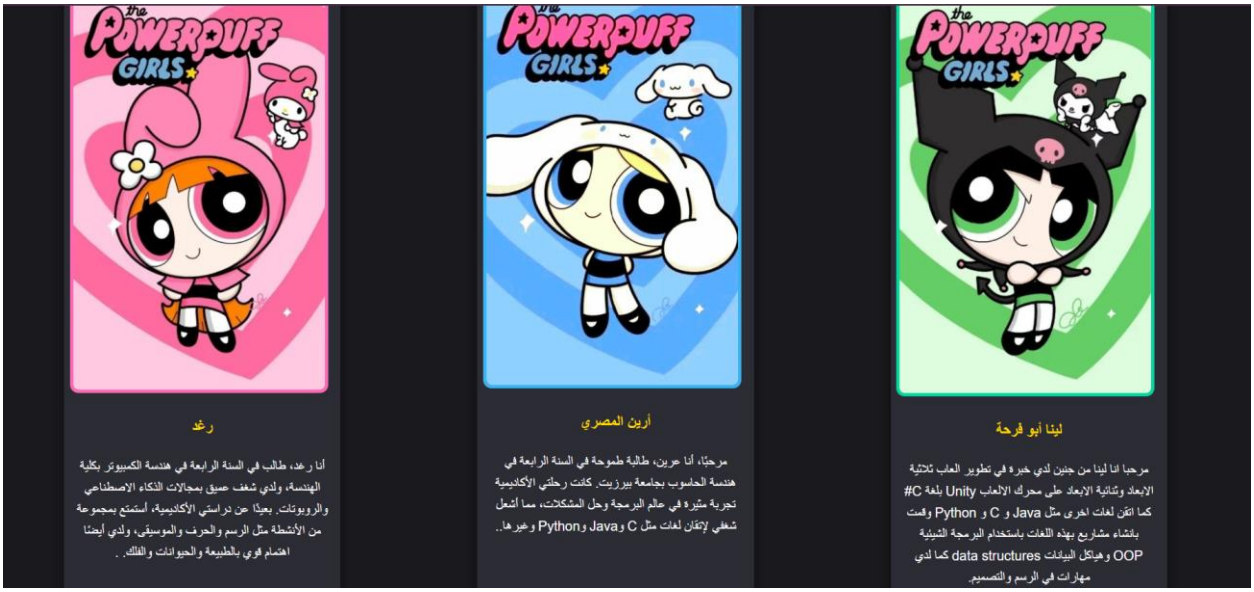
This link redirects the user to a page to search for supporting materials on the server.

A similar version of the webpage was created with support for the Arabic language. This version ensures that users who prefer Arabic can easily navigate and access all the provided resources in their native language.

Key adjustments made to support Arabic include:

1. **Text Direction**: The text is aligned from right to left (RTL) using the `dir="rtl"` attribute.
2. **Language Declaration**: The `lang="ar"` attribute is added to specify the document's language as Arabic.
3. **Content Translation**: All text, labels, and links have been translated to Arabic to provide a seamless experience for Arabic-speaking users.

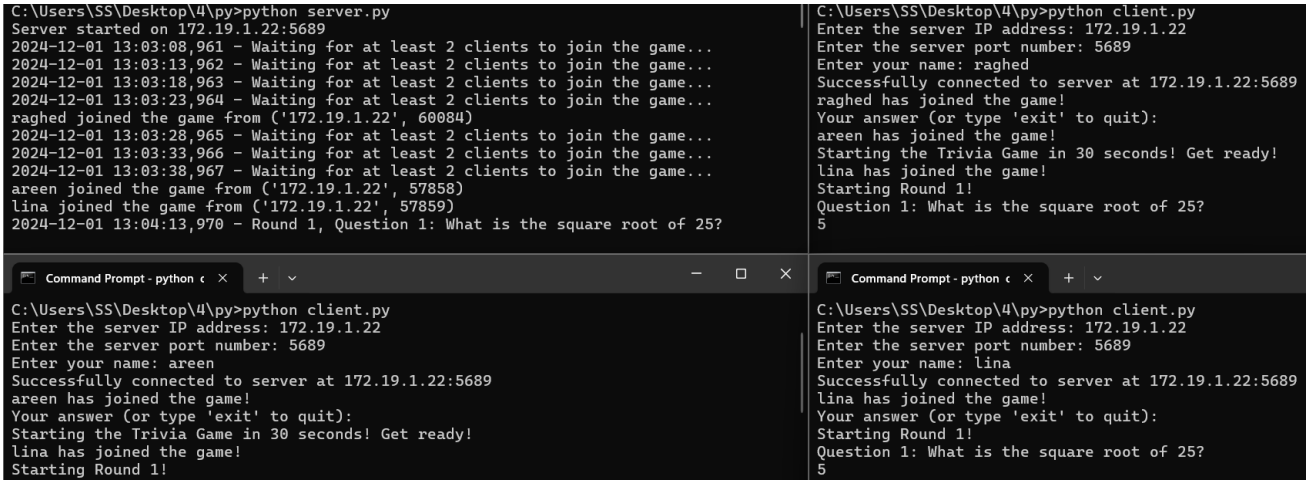




The server is running successfully, processing requests for image and video files, and displaying them in the browser. It redirects users to external search engines if the file is not found.

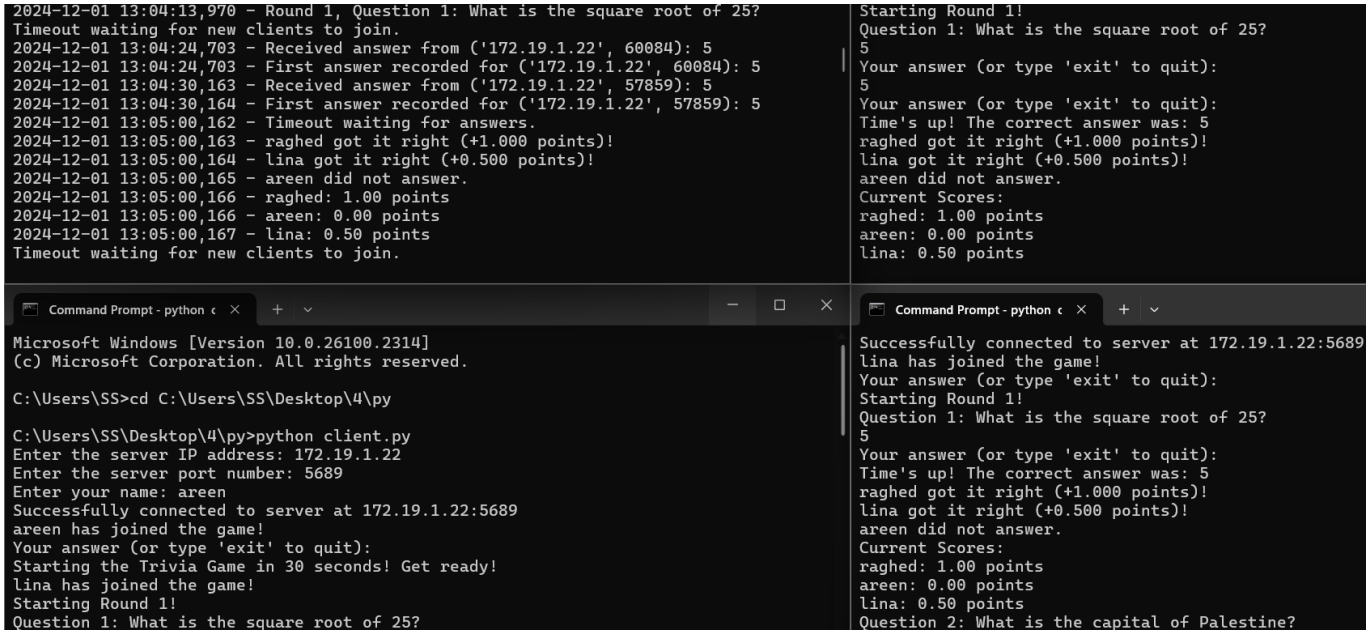
Task 3 – UDP Client-Server Trivia Game Using Socket Programming

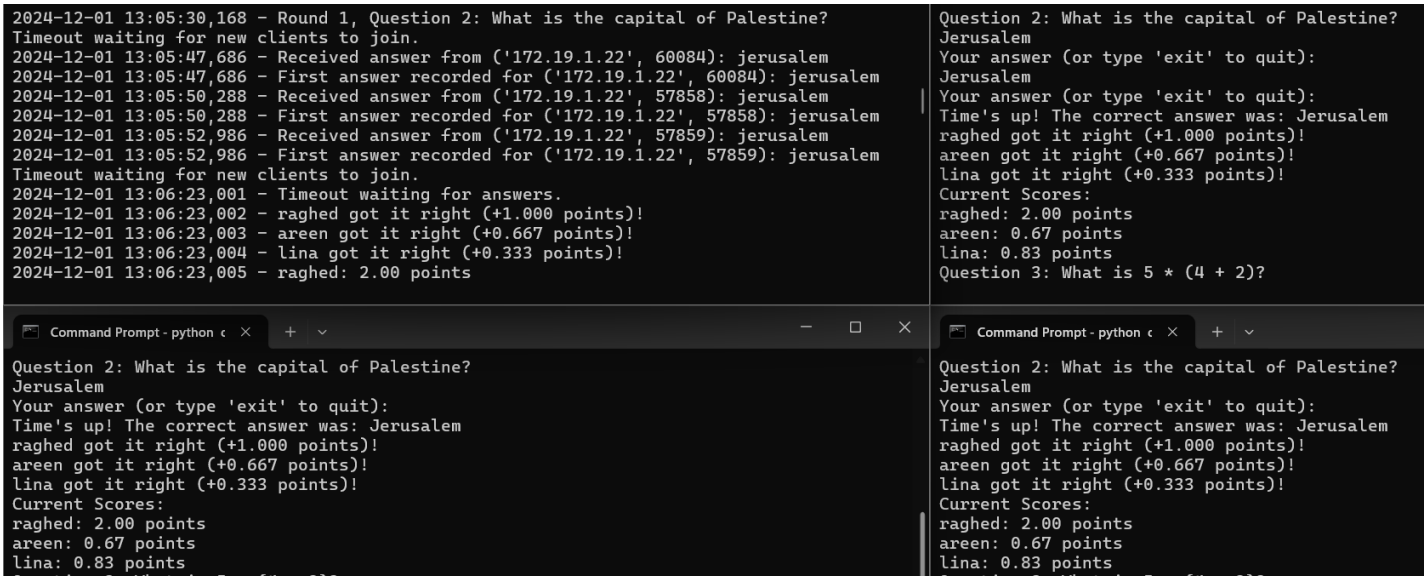
1) We started the server script (server.py) on a machine with the IP address 192.168.1.19 and port 5689 and the server waits for at least two clients to join the game before it begins.
2) Three client instances (client.py) were started, each connecting to the server using the same IP address (192.168.1.19) and port (5689) and Each client provided a username to join the game:
    - **Client 1: Username `raghed`**
    - **Client 2: Username `areen`**
    - **Client 3: Username `lina`**
3) Each client successfully connected to the server, and the server acknowledged their participation.
4) Once there is more than two clients the server began the trivia game after a 30-second scountdown.
5) we designed this trivia game to run for two rounds, but we made it flexible so that the number of rounds can easily be changed.

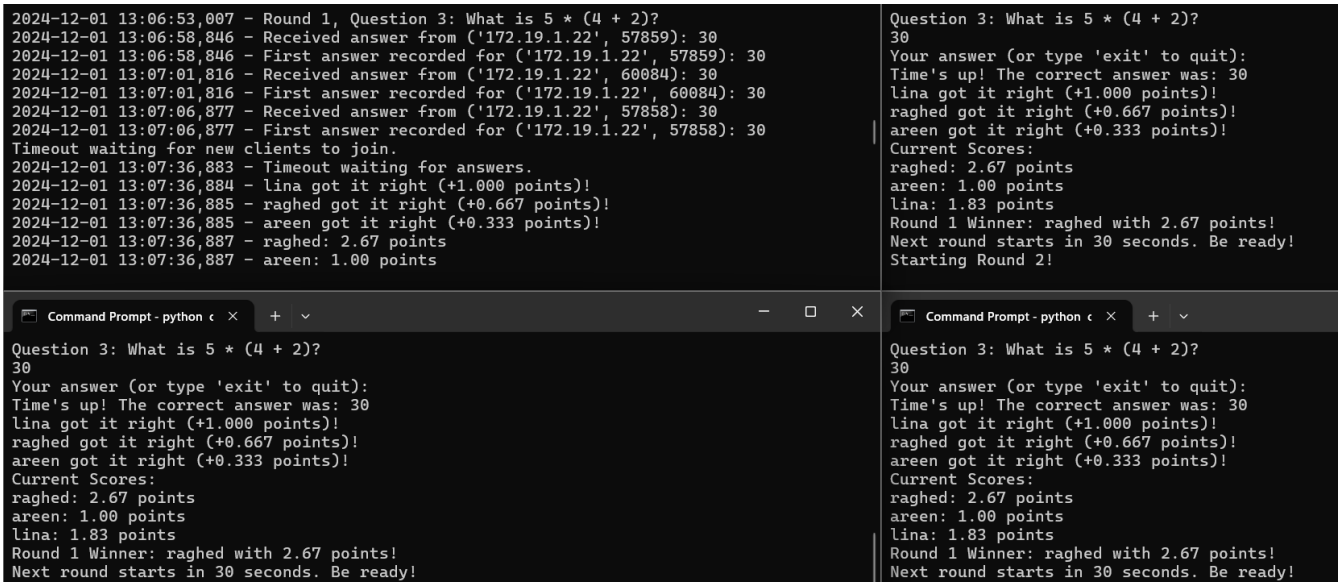


- And now the game has officially started!

- The server waits for **30 seconds** to give players enough time to respond to each question.

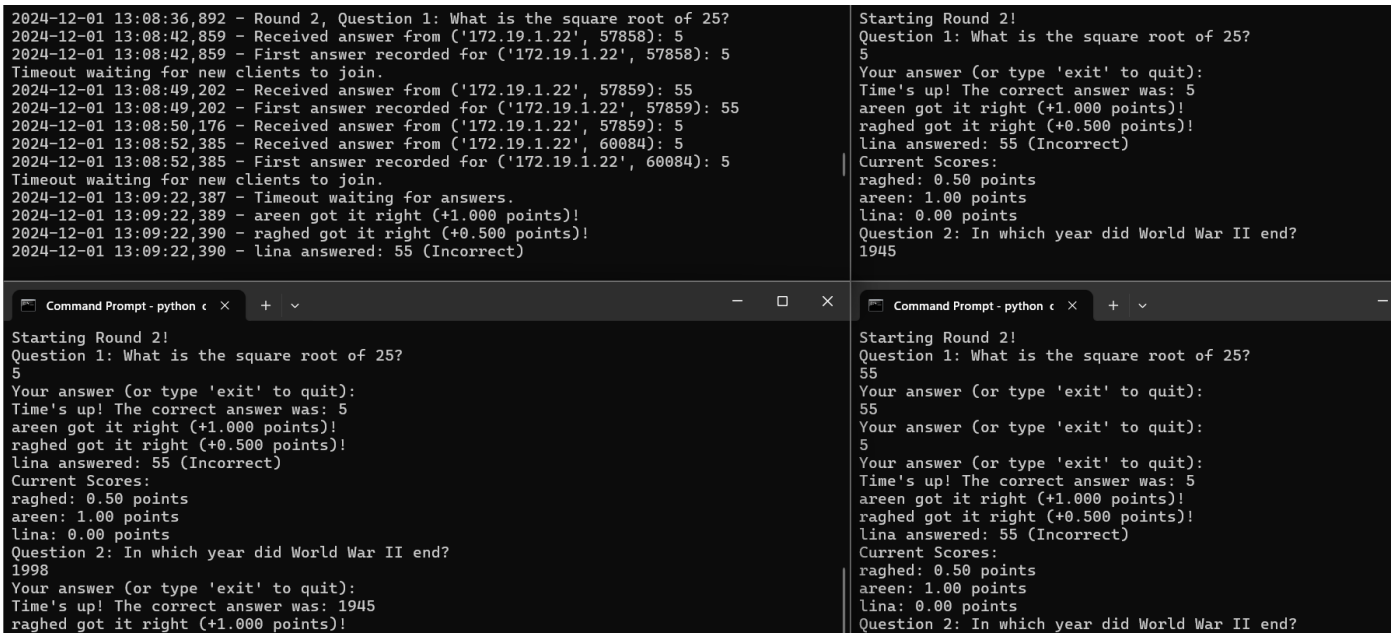




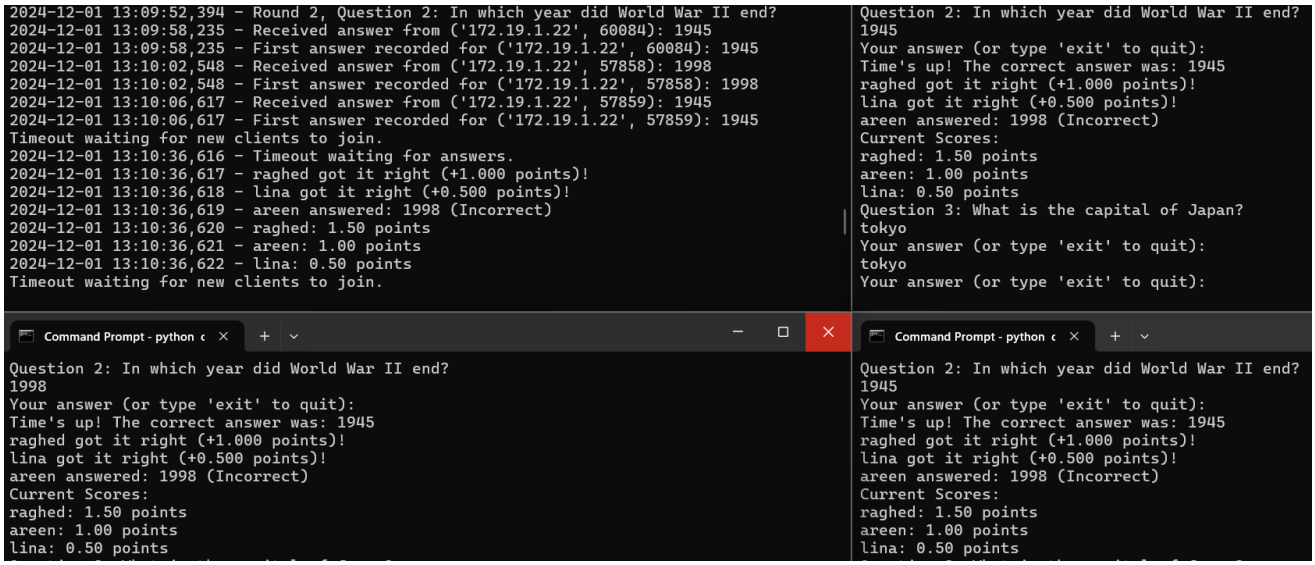


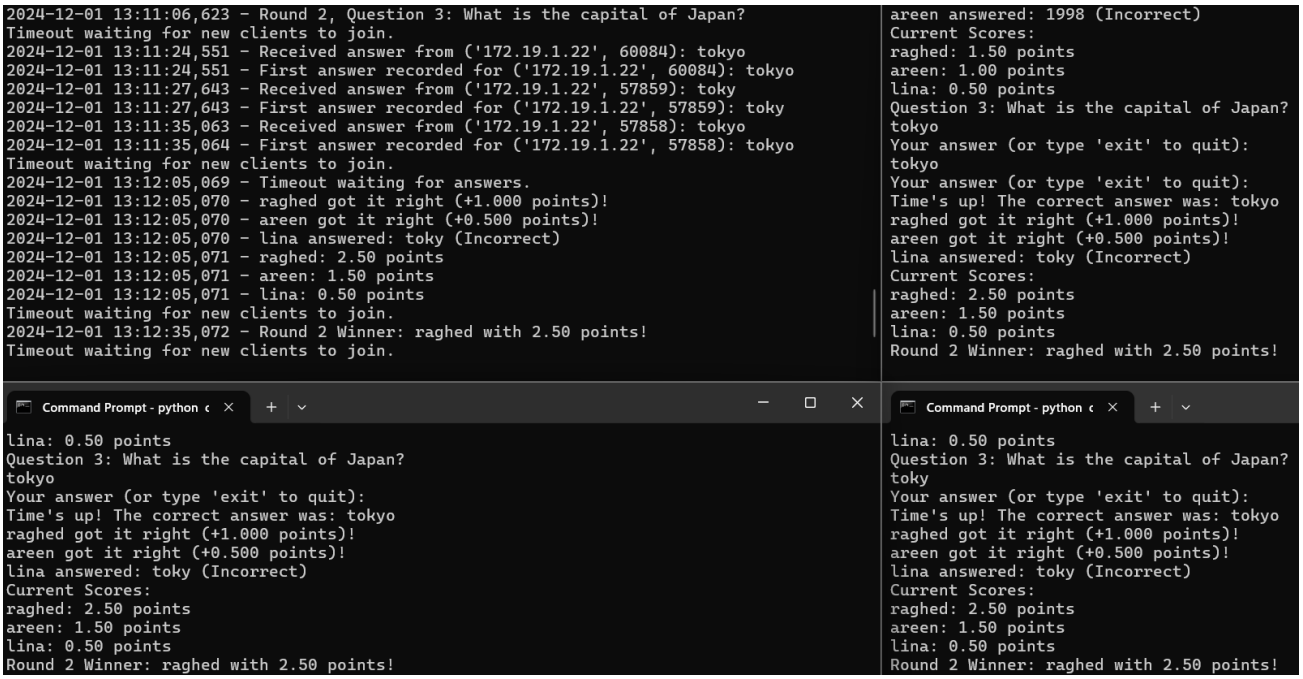
- After the first round, the winner is **raghed** with the highest score so far!
- The server pauses for **30 seconds** between rounds, giving players time to review the leaderboard and prepare for the next round and print “Next round starts in 30 seconds. Be ready!”

- Round 2



- Lina initially submitted an incorrect answer (55) and then changed it to the correct answer (5). However, the server ignored the second answer because it had already logged her first answer.



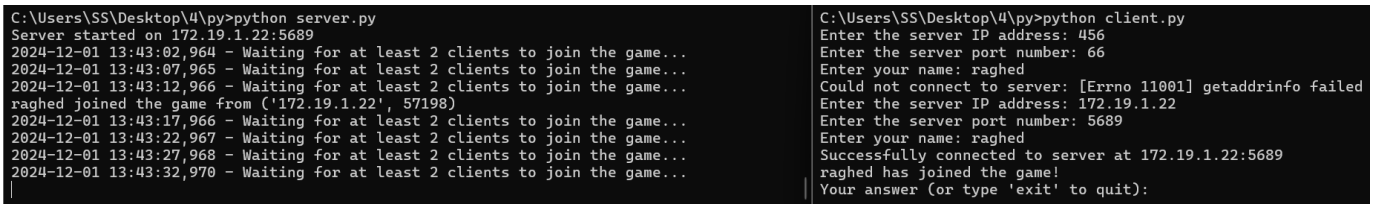


- The winner in this round is **raghed** with the highest score so far!

- So The winner of the game is **raghed** because she won in both rounds!

- The server implements a controlled shutdown mechanism. A KeyboardInterrupt (generated by pressing Ctrl+C) triggers a graceful shutdown sequence. The server broadcasts a termination message to all connected clients before closing the network connection. Clients can also initiate a disconnect by entering the command 'exit', which closes their connection to the server.

- Here's an example of what happens if a client enters an incorrect IP address:



4.Conclusion:

This project provided hands-on experience with key concepts in computer networking, focusing on protocols, troubleshooting tools, and socket programming.

In the first task, we used network tools like ping, tracert, and nslookup to explore how networks operate and diagnose issues. Wireshark allowed us to capture and analyze DNS traffic, giving us insight into real-world data exchange.

The second task involved building a simple TCP-based web server. This task taught us how to handle HTTP requests, manage files, and respond to client actions, including challenges like debugging file paths.

The third task focused on creating a UDP-based trivia game, where we applied real-time communication and multi-client interactions. This task highlighted the importance of clear protocols and teamwork.

"الرحمة والخلود للشهداء لمن هم أكرم منا جميعا ، والشفاء العاجل لكل الجرحى"