



**Faculty of Engineering and Technology**

**Department of Electrical and Computer Engineering**

**ENCS3320 – Computer Networks**

**Project #1 (Socket Programming)**

**Team Members:**

Shahd Khalaf – 1210545 – Section #1

Sama Wahidee – 1211503 – Section #1

Majd Hamarsheh – 1212306 – Section #5

**Submission Date : Dec 1, 2024**

# Table of Contents:

<b>Table of Contents:.....</b>	<b>2</b>
<b>Table of figures:.....</b>	<b>3</b>
<b>Abstract:.....</b>	<b>7</b>
<b>Task 1- Network Commands and Wireshark:.....</b>	<b>8</b>
Section A:.....	8
Section B:.....	9
Section C:.....	15
<b>Task 2- Web Server:.....</b>	<b>17</b>
Theory and Procedure:.....	17
Web Server Overview:.....	19
Section A - Main English Webpage:.....	20
Section B - Supporting Material Page:.....	35
Section C - Arabic Versions:.....	44
<b>Task 3.....</b>	<b>62</b>
UDP Client-Server Trivia Game Overview:.....	62
Theory and Procedure:.....	62
Client Responsibilities:.....	63
Game Flow and Client-Server Interaction:.....	64
Server Terminal Output.....	74
Task 3 - Server Side Code.....	77
Task 3 - Client Side Code.....	87
<b>Teamwork:.....</b>	<b>89</b>
<b>References:.....</b>	<b>90</b>

## Table of figures:

figure 1-output of ipconfig/all -Task 1.....	9
figure 2- output of ping an iphone - Task 1.....	10
figure 3-iphone IP address - Task 1.....	10
<b>figure 4: output of ping discover.engineering.utoronto.ca- Task 1.....</b>	<b>11</b>
figure 5- output of tracert discover.engineering.utoronto.ca - Task1.....	12
figure 6- output of nslookup discover.engineering.utoronto.ca - Task1.....	13
figure 7- output of telnet discover.engineering.utoronto.ca - Task1.....	13
figure 8-telnet command with different port - Task1.....	14
figure 9-output of telnet discover.engineering.utoronto.ca 80 - Task1.....	14
figure 10: Wireshark.....	15
figure 11:terminal 1 - Task 1.....	15
figure 12:DNS Query - Task 1.....	16
figure 13: DNS Response - Task 1.....	16
figure 14:Wireshark Highlights - Task 1.....	16
figure 15:HTTP protocol - Task 2.....	18
figure 16:Socket programming - Task 2.....	19
figure 17:main_en.html - Task 2.....	21
figure 18:main_en.html - Task 2.....	21
figure 19:main_en.html - Task 2.....	22
figure 20:main_en.html - Task 2.....	22
figure 21: styles_main_en.css1 - Task 2.....	23
figure 22: styles_main_en.css - Task 2.....	23
figure 23: styles_main_en.css - Task 2.....	24
figure 24: styles_main_en.css - Task 2.....	24
figure 25: styles_main_en.css - Task 2.....	25
figure 26:: styles_main_en.css - Task 2.....	25
figure 27:Web Server Interface - Task 2.....	26
figure 28:Web Server Interface - Task 2.....	27
figure 29:Web Server Interface - Task 2.....	27
figure 30:Web Server Interface - Task 2.....	28
<b>figure 31:Web Server Interface - Task 2.....</b>	<b>29</b>
figure 32:Web Server Interface - Task 2.....	29
figure 33:Web Server Interface - Task 2.....	30
figure 34: Web Server - Task 2.....	31
figure 35: Web Server - Task 2.....	31

figure 36: Web Server - Task 2.....	32
figure 37: Web Server - Task 2.....	32
figure 38: Web Server - Task 2.....	33
figure 39: Web Server - Task 2.....	33
figure 40: Web Server Error - Task 2.....	34
figure 41:supporting_material_en.html - Task 21.....	35
figure 42:supporting_material_en.html - Task 2.....	35
figure 43:supporting_material_en.html - Task 2.....	36
figure 44:style_en.css - Task 2.....	36
figure 45:style_en.css - Task 2.....	37
figure 46: style_en.css - Task 2.....	37
figure 47: Web Server Interface - Task 2.....	38
figure 48: Web Server Interface - Task 2.....	39
figure 49: HTTP Request - Task 2.....	40
figure 50: HTTP Response - Task 2.....	40
figure 51: HTTP Request - Task 2.....	41
figure 52: HTTP Response - Task 2.....	41
figure 53: HTTP Request - Task 23.....	42
figure 54: HTTP Response - Task 2.....	42
figure 55: HTTP Request - Task 2.....	43
figure 56: supporting_material_ar.html - Task 2.....	44
figure 57: supporting_material_ar.html - Task 2.....	45
figure 58: style_ar.css - Task 2.....	45
figure 59: style_ar.css - Task 2.....	46
figure 60: style_ar.css - Task 2.....	46
figure 61:Web Server Interface - Task 2.....	47
figure 62: HTTP Request - Task 2.....	47
figure 63:Web Server Interface - Task 2.....	48
figure 64: HTTP Request - Task 2.....	49
figure 65::Web Server Interface - Task 2.....	49
figure 66: HTTP Request - Task 2.....	50
figure 67:Web Server Interface - Task 2.....	51
figure 68: HTTP Response - Task 2.....	51
figure 69: HTTP Request - Task 2.....	52
figure 70::Web Server Interface - Task 2.....	52
figure 71: HTTP Response - Task 2.....	53
figure 72: HTTP Request - Task 2.....	53
figure 73:main_ar.html - Task 2.....	54
figure 74: main_ar.html - Task 2.....	54
figure 75:main_ar.html - Task 2.....	55
figure 76:main_ar.html - Task 2.....	55

figure 77:styles_main_ar.css - Task 2.....	56
figure 78:styles_main_ar.css - Task 2.....	56
figure 79:styles_main_ar.css - Task 2.....	57
figure 80:styles_main_ar.css - Task 2.....	57
figure 81:styles_main_ar.css - Task 2.....	58
figure 82:Web Server Interface - Task 2.....	59
figure 83:Web Server Interface - Task 2.....	60
figure 84:Web Server Interface - Task 2.....	60
figure 85:Web Server Interface - Task 2.....	61
figure 86:Web Server Interface - Task 2.....	61
figure 87: server listen on port 5689 - Task 3.....	64
figure 88: first client joined the game - Task 3.....	65
figure 89: Second client Joined The game - Task 3.....	66
figure 90: third client joined the game - Task 3.....	66
figure 91: Game Start and First Round Progression - Task 3.....	68
figure 92: handling correct answers - Task 3.....	68
figure 93: Handling wrong answers - Task3.....	68
figure 94: Handling Time Up Case - Task 3.....	69
figure 95:End of Round Announcement - Task 3.....	69
figure 96: Client Answered Question First Case - Task 3.....	70
figure 97: Client Answered Correct But Not The First Case - Task 3.....	71
figure 98: Handling Multiple Submissions - Task 3.....	71
figure 99: A Single Winner is Announced - Task 3.....	72
figure 100: A Tie Between Two or More Players - Task 3.....	73
figure 101: server actively listening on a 5689 port. & confirming each new client connection - Task 3.....	74
figure 102: message on the server side that enough clients joined - Task 3.....	75
figure 103: question displaying in server side and clients answers with answer result - Task 3.....	75
figure 104: message on the server side at the end of a round & announcing round winner - Task 3.....	76
figure 105: message in server side countdown in server side before starting the next round - Task 3.....	76
figure 106: server shutting down message - Task 3.....	77
figure 107: server configs & list of question definition code -Task 3.....	77
figure 108: method to broadcast messages to clients - Task 3.....	78
figure 109: method to start game and send questions to clients - Task 3.....	78
figure 110: method to initialize the game set up -Task 3.....	79
figure 111: method to ask random unique questions to client - Task 3.....	80
figure 112: method to evaluate round winner - Task 3.....	80
figure 113: end game and close server method - Task 3.....	81
figure 114: welcoming and handle client method - Task 3.....	81

figure 115: method to send data to active clients at same time - Task 3.....	82
figure 116: method to check answers and update scores - Task 3.....	83
figure 117: method to check answers , update scores & send feedbacks - Task 3. 84	
figure 118: show current score method - Task 3.....	85
figure 119: calculate final score and announce winner method - Task 3.....	85
figure 120: main method to set up UDP connection and start server - Task 3.....	86
figure 121: Server Configs Client Side Code - Task 3.....	87
figure 122: Start Client Connection Code CClient Side - Task 3.....	88

## Abstract:

The project, which is a part of the ENCS3320 Computer Networks course, uses tasks involving the Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) to investigate the fundamentals of socket programming and networking. Developing an interactive multiplayer trivia game, building a custom web server, and getting practical expertise with key network commands are among the goals. The project, which covers important ideas including the HTTP protocol, socket connectivity, and server-client interaction, places an emphasis on both theoretical comprehension and real-world application. Working together, we used our programming expertise to develop a scalable, modular solution using Wireshark and other network analysis tools. This report describes the procedures, results, and difficulties that arose throughout the project's execution.

# Task 1- Network Commands and Wireshark:

## Section A:

1. **Ipconfig - Internet Protocol configuration:** [\[7\]](#)
  - Displays details about your computer's current network setup, including IP address and TCP/IP settings. It also allows refreshing DHCP and DNS configurations to fix network issues.
2. **ping:** [\[8\]](#)
  - Checks if a device or website is reachable by sending ICMP Echo Request packets over a network and waiting for a reply (ICMP Echo Reply). It measures response time to help troubleshoot connectivity issues.
3. **tracert:** [\[9\]](#)
  - Traces the path data takes to reach a destination by showing each router or hop and the time taken for the journey. This helps diagnose network delays or routing issues.
4. **telnet - teletype network:** [\[10\]](#)
  - Allows remote access to another computer via a simple text-based interface for diagnostics, testing open ports, and monitoring network services. It typically uses port 23 but can be configured to use other ports.
5. **nslookup - DNS lookup:** [\[11\]](#)
  - Retrieves the IP address of a domain or confirms domain functionality to verify and troubleshoot DNS records and configurations. It is useful for checking DNS propagation and resolving domain issues.

## Section B:

1. Run the ipconfig /all command:

```
Wireless LAN adapter Wi-Fi:  
  
Connection-specific DNS Suffix . :  
Description . . . . . : Intel(R) Wi-Fi 6 AX201 160MHz  
Physical Address. . . . . : AA-E4-31-B3-AE-51  
DHCP Enabled. . . . . : Yes  
Autoconfiguration Enabled . . . . . : Yes  
Link-local IPv6 Address . . . . . : fe80::3981:bb15:8d39:8d68%9(PREFERRED)  
IPv4 Address. . . . . : 192.168.1.101(PREFERRED)  
Subnet Mask . . . . . : 255.255.255.0  
Lease Obtained. . . . . : 00:00:00 30 00:00:00 2024 2:23:55  
Lease Expires . . . . . : 00:00:00 30 00:00:00 2024 4:46:31  
Default Gateway . . . . . : 192.168.1.1  
DHCPv6 IAID . . . . . : 162194481  
DHCPv6 Client DUID. . . . . : 00-03-00-01-AA-E4-31-B3-AE-51  
DNS Servers . . . . . : 192.168.1.1  
NetBIOS over Tcpip. . . . . : Enabled
```

figure 1-output of ipconfig/all -Task 1

The following are the retrieved values for the primary network interface:

- IP Address: 192.168.1.101
- Subnet Mask: 255.255.255.0
- Default Gateway: 192.168.1.1
- Domain Name System: (DNS): 192.168.1.1

## 2. Ping an iPhone within the local network:

```
C:\Users\Ibrah>ping 192.168.1.100

Pinging 192.168.1.100 with 32 bytes of data:
Reply from 192.168.1.100: bytes=32 time=44ms TTL=64
Reply from 192.168.1.100: bytes=32 time=45ms TTL=64
Reply from 192.168.1.100: bytes=32 time=82ms TTL=64
Reply from 192.168.1.100: bytes=32 time=126ms TTL=64

Ping statistics for 192.168.1.100:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 44ms, Maximum = 126ms, Average = 74ms
```

figure 2- output of ping an iphone - Task 1



figure 3-iphone IP address - Task 1

The ping results demonstrate successful communication between the laptop and the iphone with the IP address [192.168.1.100](#) within the local network. The analysis is as follows:

1. **Connectivity:** All ping requests were successfully sent and received, with no packet loss (0%), confirming that the device is reachable and active on the same Wi-Fi network.

2. **Round-Trip Time (RTT):**

The time for packets to travel to the target device and back ranged from **44ms** (minimum) to **126ms** (maximum), with an average of **74ms**. This variation in latency is typical for local wireless networks and may be influenced by factors like network traffic or signal strength.

3. **TTL (Time to Live):**

The TTL value of **64** indicates direct communication with the device, as expected within a local network, without significant routing.

Notes: -If we attempt to ping a host and get a "request timed out" message, it may indicate that the host is unavailable or that it is preventing any further ping requests.

-If we ping a host and get the response, "destination host unreachable," it means that the path to the target was not found.

3. Ping **discover.engineering.utoronto.ca**:

```
C:\Users\Ibrah>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=51ms TTL=59
Reply from 23.185.0.2: bytes=32 time=70ms TTL=59
Reply from 23.185.0.2: bytes=32 time=57ms TTL=59
Reply from 23.185.0.2: bytes=32 time=55ms TTL=59

Ping statistics for 23.185.0.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 51ms, Maximum = 70ms, Average = 58ms
```

figure 4: output of ping **discover.engineering.utoronto.ca**- Task 1

Based on the results of pinging **discover.engineering.utoronto.ca**, the IP address **23.185.0.2** was identified. When searching for the location of this IP address [\[4\]](#), it was

found to be in San Francisco, California, USA. This suggests that the response does not originate from Canada. The University of Toronto may be using a third-party hosting service or a Content Delivery Network (CDN) to serve their website,

#### 4. Run tracert on [discover.engineering.utoronto.ca](http://discover.engineering.utoronto.ca):

```
C:\Users\Ibrah>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      1 ms      1 ms  192.168.1.1
  2   127 ms     8 ms      7 ms  172.20.1.1
  3     6 ms      6 ms      5 ms  213.6.240.220
  4    54 ms     50 ms     63 ms  10.74.19.21
  5    53 ms     49 ms     48 ms  10.74.19.146
  6    55 ms     53 ms     52 ms  10.74.59.250
  7     *         *         *      Request timed out.
  8    82 ms     51 ms    136 ms  23.185.0.2

Trace complete.
```

figure 5- output of tracert discover.engineering.utoronto.ca - Task1

The tracert results for [discover.engineering.utoronto.ca](http://discover.engineering.utoronto.ca) show that the server at IP [23.185.0.2](http://23.185.0.2) was reached in **8 hops**. The first few hops indicate routers within our local network and ISP, with low response times.

Hop 7 shows a "Request timed out," which is common as some routers block traceroute requests. The final hop to the server shows response times between **51ms and 136ms**, indicating a relatively short network distance from our location. This suggests the server is efficiently reachable.

#### 5. Nslookup [discover.engineering.utoronto.ca](http://discover.engineering.utoronto.ca):

```
C:\Users\Ibrah>nslookup discover.engineering.utoronto.ca
Server: UnKnown
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
```

figure 6- output of nslookup discover.engineering.utoronto.ca - Task1

The nslookup command was used to retrieve DNS information for [discover.engineering.utoronto.ca](http://discover.engineering.utoronto.ca):

- The server used for the lookup is [192.168.1.1](http://192.168.1.1), which is usually our local router or DNS server.
- The non-authoritative answer means the information came from a cached source rather than directly from the main DNS server for the domain.
- The IP addresses for [discover.engineering.utoronto.ca](http://discover.engineering.utoronto.ca) include:
  - Two IPv6 addresses ([2620:12a:8001::2](http://2620:12a:8001::2) and [2620:12a:8000::2](http://2620:12a:8000::2)).
  - One IPv4 address ([23.185.0.2](http://23.185.0.2)), which is the most common address type used on the internet.

\*This shows the addresses that can be used to access the website.

## 6. Telnet [discover.engineering.utoronto.ca](http://discover.engineering.utoronto.ca):

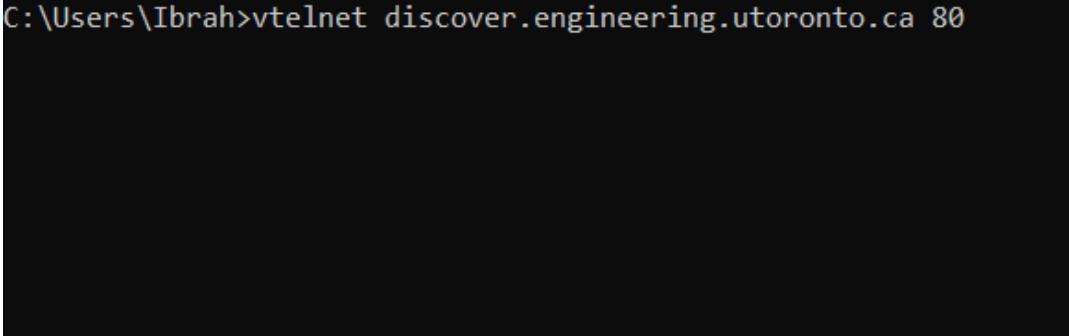
```
C:\Users\Ibrah>telnet discover.engineering.utoronto.ca
Connecting To discover.engineering.utoronto.ca...Could not open connection to the host, on port 23: Connect failed
C:\Users\Ibrah>
```

figure 7- output of telnet discover.engineering.utoronto.ca - Task1

When we tried to telnet [discover.engineering.utoronto.ca](http://discover.engineering.utoronto.ca), it failed and we received this error message "**Could not open connection to the host, on port 23: Connect failed**", it indicates

that Telnet was unable to establish a connection to [discover.engineering.utoronto.ca](http://discover.engineering.utoronto.ca) on the default port 23. This could be due to several reasons: Telnet Service Not Available on Port 23 ,Firewall Blocking Telnet,Wrong Port.

So we used different port (port=80),



```
C:\Users\Ibrah>vtelnet discover.engineering.utoronto.ca 80
```

figure 8-telnet command with different port - Task1



Telnet discover.engineering.utoronto.ca

figure 9-output of telnet discover.engineering.utoronto.ca 80 - Task1

To test connectivity to the server [discover.engineering.utoronto.ca](http://discover.engineering.utoronto.ca), we attempted to use Telnet. After specifying the hostname and port, we were able to establish a connection, but the screen remained blank. This indicates that the connection to the server was successful, but the server did not send any immediate response.

In such cases, the server may be waiting for a specific input, such as an HTTP request, before sending data back. This is common when connecting to web servers, as they require commands

like [GET / HTTP/1.1](#) to retrieve webpage content. Telnet is useful for testing simple connections, but it doesn't automatically handle specific communication protocols like HTTP or HTTPS.

## Section C:



A network protocol analyzer called Wireshark is used to record and examine packet-level communication. Among other protocols, it offers insights into DNS query-response cycles.[\[2\]](#)

figure 10: Wireshark

The hostname we choice: [www.google.com](http://www.google.com)

- After downloading and opening the wireshark we went to the terminal and wrote:  
“nslookup [www.google.com](http://www.google.com)”

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

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

PS C:\Users\samaw> nslookup www.google.com|
```

A screenshot of a Windows PowerShell window titled "Windows PowerShell". The window shows the command "nslookup www.google.com" being typed at the prompt. The output of the command is visible below the command line.

figure 11:terminal 1 - Task 1

- DNS Query (We had a Filter for DNS Traffic):

No.	Time	Source	Destination	Protocol	Length	Info
1222	72.936590	192.168.1.81	192.168.1.254	DNS	75	Standard query 0xdd1e A ssl.gstatic.com
1223	72.937064	192.168.1.81	192.168.1.254	DNS	75	Standard query 0xb5c7 HTTPS ssl.gstatic.com
1225	73.136362	192.168.1.254	192.168.1.81	DNS	91	Standard query response 0xdd1e A ssl.gstatic.com A 216.58.214.163
1227	73.136362	192.168.1.254	192.168.1.81	DNS	132	Standard query response 0xb5c7 HTTPS ssl.gstatic.com SOA ns1.google.com
1269	74.486489	fe80::77bf:50cb:87e.. fe80::1	DNS	111	Standard query 0x2c19 A settings-win.data.microsoft.com	
1270	74.519945	fe80::77bf:50cb:87e.. fe80::1	DNS	129	Standard query response 0xb5c7 A settings-win.data.microsoft.com CNAME atm-settingsfe-prod-geo2.trafficmanager.net CNAME settings-p-	
1330	79.359338	fe80::77bf:50cb:87e.. fe80::1	DNS	186	Standard query 0x1f5b A fe2cr.update.microsoft.com	
1331	79.414345	fe80::77bf:50cb:87e.. fe80::1	DNS	192	Standard query response 0x1f5b A fe2cr.update.microsoft.com CNAME fe2cr.update.msft.com.trafficmanager.net A 40.83.50.80 A 20.163.4..	
1381	82.102159	192.168.1.81	192.168.1.254	DNS	74	Standard query 0x7449 A www.google.com
1384	82.102484	192.168.1.81	192.168.1.254	DNS	74	Standard query 0xeaf1 HTTPS www.google.com
1385	82.130506	192.168.1.254	192.168.1.81	DNS	90	Standard query response 0x7449 A www.google.com A 216.239.38.120
1386	82.130506	192.168.1.254	192.168.1.81	DNS	99	Standard query response 0xeaf1 HTTPS www.google.com HTTPS
1458	85.963461	192.168.1.81	192.168.1.254	DNS	76	Standard query 0x6c78 A video.google.com
1459	85.964299	192.168.1.81	192.168.1.254	DNS	76	Standard query 0x6085 HTTPS video.google.com
1504	86.705554	192.168.1.254	192.168.1.81	DNS	114	Standard query response 0x6c78 A video.google.com CNAME video.l.google.com A 216.58.215.46
1505	86.705554	192.168.1.254	192.168.1.81	DNS	148	Standard query response 0x6085 HTTPS video.google.com CNAME video.l.google.com SOA ns1.google.com
1540	86.975155	fe80::77bf:50cb:87e.. fe80::1	DNS	186	Standard query 0x038 A download.windowsupdate.com	
1543	87.073068	192.168.1.81	192.168.1.254	DNS	86	Standard query 0xc038 A download.windowsupdate.com CNAME download.windowsupdate.com.delivery.microsoft.com CNAME wu-f-net..
1546	87.175746	fe80::1	fe80::77bf:50cb:87e.. DNS	261	Standard query response 0xc038 A download.windowsupdate.com.download.windowsupdate.com.delivery.microsoft.com CNAME wu-f-net..	

figure 12:DNS Query - Task 1

- DNS Response (We had a Filter for DNS Traffic):

1222	72.936590	192.168.1.81	192.168.1.254	DNS	75	Standard query 0xdd1e A ssl.gstatic.com
1223	72.937064	192.168.1.81	192.168.1.254	DNS	75	Standard query 0xb5c7 HTTPS ssl.gstatic.com
1225	73.136362	192.168.1.254	192.168.1.81	DNS	91	Standard query response 0xdd1e A ssl.gstatic.com A 216.58.214.163
1227	73.136362	192.168.1.254	192.168.1.81	DNS	132	Standard query response 0xb5c7 HTTPS ssl.gstatic.com SOA ns1.google.com
1269	74.486489	fe80::77bf:50cb:87e.. fe80::1	DNS	111	Standard query 0x2c19 A settings-win.data.microsoft.com	
1270	74.519945	fe80::77bf:50cb:87e.. fe80::1	DNS	129	Standard query response 0x2c19 A settings-win.data.microsoft.com CNAME atm-settingsfe-prod-geo2.trafficmanager.net CNAME settings-p..	
1330	79.359338	fe80::77bf:50cb:87e.. fe80::1	DNS	186	Standard query 0x1f5b A fe2cr.update.microsoft.com	
1331	79.414345	fe80::77bf:50cb:87e.. fe80::1	DNS	192	Standard query response 0x1f5b A fe2cr.update.microsoft.com CNAME fe2cr.update.msft.com.trafficmanager.net A 40.83.50.80 A 20.163.4..	
1381	82.102159	192.168.1.81	192.168.1.254	DNS	74	Standard query 0x7449 A www.google.com
1384	82.102484	192.168.1.81	192.168.1.254	DNS	74	Standard query 0xeaf1 HTTPS www.google.com
1385	82.130506	192.168.1.254	192.168.1.81	DNS	90	Standard query response 0x7449 A www.google.com A 216.239.38.120
1386	82.130506	192.168.1.254	192.168.1.81	DNS	99	Standard query response 0xeaf1 HTTPS www.google.com HTTPS
1458	85.963461	192.168.1.81	192.168.1.254	DNS	76	Standard query 0x6c78 A video.google.com
1459	85.964299	192.168.1.81	192.168.1.254	DNS	76	Standard query 0x6085 HTTPS video.google.com
1504	86.705554	192.168.1.254	192.168.1.81	DNS	114	Standard query response 0x6c78 A video.google.com CNAME video.l.google.com A 216.58.215.46
1505	86.705554	192.168.1.254	192.168.1.81	DNS	148	Standard query response 0x6085 HTTPS video.google.com CNAME video.l.google.com SOA ns1.google.com
1540	86.975155	fe80::77bf:50cb:87e.. fe80::1	DNS	186	Standard query 0xc038 A download.windowsupdate.com	
1543	87.073068	192.168.1.81	192.168.1.254	DNS	86	Standard query 0xc038 A download.windowsupdate.com CNAME download.windowsupdate.com.delivery.microsoft.com CNAME wu-f-net..
1546	87.175746	fe80::1	fe80::77bf:50cb:87e.. DNS	261	Standard query response 0xc038 A download.windowsupdate.com.download.windowsupdate.com.delivery.microsoft.com CNAME wu-f-net..	

figure 13: DNS Response - Task 1

- Key Wireshark Fields to Highlight:

```
> Frame 1383: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface \Device\NPF_{1B4693B9-F768-42C2-9CCF-07EE8C82A1A4}, id 0
> Ethernet II, Src: Intel_D6:22:cc:6c (f8:fe:5e:62:cc:6c), Dst: TaicangT&WEI_d8:d9:b0 (04:75:f9:d8:d9:b0)
└> Internet Protocol Version 4, Src: 192.168.1.81, Dst: 192.168.1.254
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
    > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
        Total Length: 60
        Identification: 0x6a48 (27208)
        > 000. .... = Flags: 0x0
        ...0 0000 0000 = Fragment Offset: 0
        Time to Live: 128
        Protocol: UDP (17)
        Header Checksum: 0x0000 [validation disabled]
        [Header checksum status: Unverified]
        Source Address: 192.168.1.81
        Destination Address: 192.168.1.254
        [Stream index: 4]
    > User Datagram Protocol, Src Port: 61663, Dst Port: 53
    > Domain Name System (query)
        Transaction ID: 0x7449
        > Flags: 0x0100 Standard query
            Questions: 1
            Answer RRs: 0
            Authority RRs: 0
            Additional RRs: 0
        > Queries
            > www.google.com: type A, class IN
            [Response In: 1385]
```

figure 14:Wireshark Highlights - Task 1

1. **Version:** the internet protocol addressing version is IPV4.
2. **Header Length:** the total length of the header is 20 bytes.
3. **TTL:** The packet can travel through 128 routers.

4. **Source Address:** the local IP address of my PC = 192.168.1.81.
5. **Destination Address:** the IP address of the destination = 192.168.1.254.
6. **Source Port:** the source port of the packets = 61663.
7. **Destination Port:** the destination port to the packets = 53.

## Task 2- Web Server:

This is a demo for Task2: [Task2 Demo](#)

This is a phone demo for Task2: [Task2PhoneDemo](#)

This is the source code for Task2: [Task2 Source Code](#)

## Theory and Procedure:

### HTTP protocol:

The HTTP protocol is based on the request-response paradigm. When clients (browsers) ask for resources, servers reply with error codes or content. Client-server communication is guided by status codes such as 200 OK, 404 Not Found, and 307 Temporary Redirect.[\[1\]](#)

# HTTP Connection

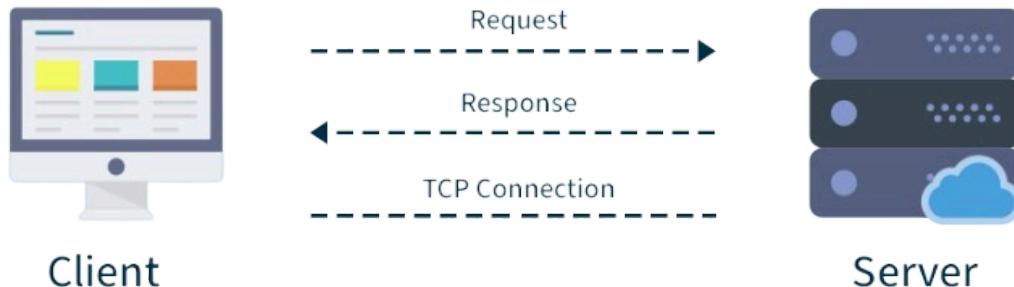


figure 15:HTTP protocol - Task 2

## Socket programming:

essential for facilitating data transmission by establishing communication over a network between a client (web browser) and a server (web server). In this project, a server that listens for incoming HTTP requests and replies with the relevant content, like HTML or CSS, is created using socket programming. The usage of the TCP/IP protocol, in which IP manages data routing and TCP guarantees dependable, connection-oriented communication, is one of the key ideas involved. The server binds to a particular local IP address and port (e.g., 5698) to listen for requests, and the socket is defined using constants like SOCK\_STREAM (for TCP) and AF\_INET (for IPv4). [\[3\]](#)

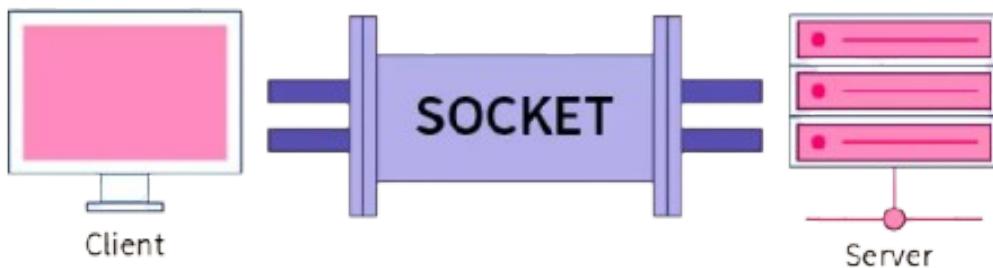


figure 16:Socket programming - Task 2

## Web Server Overview:

Using socket programming, we are building a web server that can serve several web pages and listen on port 5698. HTML, CSS, PNG, MP4, and other content types should all be supported by the server, which should also be able to respond dynamically to requests for files. The content will be handled by the server in both Arabic and English, and it will have a function that allows users to request particular photographs or movies depending on networking topics.

The following HTML pages will be served:

- Main English Webpage (main\_en.html)
- Supporting Material Page (supporting\_material\_en.html)
- Arabic Versions (main\_ar.html and supporting\_material\_ar.html)

The server will respond with the appropriate content type (e.g., text/html, text/css, image/png) and handle redirections when necessary (e.g., for unavailable files).

## Section A - Main English Webpage:

- **HTML/CSS Code:**
  - **main\_en.html:**

```

1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>ENCS3320-Webserver</title>
7      <link rel="stylesheet" href="/css/styles_main_en.css">
8  </head>
9  <body>
10     <header>
11         <h1>Welcome to ENCS3320 - Computer Networks Webserver</h1>
12     </header>
13     <a class="arabic-link" href="/main_ar.html">||ARABIC||</a>
14
15     <section class="team-section">
16         <h2>Team Members</h2>
17         <div class="team-container">
18             <div class="team-box">
19                 
20                 <h3>Shahd Khalaf</h3>
21                 <p class="id">ID: 1210545</p>
22                 <p>
23                     As a dedicated computer science student, I specialize in Java and C, with a keen interest in solving real-world problems through
24                     My internship in full-stack development allowed me to contribute to a project called Insurance Management System.
25                     Outside of my studies, I am passionate about drawing, which enhances my creativity, attention to detail, and ability to think ou
26                 </p>
27             </div>
28             <div class="team-box">
29                 
30                 <h3>Majd Hamarsheh</h3>
31                 <p class="id">ID: 1212036</p>
32                 <p>
33                     I am a fourth-year computer science student at Birzeit University, with a strong interest in technology, innovation, and AI/mach
34                     I am skilled in several programming languages, including Java and C. I thrive in team settings and value creativity and leadershi
35                     which have contributed to the success of group projects like developing an Android app for renting cars as part
36                     Outside of academics, I am committed to volunteer work.
37                 </p>
38             </div>
39         </div>

```

figure 17:main\_en.html - Task 2

```

40         <div class="team-box">
41             
42             <h3>Sama Wahidee</h3>
43             <p class="id">ID: 1211503</p> <
44             <p>
45                 I am a computer science student with strong skills in Java, C, and JavaScript, and a deep passion for technology and problem-sol
46                 I recently completed an internship in .NET development, where I contributed to building a Task Management System.
47                 I am also passionate about volunteering for causes that make a positive impact on society.
48             </p>
49         </div>
50     </div>
51 </section>
52 <section class="concept-section">
53     <h2 class="concept-title">HTTP: An Overview</h2>
54     <div class="concept-box">
55         
56         <h3 class="highlight-title">What is HTTP?</h3>
57         <h4>
58             HTTP: hypertext transfer protocol
59         </h4>
60
61         <p>
62             HTTP is an application-layer protocol for transmitting hypermedia documents, such as HTML. It was designed for communication between
63             clients and servers.
64         </p>
65         <p>
66             HTTP follows a classical client-server model, with a client opening a connection to make a request, then waiting until it receives a
67             response from the server.
68
69         <h3>HTTP request methods:</h3>
70         <p>
71             HTTP defines a set of request methods to indicate the purpose of the request and what is expected if the request is successful. Alth
72         </p>
73         <ul>
74             <li><strong>GET : </strong>
75                 The GET method requests a representation of the specified resource. Requests using GET should only retrieve data and should not

```

figure 18:main\_en.html - Task 2

```

75      <li><strong>HEAD : </strong>
76          The HEAD method asks for a response identical to a GET request, but without a response body.</li>
77      <li><strong>POST : </strong>
78          The POST method submits an entity to the specified resource, often causing a change in state or side effects on the server.</li>
79      <li><strong>PUT : </strong>
80          The PUT method replaces all current representations of the target resource with the request content.</li>
81
82  </ul>
83
84  <h3>HTTP response status codes:</h3>
85  <p>
86      HTTP response status codes indicate whether a specific HTTP request has been successfully completed. Responses are grouped in five categories:
87  </p>
88  <ol>
89      <li>Informational responses (100 – 199).</li>
90      <li>Successful responses (200 – 299).</li>
91      <li>Redirection messages (300 – 399).</li>
92      <li>Client error responses (400 – 499).</li>
93      <li>Server error responses (500 – 599).</li>
94  </ol>
95  <h3>Using HTTP cookies:</h3>
96
97
98  <p>
99      A cookie (also known as a web cookie or browser cookie) is a small piece of data a server sends to a user's web browser. The browser stores the cookie and sends it back to the server each time the user makes a request.
100  </p>
101  <h4>What cookies are used for</h4>
102
103  
104
105  <h4>Cookies are mainly used for three purposes:</h4>
106  <ul>
107      <li><strong>Management:</strong> User sign-in status, shopping cart contents, game scores, or any other user session-related details</li>
108      <li><strong>Personalization:</strong> User preferences such as display language and UI theme.</li>

```

figure 19:main\_en.html - Task 2

```

108      <li><strong>Personalization:</strong> User preferences such as display language and UI theme.</li>
109      <li><strong>Tracking:</strong> Recording and analyzing user behavior.</li>
110
111  </ul>
112
113  </div>
114  </section>
115
116
117
118 </body>
119 <footer>
120     <h2>Supporting Material</h2>
121
122     <ul>
123         <li><a href="/supporting_material_en.html">Click here to visit Supporting Material Page</a></li>
124         <li><a href="https://gaia.cs.umass.edu/kurose_ross/index.php" target="_blank">Click here to visit Textbook</a></li>
125         <li><a href="https://ritaj.birzeit.edu/" target="_blank">Click here to visit Ritaj Website</a></li>
126     </ul>
127
128 </footer>
129
130 </html>

```

figure 20:main\_en.html - Task 2

- **styles\_main\_en.css**

```

1  /* General Page Styling */
2  body {
3      font-family: Arial, sans-serif;
4      margin: 0;
5      padding: 0;
6      background-color: #fffeef; /* Light pink background */
7      color: #e63946; /* Neutral text color */
8  }
9
10 /* Header Styling */
11 header {
12     background-color: #f77f98; /* Light red for the header */
13     color: white;
14     text-align: center;
15     padding: 20px;
16     box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2); /* Subtle shadow for depth */
17 }
18
19 /* Team Section Styling */
20 .team-section {
21     padding: 40px;
22     text-align: center;
23 }
24
25 h4 {
26     font-style: italic;
27 }
28
29 h4 {
30     color: #e63946;
31     font-style: italic;
32 }
33
34 .team-container {
35     display: flex;

```

figure 21: styles\_main\_en.css1 - Task 2

```

36     flex-wrap: wrap;
37     gap: 20px;
38     justify-content: center;
39 }
40
41 .team-box {
42     background-color: white;
43     border-radius: 10px;
44     box-shadow: 0 4px 10px rgba(0, 0, 0, 0.1);
45     padding: 20px;
46     width: 400px; /* Increase the width of the box */
47     text-align: center;
48     transition: transform 0.2s;
49     display: inline-block; /* Prevent the box from stretching vertically */
50     overflow: hidden; /* Ensure no overflowing content */
51 }
52
53 .team-box:hover {
54     transform: scale(1.05); /* slightly enlarge box on hover */
55 }
56
57 /* Image Styling */
58 .team-box img {
59     width: 150px;
60     height: 150px;
61     object-fit: cover; /* Ensures the image fills the circle without distortion */
62     border-radius: 50%; /* Makes the image circular */
63     margin-bottom: 15px;
64     max-width: 100%; /* Prevent stretching */
65     max-height: 100%; /* Prevent stretching */
66 }
67
68 /* ID Styling */
69 .team-box .id {

```

figure 22: styles\_main\_en.css - Task 2

```

70     color: #f77f98; /* Blue color for the ID text */
71     font-weight: bold; /* Make the ID stand out */
72   }
73
74   /* Paragraph Styling */
75   .team-box p {
76     font-size: 0.9em;
77     line-height: 1.4;
78     color: #666; /* Subtle text color for descriptions */
79   }
80
81   .team-box h3 {
82     margin: 10px 0;
83     font-size: 1.2em;
84     color: #e63946; /* Matches the header for consistency */
85   }
86
87   /* Concept Section Styling */
88   .concept-section {
89     padding: 40px;
90     margin: 20px auto;
91     text-align: center;
92     max-width: 1200px; /* Increased max width to make the section wider */
93   }
94
95   /* Title outside the box */
96   .concept-title {
97     color: #e63946;
98     margin-bottom: 20px;
99     font-size: 1.8em;
100    text-align: center;
101  }
102
103  /* Box styling */

```

figure 23: styles\_main\_en.css - Task 2

```

104  .concept-box {
105    background-color: #ffff; /* White background for contrast */
106    border-radius: 10px;
107    box-shadow: 0 4px 10px rgba(0, 0, 0, 0.1); /* Subtle shadow for depth */
108    padding: 20px;
109    text-align: left;
110    margin: 0 auto;
111    max-width: 1200px; /* Increased max-width */
112    width: 100%; /* Ensure the box takes up the full width within the max-width */
113  }
114
115  /* Image inside the box */
116  .concept-image {
117    float: right; /* Align the image to the right */
118    margin: 0 0 20px 20px; /* Add spacing around the image */
119    max-width: 35%; /* Further reduced image width for even more space for text */
120    height: auto;
121    border-radius: 8px;
122    box-shadow: 0 4px 10px rgba(0, 0, 0, 0.1); /* Adds depth to the image */
123  }
124
125  /* Text inside the box */
126  .concept-box p {
127    font-size: 1em;
128    line-height: 1.6;
129    color: #666;
130    margin-bottom: 20px;
131  }
132
133  /* Subheadings inside the box */
134  .concept-box h3 {
135    color: #f77f98; /* A complementary shade to maintain harmony */
136    margin-top: 20px;
137  }
138

```

figure 24: styles\_main\_en.css - Task 2

```

139  /* Lists inside the box */
140 .concept-box ul, .concept-box ol {
141   margin: 20px 0;
142   padding-left: 20px;
143 }
144
145 .concept-box ul li, .concept-box ol li {
146   margin: 10px 0;
147   font-size: 0.95em;
148   color: #666; /* Darker for readability */
149 }
150
151 /* Highlight title styling */
152 .highlight-title {
153   color: #0073e6; /* Blue color for emphasis */
154   font-weight: bold; /* Make the text bold */
155   /* Slightly larger font size */
156   margin-bottom: 10px; /* Add spacing below the title */
157 }
158
159 .center-image {
160   display: block;
161   margin-left: auto;
162   margin-right: auto;
163   max-width: 100%; /* Optional: Ensures the image stays responsive */
164   height: auto; /* Optional: Maintains aspect ratio */
165
166
167 /* Styling for Arabic link */
168 .arabic-link {
169   float: left;
170   margin-top: 10px;
171   font-size: 16px;
172   color: #e63946;
173   text-decoration: none;
174 }
```

figure 25: styles\_main\_en.css - Task 2

```

174 }
175
176
177 .arabic-link:hover {
178   color: #f77f98; /* Change color on hover */
179   text-decoration: underline;
180 }
```

figure 26: styles\_main\_en.css - Task 2

- **Web Server Interface:**

- **main\_en.html:**

The following interface appears when the web browser reaches

[http://127.0.0.1:5698/main\\_en.html](http://127.0.0.1:5698/main_en.html).

The title "Welcome to ENCS3320 - Computer Networks Webserver" is shown at the top of this page, which serves as the web server's primary entrance point. The following elements make up the page's organization:

1. **Team Member Information:** The team members' names, IDs, and photos are shown along with other details about their projects, abilities, and interests.

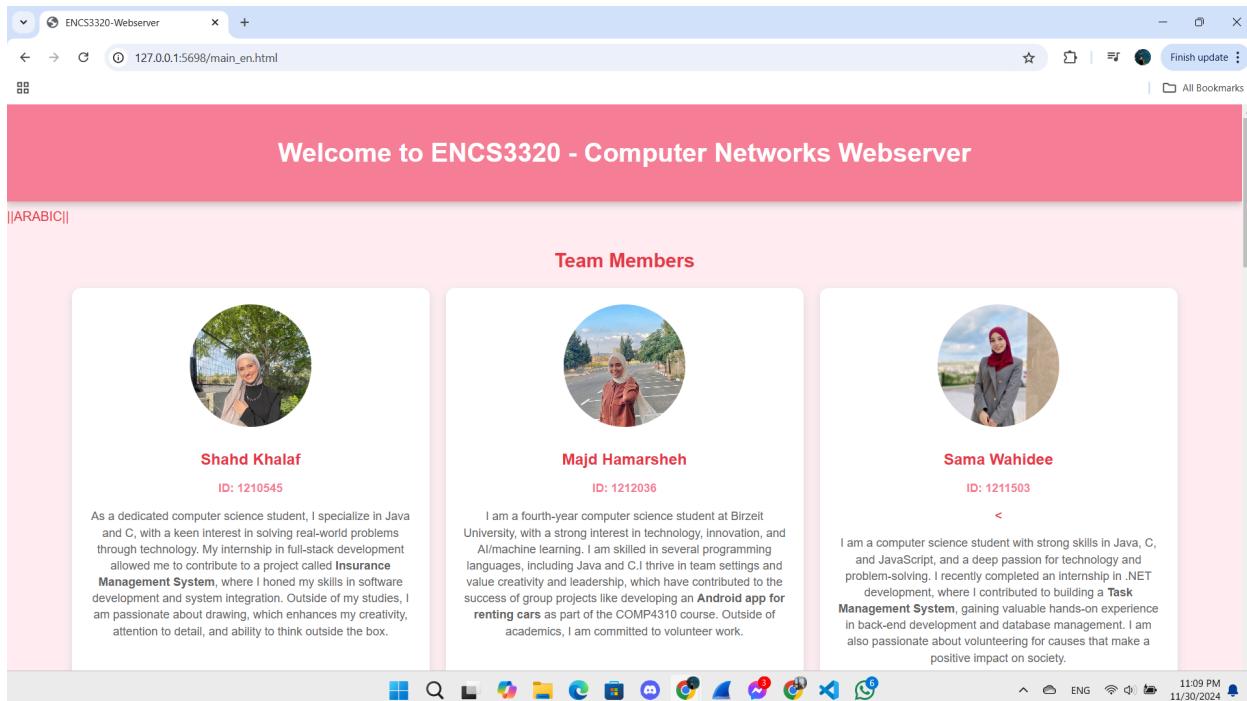


figure 27:Web Server Interface - Task 2

**2. Course Content:** Including prepared text, graphics, and lists, a topic from the course material is presented in an effective manner.[\[S\]](#)

**What is HTTP?**

**HTTP: hypertext transfer protocol**

HTTP is an application-layer protocol for transmitting hypermedia documents, such as HTML. It was designed for communication between web browsers and web servers, but it can also be used for other purposes, such as machine-to-machine communication, programmatic access to APIs, and more.

HTTP follows a classical client-server model, with a client opening a connection to make a request, then waiting until it receives a response from the server. HTTP is a stateless protocol, meaning that the server does not keep any session data between two requests, although the later addition of cookies adds state to some client-server interactions.

**HTTP request methods:**

HTTP defines a set of request methods to indicate the purpose of the request and what is expected if the request is successful. Although they can also be nouns, these request methods are sometimes referred to as HTTP verbs. Each request method has its own semantics, but some characteristics are shared across multiple methods, specifically request methods can be safe, idempotent, or cacheable.

- **GET** : The GET method requests a representation of the specified resource. Requests using GET should only retrieve data and should not contain a request content.
- **HEAD** : The HEAD method asks for a response identical to a GET request, but without a response body.
- **POST** : The POST method submits an entity to the specified resource, often causing a change in state or side effects on the server.
- **PUT** : The PUT method replaces all current representations of the target resource with the request content.

**HTTP response status codes:**

HTTP response status codes indicate whether a specific HTTP request has been successfully completed. Responses are grouped in five classes:

figure 28:Web Server Interface - Task 2

**HTTP response status codes:**

HTTP response status codes indicate whether a specific HTTP request has been successfully completed. Responses are grouped in five classes:

1. Informational responses (100 – 199).
2. Successful responses (200 – 299).
3. Redirection messages (300 – 399).
4. Client error responses (400 – 499).
5. Server error responses (500 – 599).

**Using HTTP cookies:**

A cookie (also known as a web cookie or browser cookie) is a small piece of data a server sends to a user's web browser. The browser may store cookies, create new cookies, modify existing ones, and send them back to the same server with later requests. Cookies enable web applications to store limited amounts of data and remember state information; by default the HTTP protocol is stateless.

**What cookies are used for**

figure 29:Web Server Interface - Task 2

**3. Links to Resources:** A link to a supporting local HTML file (supporting\_material\_en.html) as well as functional links to external resources (such as the Ritaj portal and the textbook website).

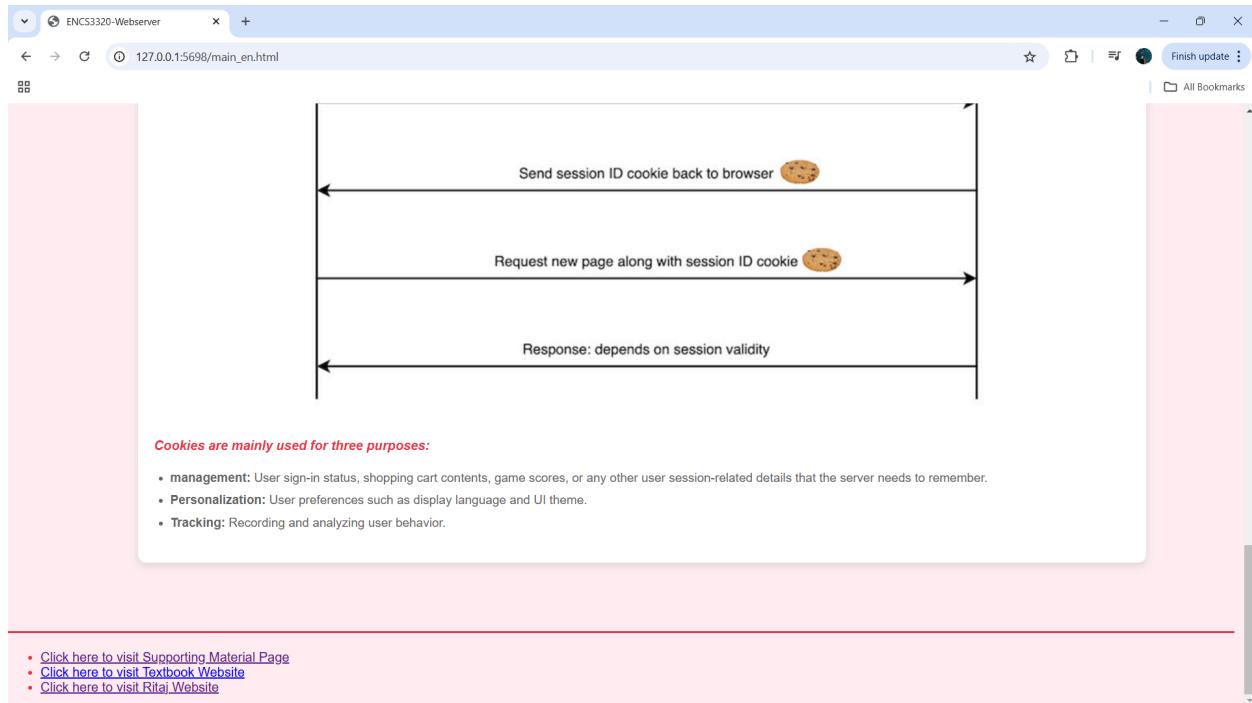


figure 30:Web Server Interface - Task 2

The server is set up to process several types of HTTP requests and, depending on the URL, reply with the relevant content. The implemented situations and the accompanying server behavior are shown below, along with images to support them:

## 1. <http://localhost:5698/>

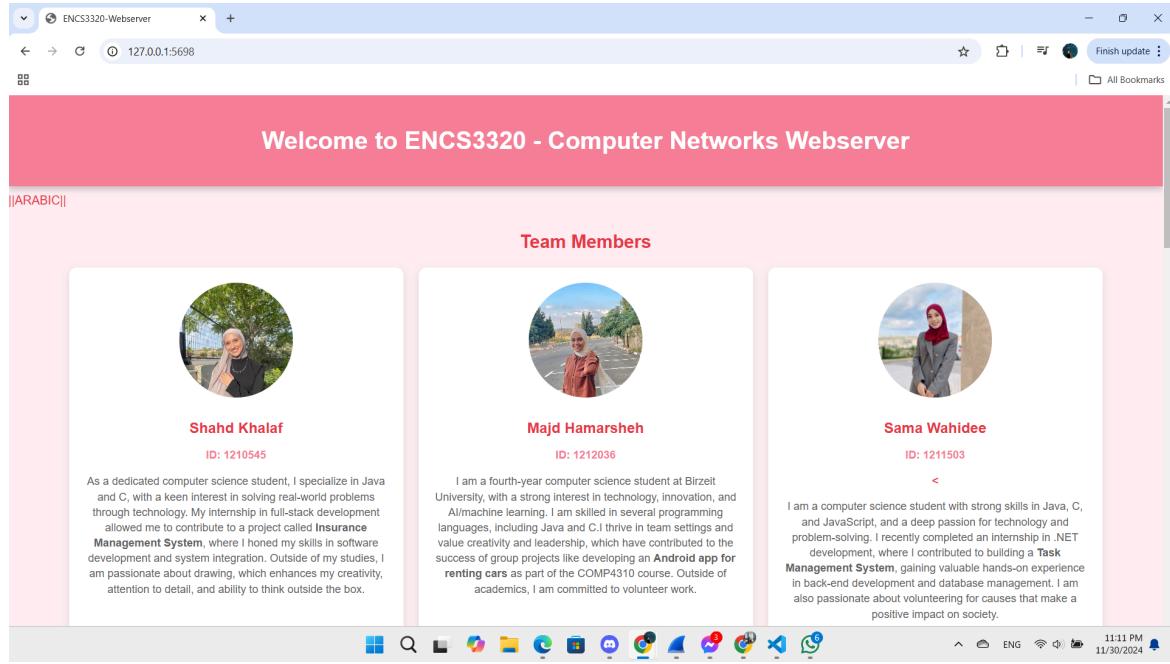


figure 31:Web Server Interface - Task 2

## 2. <http://localhost:5698/en>

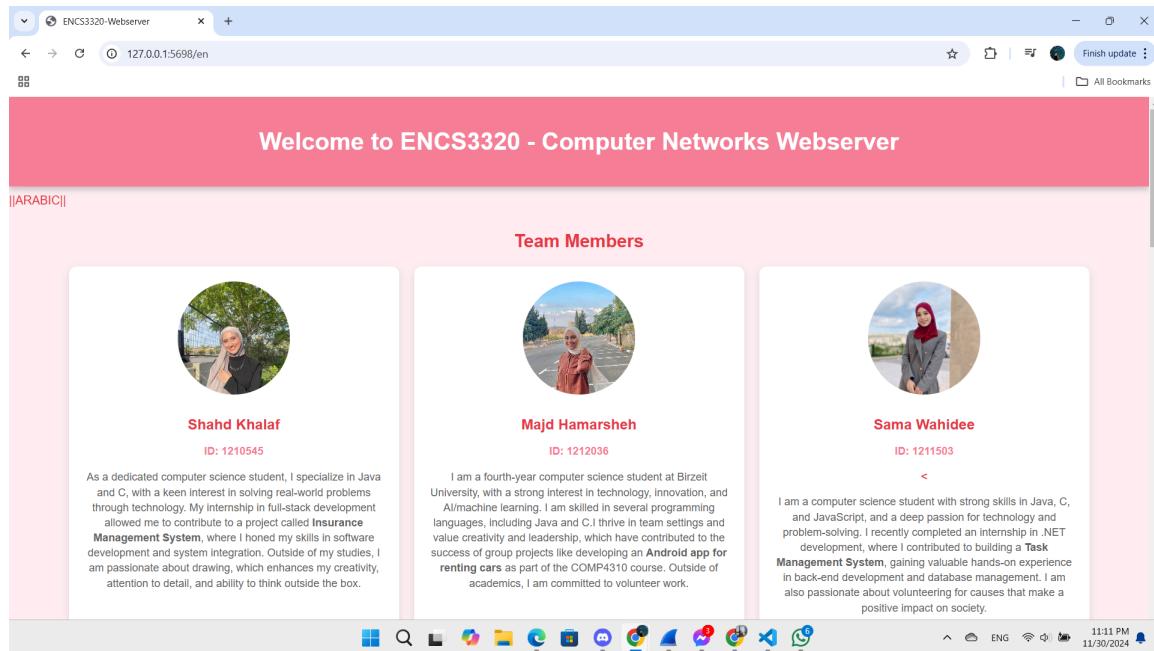


figure 32:Web Server Interface - Task 2

### 3. <http://localhost:5698/index.html>

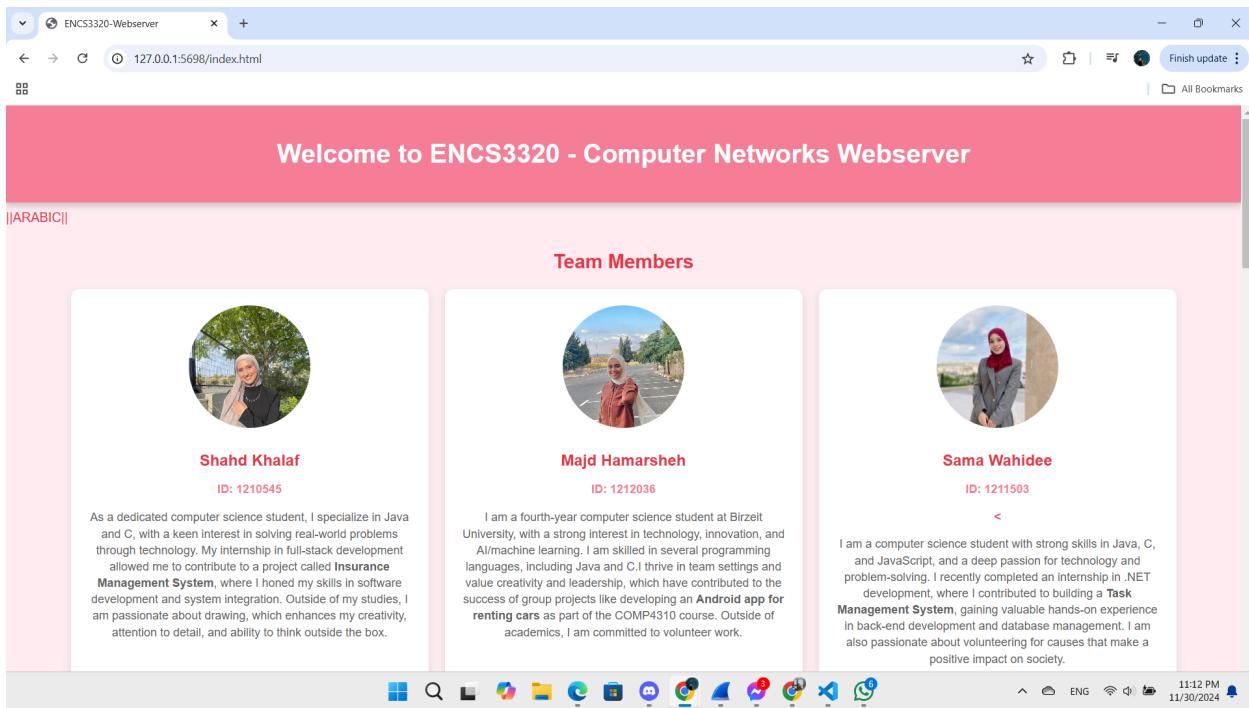


figure 33:Web Server Interface - Task 2

- **Web Server Code:**

By reading the requested file, identifying its content type (such as HTML or PNG), and creating a suitable HTTP response, the `generate_response` function dynamically responds to HTTP requests. By producing a 404 Not Found page and adding client IP and port details to the error message, it handles errors. To guarantee reliable handling, it creates a stylized fallback error page for any missing files or the `error.html`.

```

1 import os
2 import socket
3 from urllib.parse import unquote, parse_qs
4
5 # Configuration
6 HOST = '127.0.0.1'
7 PORT = 5698
8 BASE_DIR = './html' # Directory for HTML and other static files
9 IMGS_DIR = './imgs' # Directory for images
10
11 # Function to generate the HTTP response
12 def generate_response(file_path, content_type, status="200 OK", client_info=None):
13     try:
14         # Resolve the file path
15         full_path = os.path.join(os.getcwd(), file_path)
16
17         # If it's an image or video, check the file exists
18         if content_type != 'text/html' and not os.path.exists(full_path):
19             status = "404 Not Found"
20             return generate_response("html/error.html", "text/html", status=status, client_info=client_info)
21
22         # Open the file in binary mode if it's not an HTML file
23         if content_type != 'text/html':
24             with open(full_path, 'rb') as file: # Open in binary mode for image or other non-text files
25                 content = file.read()
26         else:
27             with open(full_path, 'r', encoding='utf-8') as file: # Open in text mode for HTML files
28                 content = file.read()
29
30         # For 404 errors, inject client information if provided
31         if status == "404 Not Found" and client_info:
32             content = content.replace("{client_ip}", client_info[0])
33             content = content.replace("{client_port}", str(client_info[1]))
34
35     # Build the HTTP response

```

figure 34: Web Server - Task 2

```

36     response = f"HTTP/1.1 [{status}]\r\n"
37     response += f"Content-Type: {content_type}\r\n\r\n"
38
39     # Return binary response for images or other binary files
40     if content_type != 'text/html':
41         return response.encode('utf-8') + content
42
43     # Return text response for HTML files
44     return response.encode('utf-8') + content.encode('utf-8')
45
46 except FileNotFoundError:
47     # Fallback error response if `error.html` is missing
48     fallback_message = """
49         <html>
50             <head><title>Error 404</title></head>
51             <body style="background-color:#fffeef; text-align:center; padding:50px;">
52                 <h1 style="color:#FF0000;">The file is not found</h1>
53                 <p style="color:#f77f98;">Client IP: {client_info[0]}</p>
54                 <p style="color:#f77f98;">Port: {client_info[1]}</p>
55             </body>
56         </html>
57     """
58
59     response = f"HTTP/1.1 404 Not Found\r\nContent-Type: text/html\r\n\r\n{fallback_message}"
60     return response.encode('utf-8')

```

figure 35: Web Server - Task 2

The `get_content_type` function maps well-known file extensions, such as `.html`, `.css`, and `.png`, to their corresponding MIME types in order to determine the HTTP Content-Type depending on the file's extension. To securely handle binary files, it falls back to application/octet-stream for unsupported or unknown extensions.

```

62 # Function to get the content type based on file extension
63 def get_content_type(file_path):
64     # Determine the content type based on the file extension
65     extension = os.path.splitext(file_path)[1].lower()
66     return {
67         '.html': 'text/html',
68         '.css': 'text/css',
69         '.png': 'image/png',
70         '.jpg': 'image/jpeg',
71         '.jpeg': 'image/jpeg',
72         '.gif': 'image/gif',
73         '.mp4': 'video/mp4',
74         '.avi': 'video/avi',
75         '.mov': 'video/quicktime',
76     }.get(extension, 'application/octet-stream') # Default to binary for unknown types

```

figure 36: Web Server - Task 2

Incoming HTTP requests are parsed by the `handle_request` function to identify the proper answer. It delivers static files (like CSS and images), predefined pages (like `main_en.html` and `main_ar.html`), and dynamically looks for files in the `imgs/` directory. Malformed requests yield a 400 Bad Request response, whereas missing files send a 404 Not Found page.

```

78 # Function to handle requests
79 def handle_request(request, client_address):
80     # Parse the request URL
81     lines = request.split('\n')
82     if len(lines) > 0:
83         request_line = lines[0]
84         parts = request_line.split(' ')
85         if len(parts) > 1:
86             path = parts[1].lstrip('/') # Remove leading '/' for paths
87             print(f'Requested path: {path}')
88
89             # Serve main English page
90             if path in ['', 'en', 'index.html', 'main_en.html']:
91                 return generate_response("html/main_en.html", "text/html")
92
93             # Serve supporting material page (English)
94             elif path == 'supporting_material_en.html':
95                 return generate_response("html/supporting_material_en.html", "text/html")
96
97             # Serve supporting material page (Arabic)
98             elif path == 'supporting_material_ar.html':
99                 return generate_response("html/supporting_material_ar.html", "text/html")
100
101             # Serve main Arabic page
102             elif path in ['ar', 'main_ar.html']:
103                 return generate_response("html/main_ar.html", "text/html")
104
105             # Check if the file exists in the imgs directory for images
106             elif path.startswith('imgs/'):
107                 file_name = path.split('/')[1] # Extract the image name from the path
108                 full_file_path = os.path.join(IMG_DIR, file_name)
109
110                 if os.path.exists(full_file_path):
111                     content_type = get_content_type(path)
112                     return generate_response(path, content_type)
113                 else:
114                     # If the file does not exist, return a 404
115                     return generate_response("html/error.html", "text/html", status="404 Not Found", client_info=client_address)
116

```

figure 37: Web Server - Task 2

```

117     # Serve static files (css, images)
118     elif path.startswith('css/') or path.startswith('imgs/'):
119         content_type = get_content_type(path)
120         return generate_response(path, content_type)
121
122     # If the file is not found, redirect to error.html
123     else:
124         return generate_response("html/error.html", "text/html", status="404 Not Found", client_info=client_address)
125
126     # Default case for malformed requests
127     return "HTTP/1.1 400 Bad Request\n\nInvalid request"

```

figure 38: Web Server - Task 2

In addition to handling requests and listening for new client connections, the start\_server function initializes the server. It uses the handle\_request function to process HTTP requests, accepts client connections, and makes sure the required directories are present. Clients receive responses in either text or binary format.

```

129 # Function to start the server
130 def start_server():
131     # Ensure required directories exist
132     os.makedirs(BASE_DIR, exist_ok=True)
133     os.makedirs(IMGS_DIR, exist_ok=True)
134
135     server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
136     server_socket.bind("", PORT)
137     server_socket.listen(100)
138     print(f"Server running at http://[{HOST}]:{PORT}/")
139
140     while True:
141         client_socket, addr = server_socket.accept() # Accept a new client connection
142         print(f"Connection received from {addr}")
143
144         # Receive the request from the client
145         request = client_socket.recv(1024).decode('utf-8')
146         print("HTTP Request:")
147         print(request) # Log the request details
148
149         # Handle the request and generate a response
150         response = handle_request(request, addr)
151
152         # Send the response back to the client
153         # Check if the response is bytes; send directly if it is
154         if isinstance(response, bytes):
155             client_socket.sendall(response)
156         else: # Otherwise, encode it as utf-8 and send
157             client_socket.sendall(response.encode('utf-8'))
158
159         client_socket.close()
160
161     # Start the server
162     if __name__ == '__main__':
163         start_server()

```

figure 39: Web Server - Task 2

When a client requests a file that is not on the server—for example, because of an incorrect URL or a missing resource—the 404 error page is displayed. When the server is unable to locate the requested file, it returns a "404 Not Found" status, displaying the error message and, for debugging purposes, the client's IP address and port.

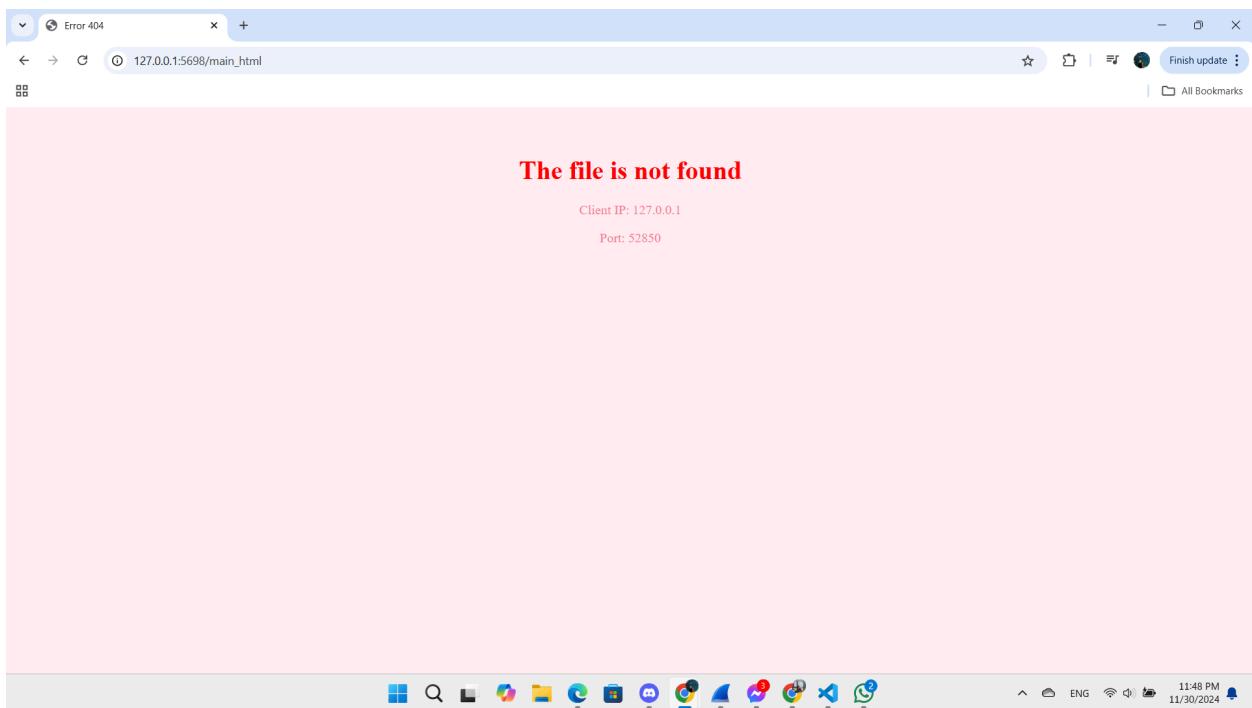


figure 40: Web Server Error - Task 2

## Section B - Supporting Material Page:

- **Code:**

- **supporting\_material\_en.html:**

```

1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="utf-8">
5      <title>Supporting Materials</title>
6      <link rel="stylesheet" href="/css/style_en.css">
7  </head>
8  <body>
9  <a class="top-right-link" href="supporting_material_ar.html"|| ARABIC ||></a>
10
11 <div class="left-half">
12     <!-- Form to input file name -->
13     <div class="image-form">
14         <form id="fileForm">
15             <h2>Request an Image or Video</h2>
16             <input type="text" id="fileName" class="image-input" placeholder="Enter the file name (e.g., http_protocol)">
17             <button type="submit" class="submit-btn">Submit</button>
18         </form>
19     </div>
20
21     <!-- Container to display the image or video -->
22     <div class="file-container" id="fileContainer"></div>
23 </div>
24
25 <div class="right-half">
26     <div class="center-text">
27         <button class="button" onclick="window.location.href='main_en.html'">Go Back To The Main Page</button>
28     </div>
29 </div>
30
31 <script>
32     function handleFileRequest(event) {
33         event.preventDefault(); // Prevent default form submission
34         const fileName = document.getElementById('fileName').value.trim();
35

```

figure 41:supporting\_material\_en.html - Task 21

```

36     if (fileName) {
37         const serverUrl = `http://127.0.0.1:5698/imgs/${encodeURIComponent(fileName)}`; // Adjusted URL format
38
39         // Check if the file exists using fetch
40         fetch(serverUrl, { method: 'GET' })
41             .then(response => {
42                 const fileContainer = document.getElementById('fileContainer');
43                 const fileExtension = fileName.split('.').pop().toLowerCase();
44
45                 if (response.status === 200) {
46                     // Serve the image or video
47                     if(['png', 'jpg', 'jpeg', 'gif'].includes(fileExtension)) {
48                         fileContainer.innerHTML = ``;
49                     } else if(['mp4', 'avi', 'mov'].includes(fileExtension)) {
50                         fileContainer.innerHTML = `<video controls><source src="${serverUrl}" type="video/${fileExtension}">Your browser does not support this file type.</p>`;
51                     } else {
52                         fileContainer.innerHTML = `<p style="color: red;">Unsupported file type.</p>`;
53                     }
54                 } else if (response.status === 404) {
55                     // Redirect if the file is not found
56                     const searchUrl = fileName.endsWith('.mp4') || fileName.endsWith('.avi')
57                         ? `https://www.youtube.com/results?search_query=${encodeURIComponent(fileName)}`
58                         : `https://www.google.com/search?q=${encodeURIComponent(fileName)}&tbo=isch`;
59
60                     window.location.href = searchUrl;
61                 }
62             })
63             .catch(error => console.error('Error:', error));
64         } else {
65             alert('Please enter a valid file name.');
66         }
67     }
68
69 
```

figure 42:supporting\_material\_en.html - Task 2

```

66     }
67   }
68   document.getElementById('fileForm').addEventListener('submit', handleFileRequest);
69 
```

figure 43:supporting\_material\_en.html - Task 2

### ○ style\_en.css

```

1  body {
2    font-family: 'Poppins', sans-serif;
3    background-color: #fffee2; /* Soft pink background */
4    color: #4a4a4a;
5    margin: 0;
6    padding: 0;
7    display: flex;
8    justify-content: center;
9    align-items: center;
10   height: 100vh;
11 }
12 
```

```

13 .top-right-link {
14   position: absolute;
15   top: 20px;
16   right: 20px;
17   color: #f77f98;
18   text-decoration: none;
19   font-size: 25px;
20 }
21 
```

```

22 /* Layout styling */
23 .left-half, .right-half {
24   width: 50%;
25   padding: 20px;
26   box-sizing: border-box;
27 }
28 
```

```

29 .left-half {
30   background-color: #ffff; /* White background for form */
31   border-radius: 20px;
32   box-shadow: 0 4px 10px rgba(0, 0, 0, 0.1);
33   margin-right: 10px;
34 }
35 
```

```

36 .right-half {
37   display: flex;
38   justify-content: center;
39 }
```

figure 44:style\_en.css - Task 2

```

40  /* Form styling */
41  .image-form {
42    text-align: center;
43    margin-bottom: 20px;
44  }
45
46  .image-form h2 {
47    font-size: 1.5rem;
48    color: #e63946; /* Cute pink heading */
49  }
50
51  .image-input {
52    width: 80%;
53    padding: 10px;
54    border: 2px solid #f8b4c8;
55    border-radius: 10px;
56    margin-bottom: 10px;
57    outline: none;
58    font-size: 1rem;
59  }
60
61  .image-input:focus {
62    border-color: #e63946;
63    box-shadow: 0 0 10px #f8b4c8;
64  }
65
66  .submit-btn {
67    padding: 10px 20px;
68    background-color: #e63946; /* Vibrant pink */
69    color: white;
70    border: none;
71    border-radius: 15px;
72    font-size: 1rem;
73    cursor: pointer;
74    transition: 0.3s;
75  }
76
77  .submit-btn:hover {

```

figure 45: style\_en.css - Task 2

```

79  background-color: #f77f98; /* Softer pink */
80  transform: scale(1.1);
81 }
82
83 /* Image/Video container styling */
84 .file-container {
85   text-align: center;
86   margin-top: 20px;
87 }
88
89 .file-container img {
90   max-width: 100%;
91   border-radius: 15px;
92   margin-top: 20px;
93   box-shadow: 0 4px 10px rgba(0, 0, 0, 0.1);
94 }
95
96 /* Button styling */
97 .center-text .button {
98   padding: 15px 25px;
99   background-color: #f77f98; /* Light pink */
100  color: white;
101  border: none;
102  border-radius: 20px;
103  font-size: 1.2rem;
104  font-weight: bold;
105  cursor: pointer;
106  text-align: center;
107  transition: 0.3s;
108 }
109
110 .center-text .button:hover {
111   background-color: #e63946;
112   transform: scale(1.1);
113 }

```

figure 46: style\_en.css - Task 2

- **Web Server Interface:**

Users can request a picture or video on computer networking topics covered in the Main English Webpage (main\_en.html) via this page. In an input box, users type the name of the desired file (video or image), and the server offers the necessary redirection based on the request.

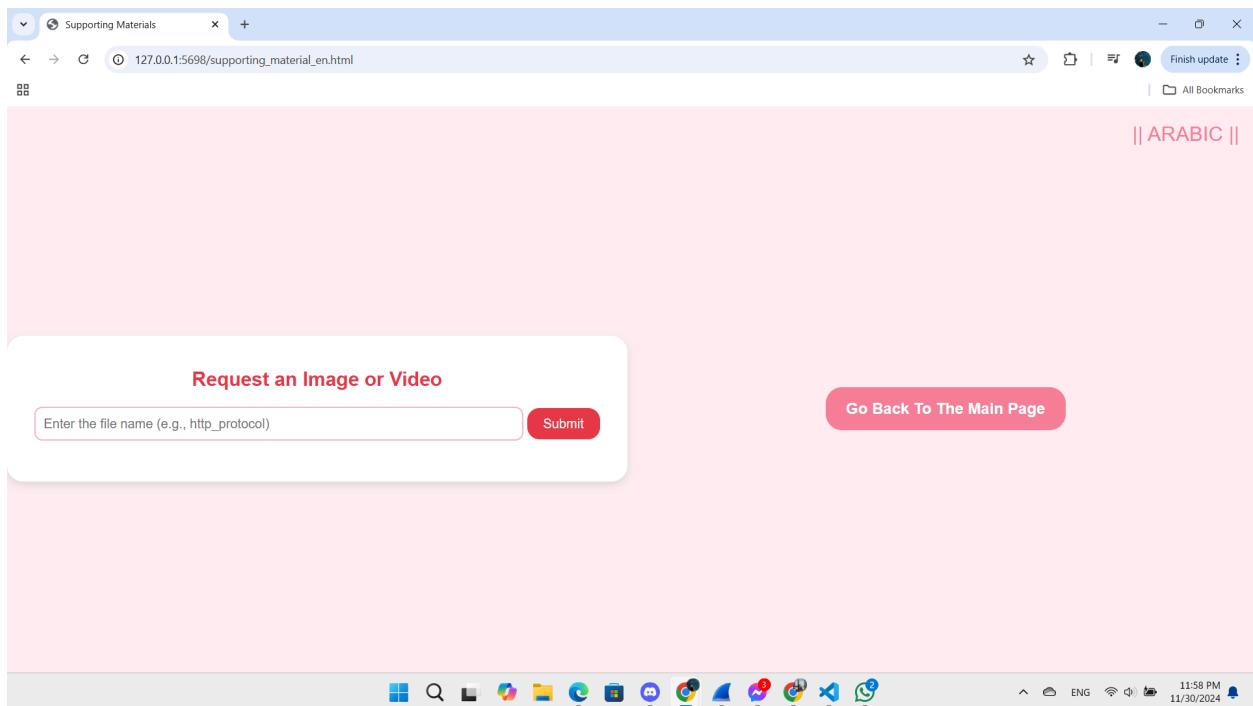


figure 47: Web Server Interface - Task 2

1. The user request an image that exist on the server:

The server will serve the image directly to the client

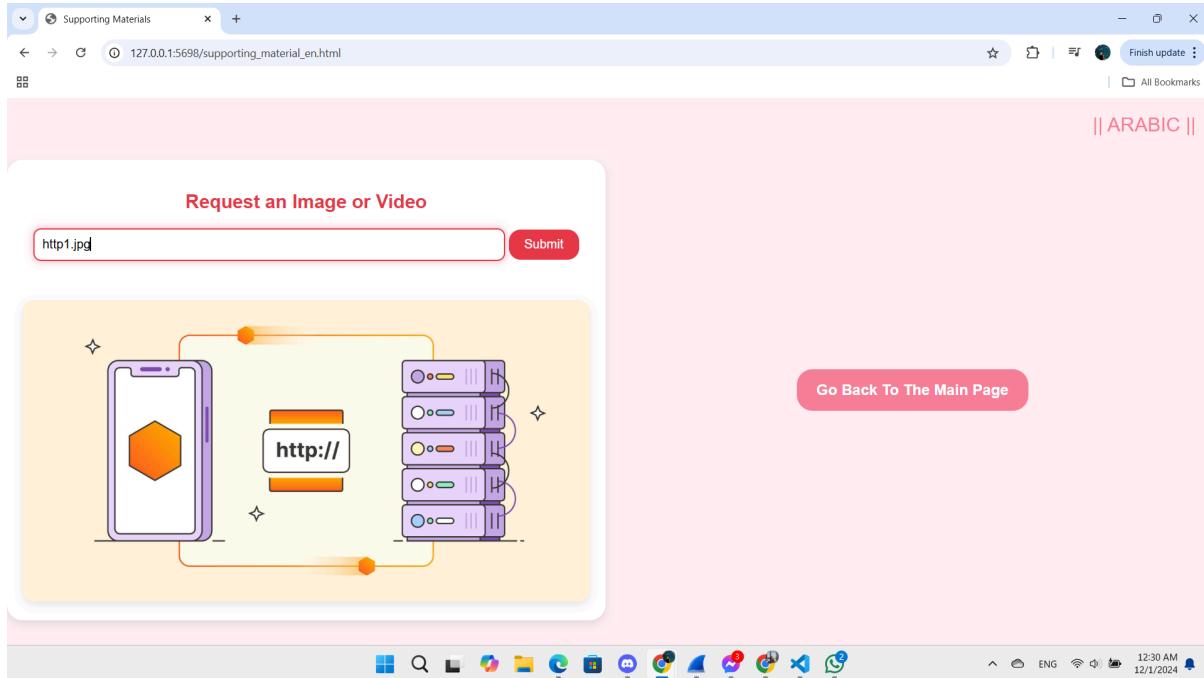


figure 48: Web Server Interface - Task 2

- **Data from the HTTP Request:**

- Requested Path:** imgs/http1.jpg (client requests an image).
- Client Connection:** 127.0.0.1:53320 (local machine, port 53320).
- HTTP Request:** GET /imgs/http1.jpg HTTP/1.1 (client requests the image).
- Accept:** Accepting image formats: image/avif, image/webp, image/png.
- Referrer:** http://127.0.0.1:5698/supporting\_material\_en.html (page requesting the image).

```

Requested path: imgs/http1.jpg
Connection received from ('127.0.0.1', 53320)
HTTP Request:
GET /imgs/http.jpg HTTP/1.1
Host: 127.0.0.1:5698
Connection: keep-alive
sec-ch-ua-platform: "Windows"
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-ch-ua: "Google Chrome";v="131", "Chromium";v="131", "Not_A_Brand";v="24"
sec-ch-ua-mobile: ?0
Accept: /*
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://127.0.0.1:5698/supporting_material_en.html
Accept-Encoding: gzip, deflate, br, zstd
Accept-Language: en-US,en;q=0.9,ar;q=0.8

```

figure 49: HTTP Request - Task 2

## 2. The user request an image that does not exist on the server:

The server will redirect to a Google search URL that filters results to images based on the user's input

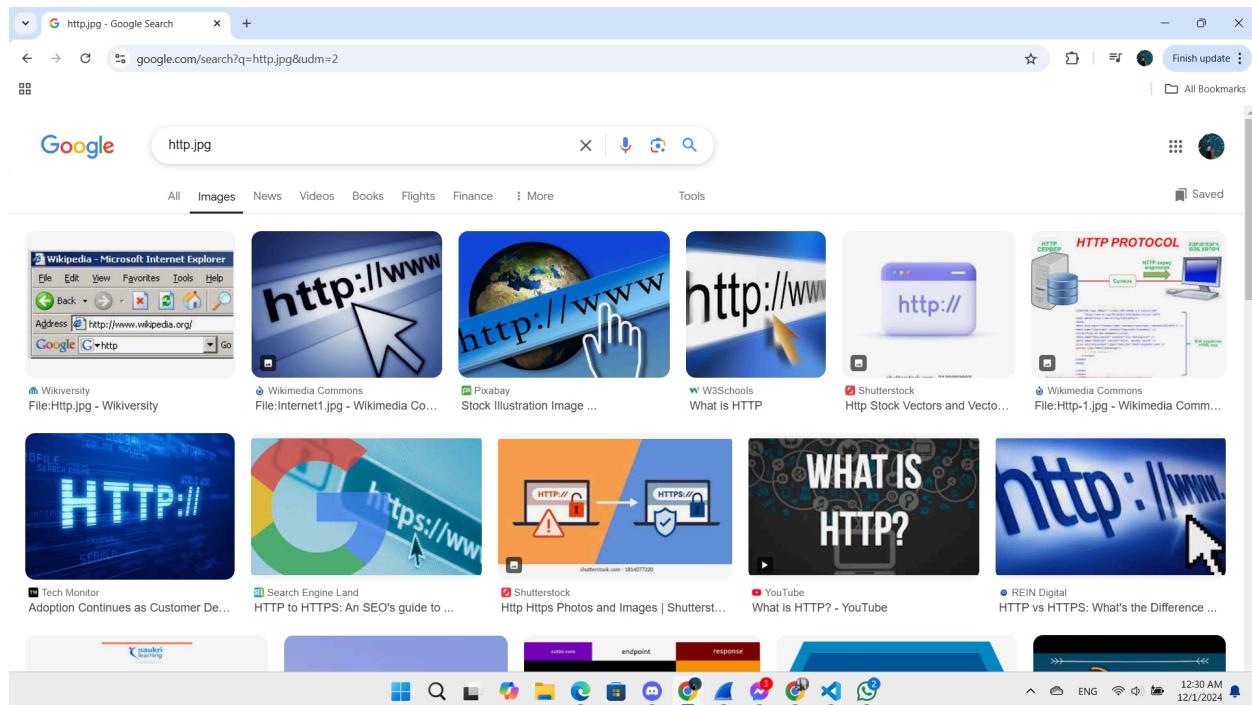


figure 50: HTTP Response - Task 2

- **Data from the HTTP Request:**

- Requested Path:** imgs/http.jpg (client requests an image).
- Client Connection:** 127.0.0.1:53766 (local machine, port 53766).
- HTTP Request:** GET /imgs/http.jpg HTTP/1.1 (client requests the image).
- Accept:** Accepting all types of content (\*/\*).
- Referrer:** http://127.0.0.1:5698/supporting\_material\_en.html (page initiating the request).

```
Requested path: imgs/Task3.mp4
Connection received from ('127.0.0.1', 53766)
HTTP Request:
GET /favicon.ico HTTP/1.1
Host: 127.0.0.1:5698
Connection: keep-alive
sec-ch-ua-platform: "Windows"
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-ch-ua: "Google Chrome";v="131", "Chromium";v="131", "Not_A_Brand";v="24"
sec-ch-ua-mobile: ?
Accept: image/avif,image/webp,image/apng,image/svg+xml,image/*,*/*;q=0.8
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: no-cors
Sec-Fetch-Dest: image
Referer: http://127.0.0.1:5698/supporting_material_en.html
Accept-Encoding: gzip, deflate, br, zstd
Accept-Language: en-US,en;q=0.9,ar;q=0.8
```

figure 51: HTTP Request - Task 2

3. The user request a video that does not exist on the server:

The server will redirect to a YouTube search URL that filters results to videos based on the user's input

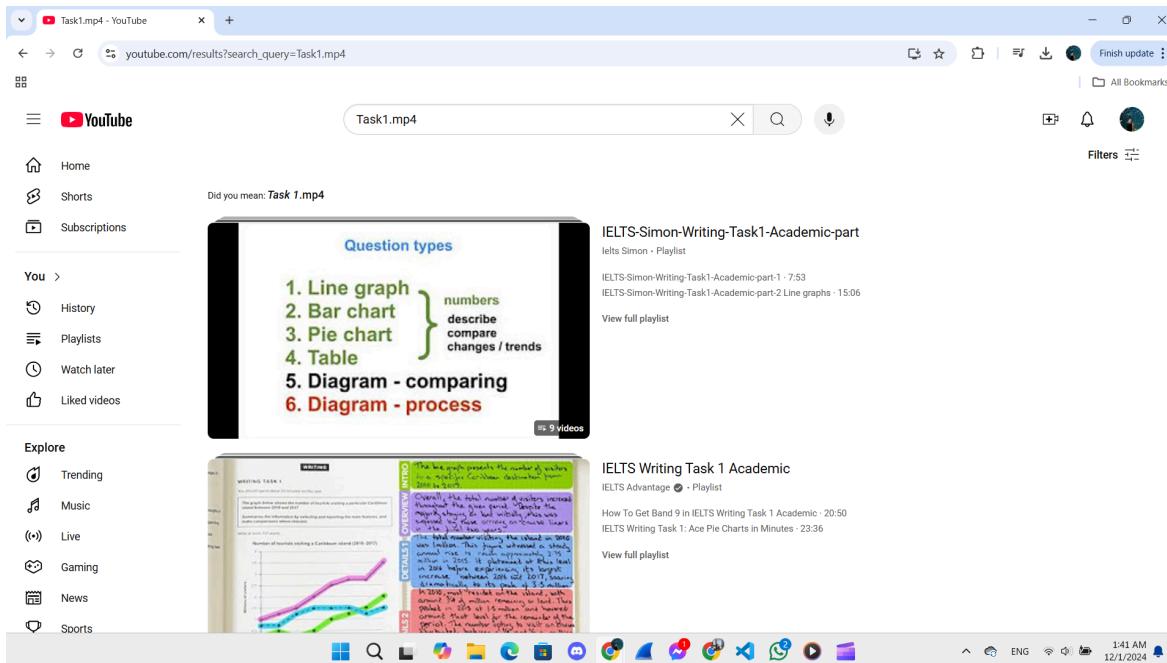


figure 52: HTTP Response - Task 2

- **Data from the HTTP Request:**

- Requested Path:** imgs/Task3.mp4 (client requests a video).
- Client Connection:** 127.0.0.1:55818 (local machine, port 55818).
- HTTP Request:** GET /imgs/Task3.mp4 HTTP/1.1 (client requests the video file).
- Accept:** Accepting all types of content (\*/\*).
- Referrer:** http://127.0.0.1:5698/supporting\_material\_en.html (page initiating the request).

```
Requested path: favicon.ico
Connection received from ('127.0.0.1', 55818)
HTTP Request:
GET /imgs/Task1.mp4 HTTP/1.1
Host: 127.0.0.1:5698
Connection: keep-alive
sec-ch-ua-platform: "Windows"
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-ch-ua: "Google Chrome";v="131", "Chromium";v="131", "Not_A_Brand";v="24"
sec-ch-ua-mobile: ?0
Accept: */*
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://127.0.0.1:5698/supporting_material_en.html
Accept-Encoding: gzip, deflate, br, zstd
Accept-Language: en-US,en;q=0.9,ar;q=0.8

Requested path: imgs/Task1.mp4
```

figure 53: HTTP Request - Task 23

4. The user request a video that exist on the server:

The server will serve the video directly to the client

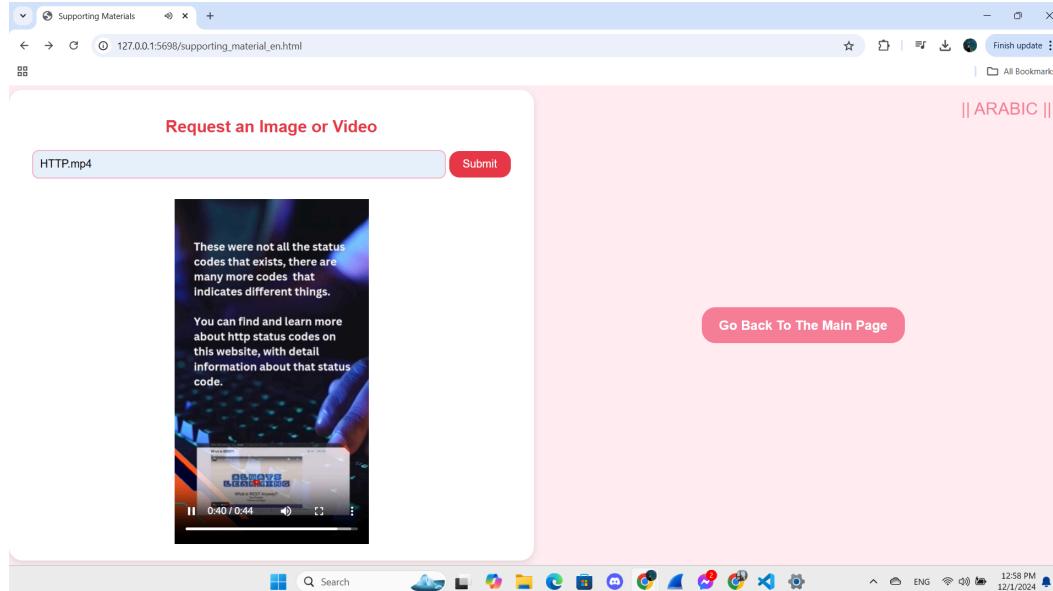


figure 54: HTTP Response - Task 2

- **Data from the HTTP Request:**
  - Requested Path:** imgs/http.mp4 (client requests a video file).
  - Client Connection:** 127.0.0.1:55788 (local machine, port 55788).
  - HTTP Request:** GET /imgs/http.mp4 HTTP/1.1 (client requests the video file).
  - Accept:** Accepting all types of content (\*/\*).
  - Referrer:** http://127.0.0.1:5698/supporting\_material\_en.html (page initiating the request)

```

Requested path: imgs/HTTP.mp4
Connection received from ('127.0.0.1', 55788)
HTTP Request:
GET /Favicon.ico HTTP/1.1
Host: 127.0.0.1:5698
Connection: keep-alive
sec-ch-ua-platform: "Windows"
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-ch-ua: "Google Chrome";v="131", "Chromium";v="131", "Not_A_Brand";v="24"
sec-ch-ua-mobile: ?0
Accept: image/avif,image/webp,image/apng,image/svg+xml,image/*,*/*;q=0.8
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: no-cors
Sec-Fetch-Dest: image
Referer: http://127.0.0.1:5698/supporting_material_en.html
Accept-Encoding: gzip, deflate, br, zstd
Accept-Language: en-US,en;q=0.9,ar;q=0.8

```

figure 55: HTTP Request - Task 2

## Section C - Arabic Versions:

The Main English Webpage and Supporting Material Page were translated into Arabic to accommodate both Arabic and English readers. Main\_ar.html and supporting\_material\_ar.html are the translated versions of the English pages main\_en.html and supporting\_material\_en.html.

- **Supporting Materia Arabic version::**

- **Code:**

- **supporting\_material\_ar.html:**

```
1  <!DOCTYPE html>
2  <html lang="ar">
3  <head>
4      <meta charset="utf-8">
5      <title>أتموا دعوة</title>
6      <link rel="stylesheet" href="/css/style_ar.css">
7  </head>
8  <body>
9
10     <a class="top-right-link" href="supporting_material_en.html" || English || </a>
11
12     <div class="left-half">
13         <!-- Form to input file name -->
14         <div class="image-form">
15             <form id="fileForm">
16                 <h2>طلب صورة أو فيديو</h2>
17                 <input type="text" id="fileName" class="image-input" placeholder="مثال: http://..."/>
18                 <button type="submit" class="submit-btn">إرسال</button>
19             </form>
20         </div>
21
22         <!-- Container to display the image or video -->
23         <div class="file-container" id="fileContainer"></div>
24     </div>
25
26     <div class="right-half">
27         <div class="center-text">
28             <button class="button" onclick="window.location.href='main_ar.html'">العودة إلى الصفحة الرئيسية</button>
29         </div>
30     </div>
31
32     <script>
33         function handleFileRequest(event) {
34             event.preventDefault(); // Prevent default form submission
35             const fileName = document.getElementById('fileName').value.trim();
36
37             if (fileName) {
38                 const serverUrl = `http://127.0.0.1:5698/imgs/${encodeURIComponent(fileName)}`; // Correct URL format
39             }
39     </script>
```

figure 56: supporting\_material\_ar.html - Task 2

```

40 // Check if the file exists using fetch
41 fetch(serverUrl, { method: 'GET' })
42 .then(response => {
43     const fileContainer = document.getElementById('fileContainer');
44     const fileExtension = fileName.split('.').pop().toLowerCase();
45
46     if (response.status === 200) {
47         // Serve the image or video
48         if(['png', 'jpg', 'gif'].includes(fileExtension)) {
49             fileContainer.innerHTML = ``;
50         } else if(['mp4', 'avi', 'mov'].includes(fileExtension)) {
51             fileContainer.innerHTML = `<video controls><source src="${serverUrl}" type="video/${fileExtension}">Your browser does not support this video format.`;
52         } else {
53             fileContainer.innerHTML = `<p style="color: red;>نوع الملف غير مدعوم</p>`;
54         }
55     } else if (response.status === 404) {
56         // Redirect if the file is not found
57         const searchUrl = fileName.endsWith('.mp4') || fileName.endsWith('.avi')
58             ? 'https://www.youtube.com/results?search_query=' + encodeURIComponent(fileName)
59             : 'https://www.google.com/search?q=' + encodeURIComponent(fileName) + '&tbo=isch';
60
61         window.location.href = searchUrl;
62     }
63 }
64 .catch(error => console.error('خطأ:', error));
65 } else {
66     alert('من فضلك أدخل اسم الملف بشكل صحيح');
67 }
68 }
69
70 document.getElementById('fileForm').addEventListener('submit', handleFileRequest);
71 </script>
72 </body>
73 </html>

```

figure 57: supporting\_material\_ar.html - Task 2

## ■ style\_ar.css:

```

1 /* General Page Styling */
2 body {
3     font-family: 'Tajawal', sans-serif; /* Arabic-friendly font */
4     background-color: #fffeef; /* Light pink background */
5     color: #4a4a4a;
6     margin: 0;
7     padding: 0;
8     display: flex;
9     justify-content: center;
10    align-items: center;
11    height: 100vh;
12    direction: rtl; /* Right to Left text direction */
13    text-align: right; /* Right-aligned text */
14 }
15
16 /* Top-right link */
17 .top-right-link {
18     position: absolute;
19     top: 20px;
20     right: 20px;
21     color: #f77f98;
22     text-decoration: none;
23     font-size: 25px;
24 }
25
26 /* Layout styling */
27 .left-half, .right-half {
28     width: 50%;
29     padding: 20px;
30     box-sizing: border-box;
31 }
32
33 .left-half {
34     background-color: #fff; /* White background for form */
35     border-radius: 20px;
36     box-shadow: 0 4px 10px rgba(0, 0, 0, 0.1);
37     margin-left: 10px;
38 }
39

```

figure 58: style\_ar.css - Task 2

```

40 .right-half {
41   display: flex;
42   justify-content: center;
43   align-items: center;
44 }
45
46 /* Form styling */
47 .image-form {
48   text-align: center;
49   margin-bottom: 20px;
50 }
51
52 .image-form h2 {
53   font-size: 1.5rem;
54   color: #e63946; /* Cute pink heading */
55 }
56
57 .image-input {
58   width: 80%;
59   padding: 10px;
60   border: 2px solid #f8b4c8;
61   border-radius: 10px;
62   margin-bottom: 10px;
63   outline: none;
64   font-size: 1rem;
65 }
66
67 .image-input:focus {
68   border-color: #e63946;
69   box-shadow: 0 0 10px #f8b4c8;
70 }
71
72 .submit-btn {
73   padding: 10px 20px;
74   background-color: #e63946; /* Vibrant pink */
75   color: white;
76   border: none;
77   border-radius: 15px;
78   font-size: 1rem;

```

figure 59: style\_ar.css - Task 2

```

79   cursor: pointer;
80   transition: 0.3s;
81 }
82
83 .submit-btn:hover {
84   background-color: #f77f98; /* Softer pink */
85   transform: scale(1.1);
86 }
87
88 /* Image/Video container styling */
89 .file-container {
90   text-align: center;
91   margin-top: 20px;
92 }
93
94 .file-container img {
95   max-width: 100%;
96   border-radius: 15px;
97   margin-top: 20px;
98   box-shadow: 0 4px 10px rgba(0, 0, 0, 0.1);
99 }
100
101 /* Button styling */
102 .center-text .button {
103   padding: 15px 25px;
104   background-color: #f77f98; /* Light pink */
105   color: white;
106   border: none;
107   border-radius: 20px;
108   font-size: 1.2rem;
109   font-weight: bold;
110   cursor: pointer;
111   text-align: center;
112   transition: 0.3s;
113 }
114

```

figure 60: style\_ar.css - Task 2

- **Web Server Interface :**

The supporting materials are also available in Arabic on this website, with the format and content modified for Arabic speakers.

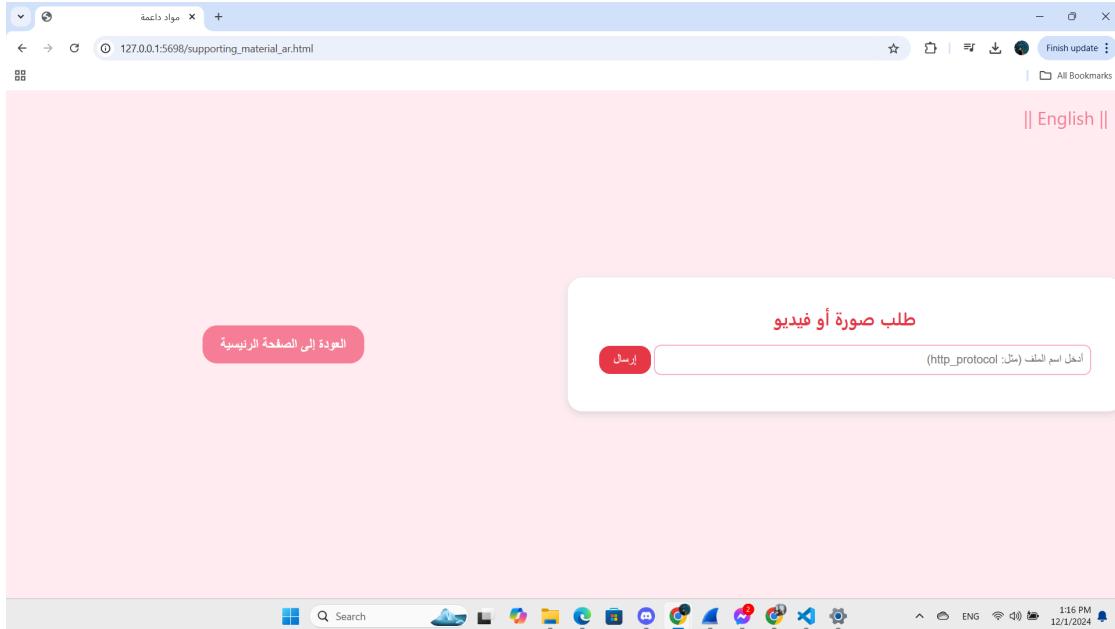


figure 61:Web Server Interface - Task 2

- **Data from the HTTP Request:**

- Requested Path:** css/style\_ar.css (client requests the Arabic CSS stylesheet).
- Client Connection:** 127.0.0.1:55890 (local machine, port 55890).
- HTTP Request:** GET /css/style\_ar.css HTTP/1.1 (client requests the CSS file).
- Accept:** Accepting text/css for CSS files and all other content (\*/\*).
- Referrer:** http://127.0.0.1:5698/supporting\_material\_ar.html (the page requesting the CSS).

```
Requested path: supporting_material_ar.html
Connection received from ('127.0.0.1', 55890)
HTTP Request:
GET /css/style_ar.css HTTP/1.1
Host: 127.0.0.1:5698
Connection: keep-alive
sec-ch-ua-platform: "Windows"
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-ch-ua: "Google Chrome";v="131", "Chromium";v="131", "Not_A_Brand";v="24"
sec-ch-ua-mobile: ?
Accept: text/css,*/*;q=0.1
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: no-cors
Sec-Fetch-Dest: style
Referer: http://127.0.0.1:5698/supporting_material_ar.html
Accept-Encoding: gzip, deflate, br, zstd
Accept-Language: en-US,en;q=0.9,ar;q=0.8

Requested path: css/style_ar.css
```

figure 62: HTTP Request - Task 2

- Same as supporting\_material\_ar.html this designed to allow users to request images or videos by entering a search term in an input form. Depending on the requested file type, the server either serves the requested content (if available) or redirects the user to external search engines



figure 63:Web Server Interface - Task 2

1. The user request an image that exist on the server:  
The server will serve the image directly to the client

- **Data from the HTTP Request:**

- Requested Path:** imgs/http1.jpg (client requests an image).
- Client Connection:** 127.0.0.1:55974 (local machine, port 55974).
- HTTP Request:** GET /imgs/http1.jpg HTTP/1.1 (client requests the image).
- Accept:** Accepting image formats like image/avif, image/webp, and image/png.
- Referrer:** http://127.0.0.1:5698/supporting\_material\_ar.html (page requesting the image).

```

Requested path: imgs/http1.jpg
Connection received from ('127.0.0.1', 55974)
HTTP Request:
GET /imgs/http1.jpg HTTP/1.1
Host: 127.0.0.1:5698
Connection: keep-alive
sec-ch-ua-platform: "Windows"
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-ch-ua: "Google Chrome";v="131", "Chromium";v="131", "Not_A_Brand";v="24"
sec-ch-ua-mobile: ?0
Accept: image/avif,image/webp,image/apng,image/svg+xml,image/*,*/*;q=0.8
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: no-cors
Sec-Fetch-Dest: image
Referer: http://127.0.0.1:5698/supporting_material_ar.html
Accept-Encoding: gzip, deflate, br, zstd
Accept-Language: en-US,en;q=0.9,ar;q=0.8

Requested path: imgs/http1.jpg
  
```

figure 64: HTTP Request - Task 2

2. The user request a video that exist on the server:

The server will serve the video directly to the client

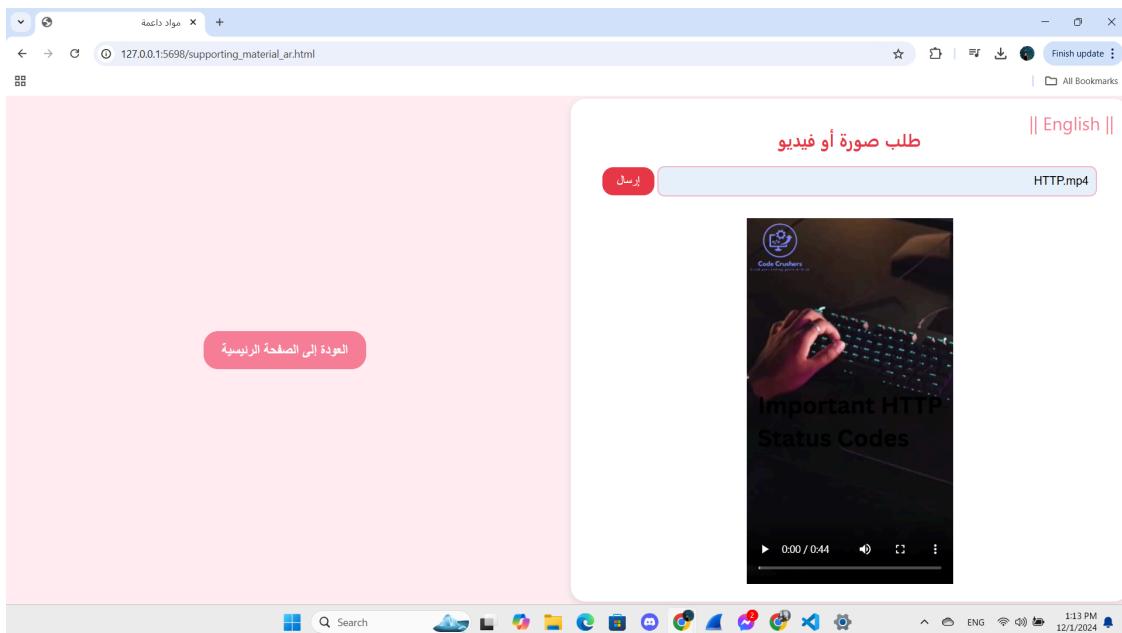


figure 65::Web Server Interface - Task 2

- **Data from the HTTP Request:**

- A. **Requested Path:** imgs/HTTP.mp4 (client requests a video file).
- B. **Client Connection:** 127.0.0.1:55951 (local machine, port 55951).
- C. **HTTP Request:** GET /imgs/HTTP.mp4 HTTP/1.1 (client requests the video file).
- D. **Accept:** Accepting all types of content (\*/\*).
- E. **Referrer:** http://127.0.0.1:5698/supporting\_material\_ar.html (the page requesting the video).

```
Requested path: imgs/HTTP.mp4
Connection received from ('127.0.0.1', 55951)
HTTP Request:
GET /imgs/HTTP.mp4 HTTP/1.1
Host: 127.0.0.1:5698
Connection: keep-alive
sec-ch-ua-platform: "Windows"
Accept-Encoding: identity;q=1, *;q=0
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-ch-ua: "Google Chrome";v="131", "Chromium";v="131", "Not_A_Brand";v="24"
sec-ch-ua-mobile: ?0
Accept: */*
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: no-cors
Sec-Fetch-Dest: video
Referer: http://127.0.0.1:5698/supporting_material_ar.html
Accept-Language: en-US,en;q=0.9,ar;q=0.8
Range: bytes=0-
Accept-Language: en-US,en;q=0.9,ar;q=0.8
Accept-Language: en-US,en;q=0.9,ar;q=0.8
Range: bytes=0-

Requested path: imgs/HTTP.mp4
```

figure 66: HTTP Request - Task 2

3. The user request a video that does not exist on the server:

The server will redirect to a YouTube search URL that filters results to videos based on the user's input

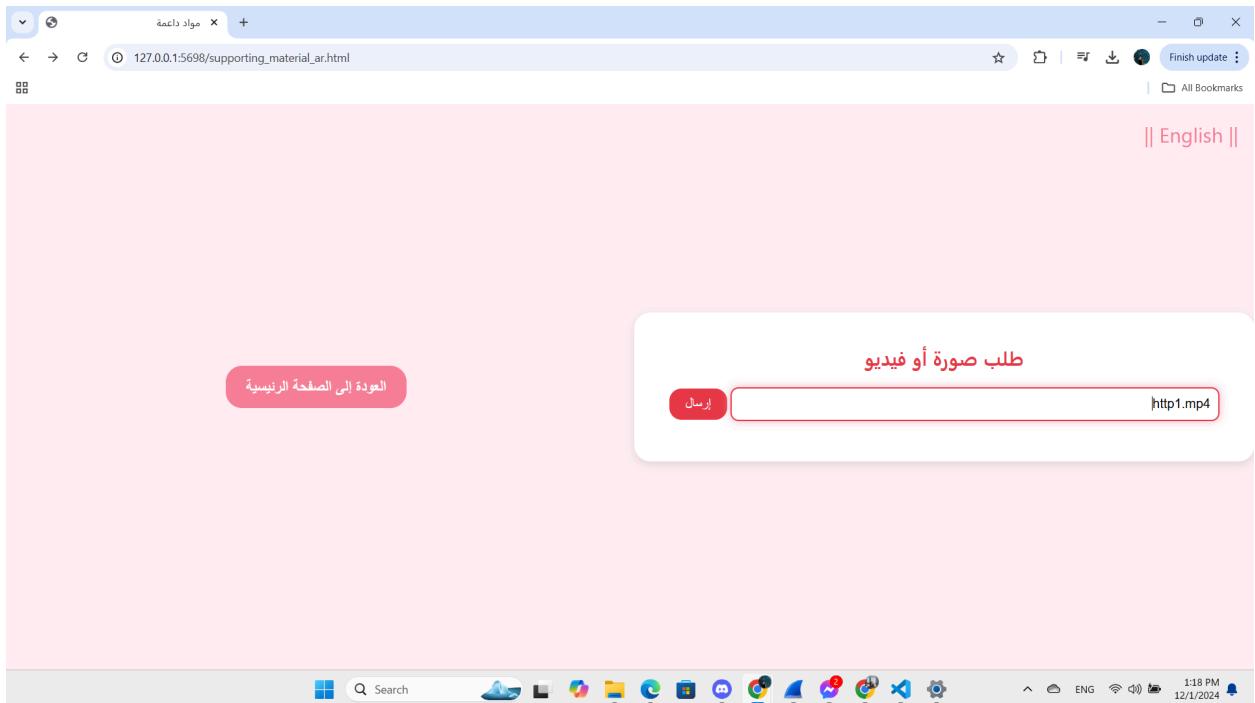


figure 67:Web Server Interface - Task 2

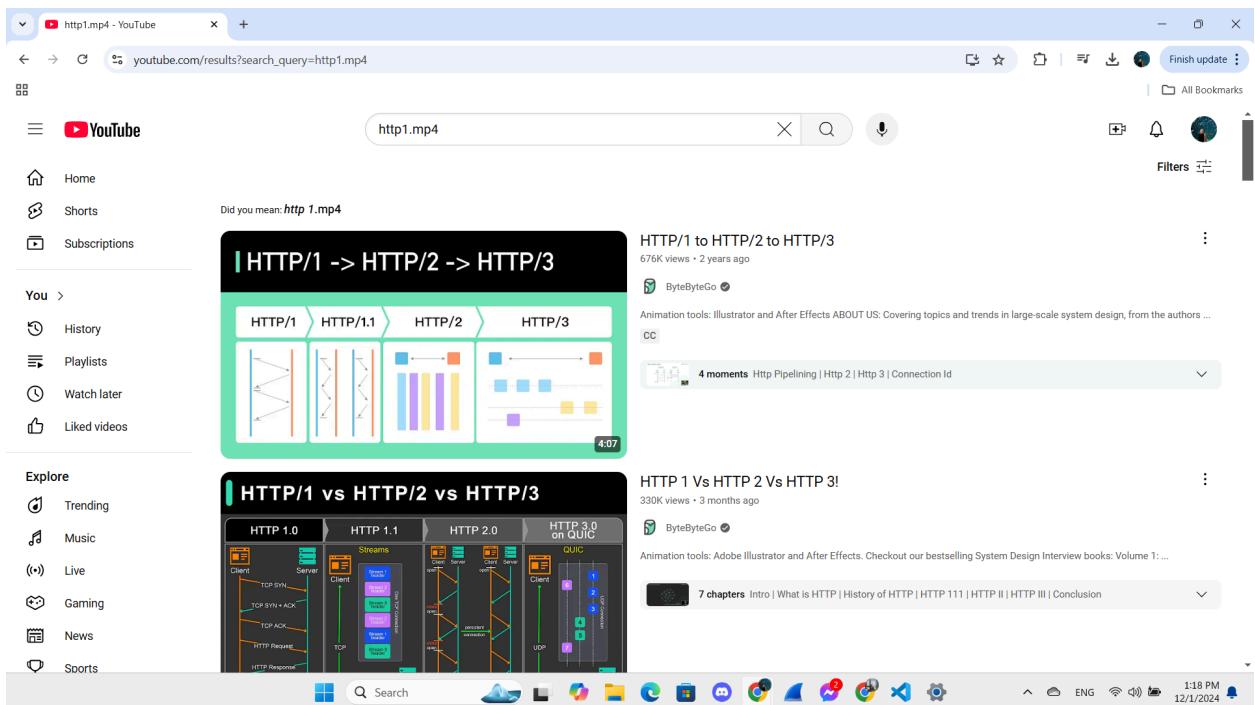


figure 68: HTTP Response - Task 2

- **Data from the HTTP Request:**

- Requested Path:** imgs/http1.mp4 (client requests a video file).
- Client Connection:** 127.0.0.1:56018 (local machine, port 56018).
- HTTP Request:** GET /imgs/http1.mp4 HTTP/1.1 (client requests the video file).
- Accept:** Accepting all types of content (\*/\*).
- Referrer:** http://127.0.0.1:5698/supporting\_material\_ar.html (the page requesting the video).

```
Requested path: imgs/http1.mp4
Connection received from ('127.0.0.1', 56018)
HTTP Request:
GET /imgs/http1.jpg HTTP/1.1
Host: 127.0.0.1:5698
Connection: keep-alive
sec-ch-ua-platform: "Windows"
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-ch-ua: "Google Chrome";v="131", "chromium";v="131", "Not_A_Brand";v="24"
sec-ch-ua-mobile: ?0
Accept: */*
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://127.0.0.1:5698/supporting_material_ar.html
Accept-Encoding: gzip, deflate, br, zstd
Accept-Language: en-US,en;q=0.9,ar;q=0.8
```

figure 69: HTTP Request - Task 2

4. The user request an image that does not exist on the server:

The server will redirect to a Google search URL that filters results to images based on the user's input

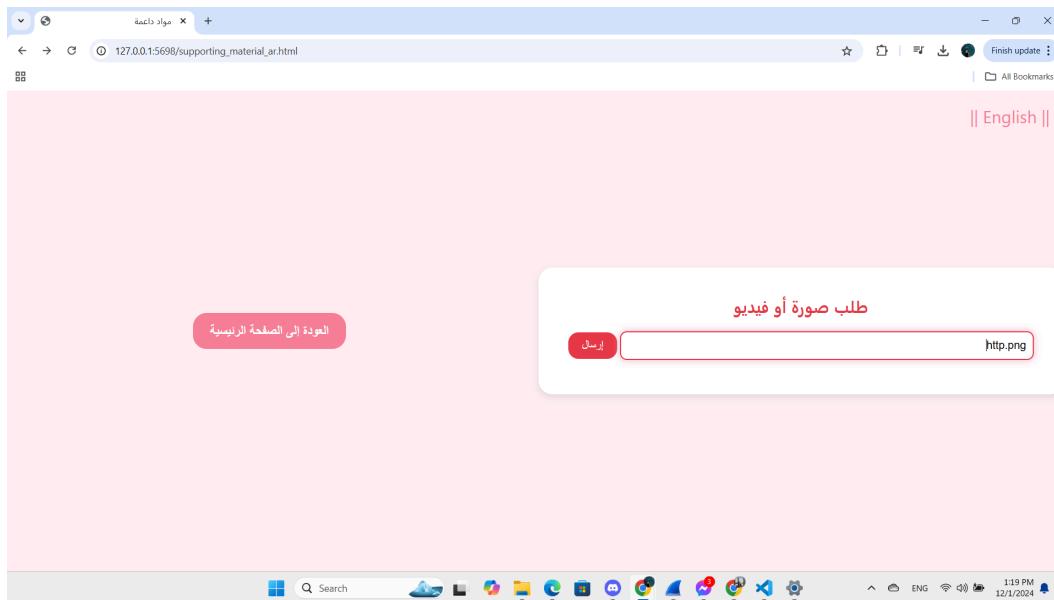


figure 70::Web Server Interface - Task 2

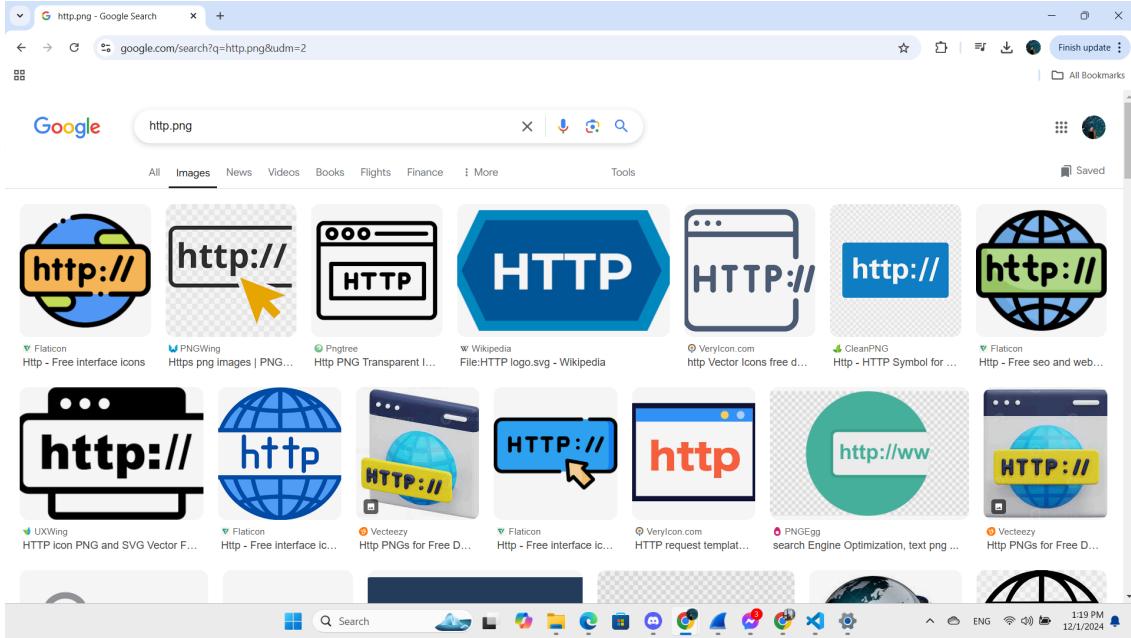


figure 71: HTTP Response - Task 2

- **Data from the HTTP Request:**

- Requested Path:** imgs/http.png (client requests an image file).
- Client Connection:** 127.0.0.1:56026 (local machine, port 56026).
- HTTP Request:** GET /imgs/http.png HTTP/1.1 (client requests the image).
- Accept:** Accepting all types of content (\*/\*).
- Referrer:** http://127.0.0.1:5698/supporting\_material\_ar.html.

```

Requested path: css/style_ar.css
Connection received from ('127.0.0.1', 56026)
HTTP Request:
GET /imgs/http.png HTTP/1.1
Host: 127.0.0.1:5698
Connection: keep-alive
sec-ch-ua-platform: "Windows"
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-ch-ua: "Google Chrome";v="131", "Chromium";v="131", "Not_A_Brand";v="24"
sec-ch-ua-mobile: ?
Accept: */*
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://127.0.0.1:5698/supporting_material_ar.html
Accept-Encoding: gzip, deflate, br, zstd
Accept-Language: en-US,en;q=0.9,ar;q=0.8

Requested path: imgs/http.png
  
```

figure 72: HTTP Request - Task 2

- **Main Page Arabic version:**

- **Code**

- **main\_ar.html:**

```

1  <!DOCTYPE html>
2  <html lang="ar">
3      <body dir="rtl">
4
5      <head>
6          <meta charset="UTF-8">
7          <meta name="viewport" content="width=device-width, initial-scale=1.0">
8          <title>ENCS3320 - خادم الويب</title>
9          <link rel="stylesheet" href="/css/styles_main_ar.css">
10     </head>
11     <body>
12         <header>
13             <h1>خادم الويب لشبكات الحاسوب - ENCS3320 بكم في</h1>
14
15         </header>
16         <a class="english-link" href="/main_en.html">|| ENGLISH ||</a>
17
18         <section class="team-section">
19             <h2>أعضاء الفريق</h2>
20             <div class="team-container">
21                 <div class="team-box">
22                     
23                     <h3>شهد خالف</h3>
24                     <p class="id">1210545</p>
25                     <p>هندام كبيير بحل مقالات العالم الحقيقي من خلال التكنولوجيا، و Java طالبة علم حاسوب في السنة الرابعة، متخصصة في لغات البرمجة كامل الأنظمة، Insurance Management System، مشاركة في مشروع يسمى full-stack development التدريب في مجال AI، خارج دراستي، أنا شغوفة بالرسم، مما يعزز إبداعي والاهتمام بالتصميم والقدرة على التفكير خارج الصندوق</p>
26
27                 </div>
28                 <div class="team-box">
29                     
30                     <h3>مجد حمارشة</h3>
31                     <p class="id">1212096</p>
32                     <p>طالبة علم حاسوب في السنة الرابعة في جامعة بيرزيت، ولدي اهتمام كبير بالتقنيات والتعلم والابتكار والقيادة، وهي وليدة الإبداع والقيادة، و Java أتقن العمل في العديد من لغات البرمجة، بما في ذلك COMP4310، والتي تساهم في نجاح المشاريع الجماعية مثل تطوير تطبيق أندويد لتأجير السيارات، وجهاز من مساق خارج المجال الأكاديمي، أنا مهتمة بالعمل التطوعي</p>
33
34
35
36
37
38
39

```

هندام كبيير بحل مقالات العالم الحقيقي من خلال التكنولوجيا، و Java طالبة علم حاسوب في السنة الرابعة، متخصصة في لغات البرمجة كامل الأنظمة، Insurance Management System، مشاركة في مشروع يسمى full-stack development التدريب في مجال AI، خارج دراستي، أنا شغوفة بالرسم، مما يعزز إبداعي والاهتمام بالتصميم والقدرة على التفكير خارج الصندوق

طالبة علم حاسوب في السنة الرابعة في جامعة بيرزيت، ولدي اهتمام كبير بالتقنيات والتعلم والابتكار والقيادة، وهي وليدة الإبداع والقيادة، و Java أتقن العمل في العديد من لغات البرمجة، بما في ذلك COMP4310، والتي تساهم في نجاح المشاريع الجماعية مثل تطوير تطبيق أندويد لتأجير السيارات، وجهاز من مساق خارج المجال الأكاديمي، أنا مهتمة بالعمل التطوعي

figure 73:main\_ar.html - Task 2

```

40
41
42         </p>
43         <div class="team-box">
44             
45             <h3>ساما وهيدي</h3>
46             <p class="id">1211503</p>
47             <p>ولدي شغف عميق بالتقنيات و حل المشكلات، وأنا طالبة علم حاسوب وأتمتع بمهارات قوية في back-end، و أكتسبت خبرة عملية قيمة في .NET ، حيث ساهمت في بناء مشروع Task Management System، لقد أكملت مؤخرًا تدريسي في تطوير JavaScript و C#، أنا أيضًا مهتمة بالعمل التطوعي في القضايا التي لها تأثير إيجابي على المجتمع</p>
48
49
50
51
52
53
54         <section class="concept-section">
55             <h2 class="concept-title">HTTP</h2>
56             <div class="concept-box">
57                 
58                 <h3 class="highlight-title">ما هو HTTP ؟</h3>
59                 <h4>
60                     HTTP:
61                     بروتوكول نقل النص التشعبي</h4>
62
63
64
65                     HTTP هو بروتوكول طبقة تطبيق لنقل مستندات الوسائط التشعبية، مثل HTML</p>
66
67
68                     HTTP
69                     يتيح العميل خادم العميل الكلاسيكي، حيث يفتح العميل اتصالاً لتقديم طلب، ثم ينتظر حتى يتلقى استجابة من الخادم</p>
70
71
72                     طرق HTTP:</h3>
73
74

```

، والوصول البرمجي إلى واجهات برمجة التطبيقات، والعديد . HTML هو بروتوكول طبقة تطبيق لنقل مستندات الوسائط التشعبية، مثل HTML

خادم العميل HTTP يتيح نموذج خادم العميل الكلاسيكي، حيث يفتح العميل اتصالاً لتقديم طلب، ثم ينتظر حتى يتلقى استجابة من الخادم

figure 74: main\_ar.html - Task 2

```

74      |      |      | وما هو متوقع في حالة نجاح الطلب، على الرغم من أنها يمكن أن تكون أيضًا أسماء، إلا أنه يشار إلى طرق الطلب هذه أحيانًا باسم أفعال
75      |      |      | </p>
76      |      |      | <ul>
77      |      |      |     <li><strong>GET : </strong>
78      |      |      |     ترداد البيانات فقط ويجب ألا تحتوي على محتوى الطلب GET تمثيلًا للمورد المحدد، يجب أن تقوم الطلبات التي تستخدم
79      |      |      |     طلب طريقة GET : </strong>
80      |      |      |     <li><strong>HEAD : </strong>
81      |      |      |     تطلب طريقة
82      |      |      |     . ولكن بدون تم الاستجابة، استجابة مماثلة لطلب HEAD تطلب طريقة
83      |      |      |     <li><strong>POST : </strong>
84      |      |      |     بـرسال كبيان إلى المورد المحدد، مما يتسبب غالباً في تغيير الحالة أو حدوث آثار جانبية على الخادم
85      |      |      |     <li><strong>PUT : </strong>
86      |      |      |     جميع التمثيلات الحالية للمورد الهدف بمحظى الطلب PUT تسبيل طريقة
87      |      |      |     </ul>
88      |      |      |     رموز حالة استجابة</h3>
89      |      |      |     <p>
90      |      |      |     محدد قد تم إكماله بنجاح، يتم تجميع الإجابات في خمس فئات HTTP إلى ما إذا كان طلب تشير رموز حالة استجابة</p>
91      |      |      |     <ol>
92      |      |      |     . الردود الإلعادية (100 - 199)
93      |      |      |     . الردود الناجحة (200 - 299)
94      |      |      |     . إعادة التوجيه (300 - 399)
95      |      |      |     . استجابات خطأ العميل (400 - 499)
96      |      |      |     . استجابات خطأ الخادم (500 - 599)
97      |      |      |     </ol>
98      |      |      |     استخدام ملفات تعريف ارتباط</h3>
99      |      |      |     <p>
100     |      |      |     ، تتيح ملفات تعريف الارتباط لتطبيقات الويب تخزين كميات محدودة من البيانات وتذكر معلومات الحالة؛ بشكل افتراضي، يكون بروتوكول</p>
101     |      |      |     <h4>
102
103
104
105
106
107

```

figure 75:main\_ar.html - Task 2

```

108      |      |      | </h4> هي ملفات تعريف الارتباط المستخدمة</p>
109      |      |      | 
110
111      |      |      | <h4>؛ تُستخدم ملفات تعريف الارتباط بشكل أساسى لثلاثة أغراض</h4>
112      |      |      | <ul>
113      |      |      |     لمستخدم، أو محتويات عربة التسوق، أو نتائج اللعبة، أو أي تفاصيل أخرى متعلقة بجلسة المستخدم والتي يحتاج الخادم إلى تذكرها</li>
114      |      |      |     . التخصيم: تفضيلات المستخدم مثل لغة العرض وموضوع واجهة المستخدم</li>
115      |      |      |     . التتبع: تسجيل وتحديث حملة المستخدم</li>
116      |      |      |     </ul>
117
118
119
120
121      |      |      | </div>
122      |      |      | </section>
123
124
125      |      |      | </body>
126      |      |      | <footer>
127      |      |      |     <h2>_
128
129      |      |      |     <ul>
130      |      |      |     <li><a href="/supporting_material_ar.html">انقر هنا لزيارة صفحة المواد الداعمة</a></li>
131      |      |      |     <li><a href="https://gaia.cs.umass.edu/kurose_ross/index.php" target="_blank">
132      |      |      |     اقتصر هنا لزيارة موقع الكتاب</a></li>
133      |      |      |     <li><a href="https://ritaj.birzeit.edu/" target="_blank">اضغط هنا لزيارة موقع ريتاج</a></li>
134      |      |      |     </ul>
135
136
137
138
139      |      |      | </footer>

```

figure 76:main\_ar.html - Task 2

## ■ styles\_main\_ar.css:

```
1 /* General Page Styling */
2 body {
3     font-family: Arial, sans-serif;
4     margin: 0;
5     padding: 0;
6     background-color: #fffeef; /* Light pink background */
7     color: #e63946; /* Neutral text color */
8     direction: rtl;
9 }
10
11 /* Header Styling */
12 header {
13     background-color: #f77f98; /* Light red for the header */
14     color: white;
15     text-align: center;
16     padding: 20px;
17     box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2); /* Subtle shadow for depth */
18 }
19
20 /* Team Section Styling */
21 .team-section {
22     padding: 40px;
23     text-align: center;
24 }
25
26 h4 {
27     font-style: italic;
28 }
29
30 h4 {
31     color: #e63946;
32     font-style: italic;
33 }
34
35 .team-container {
36     display: flex;
37     flex-wrap: wrap;
38     gap: 20px;
39 }
```

figure 77:styles\_main\_ar.css - Task 2

```
40     justify-content: center;
41 }
42
43 .team-box {
44     background-color: white;
45     border-radius: 10px;
46     box-shadow: 0 4px 10px rgba(0, 0, 0, 0.1);
47     padding: 20px;
48     width: 400px;
49     text-align: center;
50     transition: transform 0.2s;
51     display: inline-block;
52     overflow: hidden;
53 }
54
55 .team-box:hover {
56     transform: scale(1.05);
57 }
58
59 /* Image Styling */
60 .team-box img {
61     width: 150px;
62     height: 150px;
63     object-fit: cover;
64     border-radius: 50%;
65     margin-bottom: 15px;
66     max-width: 100%;
67     max-height: 100%;
68 }
69
70 /* ID Styling */
71 .team-box .id [
72     color: #f77f98; /* Blue color for the ID text */
73     font-weight: bold;
74 ]
75
76 /* Paragraph styling */
77 .team-box p {
```

figure 78:styles\_main\_ar.css - Task 2

```

78     font-size: 0.9em;
79     line-height: 1.4;
80     color: #666; /* Subtle text color for descriptions */
81   }
82
83   .team-box h3 {
84     margin: 10px 0;
85     font-size: 1.2em;
86     color: #e63946; /* Matches the header for consistency */
87   }
88
89   /* Concept Section Styling */
90   .concept-section {
91     padding: 40px;
92     margin: 20px auto;
93     text-align: center;
94     max-width: 1200px; /* Increased max width to make the section wider */
95   }
96
97   /* Title outside the box */
98   .concept-title {
99     color: #e63946;
100    margin-bottom: 20px;
101    font-size: 1.8em;
102    text-align: center;
103  }
104
105  /* Box styling */
106  .concept-box {
107    background-color: #ffff; /* White background for contrast */
108    border-radius: 10px;
109    box-shadow: 0 4px 10px rgba(0, 0, 0, 0.1); /* Subtle shadow for depth */
110    padding: 20px;
111    text-align: right;
112    margin: 0 auto;
113    max-width: 1200px; /* Increased max-width */
114    width: 100%; /* Ensure the box takes up the full width within the max-width */
115  }

```

figure 79:styles\_main\_ar.css - Task 2

```

116  /* Image inside the box */
117  .concept-image {
118    float: left; /* Align the image to the right */
119    margin: 0 0 20px 20px; /* Add spacing around the image */
120    max-width: 35%; /* Further reduced image width for even more space for text */
121    height: auto;
122    border-radius: 8px;
123    box-shadow: 0 4px 10px rgba(0, 0, 0, 0.1); /* Adds depth to the image */
124  }
125
126  /* Text inside the box */
127  .concept-box p {
128    font-size: 1em;
129    line-height: 1.6;
130    color: #666;
131    margin-bottom: 20px;
132  }
133
134  /* Subheadings inside the box */
135  .concept-box h3 {
136    color: #f77f98; /* A complementary shade to maintain harmony */
137    margin-top: 20px;
138  }
139
140  /* Lists inside the box */
141  .concept-box ul, .concept-box ol {
142    margin: 20px 0;
143    padding-right: 20px;
144  }
145
146  .concept-box ul li, .concept-box ol li {
147    margin: 10px 0;
148    font-size: 0.95em;
149    color: #666; /* Darker for readability */
150  }
151
152  /* Highlight title styling */
153  .highlight-title {

```

figure 80:styles\_main\_ar.css - Task 2

```
155 |     color: #0073e6; /* Blue color for emphasis */
156 |     font-weight: bold;
157 |     margin-bottom: 10px;
158 |
159 | .center-image {
160 |     display: block;
161 |     margin-left: auto;
162 |     margin-right: auto;
163 |     max-width: 100%;
164 |     height: auto;
165 |
166 |
167 | /* Styling for Arabic link */
168 | .english-link {
169 |     float: left;
170 |     margin-top: 10px;
171 |     font-size: 16px;
172 |     color: #e63946;
173 |     text-decoration: none;
174 |
175 | }
176 |
177 | .english-link:hover {
178 |     color: #f77f98;
179 |     text-decoration: underline;
180 |
181 | }
182 |
```

figure 81:styles\_main\_ar.css - Task 2

- **Web Server Interface :**

The following interface appears when the web browser reaches [http://127.0.0.1:5698/main\\_ar.html](http://127.0.0.1:5698/main_ar.html). The title "Welcome to ENCS3320 - Computer Networks Webserver" is shown at the top of the page, which serves as the web server's primary entrance point. This page is fully translated into Arabic for Arabic-speaking users. The following elements make up the page's organization:

1. **Team Member Information:** The team members' names, IDs, and photos are displayed along with other details about their projects, skills, and interests, all presented in Arabic to ensure accessibility for Arabic-speaking users.

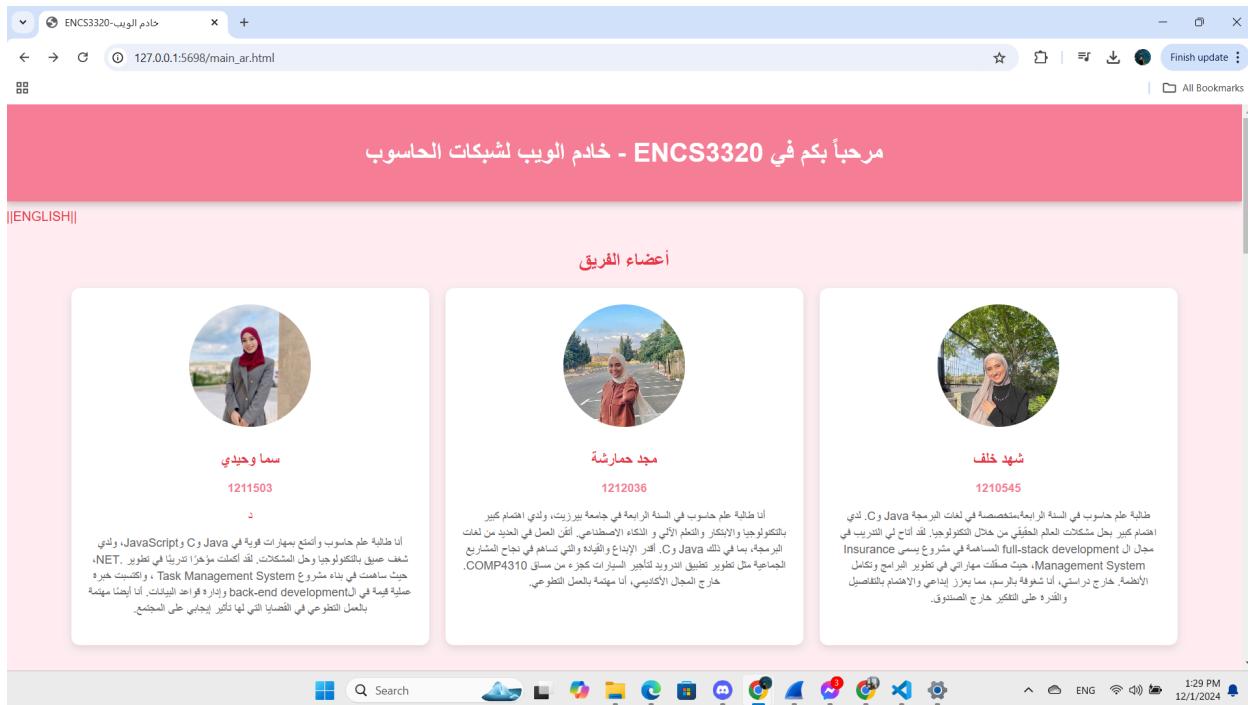


figure 82:Web Server Interface - Task 2

2. **Course Content:** A topic from the course material is presented effectively, including prepared text, graphics. This section includes key concepts from the first chapters of the textbook, with content formatted in a clear and accessible manner for Arabic-speaking users



figure 83:Web Server Interface - Task 2



figure 84:Web Server Interface - Task 2

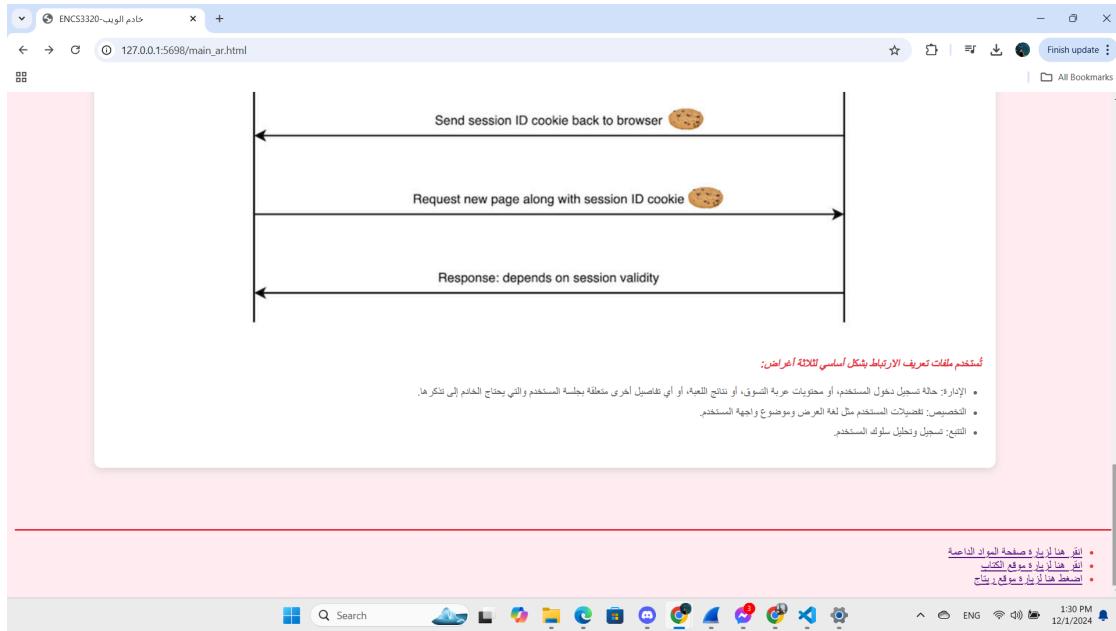


figure 85:Web Server Interface - Task 2

The server is set up to process several types of HTTP requests and, depending on the URL, reply with the relevant content. The implemented situations and the accompanying server behavior are shown below, along with images to support them:

### 1. <http://localhost:5698/ar>

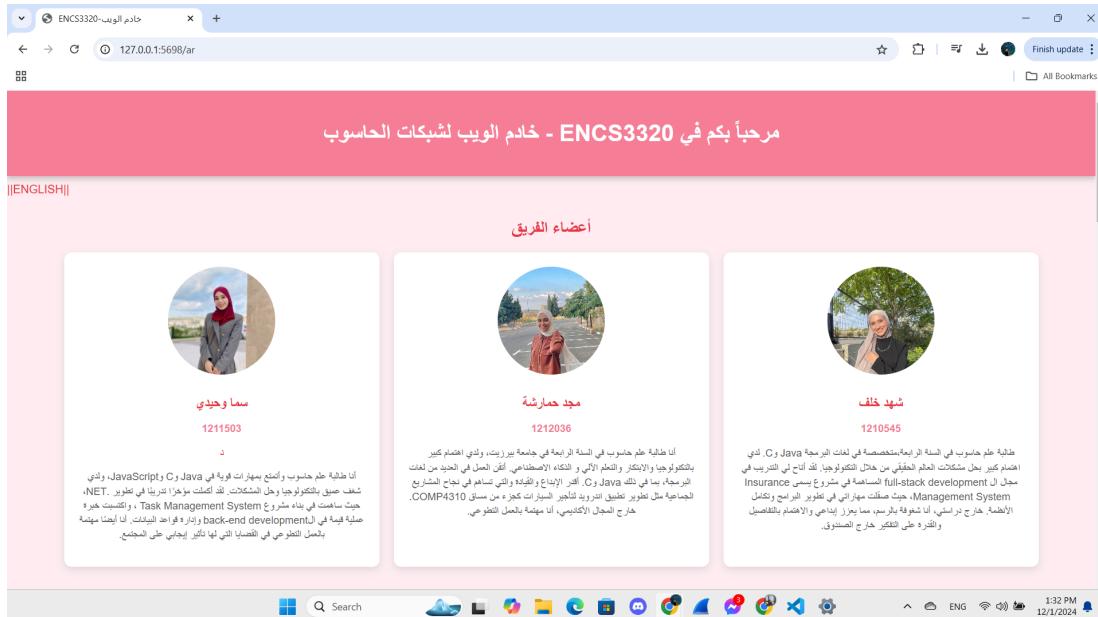


figure 86:Web Server Interface - Task 2

# Task 3

## UDP Client-Server Trivia Game Overview:

using UDP socket programming, we developed an interactive multiplayer trivia game that allows clients to connect to a server and compete in answering trivia questions. The server listens for client connections and manages the game flow, including broadcasting questions, collecting answers, tracking scores, and announcing winners. The game supports multiple rounds, and the server handles communication between clients in real time. The server also ensures fair game play by awarding points based on the speed and correctness of answers. This task highlights the use of **UDP (User Datagram Protocol)** for fast, efficient, and low-latency communication in a client-server model, making it ideal for real-time multiplayer games.

## Theory and Procedure:

### UDP vs. TCP:

For the trivia game, **UDP** (User Datagram Protocol) was chosen over **TCP** because of its low-latency, connectionless nature, which is ideal for real-time applications. UDP allows fast transmission of data without the overhead of establishing and maintaining a connection, making it suitable for time-sensitive interactions like answering trivia questions. Unlike TCP, UDP does not guarantee delivery, but for this game, occasional lost packets (like missed answers) are acceptable in exchange for speed.

### Client-Server Model:

The **client-server model** is used to structure communication between the server and multiple clients. The **server** handles the game by managing client connections, broadcasting trivia questions, collecting answers, and updating scores. **Clients** (the players) connect to the server, receive questions, and submit answers. The server is responsible for synchronizing the game flow and ensuring that all players are on the same page.

### Key Networking Concepts:

- **UDP Packets:** The game uses **UDP packets** to transmit small units of data (such as questions and answers) between the server and clients quickly.
- **IP Addresses and Ports:** Each client and the server are uniquely identified using their **IP addresses** and **port numbers**, ensuring that communication occurs between the correct devices.

- **Broadcasting Messages:** The server broadcasts trivia questions and updates to all connected clients simultaneously, keeping the game synchronized across all players.

### **Server Responsibilities:**

The server plays a central role in managing the game and ensuring smooth gameplay:

- **Client Connection Management:** The server listens on **port 5689** for incoming connections. Clients are identified by their **IP addresses** and **port numbers**, and the server maintains a list of active clients. A minimum of two clients is required to start a round.
- **Round Initialization:** Once enough clients are connected, the server broadcasts a **welcome message** to notify players that the round will begin.
- **Question Broadcast:** The server randomly selects questions and broadcasts them to all clients with a **60-second preparation period** before each question.
- **Answer Collection and Scoring:** Clients submit answers within a set time (e.g., 90 seconds). The server checks the answers, awards points, and accepts only the first answer submitted.
- **Score Tracking and Winner Announcement:** After each question, the server broadcasts the correct answer and updates the scores. At the end of each round, the server announces the leading player.
- **Round Pauses:** The server introduces short **pauses** between rounds, giving players time to review the leaderboard and prepare for the next round.

### **Client Responsibilities:**

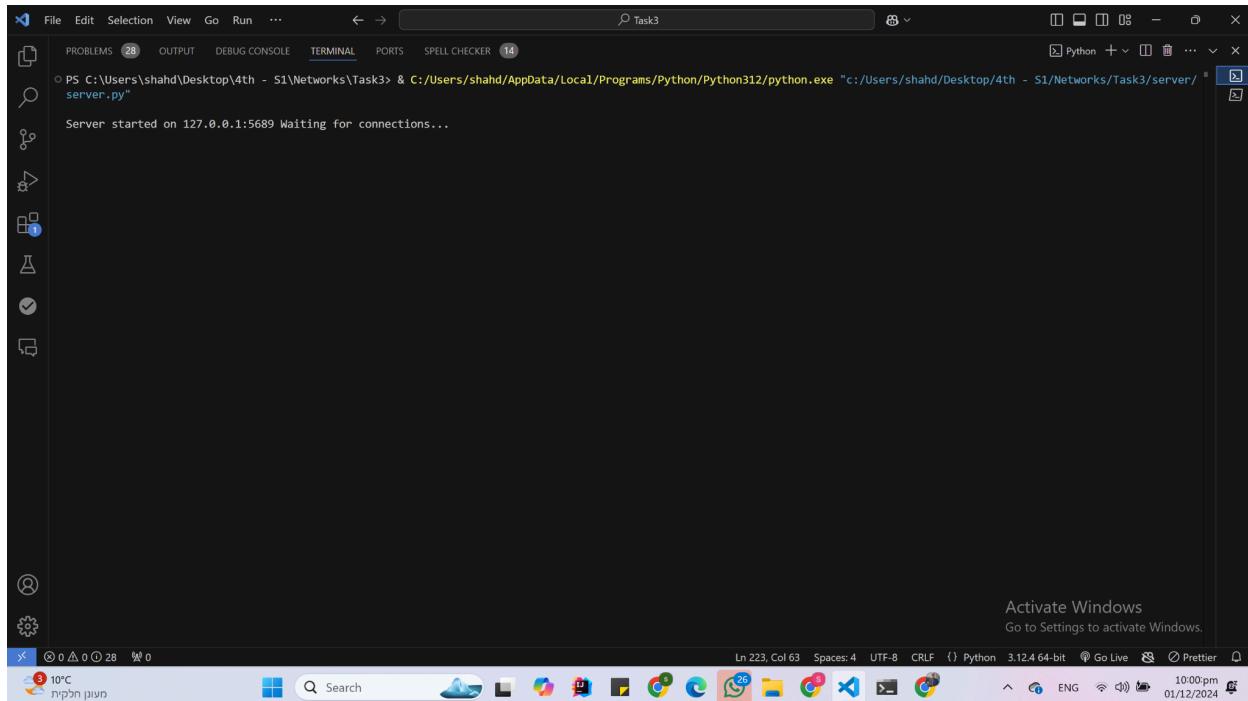
The client's role in the trivia game is to connect to the server, receive game data, and submit answers:

- **Connection Management:** The client connects to the server using the provided **IP address** and **port 5689**, submitting a **username** for the leaderboard.
- **Receiving Notifications:** The client listens for notifications from the server, including **round start**, **questions**, and **score updates**.
- **Answer Submission:** After receiving a question, the client has a **90-second window** to submit an answer to the server. The client receives feedback on whether their answer was correct.

## Game Flow and Client-Server Interaction:

This is the source code for Task3: [Task3 Source Code](#)

### 1. Server Start and Listening on Port 5689:



A screenshot of a terminal window titled "Task3". The window shows a command-line interface with the following text:  
PS C:\Users\shahd\Desktop\4th - S1\Networks\Task3> & C:/Users/shahd/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/shahd/Desktop/4th - S1/Networks/Task3/server.py"  
Server started on 127.0.0.1:5689 Waiting for connections...

figure 87: server listen on port 5689 - Task 3

The server initializes by creating a UDP socket and binding it to the local host (127.0.0.1) and port number (5689). It enters a listening state, waiting to accept incoming client connections. This stage ensures that the server is prepared to interact with clients and coordinate the trivia game.

### 2. Server Handling Multiple Clients Connections for Game Start (3 Clients):

In this step, the server successfully manages connections from three clients, ensuring all participants are ready before starting the game. Each client is identified by their IP address and port number. After a client connects and submits their name, the server displays the game rules to them. Once at least two clients are connected, the server begins a 90-second waiting period to allow additional clients to join. During this time, the server keeps all connected clients informed of the status. After the waiting period or when three clients have joined, the server finalizes the participant list and prepares to initiate the trivia game, ensuring synchronization and readiness for the upcoming rounds.

## => First client:

- The server receives a connection request from the first client.
- It prompts the client to enter their name and saves their details.
- After registering the client, the server sends a welcome message and displays the game rules to the client.
- The server informs the client that it is waiting for additional players before starting the game.

The screenshot shows a terminal window titled "python - server". The terminal output is as follows:

```
PROBLEMS 28 OUTPUT DEBUG CONSOLE TERMINAL PORTS SPELL CHECKER 14
PS C:\Users\shahd\Desktop\4th - S1\Networks\Task3> cd server
PS C:\Users\shahd\Desktop\4th - S1\Networks\Task3\server> python client.py

Connected to the server on 127.0.0.1:5689

Please enter your name:
Shahd

Welcome, Shahd! Waiting for enough players to start the game...

***** Trivia Game Rules *****
1. The game consists of 3 rounds, with 3 questions in each round.
2. You will have only 60 seconds to answer each question.
3. The faster you answer correctly, the higher your score 🎖️.
4. Players who fail to answer within the time limit will be marked as "Timeout".
5. The player with the highest total score at the end of all rounds wins!
***** Are you ready? 🎉
The game will start once we have at least 2 players. Please wait...

Waiting for at least 2 clients to join the game...
```

The terminal window is part of a larger interface with a status bar at the bottom showing "Activate Windows Go to Settings to activate Windows.", file icons, and system information like "Ln 450, Col 28", "Spaces: 4", "UTF-8", "Python 3.12.4 64-bit", "Go Live", "Prettier", and a timestamp "8:37pm 01/12/2024".

figure 88: first client joined the game - Task 3

## => Second client:

- When the second client connects, the server prompts them for their name and registers their information.
- Like the first client, the second client receives a welcome message, the **game rules as shown in the figure 89**, and an update that the server is now ready to start but will wait for more players.
- Once two clients are connected, the server initiates a 90-second waiting period for additional players to join.
- During this time, the server continues listening for new connections while keeping the current clients updated about the status.

The screenshot shows a terminal window titled "python - server". The session log is as follows:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SPELL CHECKER 14
PS C:\Users\shahd\Desktop\4th - $1\Networks\Task3> cd server
PS C:\Users\shahd\Desktop\4th - $1\Networks\Task3\server> python client.py

Connected to the server on 127.0.0.1:5689

Please enter your name:
Majd

Welcome, Majd! Waiting for enough players to start the game...

***** Trivia Game Rules *****
1. The game consists of 3 rounds, with 3 questions in each round.
2. You will have only 60 seconds to answer each question.
3. The faster you answer correctly, the higher your score 😊.
4. Players who fail to answer within the time limit will be marked as "Timeout".
5. The player with the highest total score at the end of all rounds wins!
*****
Are you ready? 🌟
The game will start once we have at least 2 players. Please wait...

***** Round 1 *****
```

The terminal is running on Windows, as indicated by the taskbar at the bottom which includes icons for File Explorer, Edge, and other applications. The system tray shows the date and time as 01/12/2024 8:37pm.

figure 89: Second client Joined The game - Task 3

## Third client:

The screenshot shows a terminal window titled "python - server". The session log is as follows:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SPELL CHECKER 14
PS C:\Users\shahd\Desktop\4th - $1\Networks\Task3> cd server
PS C:\Users\shahd\Desktop\4th - $1\Networks\Task3\server> python client.py

Connected to the server on 127.0.0.1:5689

Please enter your name:
Sama

Welcome, Sama! Waiting for enough players to start the game...

***** Trivia Game Rules *****
1. The game consists of 3 rounds, with 3 questions in each round.
2. You will have only 60 seconds to answer each question.
3. The faster you answer correctly, the higher your score 😊.
4. Players who fail to answer within the time limit will be marked as "Timeout".
5. The player with the highest total score at the end of all rounds wins!
*****
Are you ready? 🌟
The game will start once we have at least 2 players. Please wait...

***** Round 1 *****
```

The terminal is running on Windows, as indicated by the taskbar at the bottom which includes icons for File Explorer, Edge, and other applications. The system tray shows the date and time as 01/12/2024 8:38pm.

figure 90: third client joined the game - Task 3

### 3. Game Start and First Round Progression

Once the waiting period ends and the game starts, the server transitions into the first round. The following sequence describes how the server handles this phase:

#### 3.1. Broadcasting a Randomly Selected Question:

At the beginning of each question, the server selects a random unique question that hasn't been asked before. This question is broadcasted simultaneously to all active players (at the same time).

The screenshot shows a terminal window titled "Task3" with the following content:

```
***** Round 1 *****
-> Question 1: DNS is used to convert domain names into IP addresses. (True OR False)
-----
true
✓ Correct! You were the fastest! 🎉
Your current score: 1

----- Current Scores -----
Shahd: 1
Majd: 0.6453823804855346
Sama: 0

-----
-> Question 2: Which network device forwards data based on MAC addresses?
-----
⏰ Time's up! ⏰ The correct answer was: Switch

----- Current Scores -----
Shahd: 1
Majd: 1.1888142824172974
Sama: 1

-----
-> Question 3: Will we get the full mark in this project?
-----
maybe
✗ Incorrect. The correct answer was: Yes, of course! (Inshallah)
Your current score: 1

----- Current Scores -----
Shahd: 1
Majd: 1.1888142824172974
Sama: 1

-----
🎉 End of Round 1! 🎉
The winner of Round 1 is 'Majd' with 1.1888142824172974 points!
The Next Round Will Start Within 20 Seconds.. 🔥

Activate Windows
Go to Settings to activate Windows.
```

The terminal window also displays various system icons and status bars at the bottom, including file navigation, search, and system information like battery level and date/time.

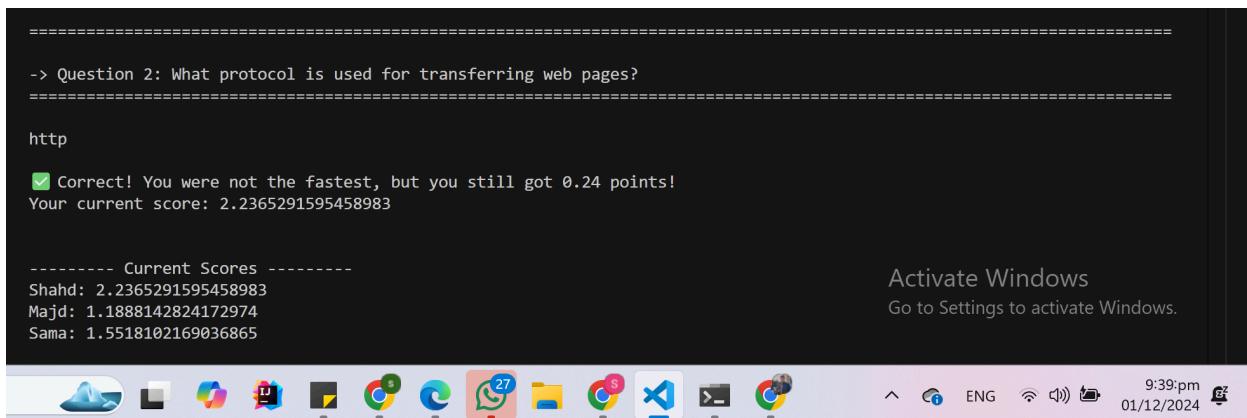
figure 91: Game Start and First Round Progression - Task 3

### 3.2. Minute Response Timer:

Each client is given 60 seconds to submit their response. During this time, the server listens for answers from all connected players. Players who respond faster and correctly are scored higher, while those who answer incorrectly or fail to respond within the time limit receive no points.

### 3.3. Handling Responses:

**Correct Answers:** Players submitting correct answers within the time limit earn points based on their response time, with faster responses receiving higher scores.



```
=====
-> Question 2: What protocol is used for transferring web pages?
=====

http

✓ Correct! You were not the fastest, but you still got 0.24 points!
Your current score: 2.2365291595458983

----- Current Scores -----
Shahd: 2.2365291595458983
Majd: 1.1888142824172974
Sama: 1.5518102169036865

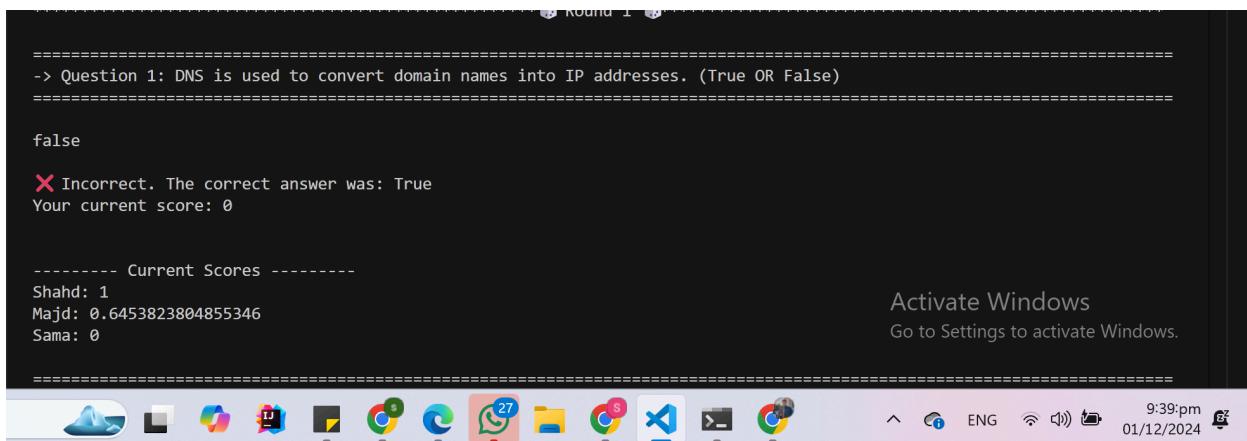
-----
```

Activate Windows  
Go to Settings to activate Windows.

9:39:pm 01/12/2024

figure 92: handling correct answers - Task 3

**Wrong Answers:** Players who submit incorrect answers are informed of the mistake and provided with the correct answer.



```
=====
-> Question 1: DNS is used to convert domain names into IP addresses. (True OR False)
=====

false

✗ Incorrect. The correct answer was: True
Your current score: 0

----- Current Scores -----
Shahd: 1
Majd: 0.6453823804855346
Sama: 0

-----
```

Activate Windows  
Go to Settings to activate Windows.

9:39:pm 01/12/2024

figure 93: Handling wrong answers - Task3

**Timeouts:** Players who fail to respond within the time limit are notified that the time is up and provided with the correct answer.

```
=====
-> Question 3: Will we get the full mark in this project?
=====

⏰ Time's up! 🕒 The correct answer was: Yes, of course! (Inshallah)

----- Current Scores -----
Shahd: 1
Majd: 1.1888142824172974
Sama: 1
```

Activate Windows  
Go to Settings to activate Windows.



figure 94: Handling Time Up Case - Task 3

### **3.4. End of Round Announcement And Transition to the Next Round:**

After all questions for the current round are completed, the server evaluates and announces the winner of the round. If there is a tie, it highlights the top-scoring players. The server waits for 20 seconds before starting the next round. This short break ensures players have time to prepare for the upcoming questions. During this time, the server broadcasts a message indicating that the next round will begin shortly.

```
🎉 End of Round 1! 🎉
The winner of Round 1 is 'Majd' with 1.1888142824172974 points!
The Next Round Will Start Within 20 Seconds..🔥
```

Activate Windows  
Go to Settings to activate Windows.

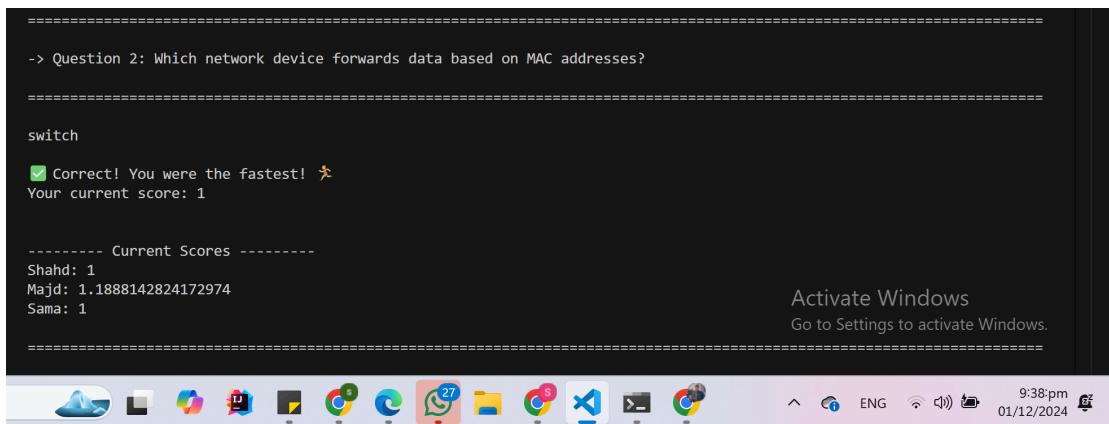


figure 95:End of Round Announcement - Task 3

### **3.5. Scoring Based on Response Speed**

- In this scenario, two clients answer the same question correctly but at different times. The score for each client is determined based on their speed of response.

In this screenshot, we see Client 1's response to the trivia question. Client 1 answered correctly and was the first to respond among all active participants. Since Client 1 was the fastest to submit a correct answer, they receive the full 1 point for the round, and feedback is sent showing the answer was correct and they were the fastest, as shown in figure 96.



```
=====  
-> Question 2: Which network device forwards data based on MAC addresses?  
=====  
switch  
 Correct! You were the fastest! 🏆  
Your current score: 1  
  
----- Current Scores -----  
Shahd: 1  
Majd: 1.1888142824172974  
Sama: 1  
=====
```

The screenshot shows a terminal window on a Windows operating system. The terminal output displays a trivia question about network devices and the user's correct answer. It also shows the current scores for three participants: Shahd, Majd, and Sama. The Majd account has a very high score (1.1888142824172974). The taskbar at the bottom of the screen includes icons for various applications like File Explorer, Google Chrome, and Microsoft Edge, along with system status indicators for battery, signal, and volume. A watermark for "Activate Windows" is visible in the top right corner of the desktop.

figure 96: Client Answered Question First Case - Task 3

In this second screenshot, Client 2 answers correctly but later than Client 1, who was the fastest. Due to the delay, Client 2's score is reduced by 10% for each second after the fastest response. For example, if Client 2 took 2 seconds longer, their score would be reduced by 20%. Feedback is sent showing the correct answer, but with a reduced score as shown in figure 97.

```

=====
-> Question 2: Which network device forwards data based on MAC addresses?
=====

switch

 Correct! You were not the fastest, but you still got 0.54 points!
Your current score: 1.1888142824172974

----- Current Scores -----
Shahd: 1
Majd: 1.1888142824172974
Sama: 1

Activate Windows
Go to Settings to activate Windows.

Ln 318, Col 42 (52 selected) Spaces: 4 UTF-8 CRLF () Python 3.12.4 64-bit ⓘ Go Live 🔍 ⓘ Prettier
0 0 ⌂ 0 28 ⌂ 0
Clouds File Explorer This PC File Types Task View Start Taskbar 9:38pm 01/12/2024

```

figure 97: Client Answered Correct But Not The First Case - Task 3

### 3.5. Handling Multiple Submissions: Only the First Answer Counts

In this screenshot, Client 3 submitted multiple answers for the same trivia question. The first answer they submitted was "oo" which was incorrect, so it was mClient 3 submitted the correct answer "HTTP," but this second answer was not considered for scoring. Only the first answer is used to determine the score, so despite the correct second submission, the score remains unchanged based on the first incorrect response. Feedback is sent to the client, showing their first answer as incorrect and updating the score accordingly.

marked as wrong. The system then informed Client 3 of the correct answer. Later,

```

=====
-> Question 2: What protocol is used for transferring web pages?
=====

oo
http

✗ Incorrect. The correct answer was: HTTP
Your current score: 1.1888142824172974

----- Current Scores -----
Shahd: 2.2365291595458983
Majd: 1.1888142824172974
Sama: 1.5518102169036865

Activate Windows
Go to Settings to activate Windows.

Ln 318, Col 42 (52 selected) Spaces: 4 UTF-8 CRLF () Python 3.12.4 64-bit ⓘ Go Live 🔍 ⓘ Prettier
0 0 ⌂ 0 28 ⌂ 0
Clouds File Explorer This PC File Types Task View Start Taskbar 9:37pm 01/12/2024

```

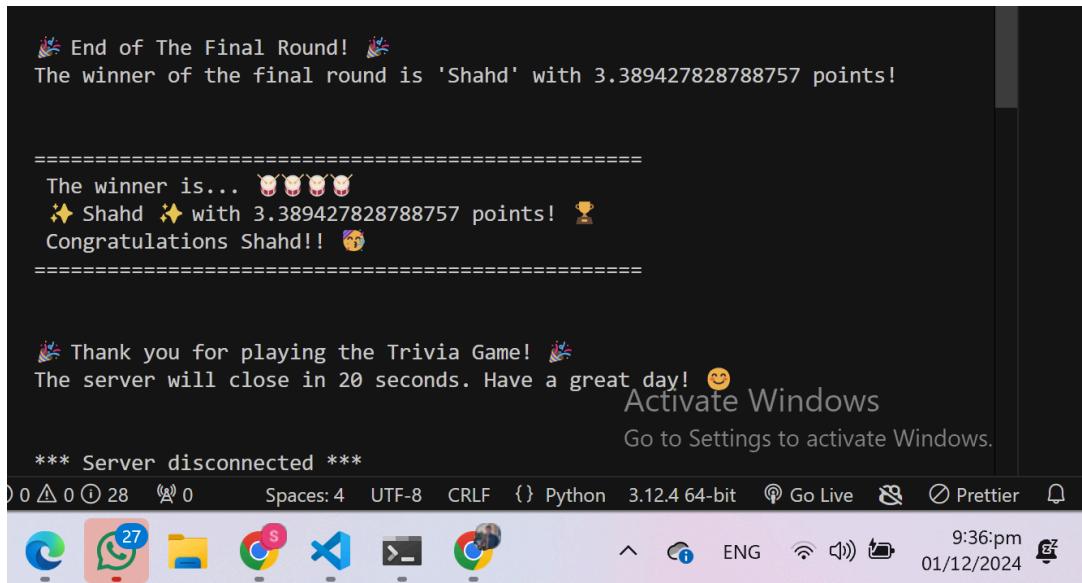
figure 98: Handling Multiple Submissions - Task 3

## 4. Final Round: Announcing the Winner and Handling No Winner Case

At the end of the final round, the player with the highest total score is announced as the winner. In case of a tie, multiple winners are declared. If no player scores points, there is no winner. After displaying the results, the server sends a final message and waits 20 seconds before closing the connection, ensuring a smooth game conclusion.

### 4.1. Case 1: A Single Winner is Announced

In this case, at the end of the final round, we calculate the scores from all the rounds to determine the player with the maximum total score. If one player has the highest score, they are declared the winner



```
🎉 End of The Final Round! 🎉
The winner of the final round is 'Shahd' with 3.389427828788757 points!

=====
The winner is... 🎉🎉🎉🎉
◆ Shahd ◆ with 3.389427828788757 points! 🏆
Congratulations Shahd!! 🎉

=====

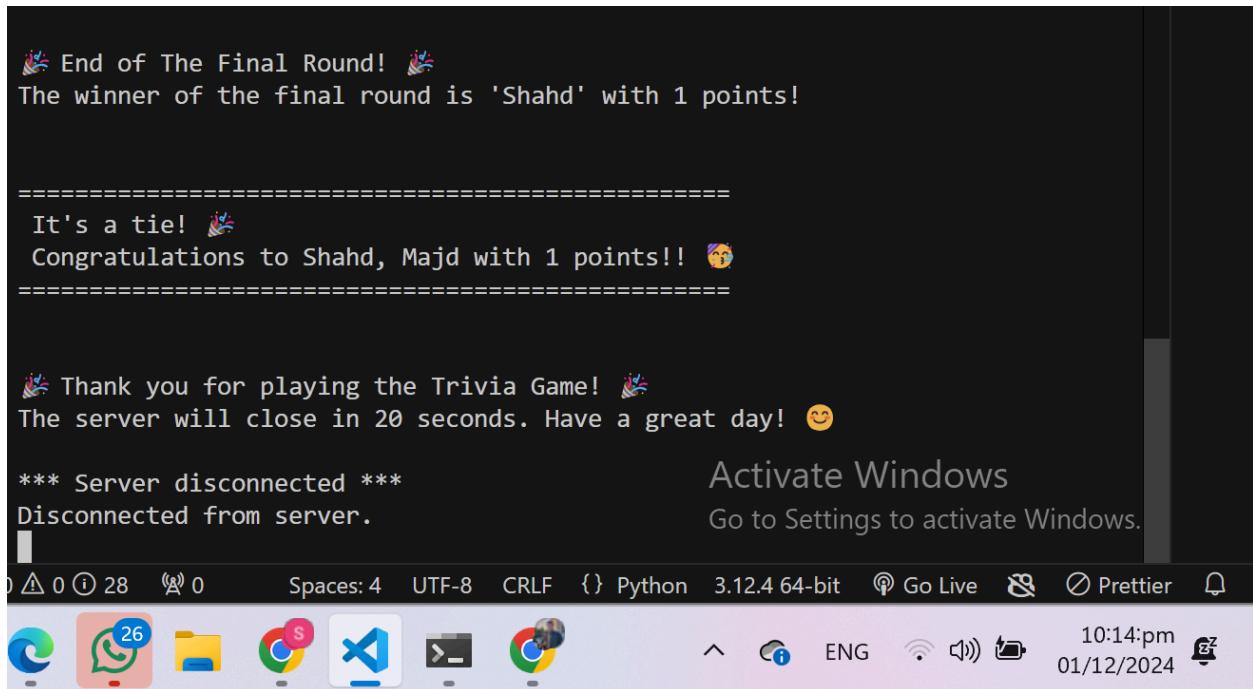
🎉 Thank you for playing the Trivia Game! 🎉
The server will close in 20 seconds. Have a great day! 😊
Activate Windows
Go to Settings to activate Windows.

*** Server disconnected ***
0 0 0 28 0 0 Spaces: 4 UTF-8 CRLF {} Python 3.12.4 64-bit ⚡ Go Live 🔍 Prettier ⌂
🕒 9:36:pm 01/12/2024 🔍
```

figure 99: A Single Winner is Announced - Task 3

#### 4.2. Case 2: A Tie Between Two or More Players

In this scenario, two or more players tie with the same maximum score at the end of the final round. In such a case, the tie is recognized, and feedback is sent indicating the tied players. The final message may state something like, "We have a tie between Client 1 and Client 2, both scoring the highest points."



```
🎉 End of The Final Round! 🎉
The winner of the final round is 'Shahd' with 1 points!

=====
It's a tie! 🎉
Congratulations to Shahd, Majd with 1 points!! 😊
=====

🎉 Thank you for playing the Trivia Game! 🎉
The server will close in 20 seconds. Have a great day! 😊

*** Server disconnected ***
Disconnected from server.
```

The screenshot shows a terminal window with the following text output:

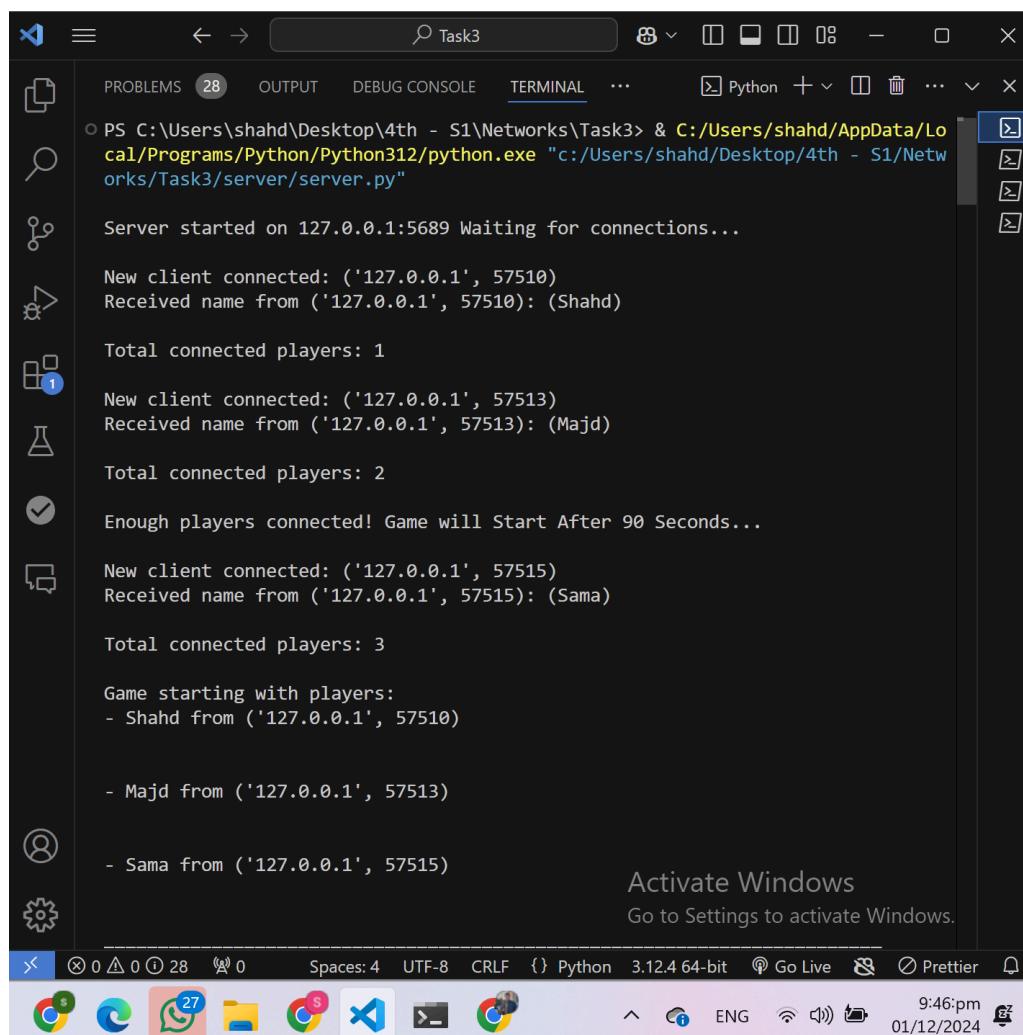
- 🎉 End of The Final Round! 🎉
- The winner of the final round is 'Shahd' with 1 points!
- =====
- It's a tie! 🎉
- Congratulations to Shahd, Majd with 1 points!! 😊
- =====
- 🎉 Thank you for playing the Trivia Game! 🎉
- The server will close in 20 seconds. Have a great day! 😊
- \*\*\* Server disconnected \*\*\*
- Disconnected from server.

Below the terminal window, the Windows taskbar is visible, showing various icons and system status information.

figure 100: A Tie Between Two or More Players - Task 3

## Server Terminal Output

In this screenshot, we see a message on the server side indicating that enough clients have joined the game and that a new round will begin. The server will print a message confirming that it has started and is actively listening on a specific port. Additionally, for each new client connection, a message will display the client's IP address and port number to confirm their successful connection to the game.



The screenshot shows a terminal window titled "Task3" with the following log output:

```
PS C:\Users\shahd\Desktop\4th - S1\Networks\Task3 & C:/Users/shahd/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/shahd/Desktop/4th - S1/Networks/Task3/server/server.py"
Server started on 127.0.0.1:5689 Waiting for connections...
New client connected: ('127.0.0.1', 57510)
Received name from ('127.0.0.1', 57510): (Shahd)
Total connected players: 1
New client connected: ('127.0.0.1', 57513)
Received name from ('127.0.0.1', 57513): (Majd)
Total connected players: 2
Enough players connected! Game will Start After 90 Seconds...
New client connected: ('127.0.0.1', 57515)
Received name from ('127.0.0.1', 57515): (Sama)
Total connected players: 3
Game starting with players:
- Shahd from ('127.0.0.1', 57510)

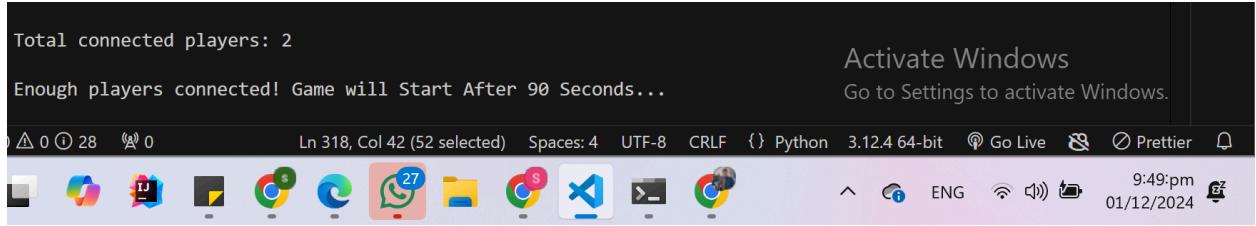
- Majd from ('127.0.0.1', 57513)

- Sama from ('127.0.0.1', 57515)
```

The terminal window also displays a "Activate Windows" message at the bottom right: "Activate Windows Go to Settings to activate Windows."

figure 101: server actively listening on a 5689 port. & confirming each new client connection - Task 3

In this screenshot, we see a message on the server side indicating that enough clients have joined the game and a new round will begin.

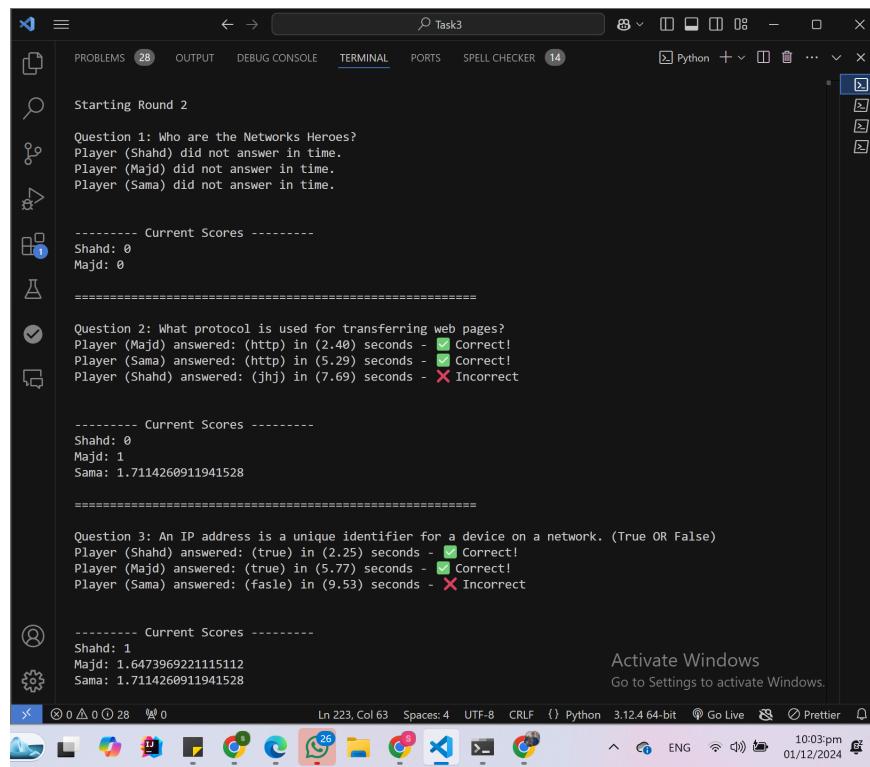


```
Total connected players: 2
Enough players connected! Game will Start After 90 Seconds...
Activate Windows
Go to Settings to activate Windows.

0 △ 0 ⚡ 28 Ln 318, Col 42 (52 selected) Spaces: 4 UTF-8 CRLF {} Python 3.12.4 64-bit ⚡ Go Live ⚡ Prettier ⚡
File Explorer Home Recent Taskbar 9:49:pm 01/12/2024
```

figure 102: message on the server side that enough clients joined - Task 3

In this screenshot, we see the server displaying the question number and text for each question. As answers are received from the clients, the server also displays each client's response along with their identifier (IP and port), and whether the answer is correct or incorrect. This provides transparency and allows both the players and server to track the progress of the game in real-time.



```
Starting Round 2

Question 1: Who are the Networks Heroes?
Player (Shahd) did not answer in time.
Player (Majd) did not answer in time.
Player (Sama) did not answer in time.

----- Current Scores -----
Shahd: 0
Majd: 0
-----

Question 2: What protocol is used for transferring web pages?
Player (Majd) answered: (http) in (2.40) seconds - ✅ Correct!
Player (Sama) answered: (http) in (5.29) seconds - ✅ Correct!
Player (Shahd) answered: (jhj) in (7.69) seconds - ❌ Incorrect

----- Current Scores -----
Shahd: 0
Majd: 1
Sama: 1.7114260911941528
-----

Question 3: An IP address is a unique identifier for a device on a network. (True OR False)
Player (Shahd) answered: (true) in (2.25) seconds - ✅ Correct!
Player (Majd) answered: (true) in (5.77) seconds - ✅ Correct!
Player (Sama) answered: (fasle) in (9.53) seconds - ❌ Incorrect

----- Current Scores -----
Shahd: 1
Majd: 1.6473969221115112
Sama: 1.7114260911941528

Activate Windows
Go to Settings to activate Windows.

0 △ 0 ⚡ 28 Ln 223, Col 63 Spaces: 4 UTF-8 CRLF {} Python 3.12.4 64-bit ⚡ Go Live ⚡ Prettier ⚡
File Explorer Home Recent Taskbar 10:03:pm 01/12/2024
```

figure 103: question displaying in server side and clients answers with answer result - Task 3

**In this screenshot, we see a message on the server side** at the end of a round and announcing the winner of the round.

```
=====
Question 3: Will we get the full mark in this project?
Player (Sama) did not answer in time.
Player (Majd) answered: (no) in (4.51) seconds
Player (Shahd) answered: (maybe) in (10.97) seconds

----- Current Scores -----
Shahd: 1
Majd: 1.1888142824172974
Sama: 1

=====
Round 1 winner: Majd with 1.1888142824172974 points
```

Activate Windows  
Go to Settings to activate Windows.



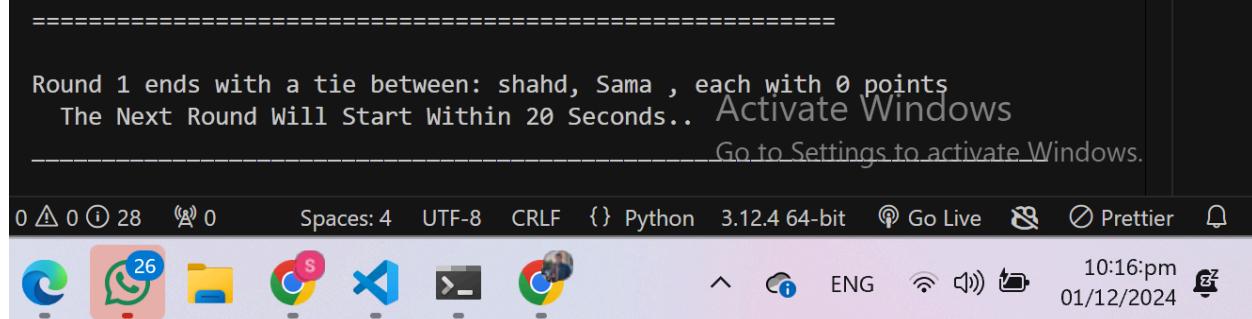
The screenshot shows a terminal window with a black background. It displays the results of a round, including player names, their answers, and the time taken. It then shows the current scores for Shahd, Majd, and Sama. Finally, it announces the round 1 winner, Majd, with a score of 1.1888142824172974 points. The terminal window has a light blue header bar with various icons and text like 'Ln 318, Col 42 (52 selected)', 'Spaces: 4', 'UTF-8', 'CRLF', 'Python', '3.12.4 64-bit', 'Go Live', 'Prettier', and a date/time stamp '01/12/2024 9:48:pm'. A watermark for 'Activate Windows' is visible in the top right corner.

figure 104: message on the server side at the end of a round & announcing round winner - Task 3

In this screenshot i Indicate the countdown in server side (20 sec) before starting the next round.

```
=====
Round 1 ends with a tie between: shahd, Sama , each with 0 points
The Next Round Will Start Within 20 Seconds..
```

Activate Windows  
Go to Settings to activate Windows.



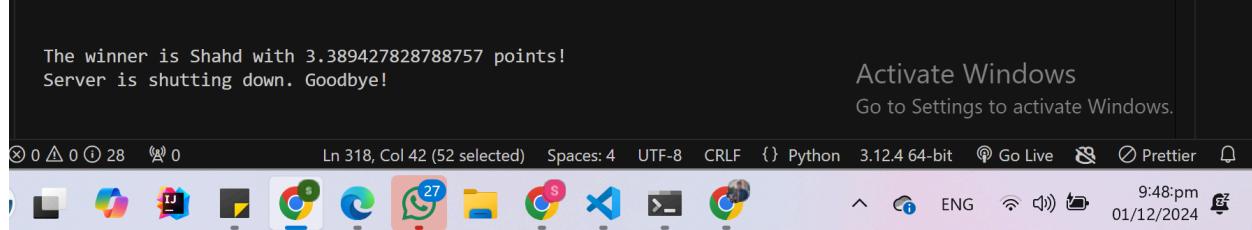
The screenshot shows a terminal window with a black background. It announces a tie between Shahd and Sama with 0 points. It then indicates that the next round will start within 20 seconds. The terminal window has a light blue header bar with various icons and text like 'Ln 318, Col 42 (52 selected)', 'Spaces: 4', 'UTF-8', 'CRLF', 'Python', '3.12.4 64-bit', 'Go Live', 'Prettier', and a date/time stamp '01/12/2024 10:16:pm'. A watermark for 'Activate Windows' is visible in the top right corner.

figure 105: message in server side countdown in server side before starting the next round - Task 3

**In this screenshot, we see a message on the server side** to handle if the server is stopped, by displaying a message confirming the shutdown.

```
The winner is Shahd with 3.389427828788757 points!
Server is shutting down. Goodbye!
```

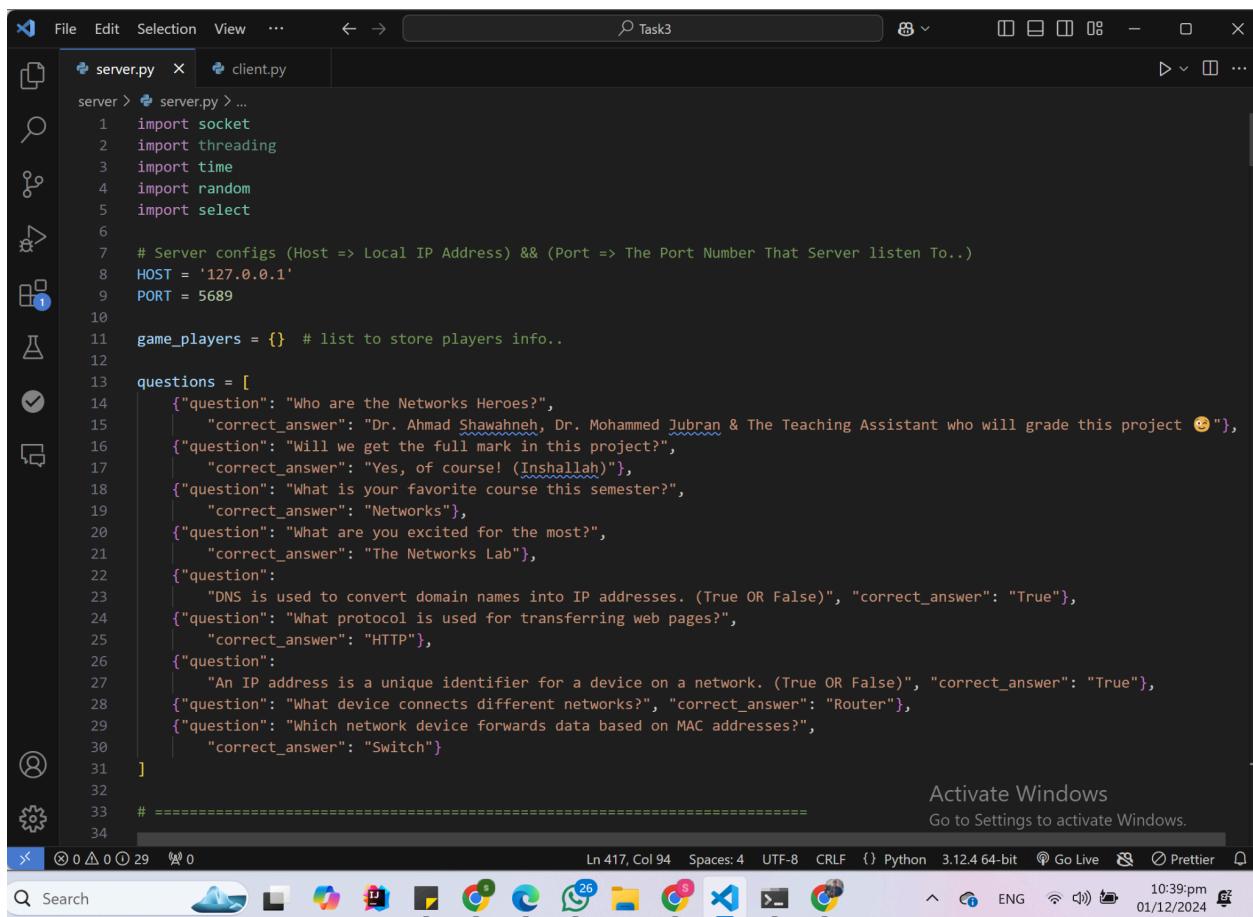
Activate Windows  
Go to Settings to activate Windows.



The screenshot shows a terminal window with a black background. It confirms that Shahd is the winner with 3.389427828788757 points and that the server is shutting down. The terminal window has a light blue header bar with various icons and text like 'Ln 318, Col 42 (52 selected)', 'Spaces: 4', 'UTF-8', 'CRLF', 'Python', '3.12.4 64-bit', 'Go Live', 'Prettier', and a date/time stamp '01/12/2024 9:48:pm'. A watermark for 'Activate Windows' is visible in the top right corner.

figure 106: server shutting down message - Task 3

## Task 3 - Server Side Code



A screenshot of a code editor window titled "Task3". The editor shows two files: "server.py" and "client.py". The "server.py" file contains Python code for a network server. It imports socket, threading, time, random, and select modules. It defines constants HOST = '127.0.0.1' and PORT = 5689. A list "game\_players" is initialized to store player info. A list "questions" contains 30 questions with their correct answers. The code ends with a comment "# =====". The status bar at the bottom shows the current line (Ln 417), column (Col 94), spaces (Spaces: 4), encoding (UTF-8), and file type (Python 3.12.4 64-bit). It also displays "Activate Windows Go to Settings to activate Windows." and the date/time (10:39pm 01/12/2024).

```
server.py > server.py > ...
1 import socket
2 import threading
3 import time
4 import random
5 import select
6
7 # Server configs (Host => Local IP Address) && (Port => The Port Number That Server listen To...)
8 HOST = '127.0.0.1'
9 PORT = 5689
10
11 game_players = {} # list to store players info..
12
13 questions = [
14     {"question": "Who are the Networks Heroes?", "correct_answer": "Dr. Ahmad Shawahneh, Dr. Mohammed Jubran & The Teaching Assistant who will grade this project 😊"}, 
15     {"question": "Will we get the full mark in this project?", "correct_answer": "Yes, of course! (Inshallah)"}, 
16     {"question": "What is your favorite course this semester?", "correct_answer": "Networks"}, 
17     {"question": "What are you excited for the most?", "correct_answer": "The Networks Lab"}, 
18     {"question": "DNS is used to convert domain names into IP addresses. (True OR False)", "correct_answer": "True"}, 
19     {"question": "What protocol is used for transferring web pages?", "correct_answer": "HTTP"}, 
20     {"question": "An IP address is a unique identifier for a device on a network. (True OR False)", "correct_answer": "True"}, 
21     {"question": "What device connects different networks?", "correct_answer": "Router"}, 
22     {"question": "Which network device forwards data based on MAC addresses?", "correct_answer": "Switch"}]
23
24 # =====
25
26
27
28
29
30
31 ]
32
33 # =====
34
```

figure 107: server configs & list of question definition code -Task 3

```
def broadcast_message(message): # msg to all connected active players
    for player in game_players.values():
        try:
            server_socket.sendto(message.encode(), player['address'])
        except Exception as e:
            print(f"**ERROR** -> Error sending message to {player['name']}: {e}")
            del game_players[player['address']]
```

figure 108: method to broadcast messages to clients - Task 3

```
# =====
game_started = False # flag to check if the game started or not
# to track all asked questions to make sure no questions are repeated..
asked_questions_set = set()

def start_game():
    global game_started, asked_questions_set
    if game_started:
        return # do not restart the game if it has already begun

    game_started = True
    print("Enough players connected! Game will Start After 90 Seconds...\n")

    # delay 90 seconds before starting the game
    time.sleep(12)

    print("Game starting with players:")
    for player in game_players.values():
        print(
            f"- {player['name']} from {player['connection'].getpeername()}\n\n")

    totalRounds = 3

    for currRound in range(totalRounds):
        initialize_game_round(currRound)

        for question_number in range(3): # 3 questions per round
            ask_question(question_number, currRound)

        # evaluate the current round winner
        evaluate_round_winner(currRound)

        # announce the final winner in the final round
        if currRound == totalRounds - 1:
            end_game(currRound)
# =====
```

Activate Windows  
Go to Settings to activate Windows.



figure 109: method to start game and send questions to clients - Task 3

```

# =====

def initialize_game_round(currRound):
    # making sure there are at least 2 active clients (players) to start the round
    while len(game_players) < 2:
        print("Not enough players to start the round. Waiting for more players...")
        broadcast_message(
            f"Not enough players to start Round {currRound + 1}. Waiting for more players to join...\n"
        )
        time.sleep(3) # check new connected players every 10 seconds
    print("\nStarting Round {currRound + 1}\n")
    broadcast_message(
        f"\n{'*' * 53} Round {currRound + 1} {'*' * 53}\n"
    )

# =====

```

Activate Windows  
Go to Settings to activate Windows.

① 29 ④ 0 Ln 73, Col 53 Spaces: 4 UTF-8 CRLF () Python 3.12.4 64-bit ⚡ Go Live 🌐 Prettier 🔍 10:40pm 01/12/2024

figure 110: method to initialize the game set up -Task 3

```

server.py x client.py
server > server.py > initialize_game_round
102 def ask_question(question_number, currRound):
103     available_questions = [q for q in questions if q['question'] not in asked_questions_set]
104     if not available_questions:
105         return
106
107     question_info = random.choice(available_questions)
108     asked_questions_set.add(question_info['question'])
109     question = question_info['question']
110     correct_answer = question_info['correct_answer']
111
112     broadcast_message(f"Question {question_number + 1}: {question}")
113     print(f"Question {question_number + 1}: {question}")
114
115     player_answers = {}
116     question_start_time = time.time()
117
118     # Wait for player answers within 15 seconds
119     while time.time() - question_start_time < 15:
120         readable_sockets, _, _ = select.select([player['address'][0] for player in game_players.values()], [], [], 0.1)
121         for client_socket in readable_sockets:
122             try:
123                 response, _ = server_socket.recvfrom(1024)
124                 player_address = next(player_address for player_address, player in game_players.items() if player['address'][0] == client_socket)
125                 if player_address not in player_answers:
126                     player_answers[player_address] = {
127                         "response": response.decode().strip(),
128                         "time": time.time() - question_start_time
129                     }
130             except Exception as e:
131                 print(f"Receiving data error: {e}")
132             continue
133
134     # Notify players who didn't answer in time
135     for player_address, player_data in game_players.items():
136         if player_address not in player_answers or player_answers[player_address]['response'] is None:
137             print(f"Player {game_players[player_address]['name']} did not answer in time.")
138             player_data["connection"].sendto(f"Time's up! The correct answer was: {correct_answer}\n".encode(), player_data['address'])
139
140     checkAnswers_UpdateScores(player_answers, correct_answer)
141     time.sleep(5)
142
143     player_answers = {} # reset players answers list to the next auction

```

Activate Windows  
Go to Settings to activate Windows.

① 0 ③ 0 ④ 29 ⑤ 0 Ln 92, Col 70 Spaces: 4 UTF-8 CRLF () Python 3.12.4 64-bit ⚡ Go Live 🌐 Prettier 🔍 10:40pm 01/12/2024

figure 111: method to ask random unique questions to client - Task 3

```

147
148     def evaluate_round_winner(currRound):
149         round_winner = max(game_players.items(), key=lambda x: x[1]['score'])
150         round_winner_name = round_winner[1]['name']
151         round_winner_score = round_winner[1]['score']
152
153         if currRound + 1 < 3: # If it's not the last round (1st and 2nd rounds)
154             round_scores = [player['score']
155                             # all player scores in one round
156                             for player in game_players.values()]
157             roundMaxScore = max(round_scores)
158
159             # find players (one or more) with the max score in the round
160             roundTop_Players = [
161                 roundplayer for roundplayer in game_players if game_players[roundplayer]['score'] == roundMaxScore]
162
163             winnersNames_Round = []
164
165             if len(roundTop_Players) > 1: # tie case
166
167                 winnersNames_Round = ", ".join(
168                     game_players[roundplayer]['name'] for roundplayer in roundTop_Players)
169                 round_winner_message = f"\nEnd of Round {currRound + 1}! 🎉\nRound {currRound + 1} ends with a thrilling tie! The top scorers are: {winnersNames_Round}, each with {round_winner_score} points\n"
170
171                 print(f"Round {currRound + 1} ends with a tie between: {winnersNames_Round} , each with {round_winner_score} points")
172
173             else:
174                 print(f"Round {currRound + 1} winner: {round_winner_name} with {round_winner_score} points")
175
176                 round_winner_message = f"\nEnd of Round {currRound + 1}! 🎉\nThe winner of Round {currRound + 1} is '{round_winner_name}' with {round_winner_score} points!\n"
177
178                 broadcast_message(round_winner_message)
179                 broadcast_message("The Next Round Will Start Within 20 Seconds... 🚨")
180                 print(" The Next Round Will Start Within 20 Seconds..")
181                 time.sleep(2)
182
183             else: # final winner
184                 broadcast_message(f"\nEnd of The Final Round! 🎉\nThe winner of the final round is '{round_winner_name}' with {round_winner_score} points!\n")
185
186
187
188

```

Activate Windows  
Go to Settings to activate Windows.

figure 112: method to evaluate round winner - Task 3

```

190
191     # this method to end game & close server after rounds end and announce the final winner..
192
193
194     def end_game(currRound):
195         RevealFinalWinner() # Reveal the final winner after all rounds end
196
197         # Notify players the server will close soon
198         closing_message = (
199             "\n\u26a1 Thank you for playing the Trivia Game! \ud83d\udcbb\n"
200             "The server will close in 20 seconds. Have a great day! \ud83d\udcbb\n"
201         )
202         broadcast_message(closing_message)
203
204         # Delay for 20 seconds before shutting down the server
205         time.sleep(20)
206
207         # Close all player connections
208         for player in game_players.values():
209             try:
210                 player['connection'].close()
211             except Exception as e:
212                 print(f"Error closing connection for {player['name']}: {e}")
213
214         print("Server is shutting down. Goodbye!")
215         exit(0) # Exit the server
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258

```

Activate Windows  
Go to Settings to activate Windows.

figure 113: end game and close server method - Task 3

```

219     # this method to handle players connections
220
221     def handle_client(message, player_address):
222         print(f"New client connected: {player_address}")
223
224         # Decode the message (the name of the player)
225         name = message.decode().strip()
226
227         # Add player to the game_players dictionary
228         game_players[player_address] = {"name": name, "score": 0}
229
230         # Send a welcome message to the player
231         welcome_message = f"\nWelcome, {name}! Waiting for enough players to start the game...\n"
232         server_socket.sendto(welcome_message.encode(), player_address)
233
234         print(f"Received name from {player_address}: {name}\n")
235         print(f"Total connected players: {len(game_players)}\n")
236
237         # Send game rules to the player
238         rules_message = """
239             ***** Trivia Game Rules *****
240             1. The game consists of 3 rounds, with 3 questions in each round.
241             2. You will have only 60 seconds to answer each question.
242             3. The faster you answer correctly, the higher your score 😊.
243             4. Players who fail to answer within the time limit will be marked as "Timeout".
244             5. The player with the highest total score at the end of all rounds wins!
245             ****
246             Are you ready? 🎉
247             The game will start once we have at least 2 players. Please wait...
248             """
249
250         server_socket.sendto(rules_message.encode(), player_address)
251
252         # Check if at least 2 players are connected
253         if len(game_players) >= 2 and not game_started:
254             start_game()
255         elif len(game_players) < 2:
256             server_socket.sendto("Waiting for at least 2 clients to join the game...\n".encode(), player_address)
257
258

```

Activate Windows  
Go to Settings to activate Windows.

figure 114: welcoming and handle client method - Task 3

```
# this method used to send the same question to all active players in the game at same time..
def send_to_all_players(message):
    for player in game_players.values():
        player['connection'].send(message.encode())
```

figure 115: method to send data to active clients at same time - Task 3

```

# => This method checks player answers (game participants), sorts them by response time (fastest first),
# => updates scores based on correctness and speed & sends results to players.
# => Faster correct answers get higher scores, and players who answered wrong or didn't respond in time get ZERO
# => For correct answers, the score is calculated based on how quickly the player responded.
# => (score reduces as time increases) -> reducing by 10% per second after the fastest time..

def checkAnswers_UpdateScores(SubmittedAnswers, CorrectAnswer):
    # ffirst, sort players by response time - only correct answers included
    sorted_playerAnswers = sorted(
        SubmittedAnswers.items(),
        key=lambda x: x[1]['time'] if x[1]['response'] == CorrectAnswer else float(
            'inf')
    )

    # Track the time of the fastest player
    fastest_time = None
    if sorted_playerAnswers:
        fastest_time = sorted_playerAnswers[0][1]['time']

    # check each player's submitted answer
    for index, (playerAddress, response_data) in enumerate(sorted_playerAnswers):
        response = response_data['response']
        playerName = game_players[playerAddress]['name']

        if response is None:  # If there is no answer submitted -> show the "Time's up!" message
            result = ""
            game_players[playerAddress]['connection'].send(
                f"{result}\n".encode())
            game_players[playerAddress]['connection'].send(
                f"Your current score: {game_players[playerAddress]['score']}\n".encode())
            print(f"Updated score for {game_players[playerAddress]['name']}: {game_players[playerAddress]['score']}")
            continue

        # If the answer is correct -> i calculated the score based on answer submitted speed
        if response.lower() == CorrectAnswer.lower():
            time_taken = response_data['time']

            # the first player answers correctly gets full point (1 point)

```

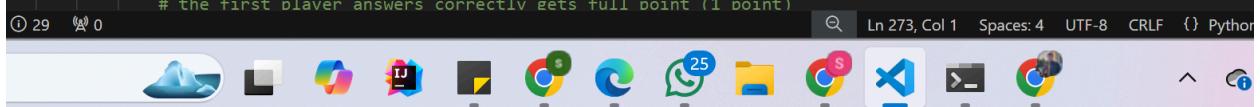


figure 116: method to check answers and update scores - Task 3

File Edit Selection ... ← → ⌂ Task3

server.py X client.py

```

server > server.py > checkAnswers_UpdateScores
275 def checkAnswers_UpdateScores(SubmittedAnswers, CorrectAnswer):
304     # If the answer is correct -> i calculated the score based on answer submitted speed
305     if response.lower() == CorrectAnswer.lower():
306         time_taken = response_data['time']
307
308         # the first player answers correctly gets full point (1 point)
309         if time_taken == fastest_time:
310             score = 1
311             result = "\n✓ Correct! You were the fastest! 🏃"
312         else: # the 2nd player answers correctly
313
314             # calculate the score based on time difference..
315             time_diff = max(0, time_taken - fastest_time)
316
317             # decrement by 10% per second after fastest time
318             score = max(0, 1 - time_diff * 0.1)
319             result = f"\n✓ Correct! You were not the fastest, but you still got {score:.2f} points!"
320
321     else: # wrong answer
322
323         score = 0
324         result = f"\n✗ Incorrect. The correct answer was: {CorrectAnswer}"
325
326         # update player's score
327         game_players[playerAddress]['score'] += score
328
329         # send result message to clients
330         game_players[playerAddress]['connection'].send(result.encode())
331         game_players[playerAddress]['connection'].send(
332             f"Your current score: {game_players[playerAddress]['score']} \n".encode())
333
334         # Check if the answer is correct or not
335         if response.lower() == CorrectAnswer.lower():
336             answer_status = "✓ Correct!"
337         else:
338             answer_status = "✗ Incorrect"
339
340         # Log player answers, response times, and correctness
341         print(f"Player ({game_players[playerAddress]['name']}) answered: ({response}) in ({response_data['time']:.2f}) seconds - {answer_status}")
342
343
344     showClientsCurrScore() # show the current scores to all game players (clients)

```

0 Δ 0 ⓘ 29 ⌂ 0 Q Ln 332, Col 70 Spaces: 4 UTF-8

search

figure 117: method to check answers , update scores & send feedbacks - Task 3

```

349
350
351     def showClientsCurrScore():
352         score_message = "\n\n----- Current Scores -----"
353         for player in game_players.values():
354             score_message += f"{player['name']}: {player['score']}\n"
355         broadcast_message(score_message)
356         print(score_message)
357         print("======\n")
358     # ======

```

figure 118: show current score method - Task 3

The screenshot shows a code editor window with two tabs: 'server.py' and 'client.py'. The 'server.py' tab is active, displaying the following code:

```

File Edit Selection ... ← → ⌂ Task3
server.py x client.py
server > server.py > RevealFinalWinner
361     def RevealFinalWinner(): # announcing the final winner in the game
362
363         max_score = max(game_players[p]['score']
364                         for p in game_players) # Find the highest score
365
366         # find all game players winners
367         winners = [p for p in game_players if game_players[p]
368                         ['score'] == max_score]
369
370         if len(winners) == 1: # if only one winner
371             winner_name = game_players[winners[0]]['name']
372             winner_score = game_players[winners[0]]['score']
373             print(f"\nThe winner is {winner_name} with {winner_score} points!")
374             broadcast_message(
375                 f"\n=*50*\n"
376                 f" The winner is... 🎉🎉🎉\n"
377                 f" ✨ {winner_name} ✨ with {winner_score} points! 🎉\n"
378                 f" Congratulations {winner_name}!! 🎉\n"
379                 f"=*50*\n"
380             )
381         elif len(winners) > 1: # more than one winner
382             tiedPlayers = ", ".join(game_players[p]['name'] for p in winners)
383             winner_score = game_players[winners[0]]['score']
384
385             print(f"\nIt's a tie! The players with the highest score are: {tiedPlayers} with {winner_score} points!")
386
387             broadcast_message(
388                 f"\n=*50*\n"
389                 f" It's a tie! 🎉\n"
390                 f" Congratulations to {tiedPlayers} with {winner_score} points!! 🎉\n"
391                 f"=*50*\n"
392             )
393         else: # No winner case ( all players timed out OR answered wrong)
394             print("\nNo winner! Everyone timed out or answered incorrectly.")
395             broadcast_message(
396                 f"\n=*50*\n"
397                 f" No winner! 🎉\nEveryone timed out or answered incorrectly.\n"
398                 f"=*50*\n"
399             )
400
401     )

```

The code implements a 'RevealFinalWinner' function that finds the maximum score in the game\_players dictionary. If there is one winner, it prints their name and score, sends a broadcast message with celebratory emojis, and prints a summary message. If there is a tie, it prints the names of all tied players and sends a broadcast message congratulating them. If no one won (all timed out or answered wrong), it prints a message indicating no winner.

figure 119: calculate final score and announce winner method - Task 3

```
403     # this is the main methods that runs the server and accpets the players (clients) connections
404     # UDP server setup
405     server_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
406     server_socket.bind((HOST, PORT))
407
408     # Receive messages from clients
409     def receive_messages():
410         while True:
411             try:
412                 message, player_address = server_socket.recvfrom(1024)
413                 handle_client(message, player_address)
414             except Exception as e:
415                 print(f"Error receiving message: {e}")
416
417     # Main execution
418     asked_questions_set = set()
419     game_started = False
420     print(f"Server is running on {HOST}:{PORT}")
421     receive_messages()
```

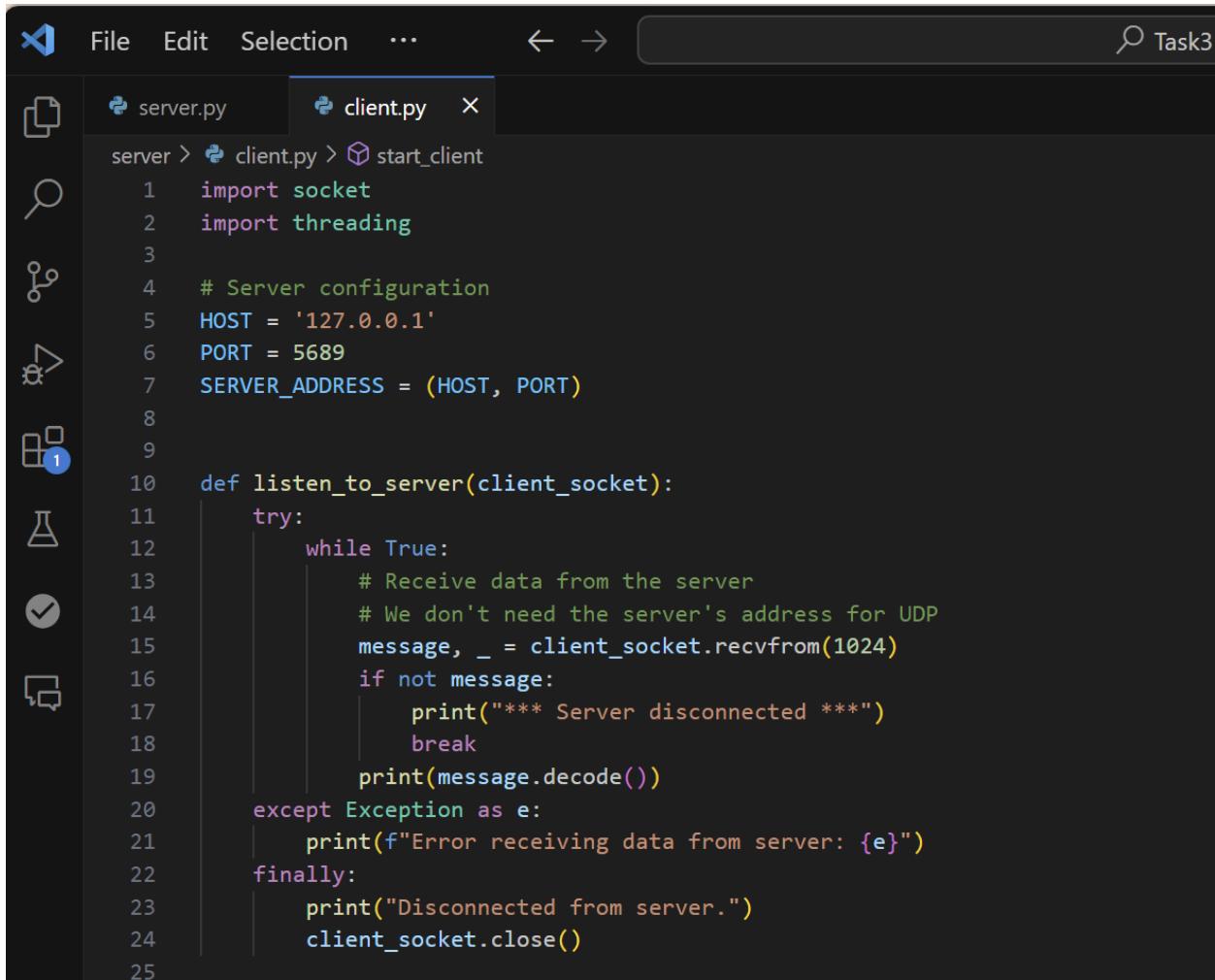
Activate Windows  
Go to Settings to activate Windows.

X 0 △ 0 ① 29 ⌂ 0 Ln 388, Col 27 Spaces: 4 UTF-8 CRLF {} Python 3.12.4 64-bit ⌂ Go Live ⌂ Prettier ⌂

Search ⌂ 10:43:pm 01/12/2024 ⌂

figure 120: main method to set up UDP connection and start server - Task 3

## Task 3 - Client Side Code



The screenshot shows a code editor interface with a dark theme. The top bar includes icons for File, Edit, Selection, and a search bar labeled "Task3". The left sidebar contains various icons for file operations like copy, paste, search, and refresh. The main workspace displays the contents of a Python file named "client.py". The code is as follows:

```
server > client.py > start_client
1 import socket
2 import threading
3
4 # Server configuration
5 HOST = '127.0.0.1'
6 PORT = 5689
7 SERVER_ADDRESS = (HOST, PORT)
8
9
10 def listen_to_server(client_socket):
11     try:
12         while True:
13             # Receive data from the server
14             # We don't need the server's address for UDP
15             message, _ = client_socket.recvfrom(1024)
16             if not message:
17                 print("**** Server disconnected ***")
18                 break
19             print(message.decode())
20     except Exception as e:
21         print(f"Error receiving data from server: {e}")
22     finally:
23         print("Disconnected from server.")
24         client_socket.close()
25
```

figure 121: Server Configs Client Side Code - Task 3

```
26
27     def start_client():
28         try:
29             # Create a UDP socket
30             client_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
31             print(f"\nConnected to the server on {HOST}:{PORT}\n")
32
33             # Start a thread to listen to server messages
34             threading.Thread(target=listen_to_server, args=(
35                 client_socket,), daemon=True).start()
36
37             # Send user input to the server
38             while True:
39                 user_input = input(" ")
40
41                 try:
42                     # Send the message to the server
43                     client_socket.sendto(user_input.encode(), SERVER_ADDRESS)
44                 except Exception as e:
45                     print(f"Error sending data: {e}")
46                     break
47             except Exception as e:
48                 print(f"Error connecting to the server: {e}")
49             finally:
50                 print("Closing connection.")
51                 client_socket.close()
52
53
54     if __name__ == "__main__":
55         start_client()
56
```

Activate Windows  
Go to Settings to activate Windows.

figure 122: Start Client Connection Code CLient Side - Task 3

## Teamwork:

Team Member	ID	Tasks Completed
Shahd Khalaf	1210545	<ul style="list-style-type: none"><li>● Task 1, section A.</li><li>● Task 3.</li><li>● Wrote the report parts for: 1.Task 1(A). 2.Task 3.</li></ul>
Majd Hamarsheh	1212036	<ul style="list-style-type: none"><li>● Task 1, section B.</li><li>● Task 2, section A+C.</li><li>● Wrote the report parts for: 1.Task 1(B). 2.Task 3.</li></ul>
Sama Wahidee	1211503	<ul style="list-style-type: none"><li>● Task 1, section C.</li><li>● Task 2, section B+C.</li><li>● Wrote the report parts for: Task 1(C). Task 2.</li></ul>

## References:

1. [Geeksforgeeks HTTP](#)
2. [Wireshark](#)
3. [Geeksforgeeks Socket programming](#)
4. [Abuseipdb](#)
5. [Developer mozilla US/docs Web HTTP](#)
6. [Helpful resources](#)
7. [IPCONFIG command](#)
8. [Ping Command](#)
9. [TRACERT Command](#)
10. [TelNet Command](#)
11. [Nslookup Command](#)