

Faculty of Engineering and Technology

Department of Electrical and Computer Engineering

ENEE3320, COMPUTER NETWORKKS

Course project 1

Prepared by:

ID:

Lama Naser

1200190

Jehad Hamayel

1200348

Sec:1

Instructor: Dr.Abd Alkareem Awad .

Abstract

This project consists of three parts. The first part is to run some commands on the terminal; ping a device in the same network, ping www.yale.edu website, tracert and nslookup it. The second part is implementing server and client application both for TCP and for UDP: A client continuously sends the numbers from 0 to 1000,000 to a server listening on port 5566. The server counts the received messages. The last part is to implement a simple but a complete web server in python that is listening on port 7788. Both html and css are used for the design of our website.

Contents

Abstract.....	1
Table Of Figures	4
Part 1	6
1- What are ping, tracert, nslookup.....	6
2- Screenshot for some commands	6
Ping a device in the same network.....	6
ping www.yale.edu	7
tracert www.yale.edu	8
nslookup www.yale.edu.....	9
Part 2: Implement a server and client application.....	10
The Same Computer	10
UDP connection.....	10
TCP connection.....	11
On different computers connected with a cable.....	12
TCP connection.....	12
UDP connection.....	14
On different computers connected on the same network.....	16
TCP connection.....	16
UDP connection.....	17
Part 3: Web Server Application.....	18
Our Code:.....	18
Main.py code:	18
Testing the application on the same computer	21
on command line (en):	21
on the browser(en):	22
main_en.html code:	23
main_en.css code:	25
When “visit my html” clicked.....	26
The request message on the command line:.....	26
form.html code (local html file):	27
3Wschools link is clicked	28

on command line(ar):.....	29
on the browser(ar):.....	30
main_ar.html code:	30
main_ar.css code:.....	31
on the command line(any html file):.....	32
on the browser(any html file):.....	32
a .html code:.....	32
on the command line(any css file):	33
on the browser(any css file):	33
a .css file:	33
on the command line(any png file):	34
on the browser(any png file):	34
on the command line(any jpg file):.....	35
on the browser(any jpg file):	35
Google Website:.....	36
Stackoverflow website:	37
Birzeit university website.....	38
Error Webpage	39
Testing from another device	41
On command line	42
Conclusion.....	43

Table Of Figures

Figure 1: ping a device on the same network.....	6
Figure 2: ping www.yale.edu.....	7
Figure 3 : tracert www.yale.edu.....	8
Figure 4: nslookup www.yale.edu	9
Figure 5: UDP connection on the same computer	10
Figure 6: TCP connection on the same computer	11
Figure 7: TCP connection on two computers connected through cable.....	12
Figure 8: the error message	13
Figure 9: UDP connection on two computers connected through cable without loss.....	14
Figure 10: UDP connection on two computers connected through cable with loss.....	15
Figure 11: TCP connection on two computers connected through a network	16
Figure 12: TCP connection on two computers connected through a network	17
Figure 13:Our Code in python For web Server.....	20
Figure 14:the request messages for / or /index.html or /main_en.html or /en.....	21
Figure 15: the result of the request in the web page in English	22
Figure 16: main_en.html code	24
Figure 17: main_en.css code	25
Figure 18:visit my html.....	26
Figure 19:request message	26
Figure 20:form.html code	27
Figure 21:w3Schools Page	28
Figure 22: the request message when we enter /ar.....	29
Figure 23:the result of the request in the web page in Arabic	30
Figure 24: main_ar.html code	31
Figure 25: main_ar.css code	31
Figure 26: any html file request	32
Figure 27: the any html file on website.....	32
Figure 28:any.html file.....	32
Figure 29: the request message for a css file	33
Figure 30: the css file on website.....	33
Figure 31: .css file.....	33
Figure 32: the request message for a png file	34
Figure 33: the png file on website.....	34
Figure 34: the request message for a jpg file	35
Figure 35:the jpg file on website.....	35
Figure 36: google website	36
Figure 37: google website request message.....	36
Figure 38: stackoverflow website	37
Figure 39:stackoverflow website requst message.....	37
Figure 40:birzeit university website.....	38

Figure 41:birzeit university website request message.....	38
Figure 42: error webpage	39
Figure 43: error webpage request message	39
Figure 44: error.html code	40
Figure 45: error.css code.....	40
Figure 46:the test on the phone.....	41
Figure 47: the request message	42

Part 1

1- What are ping, tracert, nslookup

- **ping**: is a command-line utility acts as a test to see if a networked device reachable.
- **tracert**: is a command-line utility that can be used to trace the path for a protocol packet takes to its destination.
- **nslookup**: is a command that queries to domain name and addresses servers.

2- Screenshot for some commands

Ping a device in the same network

commands:

- 1- Ping a device in the same network, e.g. from a laptop to a smartphone
- 2- ping www.yale.edu
- 3- tracert www.yale.edu
- 4- nslookup www.yale.edu

Provide a screenshot of the runs an

Part2:

Implement the following server and continuously sends the numbers from counts the received messages.

Run the programs

- 1- on same computer
- 2- on 2 different computers co
- 3- on 2 different computers co

```
C:\Users\Dell> ping 192.168.1.99

Pinging 192.168.1.99 with 32 bytes of data:
Reply from 192.168.1.99: bytes=32 time=544ms TTL=64
Reply from 192.168.1.99: bytes=32 time=6ms TTL=64
Reply from 192.168.1.99: bytes=32 time=523ms TTL=64
Reply from 192.168.1.99: bytes=32 time=315ms TTL=64

Ping statistics for 192.168.1.99:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 6ms, Maximum = 544ms, Average = 347ms

C:\Users\Dell>
```

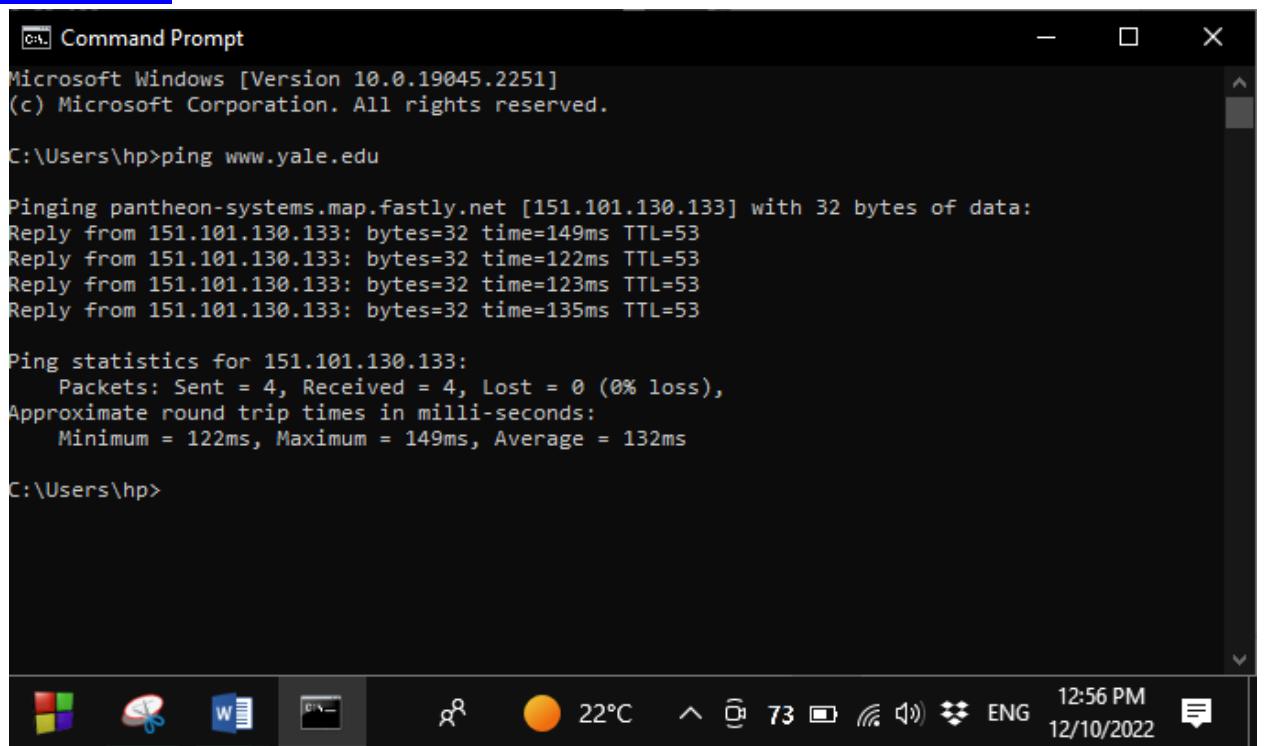
For each run, provide screenshots of the output. Also measure the time required to send the packets and the time required to receive the packets (from first packet to last packet).

When using different computers, make sure that the computers are within the same subnet for

Figure 1: ping a device on the same network

Here a successful reply from 192.168.1.99 is received, so the device is connected to the internet, and all hardware such as cables, the network card, the router and the modem are all working correctly, including the internet service provider. Four packets response were received from 192.168.1.99 with their TTL, with an average of 347ms delay.

ping www.yale.edu



```
Windows PowerShell [Version 10.0.19045.2251]
(c) Microsoft Corporation. All rights reserved.

C:\Users\hp>ping www.yale.edu

Pinging pantheon-systems.map.fastly.net [151.101.130.133] with 32 bytes of data:
Reply from 151.101.130.133: bytes=32 time=149ms TTL=53
Reply from 151.101.130.133: bytes=32 time=122ms TTL=53
Reply from 151.101.130.133: bytes=32 time=123ms TTL=53
Reply from 151.101.130.133: bytes=32 time=135ms TTL=53

Ping statistics for 151.101.130.133:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 122ms, Maximum = 149ms, Average = 132ms

C:\Users\hp>
```

Figure 2: ping www.yale.edu

Pinging a website is actually pinging the domain name and not its IP address. Because also name resolution issues related to DNS is tested, as the IP address is not memorized. Here a successful reply from www.yale.edu is received, so the device is connected to the internet, and all hardware such as cables, the network card, the router and the modem are all working correctly, including the internet service provider. Four packets response were received from 151.101.130.133 with their TTL, with an average of 132ms delay.

tracert www.yale.edu

```
2- ping www.yale.edu
3- tracert www.yale.edu
4- nslo
Provide a screen shot
Microsoft Windows [Version 10.0.19045.2251]
(c) Microsoft Corporation. All rights reserved.

Part2:
Implement the
continuously sending Tracing route to pantheon-systems.map.fastly.net [199.232.82.133]
counts the received over a maximum of 30 hops:
Run the program
1      6 ms    3 ms    53 ms superfast [192.168.1.1]
1- on same 2    23 ms   25 ms   41 ms 10.74.32.235
2- on 2 diff 3   367 ms   *     89 ms 10.74.59.186
3- on 2 diff 4   612 ms   324 ms  199 ms mrs1.decixmrs.fastly.net [185.1.47.121]
5   442 ms   405 ms  406 ms 199.232.82.133

For each run, press Enter to start the next run.
Trace complete.
Also measure the time taken for each hop.
(from first packet)
C:\Users\DELL>
```

When using different computers, for example, the IP of the first computer is 192.168.1.10, subnet mask 255.255.255.0 and the IP of the second computer is 192.168.1.11 and the subnet mask 255.255.255.0

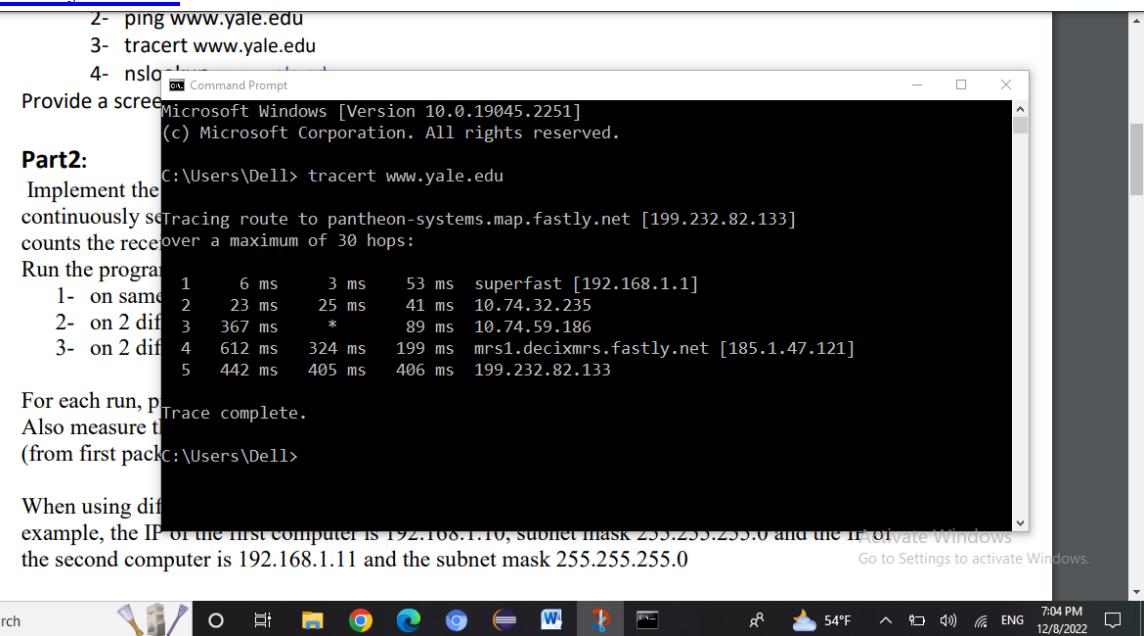
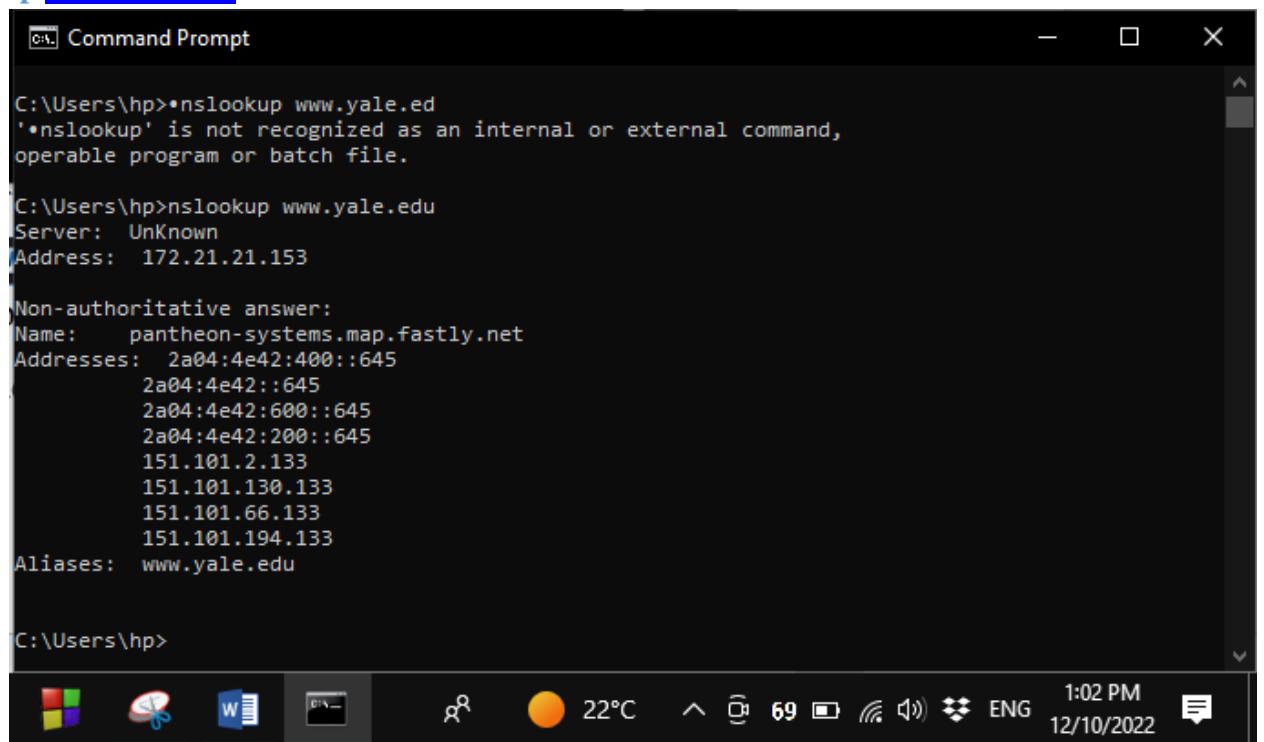


Figure 3 : tracert www.yale.edu

Tracert command (tracert) is a utility designed can be used for troubleshooting of connection issue, displays the time it takes for a packet of information to travel between a local device and a destination of IP address or domain. After performing this command for www.yale.edu, the results displayed are the hops that data packets take along their path to destination desired. When the measurements are stars (*) in request timed out and indicates that the router at that hop doesn't respond to the traceroute requests.

nslookup www.yale.edu



The screenshot shows a Windows Command Prompt window titled "Command Prompt". The command entered is "nslookup www.yale.edu". The output indicates that "nslookup" is not recognized as a command, but it then proceeds to show non-authoritative answers. The server is listed as "UnKnown" with address "172.21.21.153". The Name is "pantheon-systems.map.fastly.net" and the Addresses are multiple IPv6 and IPv4 addresses ranging from 151.101.2.133 to 151.101.194.133. TheAliases are "www.yale.edu". The system tray at the bottom shows standard icons like Start, File Explorer, Task View, and a search icon. The date and time are 12/10/2022, 1:02 PM. The weather icon shows 22°C.

```
C:\Users\hp>nslookup www.yale.edu
'nslookup' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\hp>nslookup www.yale.edu
Server: UnKnown
Address: 172.21.21.153

Non-authoritative answer:
Name: pantheon-systems.map.fastly.net
Addresses: 2a04:4e42:400::645
          2a04:4e42:600::645
          2a04:4e42:200::645
          151.101.2.133
          151.101.130.133
          151.101.66.133
          151.101.194.133
Aliases: www.yale.edu

C:\Users\hp>
```

Figure 4: nslookup www.yale.edu

Command of nslookup followed by the domain name displays a record IP address of the domain. It queries to domain name and addresses servers and gets the details. This can be shown in the figure above.

Part 2: Implement a server and client application

Server and client application was implemented both for TCP and for UDP: A client continuously sends the numbers from 0 to 1000,000 to a server listening on port 5566. The server counts the received messages.

The Same Computer

- First we run a python code at the same computer, then we estimate the time that it takes over the following connections

UDP connection

The image shows two side-by-side PyCharm IDE windows. The left window is titled 'part2_server_UDP' and contains the following Python code:

```
from socket import *
serverPort = 5566
# create UDP socket
serverSocket = socket(AF_INET, SOCK_DGRAM)
# bind socket to local port 5566
serverSocket.bind(('', serverPort))
print("The server is ready to receive")
count = 0
while True:
    # read from UDP socket into message, getting client's address
    message, clientAddress = serverSocket.recvfrom(2048)
    # increase the counter for each receive and print the count
    count = count+1
    print(count)
```

The right window is titled 'part2_client_UDP' and contains the following Python code:

```
#Jehad Hamayel 1200348
#Lama Naser 1200198
from socket import *
serverName = "localhost"
serverPort = 5566
# create UDP socket for server
clientSocket = socket(AF_INET, SOCK_DGRAM)
for x in range(0, 1000000):
    # attach server name and server port to message, send
    clientSocket.sendto(str(x).encode(), (serverName, serverPort))
#closing the socket
clientSocket.close()
```

Both windows show a 'Run' tab with a green play button icon. The left window's Run tab shows the output: 999996, 999997, 999998, 999999, 1000000. The right window's Run tab shows the output: Process finished with exit code 0.

Figure 5: UDP connection on the same computer

The time that this application took to ends its work is roughly 2 minutes and 34 seconds, without any loss of the data.

TCP connection

The screenshot shows two side-by-side code editors in a development environment. Both editors have a dark theme and are displaying Python code.

Left Editor (Server Side):

```
from socket import *
serverPort = 5566
# create TCP welcoming socket
serverSocket = socket(AF_INET, SOCK_STREAM)
serverSocket.bind(('', serverPort))
# server begins listening for incoming TCP requests
serverSocket.listen(1)
print("The server is ready to receive")
count = 0
while True:
    # server waits an accept for incoming request
    connectionSocket, addr = serverSocket.accept()
    # increase the counter for each accept and print the count
    count = count+1
    print(count)
    # closing the TCP socket Connection
    connectionSocket.close()
```

Right Editor (Client Side):

```
#Jehad Hamayel 1200348
#Lama Naser 1200199
from socket import *
#sending numbers from zero to one million
for y in range(0, 1000000):
    serverName = "localhost"
    serverPort = 5566
    # create TCP socket for server
    clientSocket = socket(AF_INET, SOCK_STREAM)
    #create connection between server and client
    clientSocket.connect((serverName, serverPort))
    #sending the message
    clientSocket.send(str(y).encode())
    #closing the socket after each request
    clientSocket.close()
```

Both editors have a 'Run' section at the bottom showing the output of their respective scripts. The server editor shows:

```
777770
999999
1000000
```

The client editor shows:

```
4:1 CRLF UTF-8 4 spaces Python 3.9 (part2_client_TCP) 14°C 100 611 PM 12/26/2022
```

Figure 6: TCP connection on the same computer

The time that this application took to end its work is roughly 9 minutes, without any loss of the data.

On different computers connected with a cable

- Second we run a python code at different computers connected with a cable, then we estimate the time that it takes over the following connections

TCP connection

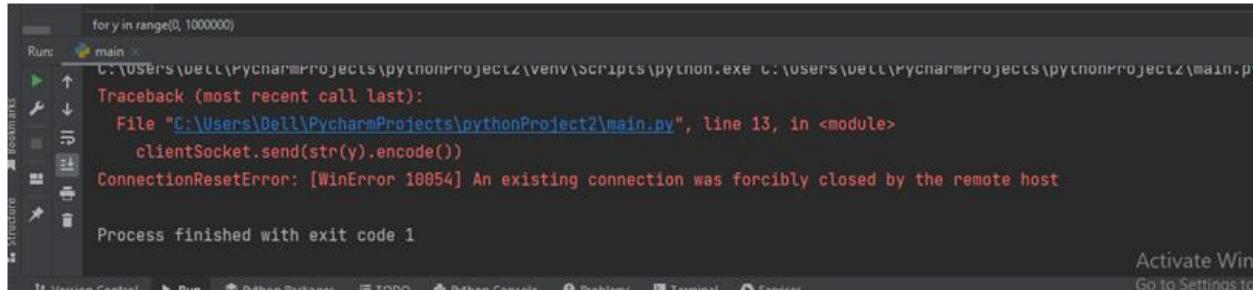
The image shows two computer screens. The top screen displays a PyCharm IDE with a Python script named 'main.py' for a TCP server. The code initializes a socket, binds it to port 5566, and enters a loop to accept connections and print a counter. The output window shows the counter increasing from 99995 to 100000. The bottom screen shows a PyCharm IDE with a client script 'main.py' that connects to the server at 192.168.1.13:5566 and sends numbers from 0 to 100,000. The terminal window shows the command to run the script and the message 'Process finished with exit code 0'. A stopwatch application is running on the bottom screen, showing a time of 02:01.73.

```
serverSocket = socket(AF_INET, SOCK_STREAM)
serverSocket.bind(('', serverPort))
# server begins listening for incoming TCP requests
serverSocket.listen(1)
print("The server is ready to receive")
count = 0
while True:
    # server waits an accept for incoming request
    connectionSocket, addr = serverSocket.accept()
    # increase the counter for each accept and print the counter
    count = count+1
    print(count)
    # closing the TCP socket Connection
    connectionSocket.close()
```

```
Jehad Hamayel 1200348
Lama Naser 1200190
from socket import *
#sending numbers from zero to one million
for y in range(0, 100000):
    serverName = '192.168.1.13'
    serverPort = 5566
    # create TCP socket for server
    clientSocket = socket(AF_INET, SOCK_STREAM)
    #create connection between server and client
    clientSocket.connect((serverName, serverPort))
    #sending the message
    clientSocket.send(str(y).encode())
for y in range(0, 100000)
```

Figure 7: TCP connection on two computers connected through cable

As note in the above pictures for the server and client just 100,000 messages were sent in 2 minutes, we tried to send 1 million messages through the cable but every time the following error was appear will sending-receiving messages.



A screenshot of the PyCharm IDE interface. The main window shows a code editor with Python code. A red error message is displayed in the terminal pane:

```
for y in range(0, 1000000)
Run: main
C:\Users\DELL\PycharmProjects\pythonProject2\venv\Scripts\python.exe C:\Users\DELL\PycharmProjects\pythonProject2\main.py
Traceback (most recent call last):
File "C:\Users\DELL\PycharmProjects\pythonProject2\main.py", line 13, in <module>
    clientSocket.send(str(y).encode())
ConnectionResetError: [WinError 10054] An existing connection was forcibly closed by the remote host
Process finished with exit code 1
```

The terminal also shows the command used to run the script: `C:\Users\DELL\PycharmProjects\pythonProject2\venv\Scripts\python.exe C:\Users\DELL\PycharmProjects\pythonProject2\main.py`.

Figure 8: the error message

We searched about this error and we find out that it may be caused by deferent reasons such as: the network link between server and client may be temporarily going down, running out of system resources, or sending malformed data.

UDP connection

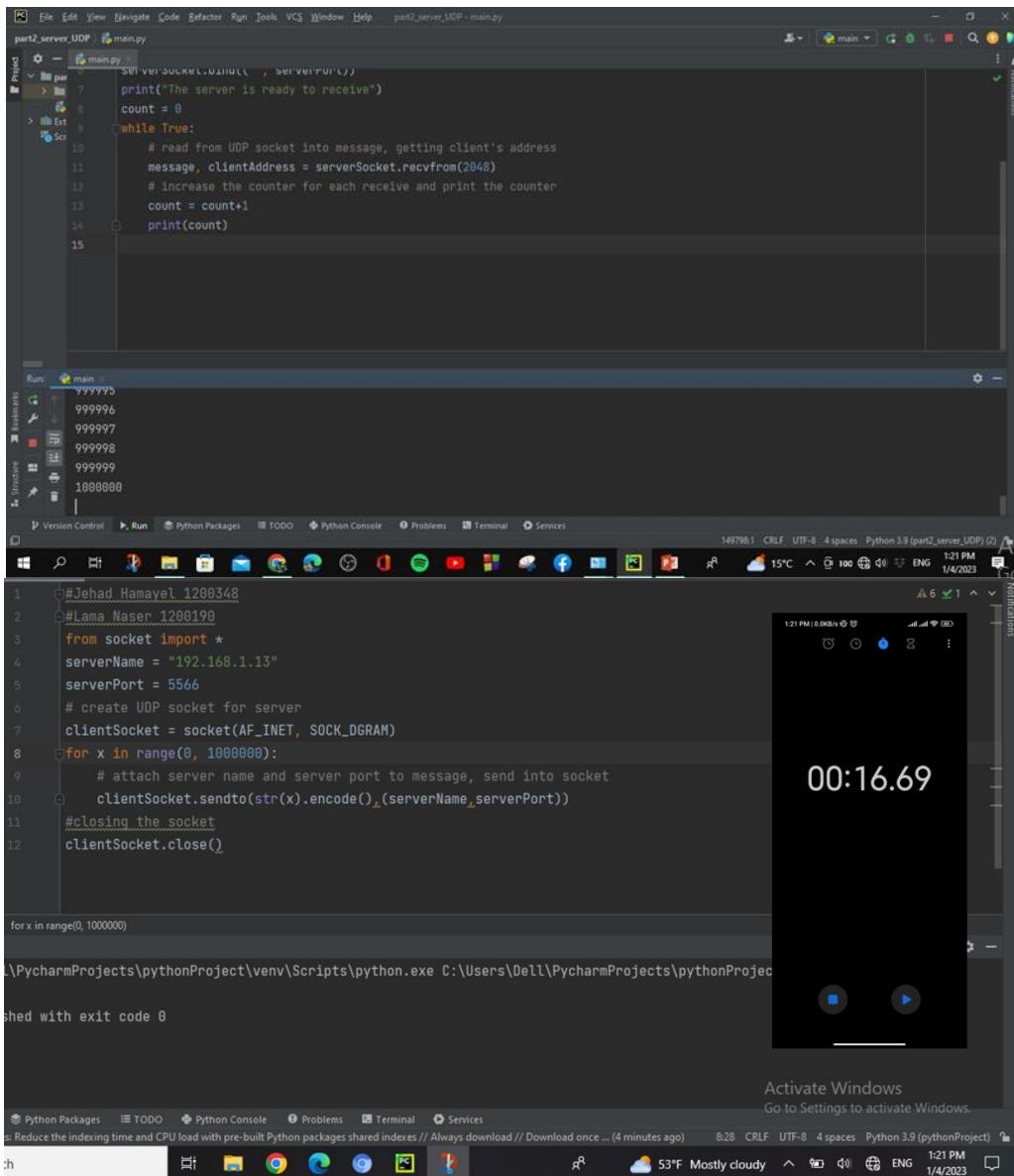


Figure 9: UDP connection on two computers connected through cable without loss

Here 1 million messages were sent in 16 seconds using UDP connection through a cable without any losing in the data.

- ❖ Also we tried this operation again and we noticed that there was some loss in the data that is because UDP connection does not give a guarantee for reliable data transfer, but on the other hand, it sends the data so fast in compare with TCP.

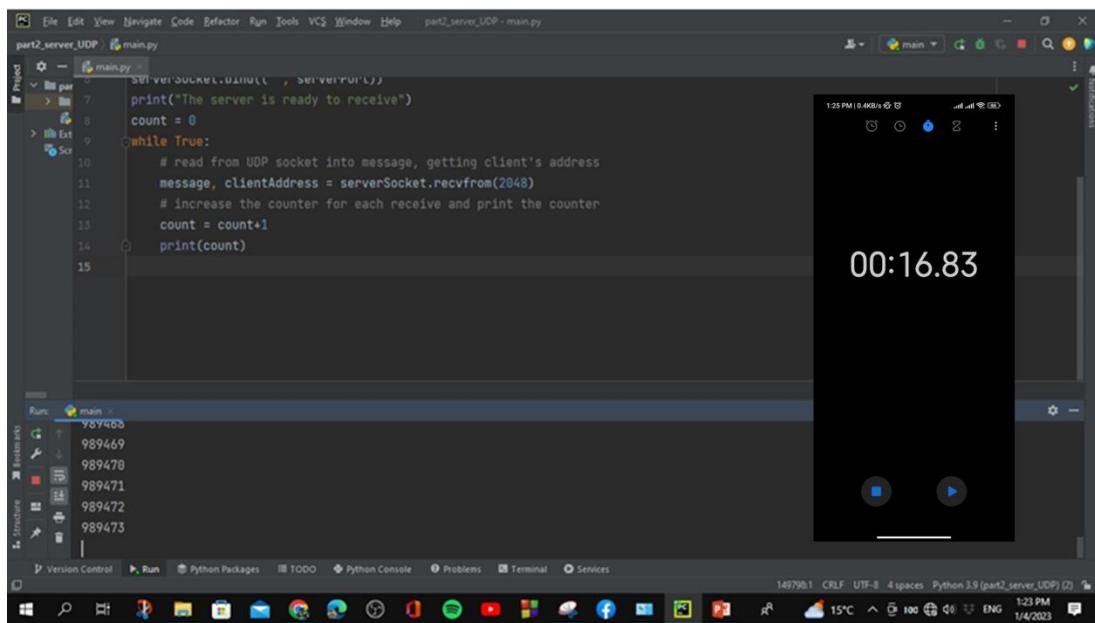


Figure 10: UDP connection on two computers connected through cable with loss

On different computers connected on the same network

- Finally, we run a python code at different computers connected on the same network, here we used the virtual box with Ubuntu operating system to be the client and the origin one as the server, then we estimate the time that it takes over the following connections

TCP connection

The image shows two side-by-side screenshots of Windows desktop environments. The top window is titled 'client.py' and contains Python code for a client socket. The bottom window is titled 'main.py' and contains Python code for a server socket. Both windows show code snippets and some output. A stopwatch application is overlaid on the bottom window, showing a time of 2:45:30.71. The taskbars at the bottom of both screens show various icons and system status.

```
1 #Jehad Hamayel 1200348
2 #Lama Naser 1200190
3 from socket import *
4 #sending numbers from zero to one mellion
5 for y in range(0, 100000):
6     serverName = '192.168.1.209'
7     serverPort = 5566
8     # create TCP socket for server
9     clientSocket = socket(AF_INET, SOCK_STREAM)
10    #create connection between server and client
11    clientSocket.connect((serverName, serverPort))
12    #sending the message
13    clientSocket.send(str(y).encode())
14    #closing the socket after each request
15    clientSocket.close()
16
```

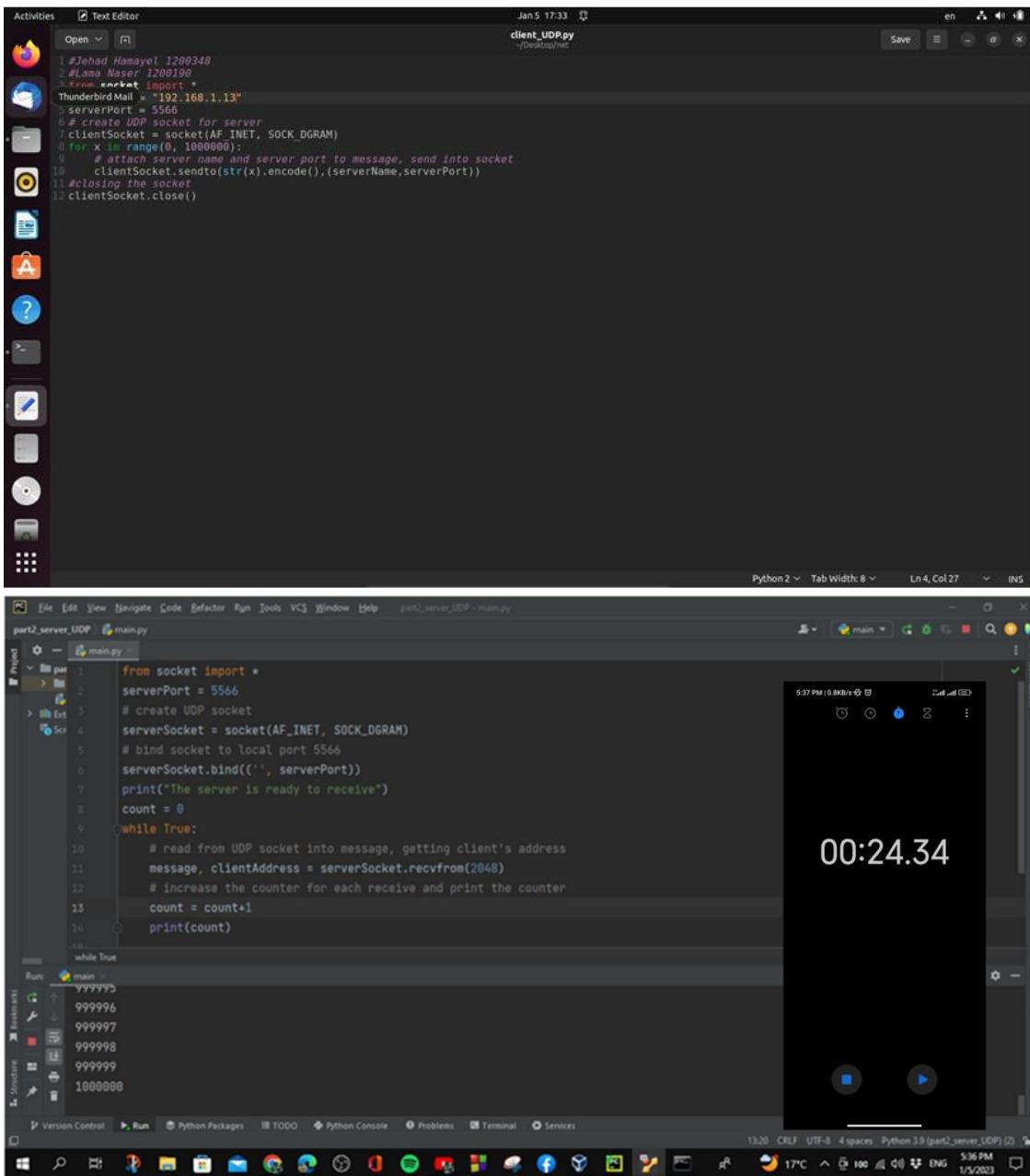


```
from socket import *
serverPort = 5566
# create TCP welcoming socket
serverSocket = socket(AF_INET, SOCK_STREAM)
serverSocket.bind(('', serverPort))
# server begins listening for incoming TCP requests
serverSocket.listen(1)
print("The server is ready to receive")
count = 0
while True:
    # server waits an accept for incoming request
    connectionSocket, addr = serverSocket.accept()
    # increase the counter for each accept and print the counter
    count += 1
    print(count)
```

Figure 11: TCP connection on two computers connected through a network

It is very clear that sending data through a network is slower than wired connection that is because the network has many instances where interference and distance can slow down the communication.

UDP connection



The screenshot shows a Linux desktop environment with two terminal windows open. The top window is a terminal window titled 'client_UDP.py' containing Python code for a UDP client. The bottom window is a terminal window titled 'part2_server_UDP - main.py' containing Python code for a UDP server. The server terminal shows a timestamp of '00:24.34'.

```
client_UDP.py
1 #Jehad Hamayel 1208348
2 #Lam Naser 1208199
3 from socket import *
ThunderbirdMail = "192.168.1.13"
serverPort = 5566
# create UDP socket for server
clientSocket = socket(AF_INET, SOCK_DGRAM)
for x in range(0, 1000000):
    # attach server name and server port to message, send into socket
    clientSocket.sendto(str(x).encode(),(serverName,serverPort))
#closing the socket
clientSocket.close()

part2_server_UDP - main.py
1 from socket import *
serverPort = 5566
# create UDP socket
serverSocket = socket(AF_INET, SOCK_DGRAM)
# bind socket to local port 5566
serverSocket.bind(('', serverPort))
print("The server is ready to receive")
count = 0
while True:
    # read from UDP socket into message, getting client's address
    message, clientAddress = serverSocket.recvfrom(2048)
    # increase the counter for each receive and print the counter
    count = count+1
    print(count)
while True:
```

Figure 12: TCP connection on two computers connected through a network

Part 3: Web Server Application.

In this part socket programming was used to implement a simple but a complete web server listening on port 7788 in python language.

Our Code:

Main.py code:

The screenshot shows a Python code editor interface with two tabs open:

- httpserver.py**: A Python script for a TCP server. It creates a socket, binds it to port 7788, and listens for incoming connections. It then reads a sentence from the client, splits it into IP and port, and extracts the file name and extension. It checks if the file exists and sends an appropriate HTTP response back to the client.
- main.py**: A Python script for a web browser. It handles different file requests based on the file extension (HTML, CSS, AR, or FORM). It opens the requested files and sends them back to the client as responses.

The status bar at the bottom displays various system and application details, including the date (12/31/2022), time (4:18 PM), and Python version (Python 3.9 (httpstest1)).

```

main.py
119     print("in pic2.png")
120     connectionSocket.send(f2.read())
121 elif t[0] == "pic1.jpg":
122     connectionSocket.send(str("HTTP/1.1 200 OK \r\n").encode())
123     connectionSocket.send(str("Content-Type: image/jpg \r\n").encode())
124     connectionSocket.send(str("\r\n").encode())
125     f2 = open("pic1.JPG", "rb")
126     print("in pic1.jpg")
127     connectionSocket.send(f2.read())
128 elif r[1] == "ar" and (len(r) == 1):
129     connectionSocket.send(str("HTTP/1.1 200 OK \r\n").encode())
130     connectionSocket.send(str("Content-Type: text/html; charset=utf-8 \r\n").encode())
131     connectionSocket.send(str("\r\n").encode())
132     f2 = open("main_ar.html", "rb")
133     print("in ar")
134     connectionSocket.send(f2.read())
135 elif t[0] == "html" and (len(r) >= 2) and (not (t[0] == "main_en.html") or t[0] == "" or t[0] == "index.html" or t[0] == "en")):
136     connectionSocket.send(str("HTTP/1.1 200 OK \r\n").encode())
137     connectionSocket.send(str("Content-Type: text/html \r\n").encode())
138     connectionSocket.send(str("\r\n").encode())
139     f2 = open("any.html", "r")
140     print("in any.html")
141     connectionSocket.send(str(f2.read()).encode())
142 elif t[0] == "error.css":
143     connectionSocket.send(str("HTTP/1.1 200 OK \r\n").encode())
144     connectionSocket.send(str("Content-Type: text/css \r\n").encode())
145     connectionSocket.send(str("\r\n").encode())
146     f2 = open("error.css", "r")
147     print("in error.css")
148     connectionSocket.send(str(f2.read()).encode())
149 elif r[1] == "css" and (len(r) >= 2) and (not (t[0] == "main_en.html") or t[0] == "" or t[0] == "index.html" or t[0] == "en")):
150     connectionSocket.send(str("HTTP/1.1 200 OK \r\n").encode())
151     connectionSocket.send(str("Content-Type: text/css \r\n").encode())
152     connectionSocket.send(str("\r\n").encode())
153 while True > elif t[0] == "pic2.png"):
154
Python Packages TODO Python Console Problems Terminal Services
70:15 CRLF UTF-8 4 spaces Python 3.9 (httpptest)

20°C 4:20 PM 12/31/2022

main.py
155     f2 = open("any.css", "r")
156     print("in any.css")
157     connectionSocket.send(str(f2.read()).encode())
158 elif r[1] == "png" and (len(r) >= 2) and (not (t[0] == "main_en.html") or t[0] == "" or t[0] == "index.html" or t[0] == "en")):
159     connectionSocket.send(str("HTTP/1.1 200 OK \r\n").encode())
160     connectionSocket.send(str("Content-Type: image/png \r\n").encode())
161     connectionSocket.send(str("\r\n").encode())
162     f2 = open("any.PNG", "rb")
163     print("in any.PNG")
164     connectionSocket.send(f2.read())
165 elif r[1] == "jpg" and (len(r) >= 2) and (not (t[0] == "main_en.html") or t[0] == "" or t[0] == "index.html" or t[0] == "en")):
166     connectionSocket.send(str("HTTP/1.1 200 OK \r\n").encode())
167     connectionSocket.send(str("Content-Type: image/jpg \r\n").encode())
168     connectionSocket.send(str("\r\n").encode())
169     f2 = open("any.JPG", "rb")
170     print("in any.jpg")
171     connectionSocket.send(f2.read())
172 elif r[1] == "go" and (len(r) == 1):
173     connectionSocket.send(str("HTTP/1.1 307 Temporary Redirect \r\n").encode())
174     connectionSocket.send(str("Content-Type: text/html \r\n").encode())
175     connectionSocket.send(str("Location: https://www.google.com \r\n").encode())
176     connectionSocket.send(str("\r\n").encode())
177 elif r[1] == "so" and (len(r) == 1):
178     connectionSocket.send(str("HTTP/1.1 307 Temporary Redirect \r\n").encode())
179     connectionSocket.send(str("Content-Type: text/html \r\n").encode())
180     connectionSocket.send(str("Location: https://stackoverflow.com \r\n").encode())
181     connectionSocket.send(str("\r\n").encode())
182 elif r[1] == "bzu" and (len(r) == 1):
183     connectionSocket.send(str("HTTP/1.1 307 Temporary Redirect \r\n").encode())
184     connectionSocket.send(str("Content-Type: text/html \r\n").encode())
185     connectionSocket.send(str("Location: https://ritaj.birzeit.edu \r\n").encode())
186     connectionSocket.send(str("\r\n").encode())
187 else:
188     connectionSocket.send(str("HTTP/1.1 404 Not Found \r\n").encode())
189
while True > elif r[1] == "css" and (len(r) >= 2) and (not (t[0] == "main_en.html") or t[0] == "" or t[0] == "index.html" or t[0] == "en")):
190     f2 = open("any.css", "r")
191     print("in any.css")
192     connectionSocket.send(str(f2.read()).encode())
193
Python Packages TODO Python Console Problems Terminal Services
104:22 CRLF UTF-8 4 spaces Python 3.9 (httpptest1)

20°C 4:21 PM 12/31/2022

```

The screenshot shows a code editor window with a dark theme. The code is written in Python and is intended to handle a connection socket. It includes sending a Content-Type header, reading from an error.html file, printing the IP and port, and then closing the connection socket.

```
137     connectionSocket.send(str("Content-Type: text/html \r\n").encode())
138     connectionSocket.send(str("\r\n").encode())
139     f2 = open("error.html", 'r')
140     print("in error.html")
141     connectionSocket.send(str(f2.read())+ "Ip:" + str(ip) + " Port:" + str(port)).encode()
142
143     connectionSocket.close()
144
```

The status bar at the bottom of the editor shows various icons and information: Python Packages, TODO, Python Console, Problems, Terminal, Services, 144:1 CRLF, UTF-8, 4 spaces, Python 3.9 (http://test), 20°C, 100%, ENG, 4:22 PM, and 12/31/2022.

Figure 13:Our Code in python For web Server

Testing the application on the same computer

- ❖ if the request is / or /index.html or /main_en.html or /en (for example localhost:7788/Or localhost:7788/en) then the server sends main_en.html file with Content-Type: text/html as shown in the figure below.

on command line (en):

```
C:\Users\hp\Pycharm\Projects\httpstest\venv\scripts\python.exe C:\Users\hp\Pycharm\Projects\httpstest\main.py
The server is ready to receive
GET / HTTP/1.1
Host: localhost:7788
Connection: keep-alive
sec-ch-ua: "Not2Brand";v="8", "Chromium";v="108", "Google Chrome";v="108"
sec-ch-ua-mobile: 70
sec-ch-ua-platform: "Windows"
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/108.0.0.0 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/png,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9

in en
GET /index_en.css HTTP/1.1
Host: localhost:7788
Connection: keep-alive
sec-ch-ua: "Not2Brand";v="8", "Chromium";v="108", "Google Chrome";v="108"
sec-ch-ua-mobile: 70
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/108.0.0.0 Safari/537.36
Accept: text/css,*/*;q=0.8
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: no-cors
Sec-Fetch-Dest: style
Referer: http://localhost:7788/
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9

in main_en.css
GET /pic2.png HTTP/1.1
Host: localhost:7788
Connection: keep-alive
sec-ch-ua: "Not2Brand";v="8", "Chromium";v="108", "Google Chrome";v="108"
sec-ch-ua-mobile: 70
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/108.0.0.0 Safari/537.36
Accept: image/avif,image/webp,image/png,image/svg+xml,image/*,*/*;q=0.8
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: no-cors
Sec-Fetch-Dest: image
Referer: http://localhost:7788/
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9

in pic2.jpg
GET /pic1.jpg HTTP/1.1
Host: localhost:7788
Connection: keep-alive
sec-ch-ua: "Not2Brand";v="8", "Chromium";v="108", "Google Chrome";v="108"
sec-ch-ua-mobile: 70
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/108.0.0.0 Safari/537.36
Accept: image/avif,image/webp,image/png,image/svg+xml,image/*,*/*;q=0.8
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: no-cors
Sec-Fetch-Dest: image
Referer: http://localhost:7788/
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9

in error.html
|
|
```

Figure 14:the request messages for / or /index.html or /main_en.html or /en

on the browser(en):

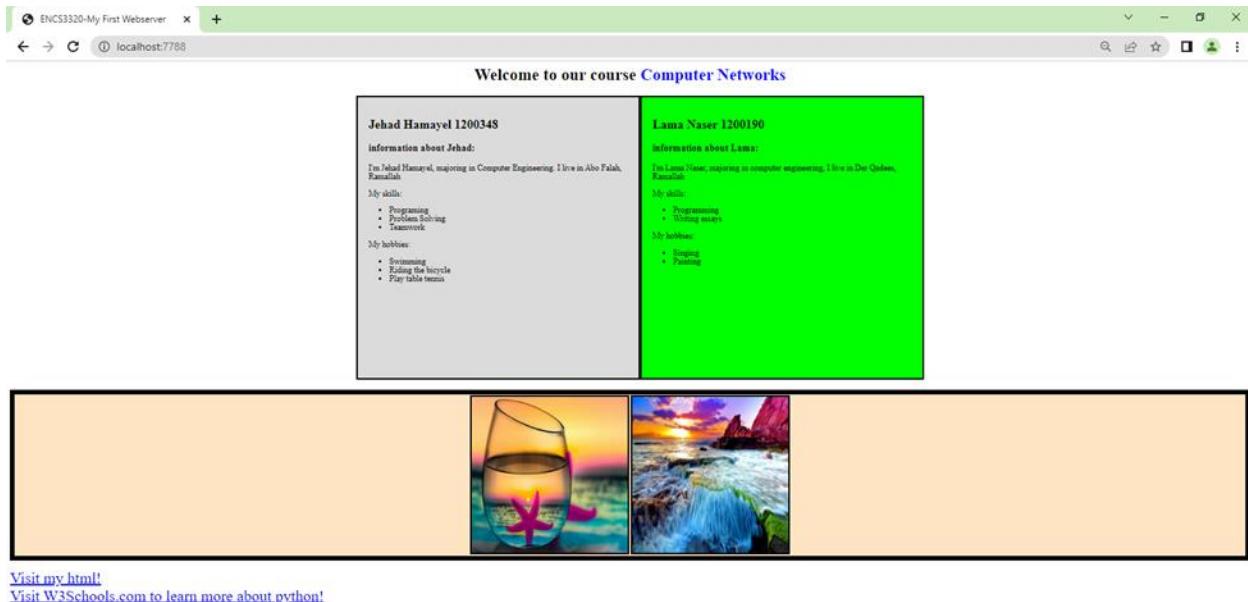


Figure 15: the result of the request in the web page in English

For the HTML code, “ENCS3320-My First Webserver” in the title, “Welcome to our course computer networks” and our names and IDs were printed on the page of our website. And some information about us. This is the page of the English version of the website. All these texts were styled in the CSS code.

main_en.html code:

```
main.py
main_en.html

1  <!DOCTYPE html>
2  <html>
3  <head>
4      <title>ENCS3320-My First Webserver</title>
5      <link rel="stylesheet" href="main_en.css">
6  </head>
7  <body>
8      <div class="row1">
9          <h1 class="class1">Welcome to our course </h1><h1 class="class2">Computer Networks</h1>
10     </div>
11     <div class="row">
12         <div class="d1">
13             <h2>Jehad Hamayel 1200348</h2>
14             <h3>information about Jehad:</h3>
15             <p>I'm Jehad Hamayel, majoring in Computer Engineering. I live in Abo Falah, Ramallah</p>
16             <p>My skills:</p>
17             <ul>
18                 <li>Programming</li>
19                 <li>Problem Solving</li>
20                 <li>Teamwork</li>
21             </ul>
22             <p>My hobbies:</p>
23             <ul>
24                 <li>Swimming</li>
25                 <li>Riding the bicycle</li>
26                 <li>Play table tennis</li>
27             </ul>
28         </div>
29         <div class="d2">
30             <h2>Lama Naser 1200190</h2>
31             <h3>information about Lama:</h3>
32             <p>I'm Lama Naser, majoring in computer engineering, I live in Der Qadees, Ramallah</p>
33         </div>
34     </div>
35     <p>My skills:</p>
36     <ul>
37         <li>Programming</li>
38         <li>Writing essays</li>
39     </ul>
40     <p>My hobbies:</p>
41     <ul>
42         <li>Singing</li>
43         <li>Painting</li>
44     </ul>
45
46     </div>
47     </div>
48     <div class="imRow">
49         
50         
51     </div>
52     <br>
53     <a href="form.html">Visit my html!</a>
54     <br>
55     <a href="https://www.w3schools.com/python/gloss_python_multi_line_strings.asp">Visit W3Schools.com to learn more about python!</a>
56
57 </body>
58 </html>
```

```
35      <p>My skills:</p>
36      <ul>
37          <li>Programming</li>
38          <li>Writing essays</li>
39      </ul>
40      <p>My hobbies:</p>
41      <ul>
42          <li>Singing</li>
43          <li>Painting</li>
44      </ul>
45
46      </div>
47      </div>
48      <div class="imRow">
49          
50          
51      </div>
52
53      <br>
54      <a href="form.html">Visit my html!</a>
55
56      <br>
57      <a href="https://www.w3schools.com/python/gloss_python_multi_line_strings.asp">Visit W3Schools.com to learn more about python!</a>
58
59  </body>
60  </html>
```

html > body > div.row > div.d2 > p

Python Packages TODO Python Console Problems Terminal Services

34:15 CRLF UTF-8 4 spaces Python 3.9 (http://test1) 427 PM 20°C ENG 12/31/2022

Figure 16: main_en.html code

main_en.css code:

The image shows a code editor interface with three vertically stacked panels, each displaying a portion of the same CSS file. The top panel shows lines 1 through 35, the middle panel shows lines 36 through 49, and the bottom panel shows lines 50 through 52. The code uses inline-block displays and flexbox properties for layout, along with various colors and border styles.

```
1 h1.class1, h1.class2{  
2     display: inline;  
3 }  
4 div.row1{  
5     text-align: center;  
6 }  
7 div.imRow{  
8     text-align: center;  
9     border: 10px solid black;  
10    background-color: bisque;  
11 }  
12 img{  
13     border: 3px solid black;  
14 }  
15 a{  
16     font-size: 30px;  
17     justify-content: center;  
18 }  
19 h1.class2{  
20     color: blue;  
21 }  
22 div.row{  
23     width: 100%;  
24     display: flex;  
25     flex-direction: row;  
26     justify-content: center;  
27     padding: 20px;  
28 }  
29  
30  
31  
32 }  
33  
34 div.d1{  
35     width: 500px;  
36     height: 500px;  
37     display: inline-block;  
38     border: 3px solid black;  
39     padding: 20px;  
40     background-color: rgb(219, 219, 219);  
41 }  
42 div.d2{  
43     width: 500px;  
44     height: 500px;  
45     display: inline-block;  
46     border: 3px solid black;  
47     padding: 20px;  
48     background-color: rgb(0, 255, 0);  
49 }  
50  
51  
52
```

Figure 17: main_en.css code

When “visit my html” clicked

When “visit my html” clicked, then a local html file will appear.

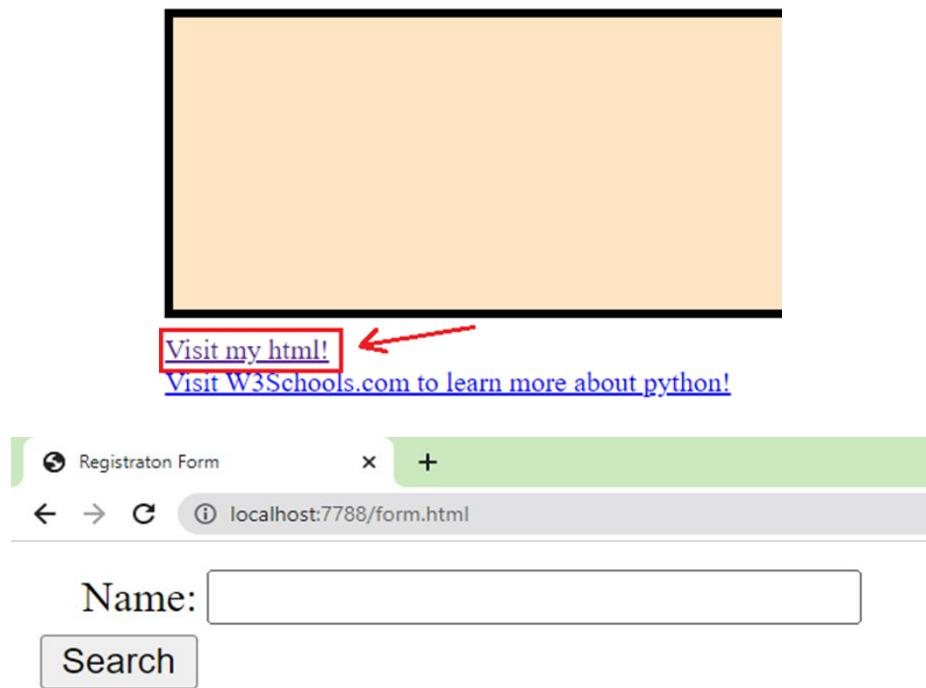


Figure 18:visit my html

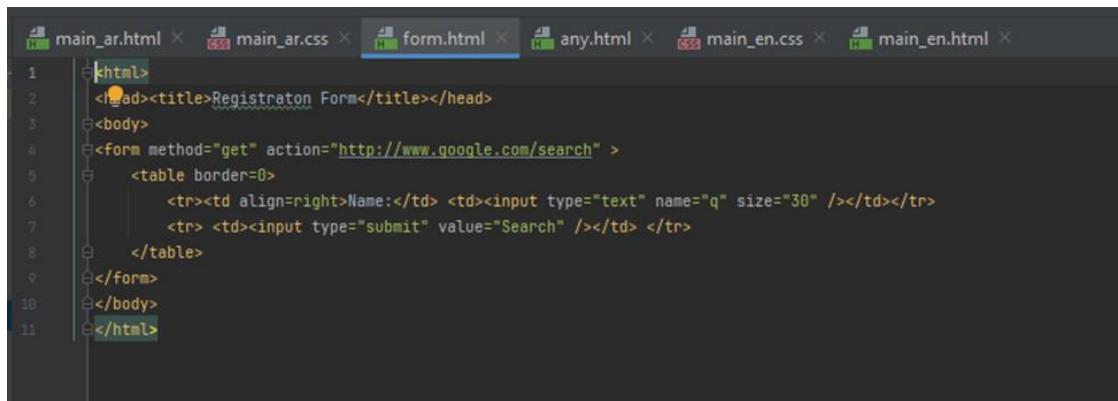
The request message on the command line:

```
GET /form.html HTTP/1.1
Host: localhost:7788
Connection: keep-alive
sec-ch-ua: "Not?A_Brand";v="8", "Chromium";v="108", "Google Chrome";v="108"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Windows"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/108.0.0.0 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: http://localhost:7788/
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
```

in form.html

Figure 19:request message

form.html code (local html file):



A screenshot of a code editor window showing the content of the 'form.html' file. The window title is 'form.html'. The code itself is as follows:

```
1 <html>
2   <head><title>Registration Form</title></head>
3   <body>
4     <form method="get" action="http://www.google.com/search" >
5       <table border=0>
6         <tr><td align=right>Name:</td> <td><input type="text" name="q" size="30" /></td></tr>
7         <tr> <td><input type="submit" value="Search" /></td> </tr>
8       </table>
9     </form>
10    </body>
11  </html>
```

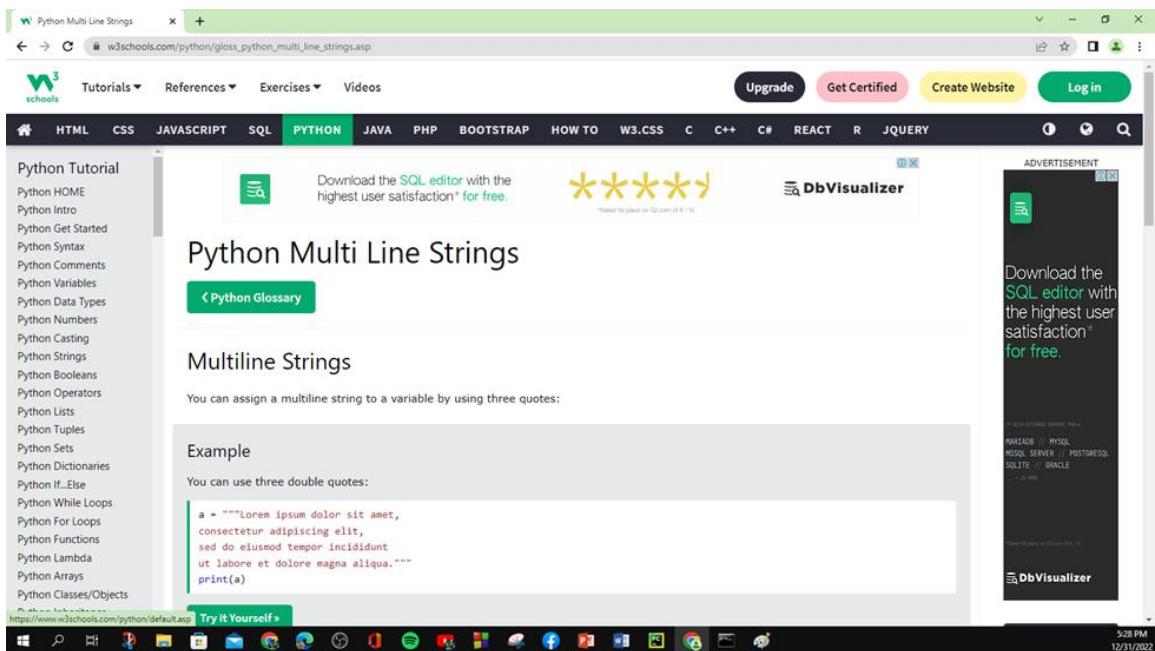
Figure 20:form.html code

3Wschools link is clicked

If this link is clicked then the following page will appear



[Visit my html!](#)
[Visit W3Schools.com to learn more about python!](#)



Python Multi Line Strings

Multiline Strings

You can assign a multiline string to a variable by using three quotes:

Example

You can use three double quotes:

```
a = """Lorem ipsum dolor sit amet,
consectetur adipiscing elit,
sed do eiusmod tempor incididunt
ut labore et dolore magna aliqua."""
print(a)
```

Figure 21:w3Schools Page

- ❖ If the request is `/ar` then the server response with `main_ar.html` which is an Arabic version of `main_en.html`

on command line(ar):

The image consists of three vertically stacked screenshots of a PyCharm terminal window. Each screenshot shows the command-line interface with the following text output:

```

main
C:\Users\hp\PycharmProjects\httptest1\venv\Scripts\python.exe C:\Users\hp\PycharmProjects\httptest1\main.py
The server is ready to receive
GET /ar HTTP/1.1
Host: localhost:7788
Connection: keep-alive
Cache-Control: max-age=0
sec-ch-ua: "NotA[Brand];v="8", "Chromium";v="108", "Google Chrome";v="108"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Windows"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/108.0.0.0 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9

in main_ar.css
GET /pic2.png HTTP/1.1
Host: localhost:7788
Connection: keep-alive
sec-ch-ua: "NotA[Brand];v="8", "Chromium";v="108", "Google Chrome";v="108"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Windows"
Accept: text/css,*/*;q=0.1
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: no-cors
Sec-Fetch-Dest: style
Referer: http://localhost:7788/ar
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9

in main_ar.css
GET /pic2.png HTTP/1.1
Host: localhost:7788
Connection: keep-alive
sec-ch-ua: "NotA[Brand];v="8", "Chromium";v="108", "Google Chrome";v="108"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Windows"
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://localhost:7788/ar
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9

in pic2.png
GET /pic1.jpg HTTP/1.1
Host: localhost:7788
Connection: keep-alive
sec-ch-ua: "NotA[Brand];v="8", "Chromium";v="108", "Google Chrome";v="108"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Windows"
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://localhost:7788/ar
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9

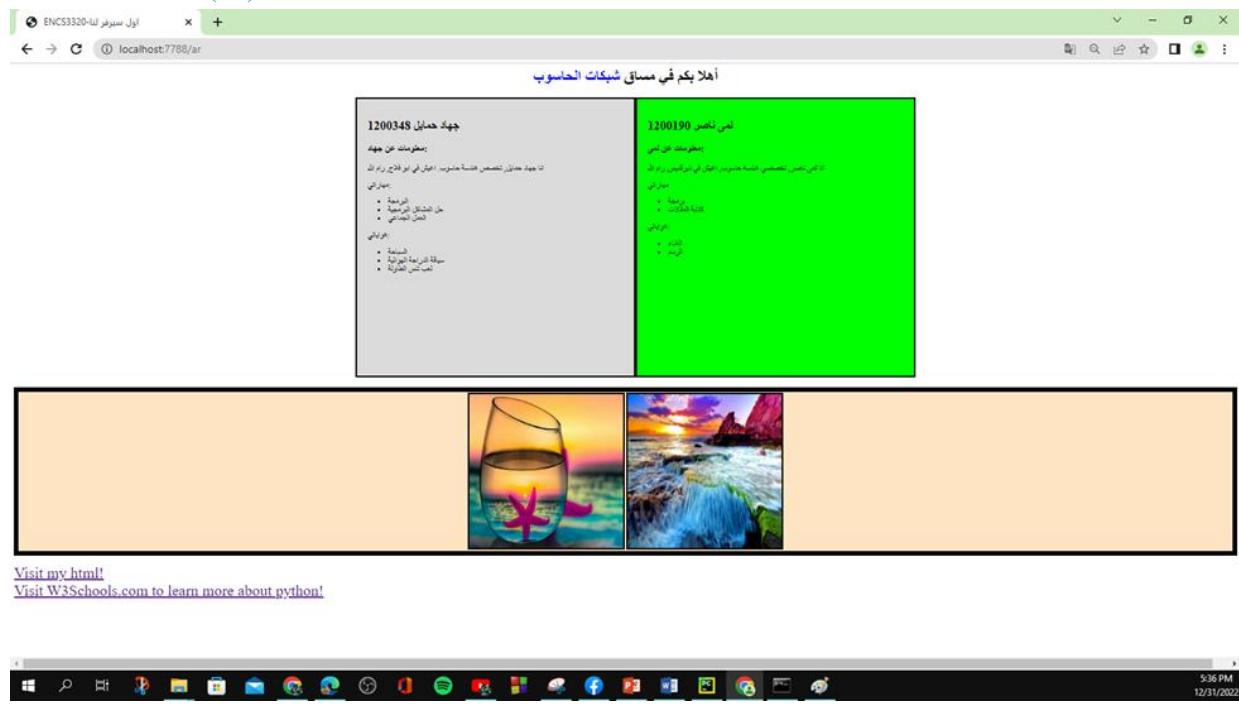
in pic1.jpg

```

The terminal window includes standard PyCharm navigation and status bars at the top and bottom.

Figure 22: the request message when we enter `/ar`

on the browser(ar):



Visit my [html!](#)
Visit [W3Schools.com](#) to learn more about python!



Figure 23:the result of the request in the web page in Arabic

main_ar.html code:

```
main.py main_ar.html main_en.css main_en.html
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <title>ENCS3320-اول سيرفر لـ LAN</title>
5      <link rel="stylesheet" href="main_ar.css">
6  </head>
7  <body>
8      <div class="row1">
9          <h1 class="class1" style="text-align: center;">أهلا بك في مساق شبكات الحاسوب</h1>
10         <div class="row2">
11             <div class="d1">
12                 <h2>جهاز حاسوب</h2>
13                 <h3>معلومات عن جهاز</h3>
14                 <p>جهاز حاسوب متعدد الأغراض، يعمل على إنشاء و إدارة المحتوى.</p>
15                 <p>مواصفات:</p>
16                 <ul>
17                     <li>الذاكرة: 8GB</li>
18                     <li>البطاقة المفتوحة: Intel Core i5</li>
19                     <li>النظام التشغيلي: Windows 10 Pro</li>
20                 </ul>
21                 <p>مميزات:</p>
22                 <ul>
23                     <li>السرعة: 500GB</li>
24                     <li>الطاقة: 1TB</li>
25                     <li>النظام التشغيلي: Windows 10 Pro</li>
26                     <li>النظام التشغيلي: Linux</li>
27                 </ul>
28             </div>
29             <div class="d2">
30                 <h2>لپ تاپ</h2>
31                 <h3>معلومات عن لپ تاپ</h3>
32                 <p>لپ تاپ متعدد الأغراض، يعمل على إنشاء و إدارة المحتوى.</p>
33                 <p>مواصفات:</p>
34                 <ul>
35                     <li>الذاكرة: 16GB</li>
36                     <li>البطاقة المفتوحة: NVIDIA GeForce RTX 3060</li>
37                     <li>النظام التشغيلي: Windows 11 Pro</li>
38                 </ul>
39                 <p>مميزات:</p>
40                 <ul>
41                     <li>السرعة: 1TB</li>
42                     <li>الطاقة: 1TB</li>
43                 </ul>
44         </div>
45     </div>
46     <img alt="Background image of a sunset over water" style="width: 100%; height: 100%; object-fit: cover; position: absolute; top: 0; left: 0; z-index: -1;">
47 </body>
48 </html>
```

```

52     <li><a href="#">Visit my html!</a>
53   </ul>
54 </div>
55 </div>
56 <div class="imRow">
57   
58   
59 </div>
60 <br>
61 <a href="form.html">Visit my html!</a>
62 <br>
63 <a href="https://www.w3schools.com/python/gloss_python_multi_line_strings.asp">Visit W3Schools.com to learn more about python!</a>
64 </body>
65 </html>

```

html > body > div.row > div.d2 > ul > li

Python Packages TODO Python Console Problems Terminal Services

41:19 CRLF UTF-8 4 spaces Python 3.9 (http://test1) 4:34 PM 12/31/2022

Figure 24: main_ar.html code

main_ar.css code:

```

1   .l.class1, .h1.class2{
2     display: inline;
3   }
4   div.row1{
5     text-align: center;
6   }
7   div.imRow{
8     text-align: center;
9     border: 1px solid black;
10    background-color: bisque;
11  }
12  img{
13    border: 3px solid black;
14  }
15  a{
16    font-size: 30px;
17    justify-content: center;
18  }
19  h1.class2{
20    color: blue;
21  }
22  div.row{
23    width: 100%;
24    display: flex;
25    flex-direction: row;
26    justify-content: center;
27    padding: 20px;
28  }
29  div.d1{
30    width: 500px;
31    height: 500px;
32    display: inline-block;
33    border: 3px solid black;
34    padding: 20px;
35    background-color: rgb(219, 219, 219);
36  }
37  div.d2{
38    width: 500px;
39
40    height: 500px;
41    display: inline-block;
42    border: 3px solid black;
43    padding: 20px;
44    background-color: rgb(0, 255, 0);
45  }

```

main.py main_ar.html main_ar.css main_en.css main_en.html

Python Packages TODO Python Console Problems Terminal Services

1:1 CRLF UTF-8 4 spaces Python 3.9 (http://test1) 4:35 PM 12/31/2022

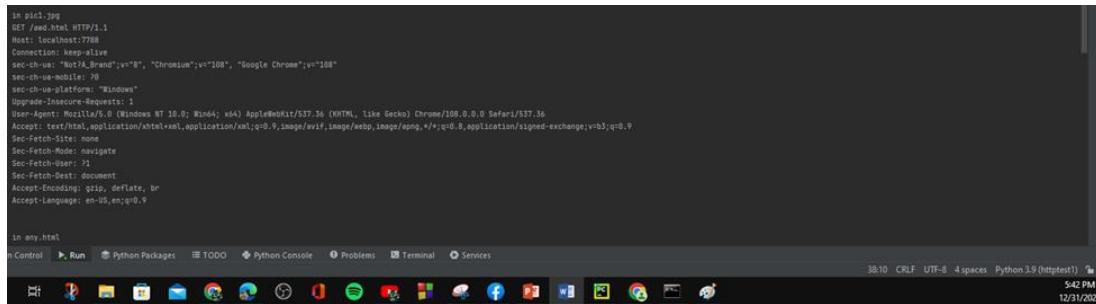
Python Packages TODO Python Console Problems Terminal Services

43:12 CRLF UTF-8 4 spaces Python 3.9 (http://test1) 4:35 PM 12/31/2022

Figure 25: main_ar.css code

- ❖ if the request is an **.html** file then the server sends the a.html file with Content-Type: text/html.

on the command line(any html file):



```
in pic1.jpg
GET /a.html HTTP/1.1
Host: localhost:7788
Connection: keep-alive
sec-ch-ua: "Not%7A,Braun";v="80", "Chromium";v="108", "Google Chrome";v="108"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Windows"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/108.0.0.0 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
```

Figure 26: any html file request

on the browser(any html file):

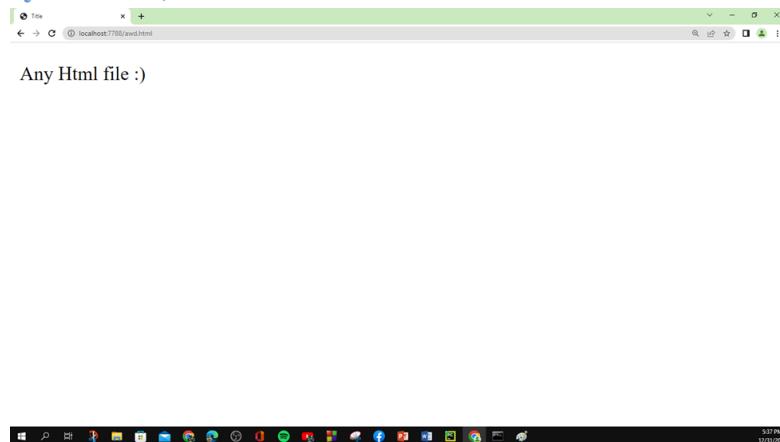
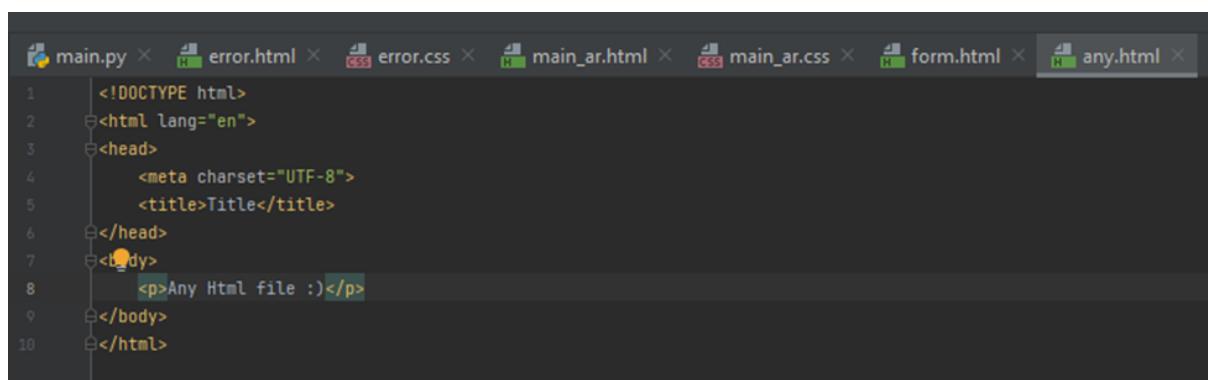


Figure 27: the any html file on website

a .html code:

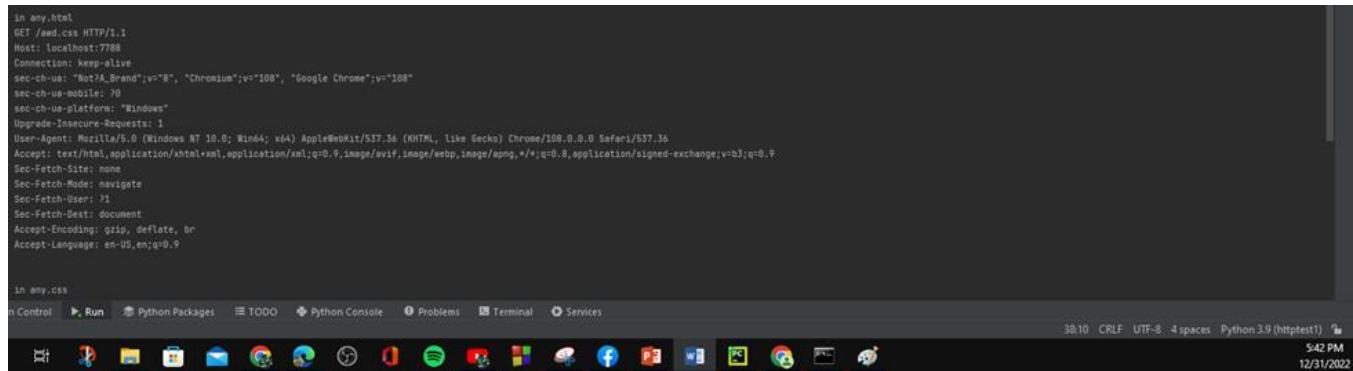


```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
    <p>Any Html file :)</p>
</body>
</html>
```

Figure 28: any.html file

- ❖ if the request is a **.css** file then the server sends the a.css file with Content-Type: text/css

on the command line(any css file):



```
in any.html
GET /awd.css HTTP/1.1
Host: localhost:7788
Connection: keep-alive
sec-ch-ua: "NotA Brand";v="8", "Chromium";v="108", "Google Chrome";v="108"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Windows"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/108.0.0.0 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
```

Figure 29: the request message for a css file

on the browser(any css file):

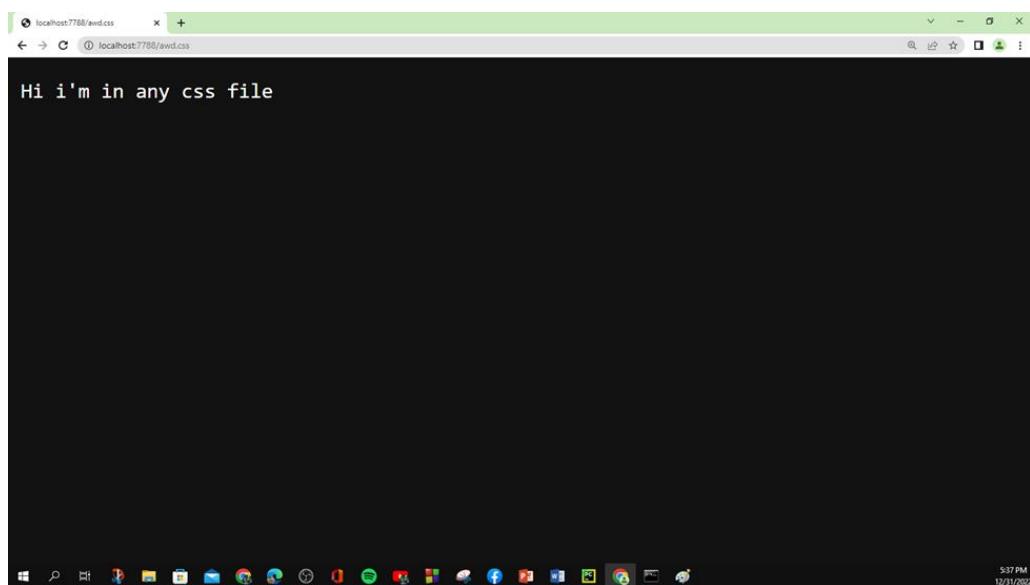
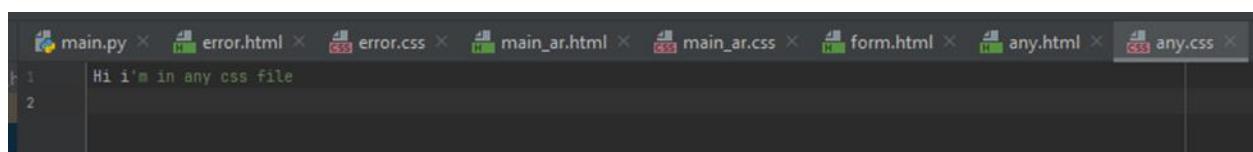


Figure 30: the css file on website

a .css file:

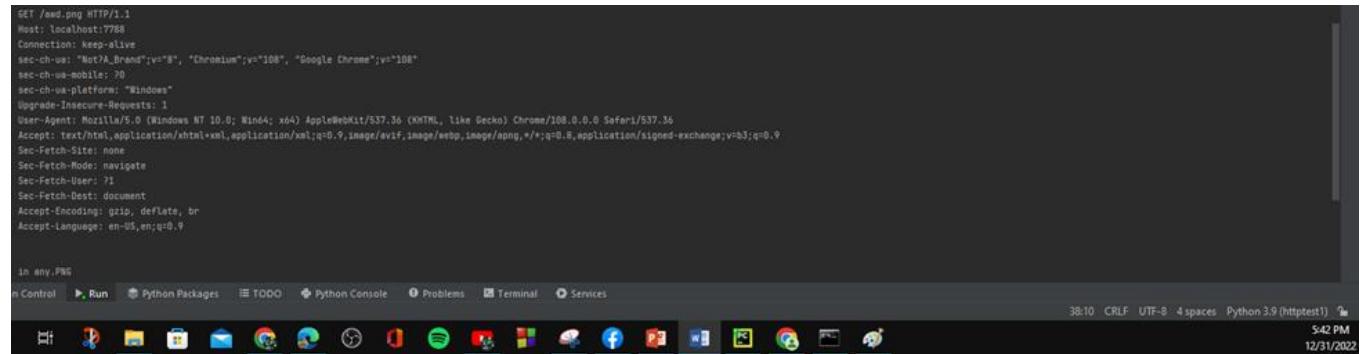


```
main.py × error.html × error.css × main_ar.html × main_ar.css × form.html × any.html × any.css ×
1 Hi i'm in any css file
2
```

Figure 31: .css file

- ❖ if the request is a .png then the server sends any.png image with Content-Type: image/png.

on the command line(any png file):



```
GET /awd.png HTTP/1.1
Host: localhost:7788
Connection: keep-alive
sec-ch-ua: "Not%2ABrand";v="8", "Chromium";v="108", "Google Chrome";v="108"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Windows"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/108.0.0.0 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
```

The screenshot shows a terminal window with the command `curl http://localhost:7788/awd.png` entered and its output displayed. The output includes the full HTTP request headers and the response body, which is a binary image file.

Figure 32: the request message for a png file

on the browser(any png file):

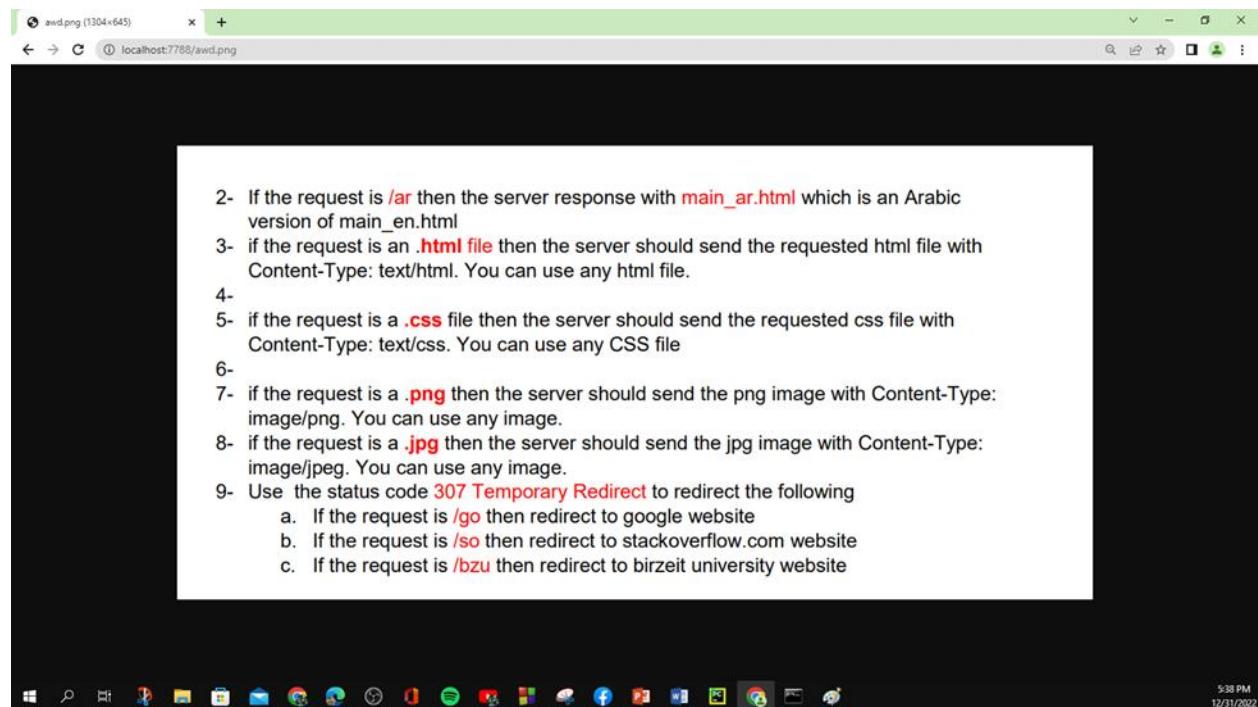


Figure 33: the png file on website

- ❖ if the request is a .jpg then the server sends any.jpg image with Content-Type: image/jpeg

on the command line(any jpg file):

```
in any.PNG
GET /swd.jpg HTTP/1.1
Host: localhost:7788
Connection: keep-alive
sec-ch-usr-id: "Not%4A_Brand";v="8", "Chromium";v="108", "Google Chrome";v="108"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Windows"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/108.0.0.0 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9

in any.JPG

in Control Run Python Packages TODO Python Console Problems Terminal Services
38:10 CRLF UTF-8 4 spaces Python 3.9 (https://) 543 PM
12/31/2022
```

Figure 34: the request message for a jpg file

on the browser(any jpg file):

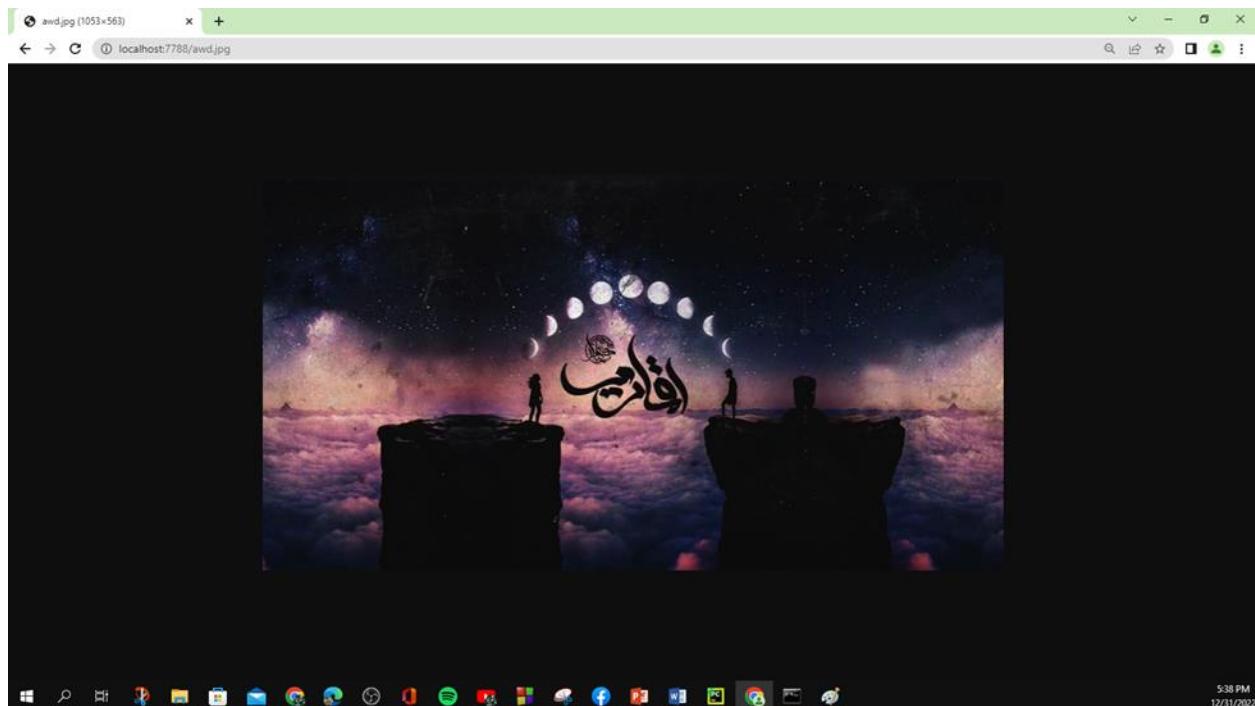


Figure 35:the jpg file on website

Google Website:

- ❖ If the request is /go then redirect to google website

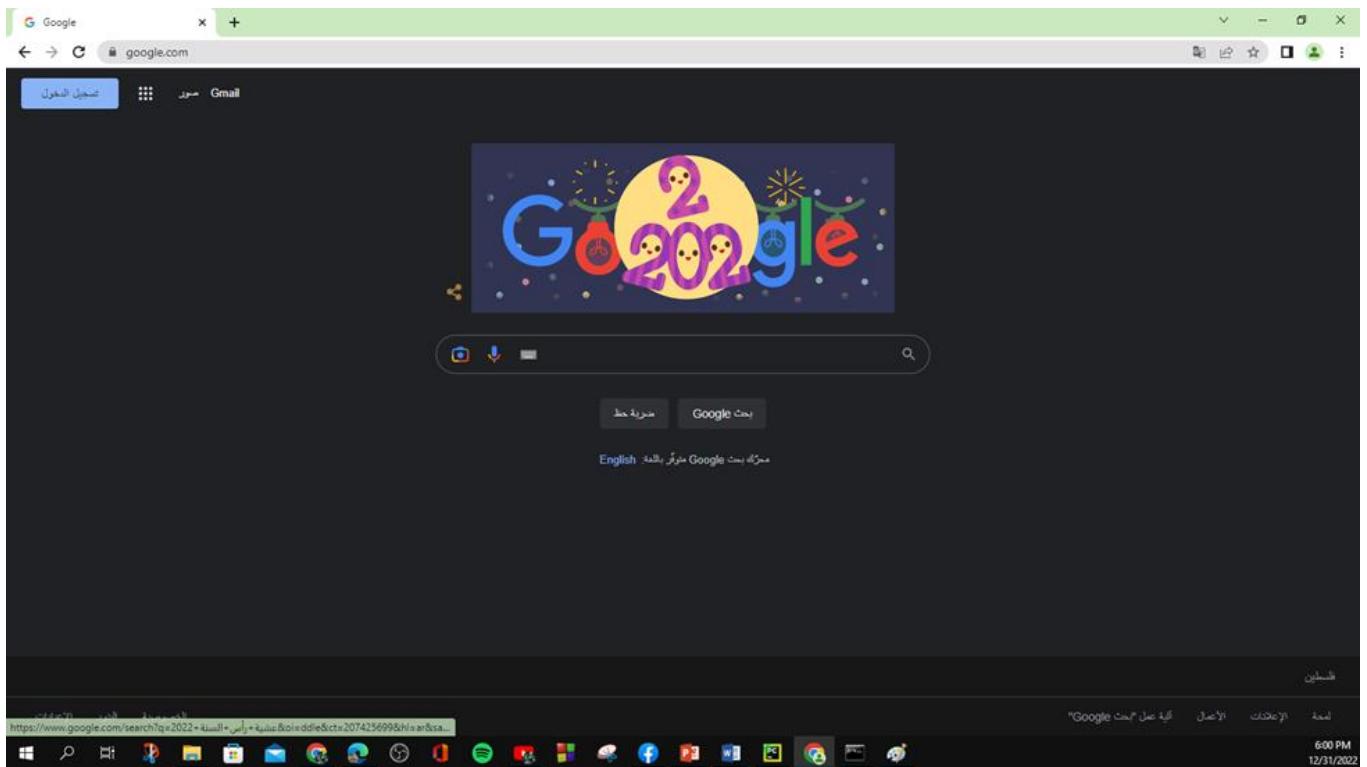


Figure 36: google website

```
GET /go HTTP/1.1
Host: localhost:7788
Connection: keep-alive
Cache-Control: max-age=0
sec-ch-ua: "Not%2ABrand";v="8", "Chromium";v="108", "Google Chrome";v="108"
sec-ch-ua-mobile: 0
sec-ch-ua-platform: "Windows"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/108.0.0.0 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
```

Figure 37: google website request message

Stackoverflow website:

- ❖ If the request is /so then redirect to stackoverflow.com website

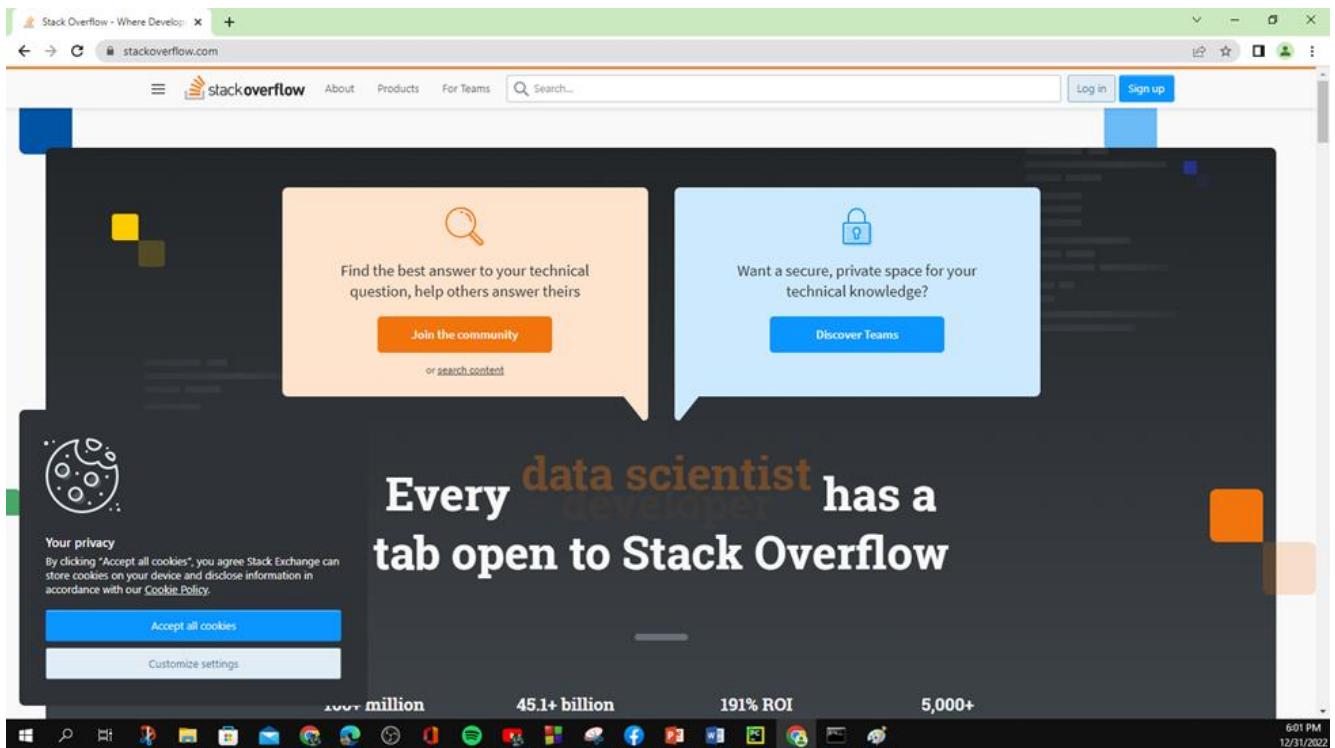


Figure 38: stackoverflow website

A screenshot of a terminal window titled 'main'. The code in the terminal is a Python script named 'main.py' which uses the 'httpx' library to send an HTTP request to 'localhost:7788'. The response shows a user agent string for Google Chrome version 108.0.5357.36. The terminal window is part of a PyCharm IDE interface, with other tabs like 'Run', 'Python Packages', and 'Terminal' visible. The status bar at the bottom right shows the date and time as '12/31/2022 6:02 PM'.

Figure 39: stackoverflow website request message

Birzeit university website

- ❖ If the request is **/bzu** then redirect to birzeit university website

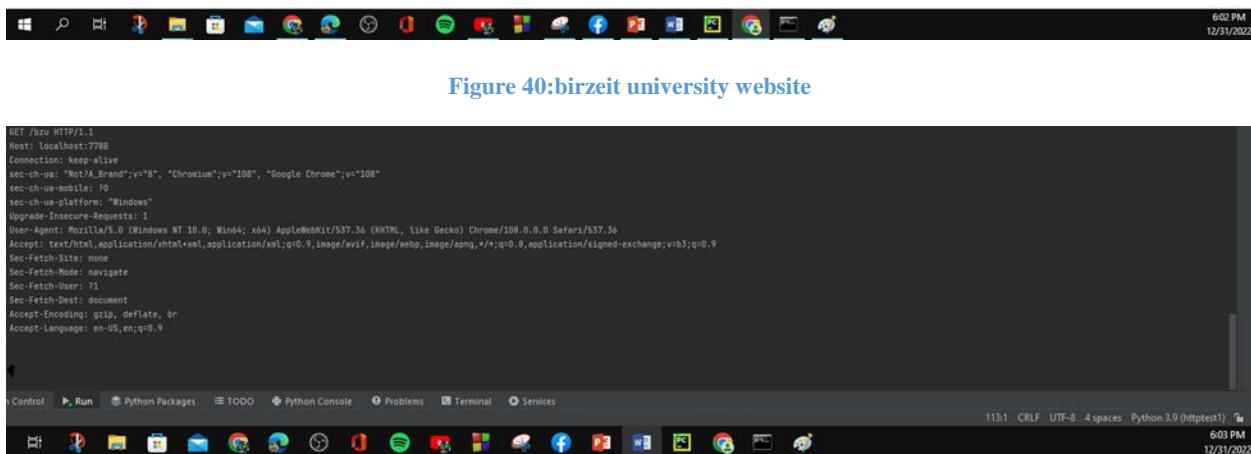


Figure 40:birzeit university website

Error Webpage

- ❖ If the request is wrong or the file doesn't exist the server returns a simple HTML Webpage with Content-Type text/html, that contains:
 - 1- "HTTP/1.1 404 Not Found" in the response status
 - 2- "Error" in the title
 - 3- "The file is not found" in the body in red
 - 4- Our names and IDs in Bold
 - 5- The IP and port number of the client

Here is an example in the figure below

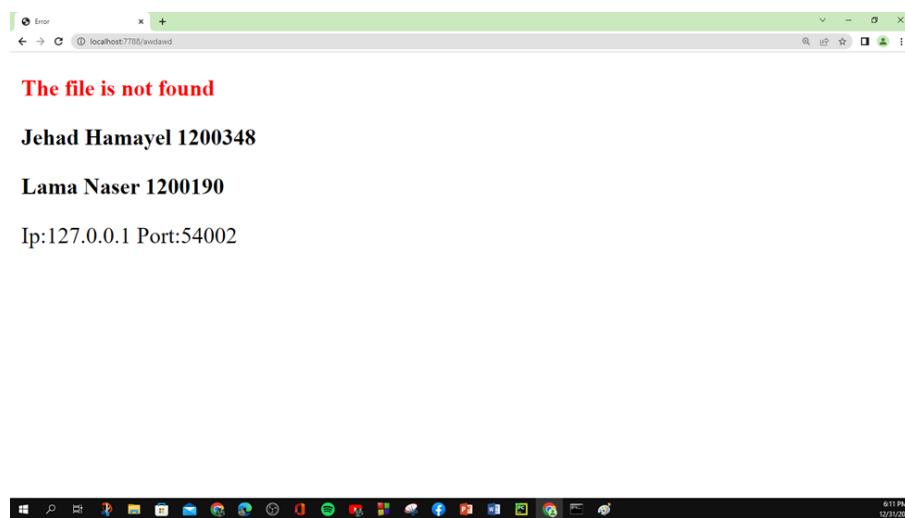


Figure 42: error webpage

On command line(Error Webpage):

A screenshot of a terminal window titled 'Windows Terminal'. The command entered is 'python C:\Users\hp\PycharmProjects\httpstest\venv\scripts\python.exe C:\Users\hp\PycharmProjects\httpstest\main.py'. The output shows the server's response to a request for 'error.html':

```
The server is ready to receive
GET /index.html HTTP/1.1
Host: localhost:7788
Connection: keep-alive
Cache-Control: max-age=0
sec-ch-ua: "Not(A Brand";v="8", "Chrome";v="108", "Google Chrome";v="108"
sec-ch-ua-platform: "Windows"
Upgrade-Insecure-Request: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/108.0.0.0 Safari/108.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9

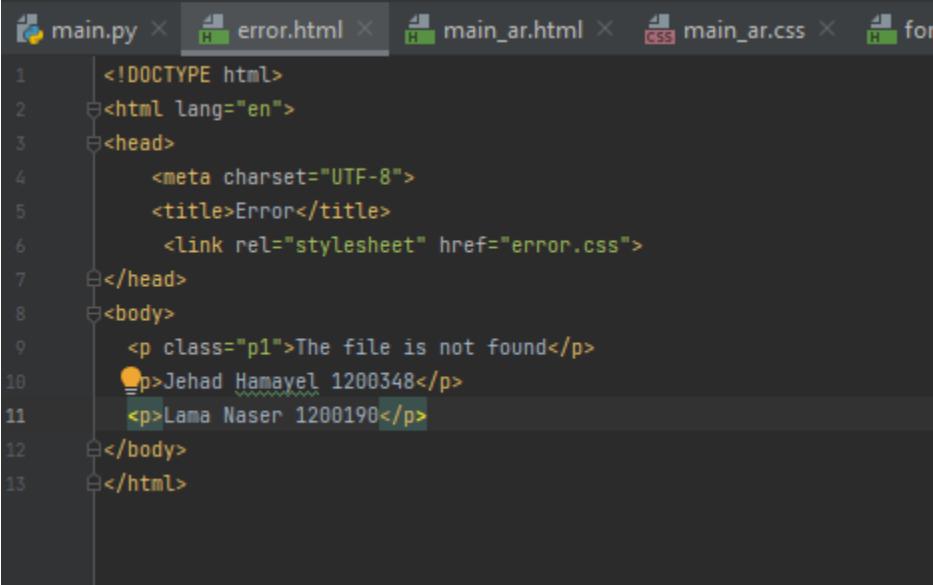
in error.html
GET /error.html HTTP/1.1
Host: localhost:7788
Connection: keep-alive
Cache-Control: max-age=0
sec-ch-ua: "Not(A Brand";v="8", "Chrome";v="108", "Google Chrome";v="108"
sec-ch-ua-platform: "Windows"
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/108.0.0.0 Safari/108.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: no-cors
Sec-Fetch-Dest: style
Referer: http://localhost:7788/index.html
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9

in error.css
```

At the bottom of the terminal window, the status bar shows '2028 CR LF UTF-8 4 spaces Python 3.9 (httpstest) 6:12 PM 12/31/2022'.

Figure 43: error webpage request message

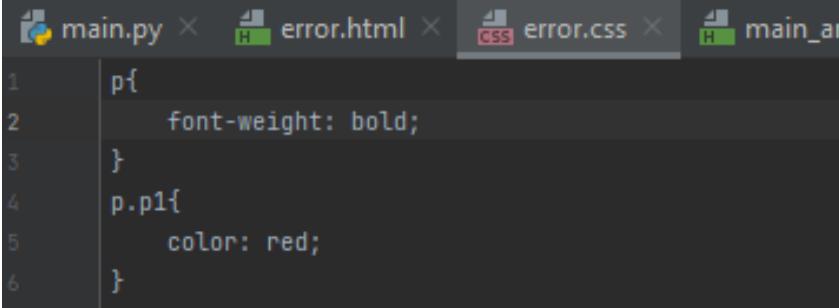
error.html code:



```
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="UTF-8">
5     <title>Error</title>
6     <link rel="stylesheet" href="error.css">
7   </head>
8   <body>
9     <p class="p1">The file is not found</p>
10    <p>Jehad Hamayel 1200348</p>
11    <p>Lama Naser 1200190</p>
12  </body>
13 </html>
```

Figure 44: error.html code

error.css code:



```
1 p{
2   font-weight: bold;
3 }
4 .p1{
5   color: red;
6 }
```

Figure 45: error.css code

Testing from another device

We tested the project from a phone, everything was going well.

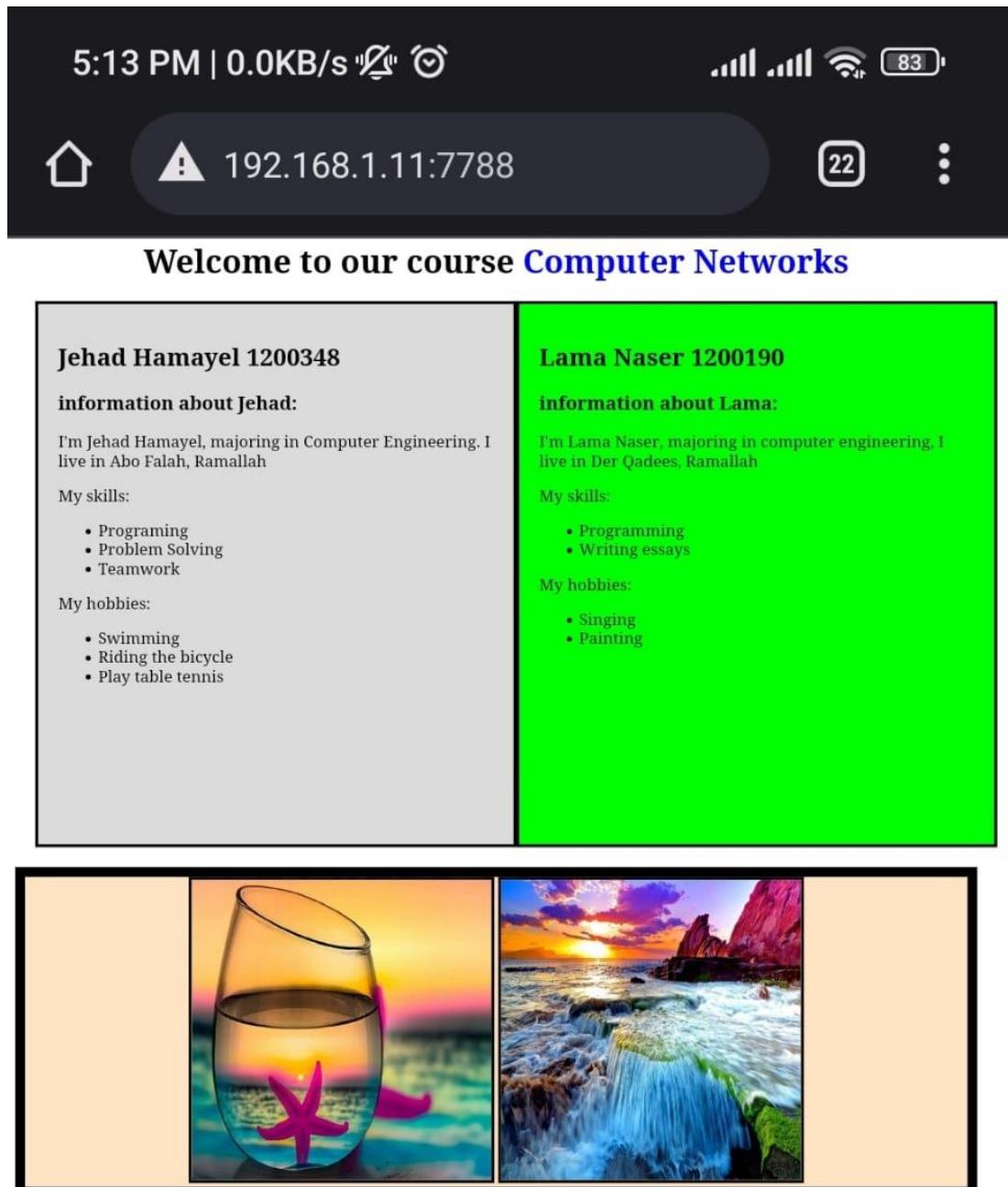
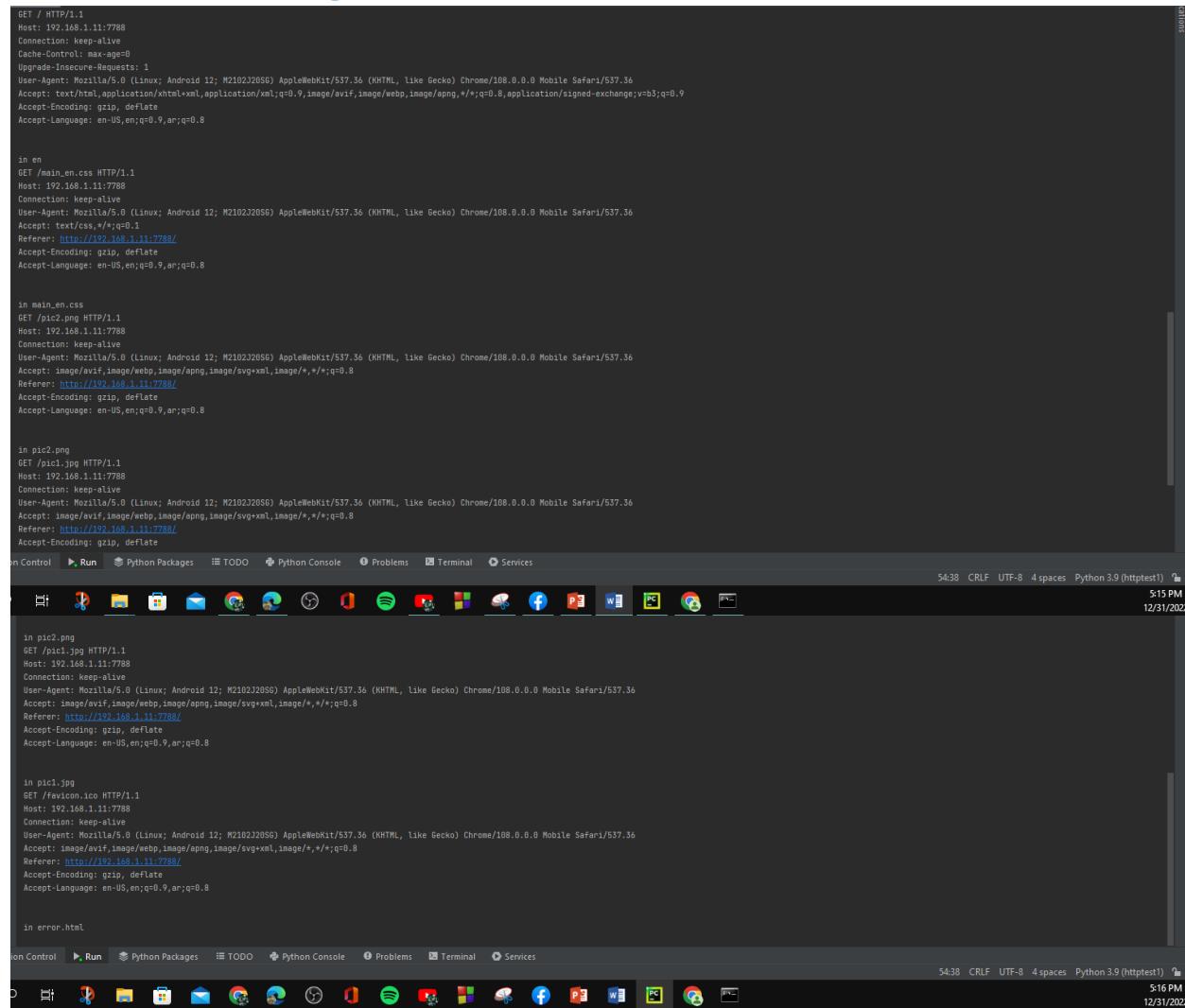


Figure 46:the test on the phone

On command line Testing from another device:



```
GET / HTTP/1.1
Host: 192.168.1.11:7788
Connection: keep-alive
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Linux; Android 12; M2102320SG) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/108.0.0.0 Mobile Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9,ar;q=0.8

in en
GET /main_en.css HTTP/1.1
Host: 192.168.1.11:7788
Connection: keep-alive
User-Agent: Mozilla/5.0 (Linux; Android 12; M2102320SG) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/108.0.0.0 Mobile Safari/537.36
Accept: text/css,*/*;q=0.1
Referer: http://192.168.1.11:7788/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9,ar;q=0.8

in main_en.css
GET /pic2.png HTTP/1.1
Host: 192.168.1.11:7788
Connection: keep-alive
User-Agent: Mozilla/5.0 (Linux; Android 12; M2102320SG) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/108.0.0.0 Mobile Safari/537.36
Accept: image/avif,image/webp,image/apng,image/svg+xml,image/*,*/*;q=0.8
Referer: http://192.168.1.11:7788/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9,ar;q=0.8

in pic2.png
GET /pic1.jpg HTTP/1.1
Host: 192.168.1.11:7788
Connection: keep-alive
User-Agent: Mozilla/5.0 (Linux; Android 12; M2102320SG) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/108.0.0.0 Mobile Safari/537.36
Accept: image/avif,image/webp,image/apng,image/svg+xml,image/*,*/*;q=0.8
Referer: http://192.168.1.11:7788/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9,ar;q=0.8

in pic1.jpg
GET /favicon.ico HTTP/1.1
Host: 192.168.1.11:7788
Connection: keep-alive
User-Agent: Mozilla/5.0 (Linux; Android 12; M2102320SG) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/108.0.0.0 Mobile Safari/537.36
Accept: image/avif,image/webp,image/apng,image/svg+xml,image/*,*/*;q=0.8
Referer: http://192.168.1.11:7788/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9,ar;q=0.8

in error.html
```

Figure 47: the request message

Conclusion

To sum up in this project, processes on different hosts communicate through network applications and we tried in this project to write simple applications in python to understand how connection happens and which type of protocols can be more appropriate for different applications