Department of Electrical and Computer Engineering

Computer Networks
ENCS3320

Project1

Prepared by:

Mahmoud Awad **Student NO.** 1212677
Instructor: Dr. Abdalkarim Awad
Section: One

Abd AlRahman Shaheen **Student NO.** 1211753
Instructor: Dr. Mohammad Jubran
Section: Two

Omar Daghlas **Student NO.** 1222500
Instructor: Dr. Imad Tartir
Section: Four

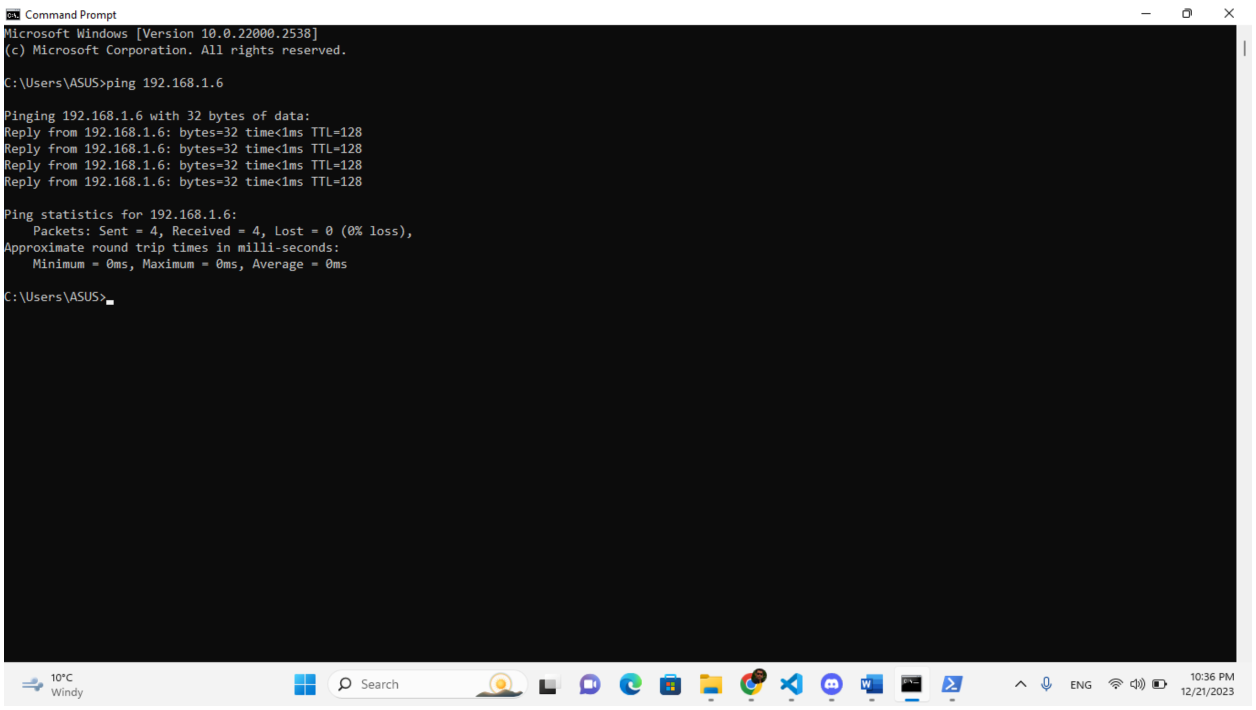
Date: 22_12_2023

Part1:

1. In your own words, what are ping, tracert, nslookup, and telnet (write one sentence for each one)
   - Ping: ping is a small packet of data sent from one computer to another to check if the recipient is reachable and how long it takes for the data to travel back.
   - Tracert: tracert is a network diagnostic tool used to trace the route that packets take from a source device to a destination device or server on an IP network.
   - Nslookup: nslookup is a command-line tool used for querying Domain Name System (DNS) servers to obtain domain name or IP address mapping, DNS records, and other DNS-related information.
   - Telnet: telnet is a network protocol and a command-line tool that allows for bidirectional text communication between two devices over a network.

2. Make sure that your computer is connected to the internet and then run the following commands:

   1. Ping a device in the same network, e.g. from a laptop to a smartphone

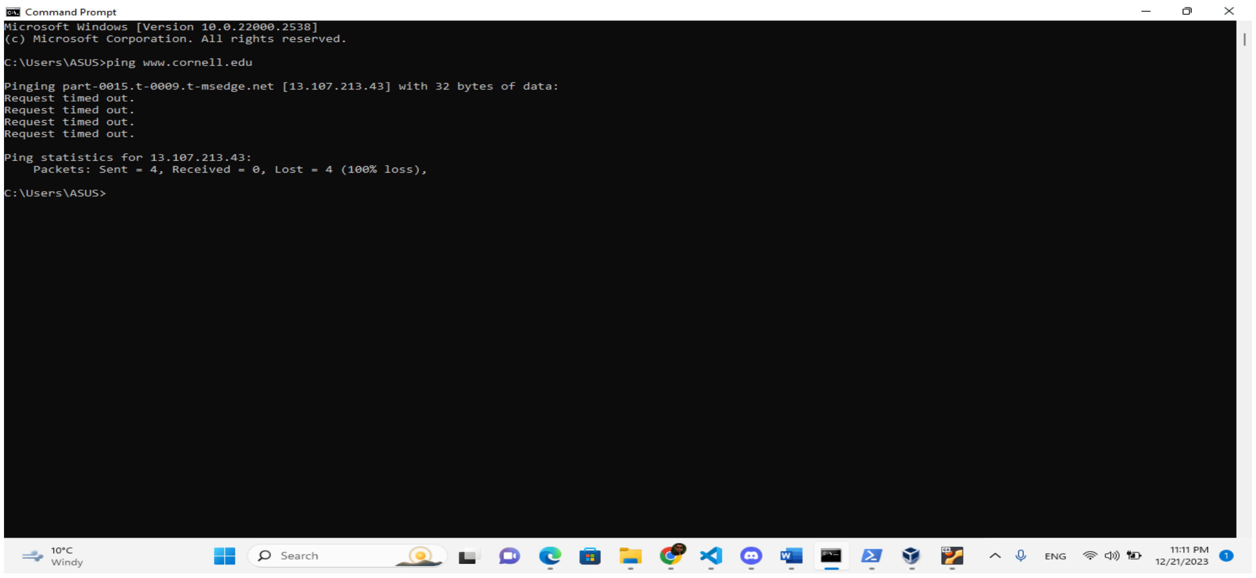


```
Microsoft Windows [Version 10.0.22000.2538]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ASUS>ping 192.168.1.6

Pinging 192.168.1.6 with 32 bytes of data:
Reply from 192.168.1.6: bytes=32 time<1ms TTL=128
Reply from 192.168.1.6: bytes=32 time<1ms TTL=128
Reply from 192.168.1.6: bytes=32 time<1ms TTL=128
Reply from 192.168.1.6: bytes=32 time<1ms TTL=128

Ping statistics for 192.168.1.6:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms

C:\Users\ASUS>
```

Explanation of the output:
This command displays the results of each ICMP echo request. For each packet sent (four in this case), the output shows a reply from the specified IP address (192.168.1.6) along with the size of the data packet (32 bytes), the round trip time (time<1ms, indicating a response time of less than 1 millisecond), and the Time to Live (TTL) value, which is 128 in this case.

2. ping www.cornell.edu

    1) Using Command Prompt:



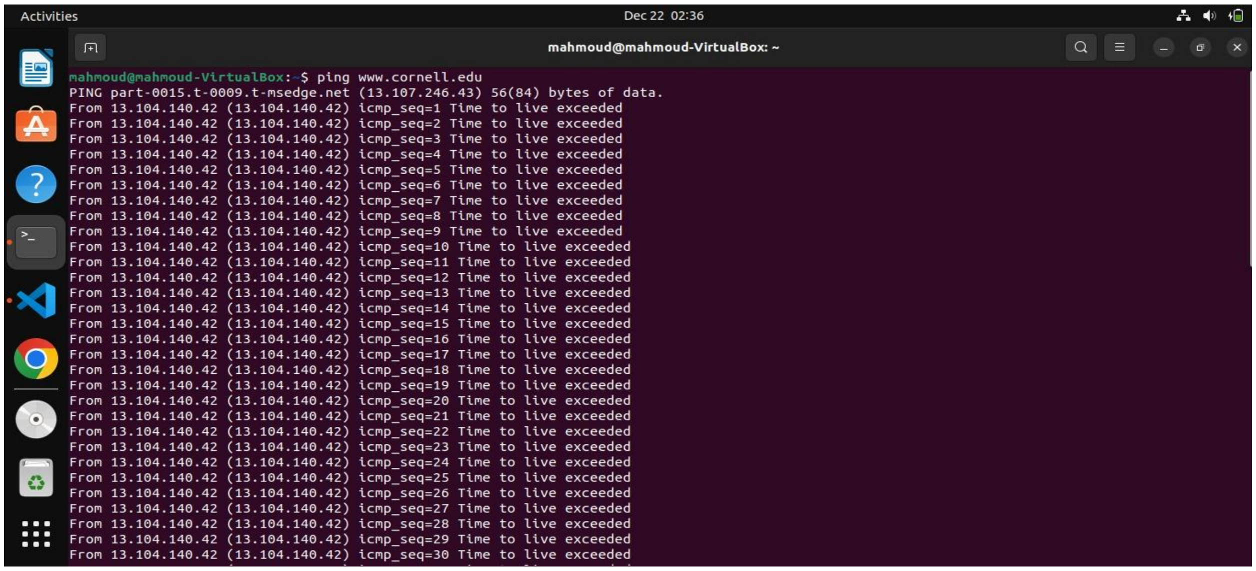
```
Microsoft Windows [Version 10.0.22000.2538]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ASUS>ping www.cornell.edu

Pinging part-0015.t-0009.t-msedge.net [13.107.213.43] with 32 bytes of data:
Request timed out.
Request timed out.
Request timed out.
Request timed out.

Ping statistics for 13.107.213.43:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),

C:\Users\ASUS>
```

Explanation of the output:

The output shows that the ping attempts to the IP address 13.107.213.43 (associated with the domain www.cornell.edu) resulted in four consecutive timeouts. Each "Request timed out" message indicates that the ping packet did not receive a response within the expected time frame.

---

    2) Using Linux:

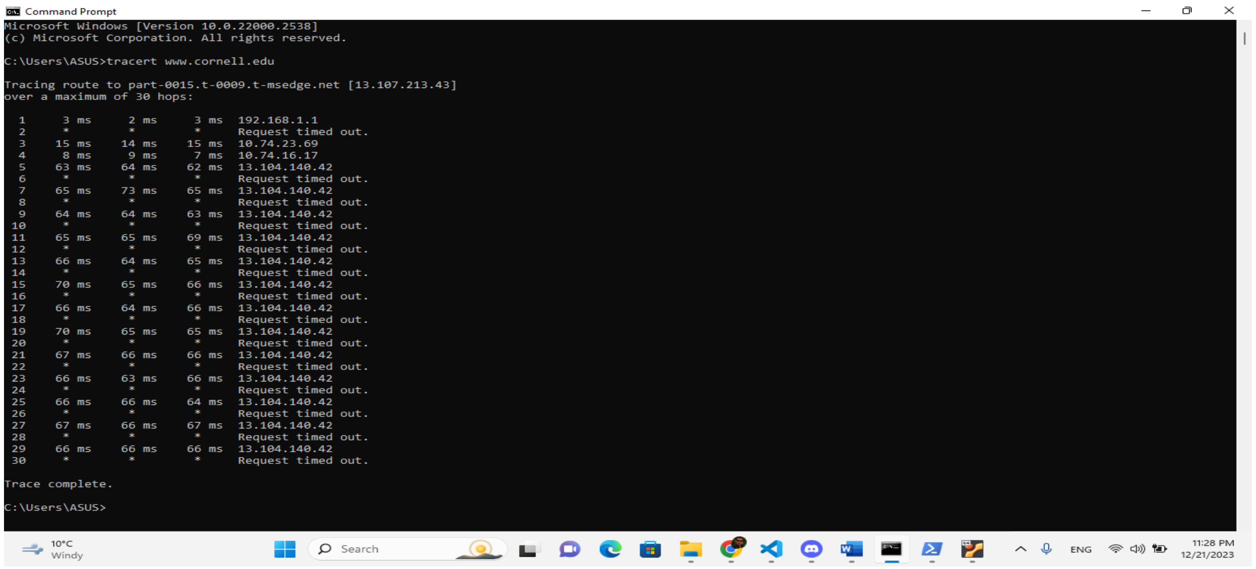


```
mahmoud@mahmoud-VirtualBox:~$ ping www.cornell.edu
PING part-0015.t-0009.t-msedge.net (13.107.246.43) 56(84) bytes of data.
From 13.104.140.42 (13.104.140.42) icmp_seq=1 Time to live exceeded
From 13.104.140.42 (13.104.140.42) icmp_seq=2 Time to live exceeded
From 13.104.140.42 (13.104.140.42) icmp_seq=3 Time to live exceeded
From 13.104.140.42 (13.104.140.42) icmp_seq=4 Time to live exceeded
From 13.104.140.42 (13.104.140.42) icmp_seq=5 Time to live exceeded
From 13.104.140.42 (13.104.140.42) icmp_seq=6 Time to live exceeded
From 13.104.140.42 (13.104.140.42) icmp_seq=7 Time to live exceeded
From 13.104.140.42 (13.104.140.42) icmp_seq=8 Time to live exceeded
From 13.104.140.42 (13.104.140.42) icmp_seq=9 Time to live exceeded
From 13.104.140.42 (13.104.140.42) icmp_seq=10 Time to live exceeded
From 13.104.140.42 (13.104.140.42) icmp_seq=11 Time to live exceeded
From 13.104.140.42 (13.104.140.42) icmp_seq=12 Time to live exceeded
From 13.104.140.42 (13.104.140.42) icmp_seq=13 Time to live exceeded
From 13.104.140.42 (13.104.140.42) icmp_seq=14 Time to live exceeded
From 13.104.140.42 (13.104.140.42) icmp_seq=15 Time to live exceeded
From 13.104.140.42 (13.104.140.42) icmp_seq=16 Time to live exceeded
From 13.104.140.42 (13.104.140.42) icmp_seq=17 Time to live exceeded
From 13.104.140.42 (13.104.140.42) icmp_seq=18 Time to live exceeded
From 13.104.140.42 (13.104.140.42) icmp_seq=19 Time to live exceeded
From 13.104.140.42 (13.104.140.42) icmp_seq=20 Time to live exceeded
From 13.104.140.42 (13.104.140.42) icmp_seq=21 Time to live exceeded
From 13.104.140.42 (13.104.140.42) icmp_seq=22 Time to live exceeded
From 13.104.140.42 (13.104.140.42) icmp_seq=23 Time to live exceeded
From 13.104.140.42 (13.104.140.42) icmp_seq=24 Time to live exceeded
From 13.104.140.42 (13.104.140.42) icmp_seq=25 Time to live exceeded
From 13.104.140.42 (13.104.140.42) icmp_seq=26 Time to live exceeded
From 13.104.140.42 (13.104.140.42) icmp_seq=27 Time to live exceeded
From 13.104.140.42 (13.104.140.42) icmp_seq=28 Time to live exceeded
From 13.104.140.42 (13.104.140.42) icmp_seq=29 Time to live exceeded
From 13.104.140.42 (13.104.140.42) icmp_seq=30 Time to live exceeded
```

Explanation of the output:

The subsequent lines show responses received from the IP address 13.104.140.42, indicating that the Time to Live (TTL) for the packets has been exceeded. This typically occurs when a packet traverses routers, and the TTL value reaches zero, preventing it from reaching the destination. The router then sends an ICMP Time to Live Exceeded message back to the source.

3. tracert www.cornell.edu


Command Prompt

```
Microsoft Windows [Version 10.0.22000.2538]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ASUS>tracert www.cornell.edu

Tracing route to part-0015.t-0009.t-msedge.net [13.107.213.43]
over a maximum of 30 hops:

  1     3 ms     2 ms     3 ms  192.168.1.1
  2     *        *        *     Request timed out.
  3    15 ms    14 ms    15 ms  10.74.23.69
  4     8 ms     9 ms     7 ms  10.74.16.17
  5    63 ms    64 ms    62 ms  13.104.140.42
  6     *        *        *     Request timed out.
  7    65 ms    73 ms    65 ms  13.104.140.42
  8     *        *        *     Request timed out.
  9    64 ms    64 ms    63 ms  13.104.140.42
 10     *        *        *     Request timed out.
 11    65 ms    65 ms    69 ms  13.104.140.42
 12     *        *        *     Request timed out.
 13    66 ms    64 ms    65 ms  13.104.140.42
 14     *        *        *     Request timed out.
 15    70 ms    65 ms    66 ms  13.104.140.42
 16     *        *        *     Request timed out.
 17    66 ms    64 ms    66 ms  13.104.140.42
 18     *        *        *     Request timed out.
 19    70 ms    65 ms    65 ms  13.104.140.42
 20     *        *        *     Request timed out.
 21    67 ms    66 ms    66 ms  13.104.140.42
 22     *        *        *     Request timed out.
 23    66 ms    63 ms    66 ms  13.104.140.42
 24     *        *        *     Request timed out.
 25    66 ms    66 ms    64 ms  13.104.140.42
 26     *        *        *     Request timed out.
 27    67 ms    66 ms    67 ms  13.104.140.42
 28     *        *        *     Request timed out.
 29    66 ms    66 ms    66 ms  13.104.140.42
 30     *        *        *     Request timed out.

Trace complete.

C:\Users\ASUS>
```

Explanation of the output:

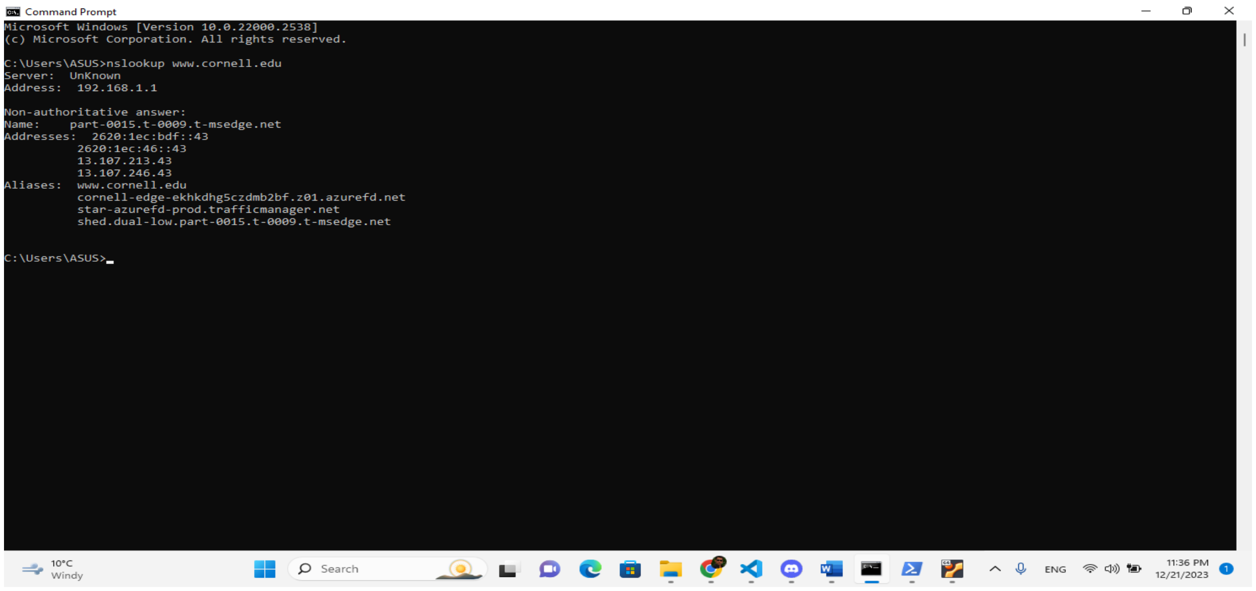
The first step (hop) is quick and takes about 3 milliseconds to reach a place labeled as 192.168.1.1, which might be like the entrance gate to the internet.

The second step didn't respond, and we don't know what happened there.

The third step took about 15 milliseconds to reach another place with the address 10.74.23.69.

---

4. nslookup www.cornell.edu


Command Prompt

```
Microsoft Windows [Version 10.0.22000.2538]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ASUS>nslookup www.cornell.edu
Server:  UnKnown
Address:  192.168.1.1

Non-authoritative answer:
Name:    part-0015.t-0009.t-msedge.net
Addresses:  2620:1ec:bdf::43
          2620:1ec:46::43
          13.107.213.43
          13.107.246.43
Aliases:  www.cornell.edu
          cornell-edge-ekhkdhg5czdmb2bf.z01.azurefd.net
          star-azurefd-prod.trafficmanager.net
          shed.dual-low.part-0015.t-0009.t-msedge.net


C:\Users\ASUS>_
```

Explanation of the output:

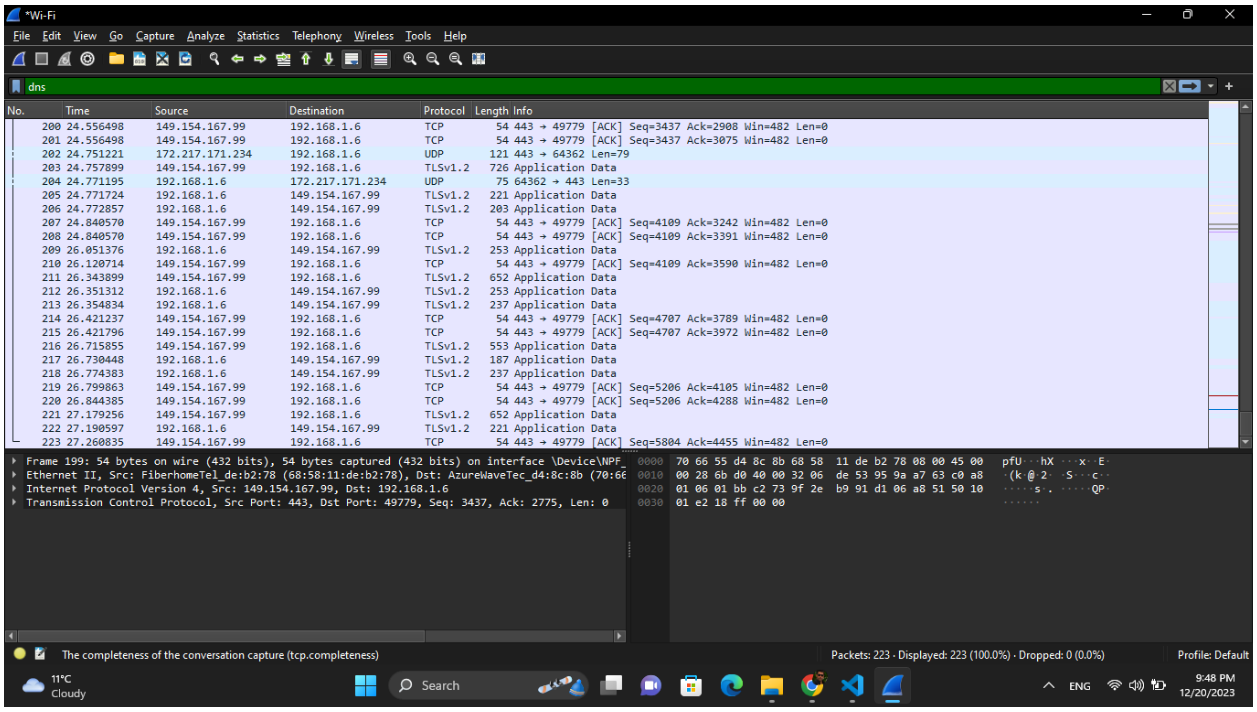
The website www.cornell.edu has other names like part-0015.t-0009.t-msedge.net.

It can be reached using different numbers (addresses) like 2620:1ec:bdf::43, 2620:1ec:46::43, 13.107.213.43, and 13.107.246.43.

The website has nicknames or aliases like www.cornell.edu.

It's associated with other names like cornell-edge-ekhkdhg5czdmb2bf.z01.azurefd.net and star-azurefd-prod.trafficmanager.net.

3. Use wireshark to capture some DNS messages.


*Wi-Fi
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
dns

```
No.     Time       Source           Destination      Protocol  Length Info
    200 24.556498  149.154.167.99   192.168.1.6      TCP         54 443 → 49779 [ACK] Seq=3437 Ack=2908 Win=482 Len=0
    201 24.556498  149.154.167.99   192.168.1.6      TCP         54 443 → 49779 [ACK] Seq=3437 Ack=3075 Win=482 Len=0
    202 24.751221  172.217.171.234  192.168.1.6      UDP        121 443 → 64362 Len=79
    203 24.757899  149.154.167.99   192.168.1.6      TLSv1.2    726 Application Data
    204 24.771195  192.168.1.6      172.217.171.234  UDP         75 64362 → 443 Len=33
    205 24.771724  192.168.1.6      149.154.167.99   TLSv1.2    221 Application Data
    206 24.772857  192.168.1.6      149.154.167.99   TLSv1.2    203 Application Data
    207 24.840570  149.154.167.99   192.168.1.6      TCP         54 443 → 49779 [ACK] Seq=4109 Ack=3242 Win=482 Len=0
    208 24.840570  149.154.167.99   192.168.1.6      TCP         54 443 → 49779 [ACK] Seq=4109 Ack=3391 Win=482 Len=0
    209 26.051376  192.168.1.6      149.154.167.99   TLSv1.2    253 Application Data
    210 26.120714  149.154.167.99   192.168.1.6      TCP         54 443 → 49779 [ACK] Seq=4109 Ack=3590 Win=482 Len=0
    211 26.343899  149.154.167.99   192.168.1.6      TLSv1.2    652 Application Data
    212 26.351312  192.168.1.6      149.154.167.99   TLSv1.2    253 Application Data
    213 26.354834  192.168.1.6      149.154.167.99   TLSv1.2    237 Application Data
    214 26.421237  149.154.167.99   192.168.1.6      TCP         54 443 → 49779 [ACK] Seq=4707 Ack=3789 Win=482 Len=0
    215 26.421796  149.154.167.99   192.168.1.6      TCP         54 443 → 49779 [ACK] Seq=4707 Ack=3972 Win=482 Len=0
    216 26.715855  149.154.167.99   192.168.1.6      TLSv1.2    553 Application Data
    217 26.730448  192.168.1.6      149.154.167.99   TLSv1.2    187 Application Data
    218 26.774383  192.168.1.6      149.154.167.99   TLSv1.2    237 Application Data
    219 26.799863  149.154.167.99   192.168.1.6      TCP         54 443 → 49779 [ACK] Seq=5206 Ack=4105 Win=482 Len=0
    220 26.844385  149.154.167.99   192.168.1.6      TCP         54 443 → 49779 [ACK] Seq=5206 Ack=4288 Win=482 Len=0
    221 27.179256  149.154.167.99   192.168.1.6      TLSv1.2    652 Application Data
    222 27.190597  192.168.1.6      149.154.167.99   TLSv1.2    221 Application Data
    223 27.260835  149.154.167.99   192.168.1.6      TCP         54 443 → 49779 [ACK] Seq=5804 Ack=4455 Win=482 Len=0

Frame 199: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface \Device\NPF_
Ethernet II, Src: FiberhomeTel_de:b2:78 (68:58:11:de:b2:78), Dst: AzureWaveTec_d4:8c:8b (70:66
Internet Protocol Version 4, Src: 149.154.167.99, Dst: 192.168.1.6
Transmission Control Protocol, Src Port: 443, Dst Port: 49779, Seq: 3437, Ack: 2775, Len: 0

0000  70 66 55 d4 8c 8b 68 58  11 de b2 78 08 00 45 00   pfU···hX ···x··E·
0010  00 28 6b d0 40 00 32 06  de 53 95 9a a7 63 c0 a8   ·(k·@·2· ·S···c··
0020  01 06 01 bb c2 73 9f 2e  b9 91 d1 06 a8 51 50 10   ·····s·. ······QP·
0030  01 e2 18 ff 00 00                                  ······

The completeness of the conversation capture (tcp.completeness)
Packets: 223 · Displayed: 223 (100.0%) · Dropped: 0 (0.0%)    Profile: Default
```

Explanation of the output:
Wireshark Interface: Displays captured network packets with details like number, time, source, destination, protocol, length, and information.
Color-coded rows represent individual packets, allowing for easy identification and analysis.
TCP Details: The bottom section shows details of a specific Transmission Control Protocol (TCP) packet.
Taskbar Icons: Includes Wi-Fi signal strength and battery level indicators.

➢ From the ping results, do you think the response you have got is from USA? Explain your answer briefly.

- When you use the "ping" command to talk to a server, you're checking if it responds back. The response time is shown, but it doesn't tell you the exact route or where the server is. It just says if the server is reachable or not.
- If you want to know the path your data takes to reach a destination, you can use tools like "traceroute" or "tracert" on your computer. These tools reveal each stop (router) along the way, helping you see the route and possibly the location of each middle server.
- depending on the previous run of tracert:
  As you notice the time diffirance on 2 specific points:
  from **(7ms)** to **(62ms)**
  this is where the response transported through the ocean and went to **America**.
  hence, the response we got is from **USA.**

Part2:

❖ Using socket programming, implement TCP client and server applications in **python**. The server should listen on port 9955. The server waits for a message from a client. If the message is with one of the students ID **(1211773, 1222500, 1212677)**. the sever should do the following:

1. display a message on the server side that the OS will lock screen after 10 seconds
2. send a message to the client that the sever will lock screen after 10 seconds
3. then wait 10 seconds
4. then call a function lock the screen of the operating system windows.

Code:

---

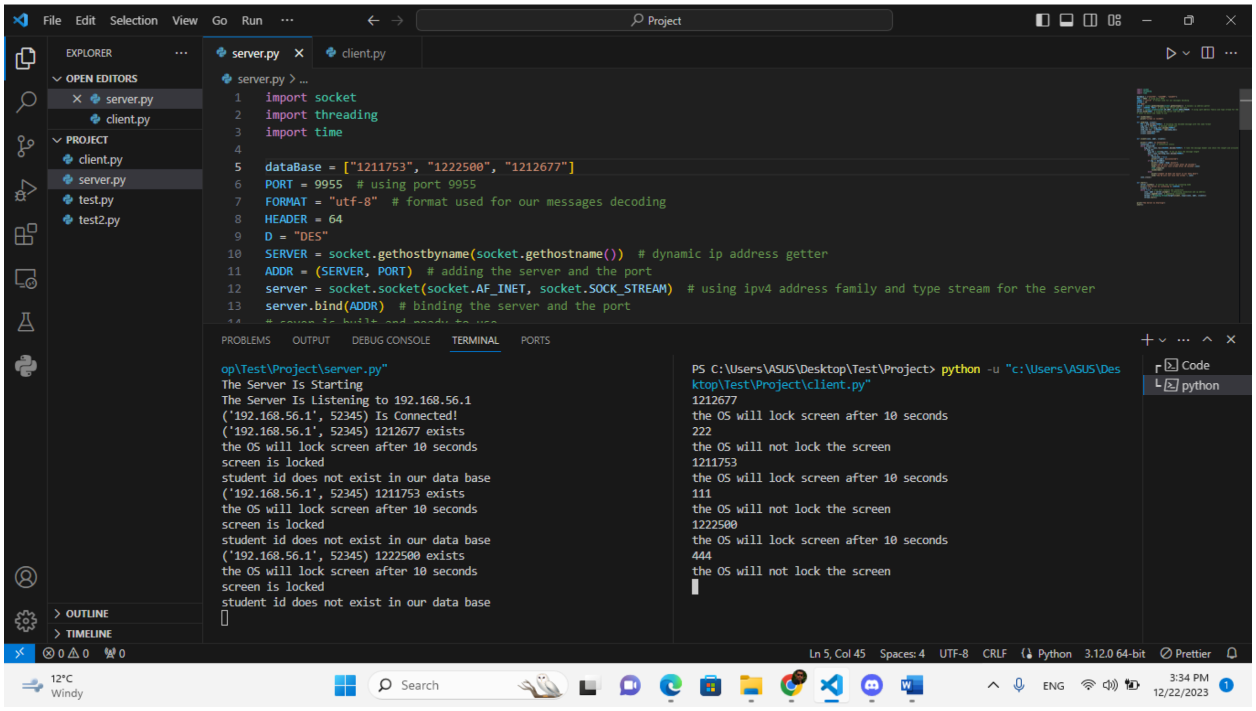
1. Server Code:

```python
import socket
import threading
import time

dataBase = ["1211753", "1222500", "1212677"]
PORT = 9955  # using port 9955
FORMAT = "utf-8"  # format used for our messages decoding
HEADER = 64
D = "DES"
SERVER = socket.gethostbyname(socket.gethostname())  # dynamic ip address getter
ADDR = (SERVER, PORT)  # adding the server and the port
server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)  # using ipv4 address family and type stream for the server
server.bind(ADDR)  # binding the server and the port
# sever is built and ready to use

def lockScreen():
    print("screen is locked")

def send(msg, client):
    MSG = msg.encode(FORMAT)  # encoding the decoded message with the same format
    msg_len = len(MSG)  # message length got
    send_len = str(msg_len).encode(FORMAT)
    send_len +=b' '*(HEADER - len(send_len))
    client.send(send_len)
    client.send(MSG)

def client(conn, addr, clients):

    print(f"{addr} Is Connected!")
    connected = True  # connection status
    while connected:
        msg_len = conn.recv(HEADER).decode(FORMAT)  # take the message header and check the length and allocate 64
        if msg_len:
            msg_len = int(msg_len)  # now we have the message length
            msg = conn.recv(msg_len).decode(FORMAT)
            if msg == D:
                connected = False
                print(f"{addr} disconnected")
            if msg in dataBase:
                print(f"{addr} {msg} exists")
                print("the OS will lock screen after 10 seconds")
                send("the OS will lock screen after 10 seconds",conn)
                time.sleep(10)
                lockScreen()
            else:
                print("student id does not exist in our data base")
                send("the OS will not lock the screen ", conn)
    conn.close()

def start():
    server.listen()  # setting the server to listening mode
    print(f"The Server Is Listening to {SERVER} ")
    clients = []
    while True:  # this will receive any connection
        conn, addr = server.accept()  # obtaining connection and ip address
        clients.append(conn)  # appending connections
        thread = threading.Thread(target=client, args=(conn, addr, clients))
        thread.start()

print("The Server Is Starting")
start()
```

2. Client Code:

```python
import socket
import threading

PORT = 9955 # using port 9955
FORMAT = "utf-8"  # format used for our messages decoding
D = "DES"
HEADER = 64
SERVER = "192.168.56.1"  # my ip address
ADDR = (SERVER, PORT)  # adding the server and the port
client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)  # using ipv4 address family and type stream for the client
client.connect(ADDR)

def send(msg):
    MSG = msg.encode(FORMAT)  # encoding the decoded message with the same format
    msg_len = len(MSG)  # message length got
    send_len = str(msg_len).encode(FORMAT)
    send_len +=b' '*(HEADER - len(send_len))
    client.send(send_len)
    client.send(MSG)

def read():
    msg_len = client.recv(HEADER).decode(FORMAT)  # take the message header and check the length and allocate 64
    if msg_len:
        msg_len = int(msg_len)  # now we have the message length
        msg = client.recv(msg_len).decode(FORMAT)
        print(f"{msg} ")

x = ''
while 1:
    x = input()
    if x!='DES':
         send(x)
    thread = threading.Thread(target=read)
    thread.start()
send(D)
```

Result:



```python
import socket
import threading
import time

dataBase = ["1211753", "1222500", "1212677"]
PORT = 9955  # using port 9955
FORMAT = "utf-8"  # format used for our messages decoding
HEADER = 64
D = "DES"
SERVER = socket.gethostbyname(socket.gethostname())  # dynamic ip address getter
ADDR = (SERVER, PORT)  # adding the server and the port
server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)  # using ipv4 address family and type stream for the server
server.bind(ADDR)  # binding the server and the port
```

```
op\Test\Project\server.py"
The Server Is Starting
The Server Is Listening to 192.168.56.1
('192.168.56.1', 52345) Is Connected!
('192.168.56.1', 52345) 1212677 exists
the OS will lock screen after 10 seconds
screen is locked
student id does not exist in our data base
('192.168.56.1', 52345) 1211753 exists
the OS will lock screen after 10 seconds
screen is locked
student id does not exist in our data base
('192.168.56.1', 52345) 1222500 exists
the OS will lock screen after 10 seconds
screen is locked
student id does not exist in our data base
```

```
PS C:\Users\ASUS\Desktop\Test\Project> python -u "c:\Users\ASUS\Des
ktop\Test\Project\client.py"
1212677
the OS will lock screen after 10 seconds
222
the OS will not lock the screen
1211753
the OS will lock screen after 10 seconds
111
the OS will not lock the screen
1222500
the OS will lock screen after 10 seconds
444
the OS will not lock the screen
```

Part3:

Code:

1. Server Code:

```python
import socket
import os
import urllib.parse
from threading import Thread

class MyHTTPServer:
    def __init__(self, host, port):
        self.host = host
        self.port = port
        self.server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        self.server_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
        self.server_socket.bind((self.host, self.port))
        self.server_socket.listen(5)

    def start(self):
        print(f"Server listening on {self.host}:{self.port}")
        while True:
            client_socket, client_address = self.server_socket.accept()
            client_handler = Thread(target=self.handle_client, args=(client_socket,))
            client_handler.start()

    def handle_client(self, client_socket):
        client_address = client_socket.getpeername()
        request_data = client_socket.recv(1024).decode("utf-8")
        if not request_data:
            return

        method, path, _ = request_data.split(" ", 2)
        path = urllib.parse.unquote(path)
        file_path = path.lstrip("/")

        if method == "GET":
            if file_path == "index.html" or file_path == "en" or file_path == "":
                status_code = "200"
                self.log_request_details(client_address, method, path, "HTML", status_code)
                self.send_html_response(client_socket, "main_en.html")
            elif file_path == "ar":
                status_code = "200"
                self.log_request_details(client_address, method, path, "HTML", status_code)
                self.send_html_response(client_socket, "main_ar.html")
            elif file_path.endswith(".html"):
                status_code = "200"
                self.log_request_details(client_address, method, path, "HTML", status_code)
                self.send_html_file(client_socket, file_path)
            elif file_path.endswith(".css"):
                status_code = "200"
                self.log_request_details(client_address, method, path, "CSS", status_code)
                self.send_css_file(client_socket, file_path)
            elif file_path.endswith(".png"):
                status_code = "200"
                self.log_request_details(client_address, method, path, "PNG", status_code)
                self.send_image(client_socket, file_path, "image/png")
            elif file_path.endswith(".jpg"):
                status_code = "200"
                self.log_request_details(client_address, method, path, "JPG", status_code)
                self.send_image(client_socket, file_path, "image/jpeg")
            elif file_path in ["cr", "so", "rt"]:
                status_code = "307"
                self.log_request_details(client_address, method, path, "Redirect", status_code)
                self.send_redirect(client_socket, file_path)
            else:
                status_code = "404"
                self.log_request_details(client_address, method, path, "Unknown", status_code)
                self.send_error_response(client_socket)
        else:
            self.send_error_response(client_socket)

        client_socket.close()

    def log_request_details(self, client_address, method, path, file_type, status_code):
        print(f"Received request from {client_address}: Method: {method}, Path: {path}, File Type: {file_type}, Status Code: {status_code}")

    def send_static_file(self, client_socket, file_path, content_type):
        abs_file_path = os.path.abspath(os.path.join(os.path.dirname(__file__), "static", file_path))
        if not os.path.exists(abs_file_path):
            self.send_error_response(client_socket)
            return

        with open(abs_file_path, "rb") as file:
            content = file.read()

        response = f"HTTP/1.1 200 OK\r\nContent-Type: {content_type}\r\nContent-Length: {len(content)}\r\n\r\n"
        client_socket.sendall(response.encode("utf-8") + content)

    def send_html_response(self, client_socket, filename):
        self.send_static_file(client_socket, filename, "text/html")

    def send_html_file(self, client_socket, file_path):
        self.send_static_file(client_socket, file_path, "text/html")

    def send_css_file(self, client_socket, file_path):
        self.send_static_file(client_socket, file_path, "text/css")

    def send_image(self, client_socket, file_path, content_type):
        self.send_static_file(client_socket, file_path, content_type)

    def send_redirect(self, client_socket, target):
        locations = {"cr": "http://cornell.edu", "so": "http://stackoverflow.com", "rt": "http://ritaj.birzeit.edu"}
        if target in locations:
            response = f"HTTP/1.1 307 Temporary Redirect\r\nLocation: {locations[target]}\r\n\r\n"
            client_socket.sendall(response.encode("utf-8"))
        else:
            self.send_error_response(client_socket)

    def send_error_response(self, client_socket):
        response = "HTTP/1.1 404 Not Found\r\nContent-Type: text/html\r\n\r\n"

        try:
            with open(os.path.join("static", "DNE.html"), "rb") as file:
                content = file.read()
            client_socket.sendall(response.encode("utf-8") + content)
        except FileNotFoundError:
            error_message = "<html><body><h1>File Not Found</h1></body></html>"
            client_socket.sendall(response.encode("utf-8") + error_message.encode("utf-8"))

if __name__ == "__main__":
    host = '0.0.0.0'
    port = 9966
    server = MyHTTPServer(host, port)

    try:
        server.start()
    except KeyboardInterrupt:
        print("Server stopped by user")

```

2. main_en.html:

```html
<!DOCTYPE html>
<html>
<head>
    <title>ENCS3320-My Tiny Webserver 23/24</title>
    <style>
        body {
            background: linear-gradient(to right, #6dd5ed, #2193b0);
            font-family: Arial, sans-serif;
            color: #fff;
            margin: 0;
            padding: 0;
        }

        .info-box,
        .student {
            background-color: #fff;
            color: #333;
            border-radius: 10px;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.2);
            margin: 20px;
            padding: 20px;
            position: relative;
            transition: all 0.3s ease;
        }

        .info-box:hover,
        .student:hover {
            box-shadow: 0 0 15px rgba(0, 0, 0, 0.3);
        }

        .students {
            display: flex;
            justify-content: space-around;
            flex-wrap: wrap;
        }

        .student-photo {
            width: 100px;
            height: 100px;
            border-radius: 50%;
            object-fit: cover;
            position: absolute;
            top: 10px;
            right: 10px;
        }

        footer {
            text-align: center;
            padding: 20px;
        }

        a {
            color: #1a75ff;
            margin: 0 10px;
            transition: color 0.3s ease;
        }

        a:hover {
            color: #ffcc29;
        }
    </style>
</head>
<body>
    <h1>Welcome to our course Computer Networks, <span style="color: blue;">This is a tiny webserver</span></h1>

    <div class="info-box">
        <p>Content-Type in HTTP is used to indicate the media type of the resource. In responses, a Content-Type header tells the client what the content type of the returned content actually is.</p>
    </div>

    <div class="student-container">
        <div class="student">
            <img src="student1.png" alt = "student1 image" style="width: 100px; height: 100px; border-radius: 50%; object-fit: cover; position: absolute; top: 10px; right: 10px;">
            <h2>Omar Daghlas</h2>
            <p>ID: 1222500</p>
            <p>Projects: Gaza- Humanitarian Information Managing System</p>
            <p>Skills: Public Speeches</p>
            <p>Hobbies: Swimming</p>
        </div>

        <div class="student">
            <img src="student2.jpg" alt="student2 image" style="width: 100px; height: 100px; border-radius: 50%; object-fit: cover; position: absolute; top: 10px; right: 10px;">
            <h2>Abd AlRahman Shaheen</h2>
            <p>ID: 1211753</p>
            <p>Projects: private accounting program (c), local web server(html)(css)(python), simple CPU(multisim)(vhdl). </p>
            <p>Skills: c,c++,css,html,python,java,verilog,embeded c, fast learning, self learning.</p>
            <p>Hobbies: Coding, gamming, smoking. </p>
        </div>

        <div class="student">
            <img src="student3.jpg" alt="student3 image" style="width: 100px; height: 100px; border-radius: 50%; object-fit: cover; position: absolute; top: 10px; right: 10px;">
            <h2>Mahmoud Awad</h2>
            <p>ID: 1212677</p>
            <p>Projects: Train Seat Reserving System</p>
            <p>Skills: Programming</p>
            <p>Hobbies: Cycling</p>
        </div>
    </div>

    <a href="main_ar.html"> Arabic Page</a>
    <a href="https://www.w3schools.com/python/python_strings.asp">Python Strings at W3Schools</a>
</body>
</html>
```

3. main_ar.html:

```html
<!DOCTYPE html>
<html lang="ar" dir="rtl">
<head>
    <meta charset="UTF-8">
    <title>ENCS3320-24/23 خادم الويب الصغير الخاص بي</title>
    <link rel="stylesheet" type="text/css" href="style.css">
</head>
<body>
    <h1> مرحبًا بكم في دورتنا لشبكات الكمبيوتر، <span style="color: blue;">هذا خادم ويب صغير</span></h1>

    <div class="info-box">
        <p>يُستخدم نوع المحتوى في HTTP للإشارة إلى نوع الوسائط للمورد. في الاستجابات، يخبر رأس نوع المحتوى العميل بنوع المحتوى الفعلي المُرجع.</p>
    </div>

    <div class="student-container">
```

```html
        <div class="student">
            <img src="student1.png" alt="صورة الطالب الاول" class="student-photo" style="width: 100px; height: 100px; border-radius: 50%; object-fit: cover; position: absolute; top: 10px; left: 10px;">
            <h2>عمر دغلس</h2>
            <p>الرقم التعريفي: 1222500</p>
            <p>المشاريع: نظام إدارة المعلومات الإنسانية في غزة</p>
            <p>المهارات: الخطابة العامة</p>
            <p>الهوايات: السباحة</p>
        </div>

        <div class="student">
            <img src="student2.jpg" alt="صورة الطالب الثاني" class="student-photo" style="width: 100px; height: 100px; border-radius: 50%; object-fit: cover; position: absolute; top: 10px; left: 10px;">
            <h2>عبد الرحمن شاهين</h2>
            <p>الرقم التعريفي: 1211753</p>
            <p> المشاريع: برنامج محاسبة خاص , خادم ويب محلي ,  معالج بسيط</p>
            <p>المهارات: البرمجة,التلعم السريع,التعلم الذاتي</p>
            <p>الهوايات: البرمجة,العاب الفيديو,التدخين</p>
        </div>

        <div class="student">
            <img src="student3.jpg" alt="صورة الطالب الثالث" class="student-photo" style="width: 100px; height: 100px; border-radius: 50%; object-fit: cover; position: absolute; top: 10px; left: 10px;">
            <h2>محمود عوض</h2>
            <p>الرقم التعريفي: 1212677</p>
            <p>المشاريع: برنامج حجز مقاعد الطائرات</p>
            <p>المهارات: البرمجة</p>
            <p>الهوايات: ركوب الدراجات </p>
        </div>
    </div>

    <a href="main_en.html">English</a>
    <a href="https://www.w3schools.com/python/python_strings.asp">سلاسل الأحرف في Python على موقع W3Schools</a>
</body>
</html>
```

4. DNE.html:

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Error 404</title>
    <style>
        body { font-family: Arial, sans-serif; }
        .error-message { color: red; }
        .bold { font-weight: bold; }
    </style>
    <script>
        document.addEventListener("DOMContentLoaded", function() {
            // Use JavaScript to get client IP address and port
            document.querySelector(".bold.client-ip").innerText = "Client IP: " + window.location.hostname;
            document.querySelector(".bold.client-port").innerText = "Client Port: " + window.location.port;
        });
    </script>
</head>
<body>
    <h1>HTTP/1.1 404 Not Found</h1>
    <p class="error-message">The file is not found</p>
    <p class="bold">Student1 Name: [Omar Daghlas]</p>
    <p class="bold">Student1 ID: [1222500]</p>
    <p class="bold">Student2 Name: [Abd AlRahman Shaheen]</p>
    <p class="bold">Student2 ID: [1211753]</p>
    <p class="bold">Student3 Name: [Mahmoud Awad]</p>
    <p class="bold">Student3 ID: [1212677]</p>
    <p class="bold client-ip"></p>
    <p class="bold client-port"></p>
</body>
</html>

```

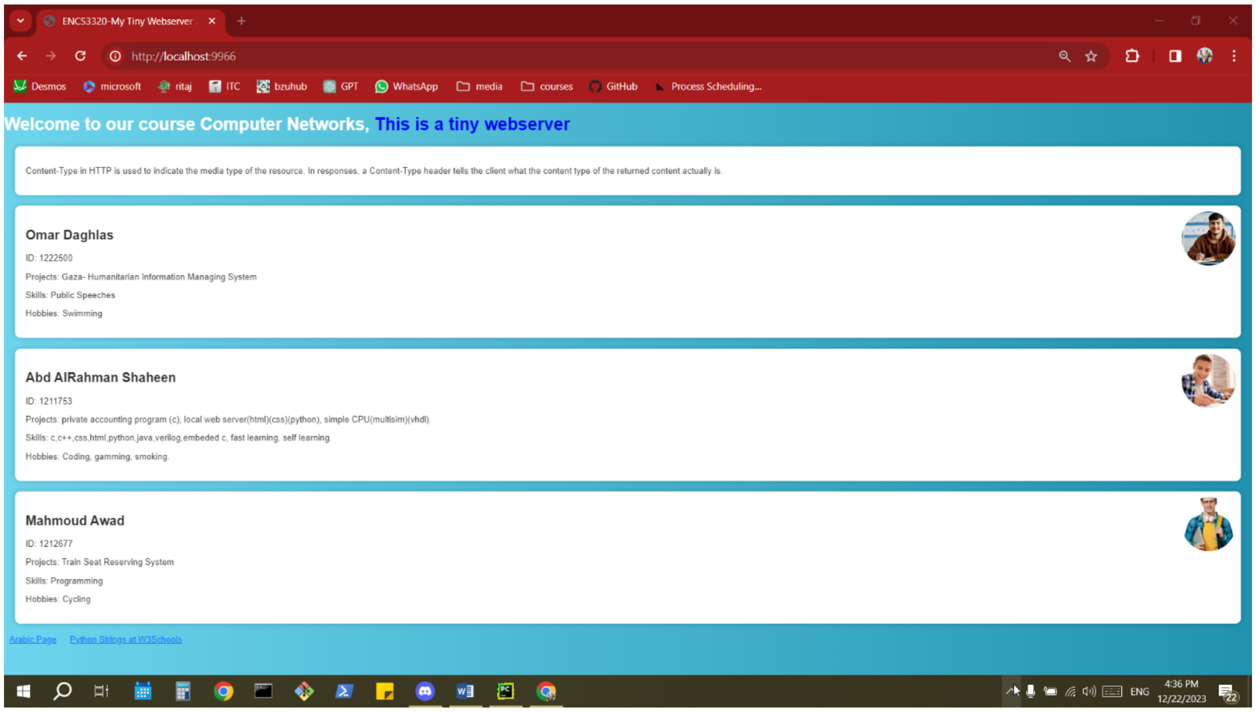
5. Style.CSS:

```css
/* style.css */
body {
    background: linear-gradient(to right, #6dd5ed, #2193b0);
    font-family: Arial, sans-serif;
    color: #fff;
    margin: 0;
    padding: 0;
}
.info-box,
.student {
    background-color: #fff;
    color: #333;
    border-radius: 10px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.2);
    margin: 20px;
    padding: 20px;
    position: relative;
    transition: all 0.3s ease;
}
.info-box:hover,
.student:hover {
    box-shadow: 0 0 15px rgba(0, 0, 0, 0.3);
}
.students {
    display: flex;
    justify-content: space-around;
    flex-wrap: wrap;
}
/* Remove styles related to the student photo */
footer {
    text-align: center;
    padding: 20px;
}

a {
    color: #1a75ff;
    margin: 0 10px;
    transition: color 0.3s ease;
}
a:hover {
    color: #ffcc29;
}

```

Results:

1. main_en.html Results:

1. http://localhost:9966/



Welcome to our course Computer Networks, This is a tiny webserver

Content-Type in HTTP is used to indicate the media type of the resource. In responses, a Content-Type header tells the client what the content type of the returned content actually is.

Omar Daghlas

ID: 1222500

Projects: Gaza- Humanitarian Information Managing System

Skills: Public Speeches

Hobbies: Swimming

Abd AlRahman Shaheen

ID: 1211753

Projects: private accounting program (c), local web server(html)(css)(python), simple CPU(multisim)(vhdl).

Skills: c,c++,css,html,python,java,verilog,embeded c, fast learning, self learning.

Hobbies: Coding, gamming, smoking.

Mahmoud Awad

ID: 1212677

Projects: Train Seat Reserving System

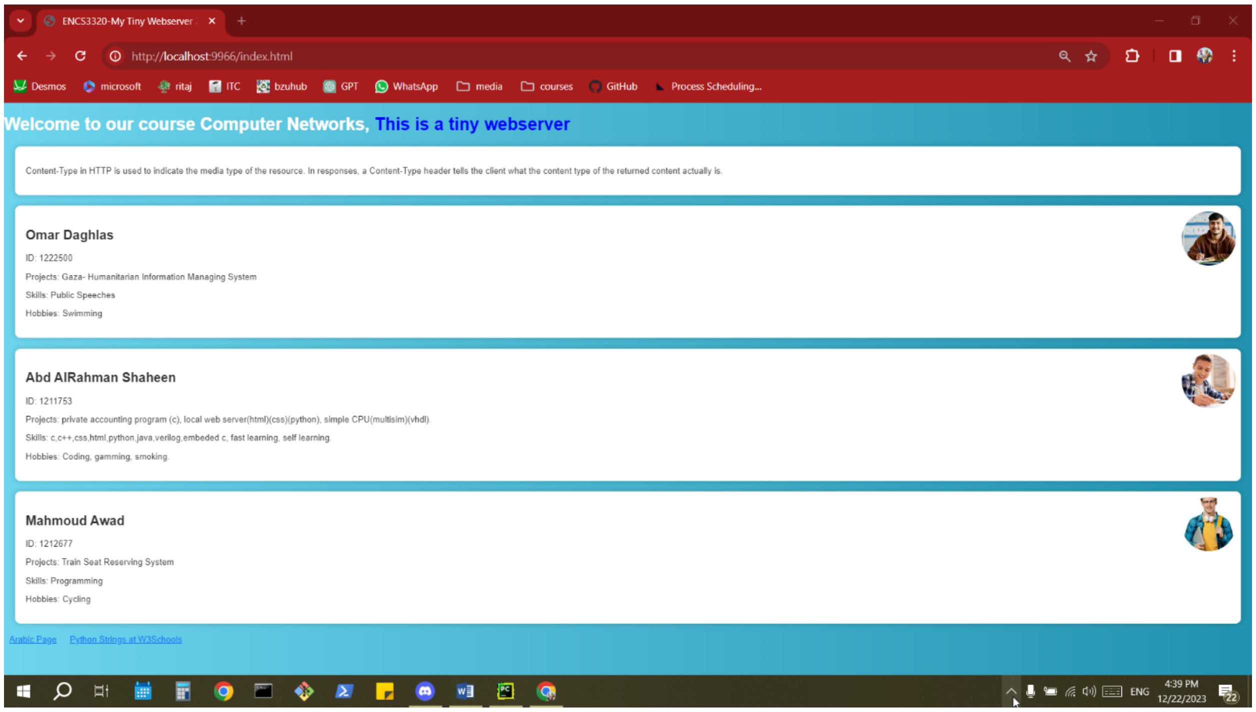
Skills: Programming

Hobbies: Cycling

Arabic Page Python Strings at W3Schools

---

2. http://localhost:9966/index.html



Welcome to our course Computer Networks, This is a tiny webserver

Content-Type in HTTP is used to indicate the media type of the resource. In responses, a Content-Type header tells the client what the content type of the returned content actually is.

Omar Daghlas

ID: 1222500

Projects: Gaza- Humanitarian Information Managing System

Skills: Public Speeches

Hobbies: Swimming

Abd AlRahman Shaheen

ID: 1211753

Projects: private accounting program (c), local web server(html)(css)(python), simple CPU(multisim)(vhdl).

Skills: c,c++,css,html,python,java,verilog,embeded c, fast learning, self learning.

Hobbies: Coding, gamming, smoking.

Mahmoud Awad

ID: 1212677

Projects: Train Seat Reserving System

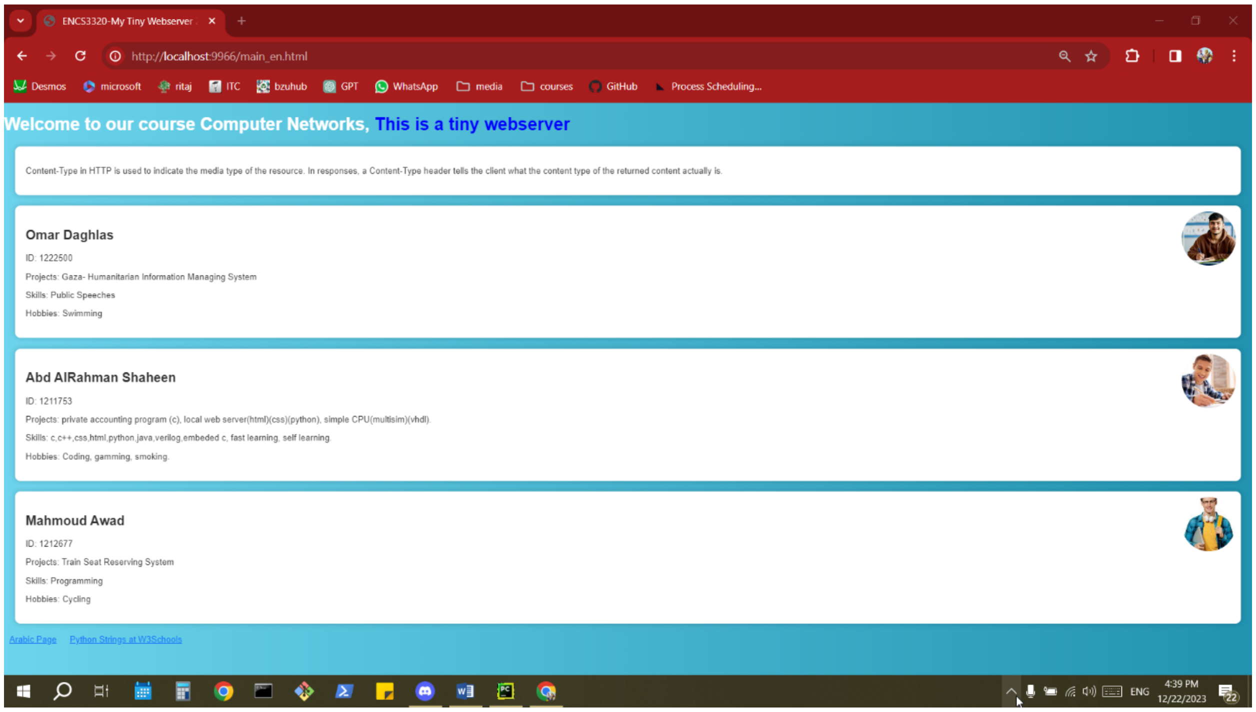
Skills: Programming

Hobbies: Cycling

Arabic Page Python Strings at W3Schools

---

3. http://localhost:9966/main_en.html


ENCS3320-My Tiny Webserver

http://localhost:9966/main_en.html

Welcome to our course Computer Networks, This is a tiny webserver

Content-Type in HTTP is used to indicate the media type of the resource. In responses, a Content-Type header tells the client what the content type of the returned content actually is.

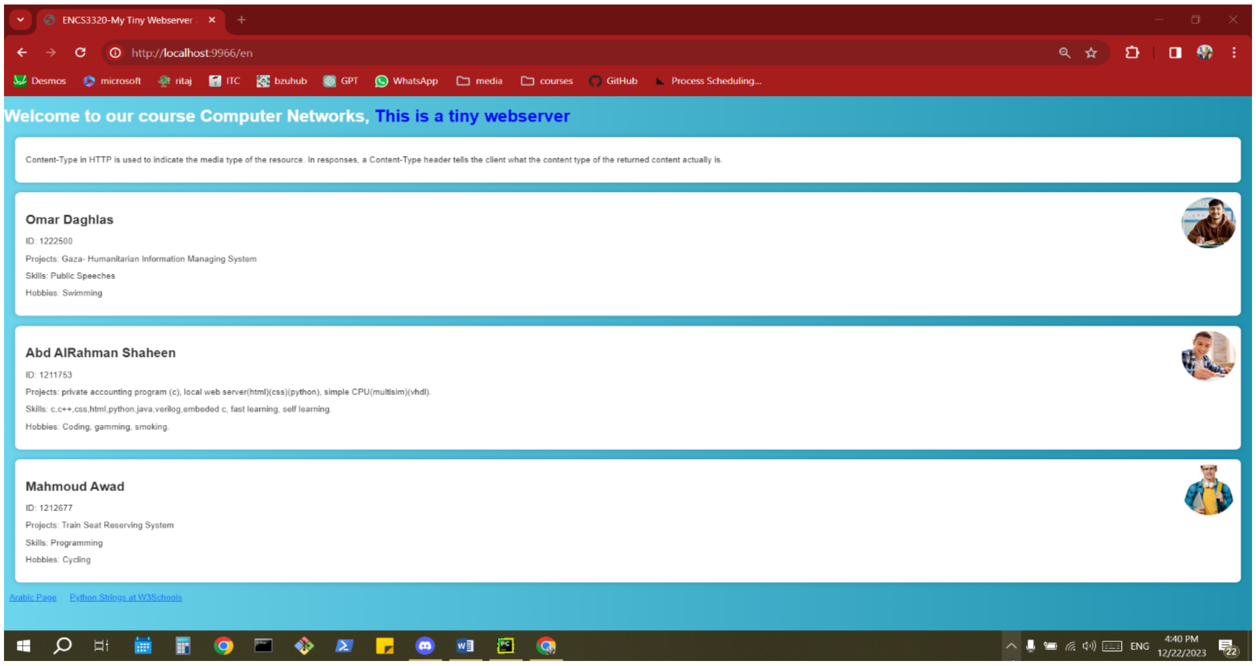
Omar Daghlas
ID: 1222500
Projects: Gaza- Humanitarian Information Managing System
Skills: Public Speeches
Hobbies: Swimming

Abd AlRahman Shaheen
ID: 1211753
Projects: private accounting program (c), local web server(html)(css)(python), simple CPU(multisim)(vhdl).
Skills: c,c++,css,html,python,java,verilog,embeded c, fast learning, self learning.
Hobbies: Coding, gamming, smoking.

Mahmoud Awad
ID: 1212677
Projects: Train Seat Reserving System
Skills: Programming
Hobbies: Cycling

Arabic Page   Python Strings at W3Schools

4. http://localhost:9966/en


ENCS3320-My Tiny Webserver

http://localhost:9966/en

Welcome to our course Computer Networks, This is a tiny webserver

Content-Type in HTTP is used to indicate the media type of the resource. In responses, a Content-Type header tells the client what the content type of the returned content actually is.

Omar Daghlas
ID: 1222500
Projects: Gaza- Humanitarian Information Managing System
Skills: Public Speeches
Hobbies: Swimming

Abd AlRahman Shaheen
ID: 1211753
Projects: private accounting program (c), local web server(html)(css)(python), simple CPU(multisim)(vhdl).
Skills: c,c++,css,html,python,java,verilog,embeded c, fast learning, self learning.
Hobbies: Coding, gamming, smoking.

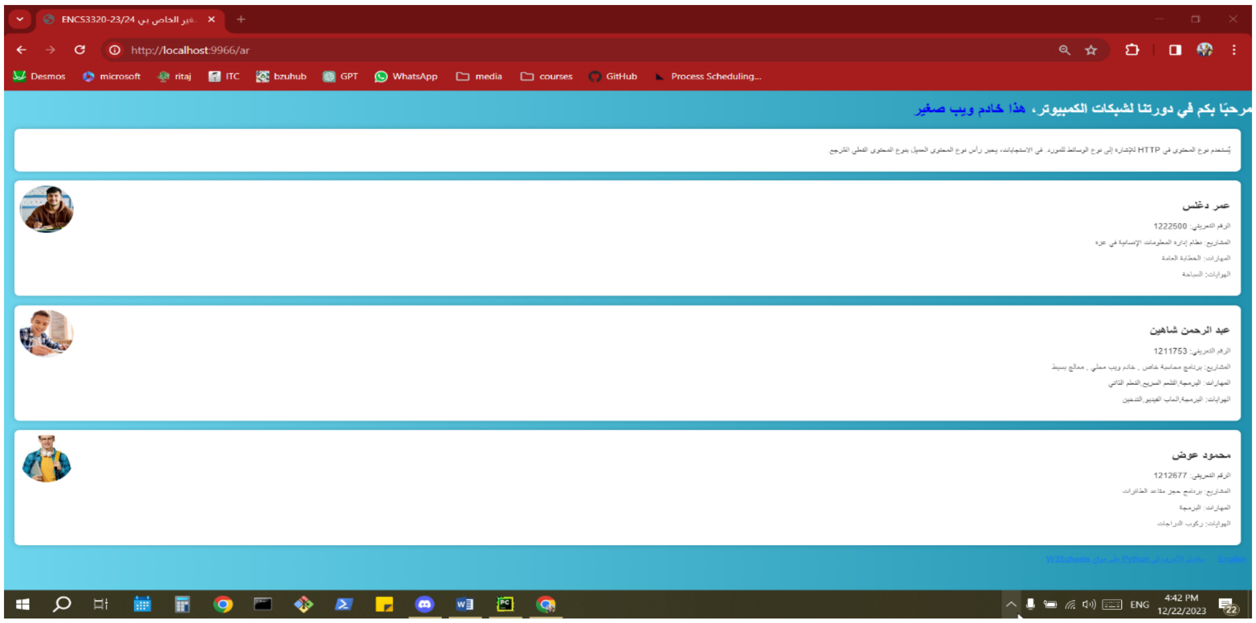
Mahmoud Awad
ID: 1212677
Projects: Train Seat Reserving System
Skills: Programming
Hobbies: Cycling

Arabic Page   Python Strings at W3Schools

- For all these four images the server receives an http request for main_en.html file then opens the html file.

2. main_ar.html Result:

❖ http://localhost:9966/ar



http://localhost:9966/ar

مرحبًا بكم في دورتنا لشبكات الكمبيوتر، هذا خادم ويب صغير

يُستخدم نوع المحتوى في HTTP للإشارة إلى نوع الوسائط للمورد. في الاستجابات، يخبر رأس نوع المحتوى العميل بنوع المحتوى الفعلي المُرجع.

عمر دغلس

الرقم التعريفي: 1222500

المشاريع: نظام إدارة المعلومات الإنسانية في غزة

المهارات: الخطابة العامة

الهوايات: السباحة

عبد الرحمن شاهين

الرقم التعريفي: 1211753

المشاريع: برنامج محاسبة خاص , خادم ويب محلي , معالج بسيط

المهارات: البرمجة,التعلم السريع,التعلم الذاتي

الهوايات: البرمجة,ألعاب الفيديو,التدخين

محمود عوض

الرقم التعريفي: 1212677

المشاريع: برنامج حجز مقاعد الطائرات

المهارات: البرمجة

الهوايات: ركوب الدراجات

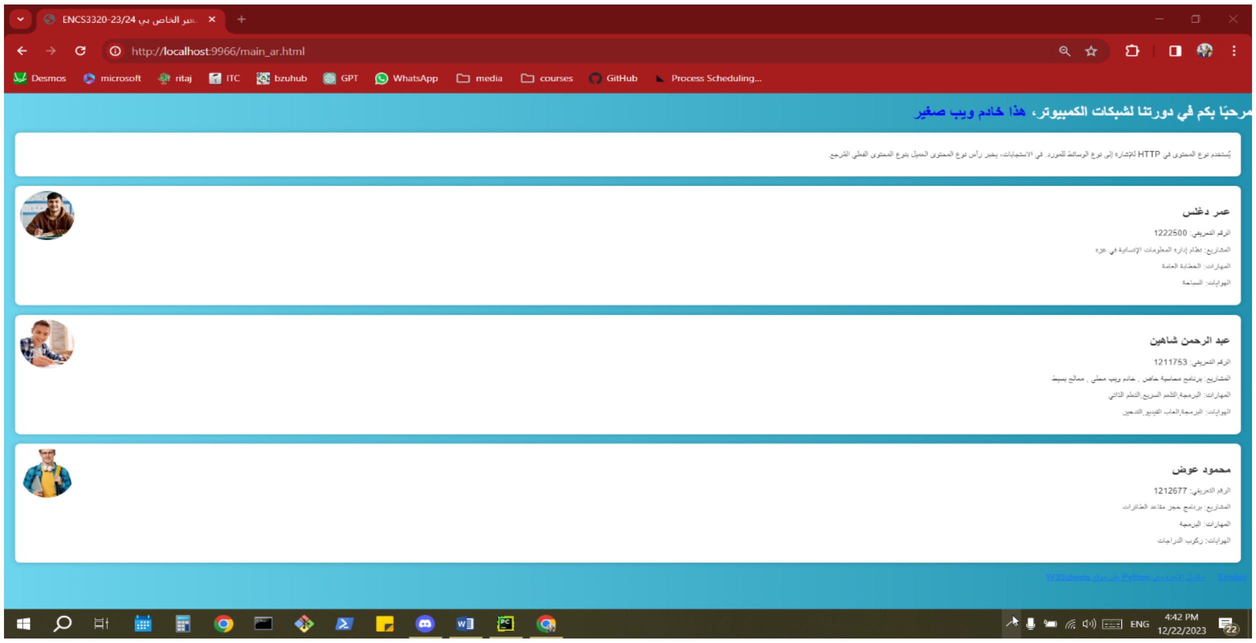
English [illegible] الأحرف في Python على موقع W3Schools

- For this image the sever receives an http request for main_ar.html then opens the html file.

---

3. ALL .html file Result:

❖ http://localhost:9966/main_ar.html OR http://localhost:9966/*.html



http://localhost:9966/main_ar.html

مرحبًا بكم في دورتنا لشبكات الكمبيوتر، هذا خادم ويب صغير

يُستخدم نوع المحتوى في HTTP للإشارة إلى نوع الوسائط للمورد. في الاستجابات، يخبر رأس نوع المحتوى العميل بنوع المحتوى الفعلي المُرجع.

عمر دغلس

الرقم التعريفي: 1222500

المشاريع: نظام إدارة المعلومات الإنسانية في غزة

المهارات: الخطابة العامة

الهوايات: السباحة

عبد الرحمن شاهين

الرقم التعريفي: 1211753

المشاريع: برنامج محاسبة خاص , خادم ويب محلي , معالج بسيط

المهارات: البرمجة,التعلم السريع,التعلم الذاتي

الهوايات: البرمجة,ألعاب الفيديو,التدخين

محمود عوض

الرقم التعريفي: 1212677

المشاريع: برنامج حجز مقاعد الطائرات

المهارات: البرمجة

الهوايات: ركوب الدراجات

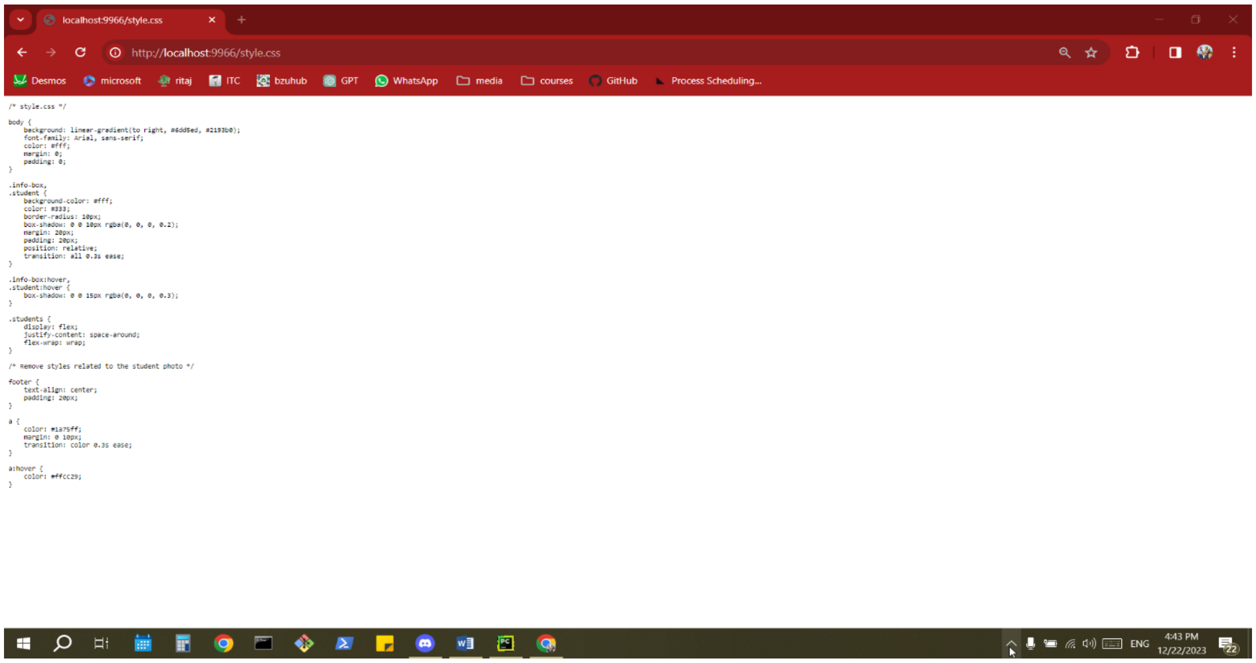
English [illegible] الأحرف في Python على موقع W3Schools

- For this image the sever receives an http request for .html file then opens the requested html file.

---

4. .CSS Result:

❖ http://localhost:9966/style.css



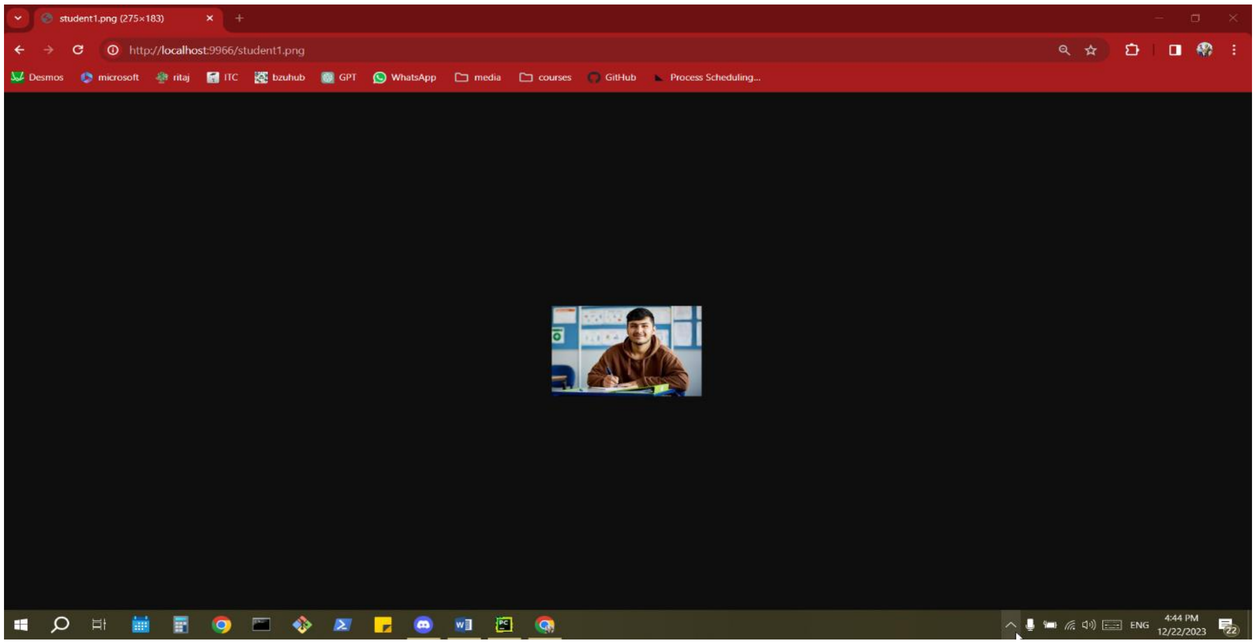
```css
/* style.css */

body {
    background: linear-gradient(to right, #6dd5ed, #2193b0);
    font-family: Arial, sans-serif;
    color: #fff;
    margin: 0;
    padding: 0;
}

.info-box,
.student {
    background-color: #fff;
    color: #333;
    border-radius: 10px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.2);
    margin: 20px;
    padding: 20px;
    position: relative;
    transition: all 0.3s ease;
}

.info-box:hover,
.student:hover {
    box-shadow: 0 0 15px rgba(0, 0, 0, 0.3);
}

.students {
    display: flex;
    justify-content: space-around;
    flex-wrap: wrap;
}

/* Remove styles related to the student photo */

footer {
    text-align: center;
    padding: 20px;
}

a {
    color: #1a75ff;
    margin: 0 10px;
    transition: color 0.3s ease;
}

a:hover {
    color: #ffcc29;
}
```

- For this image the sever receives an http request for .css file then opens the requested css file.

---

5. .png Result:

❖ http://localhost:9966/student1.png

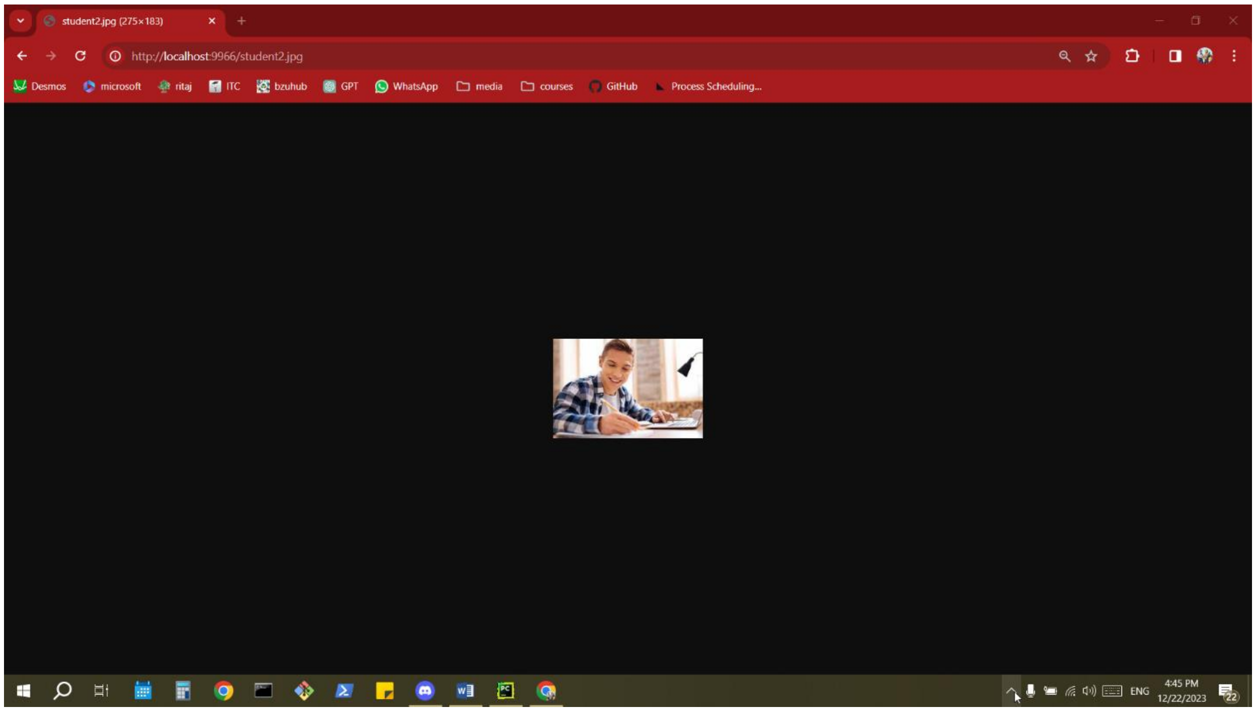


- For this image the sever receives an http request for .png file then opens the requested png file.

---

6. .jpg Result:

❖ http://localhost:9966/student2.jpg

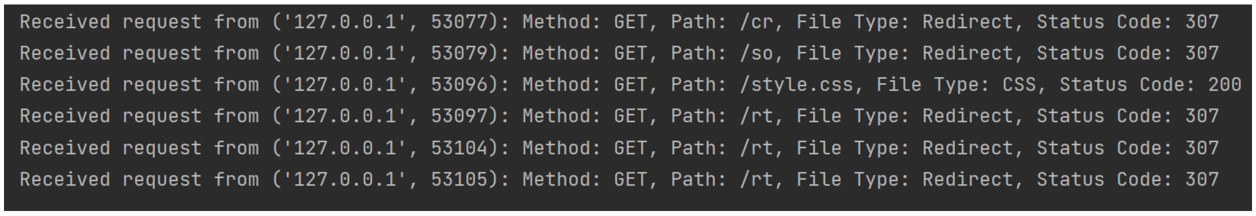


- For this image the sever receives an http request for .jpg file then opens the requested jpg file.

---

7. Status Code Result:

❖ http://localhost:9966/cr
❖ http://localhost:9966/so
❖ http://localhost:9966/rt



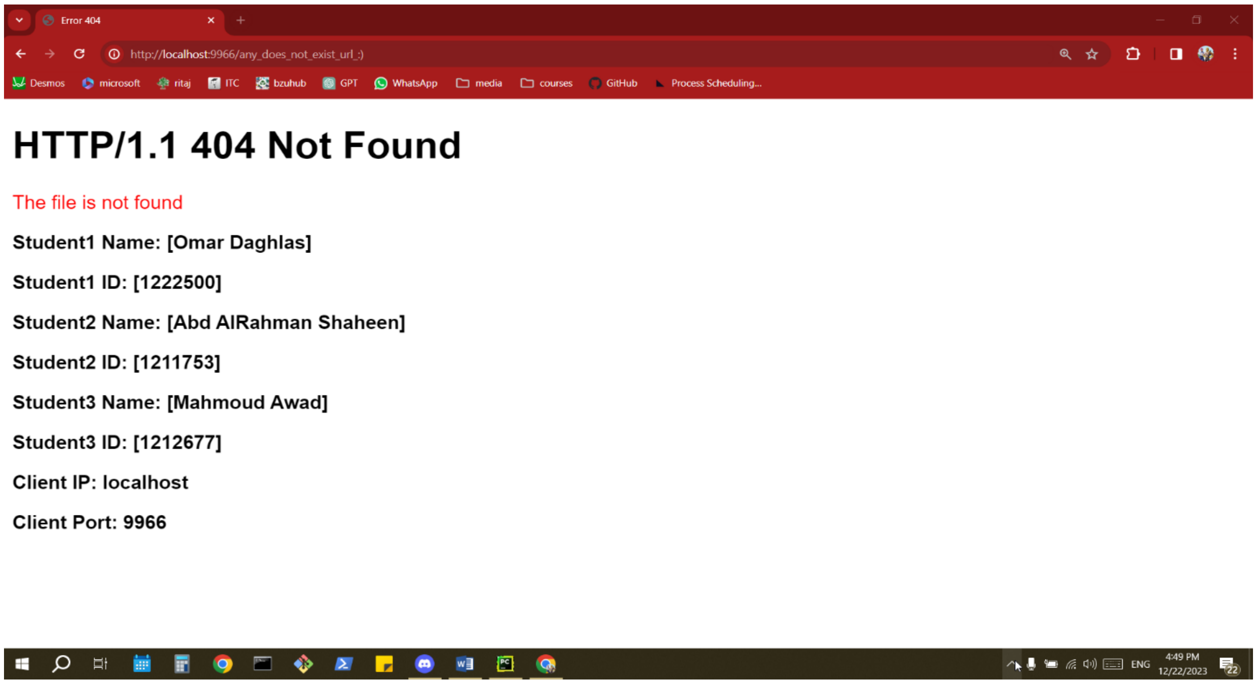
```
Received request from ('127.0.0.1', 53077): Method: GET, Path: /cr, File Type: Redirect, Status Code: 307
Received request from ('127.0.0.1', 53079): Method: GET, Path: /so, File Type: Redirect, Status Code: 307
Received request from ('127.0.0.1', 53096): Method: GET, Path: /style.css, File Type: CSS, Status Code: 200
Received request from ('127.0.0.1', 53097): Method: GET, Path: /rt, File Type: Redirect, Status Code: 307
Received request from ('127.0.0.1', 53104): Method: GET, Path: /rt, File Type: Redirect, Status Code: 307
Received request from ('127.0.0.1', 53105): Method: GET, Path: /rt, File Type: Redirect, Status Code: 307
```

- For this image the sever receivesan http request for redirection then opens the requested direction.

---

8. The Result if the File doesn't Exist:

❖ http://localhost:9966/any_does_not_exist_url_:)



HTTP/1.1 404 Not Found

The file is not found

Student1 Name: [Omar Daghlas]

Student1 ID: [1222500]

Student2 Name: [Abd AlRahman Shaheen]

Student2 ID: [1211753]

Student3 Name: [Mahmoud Awad]

Student3 ID: [1212677]

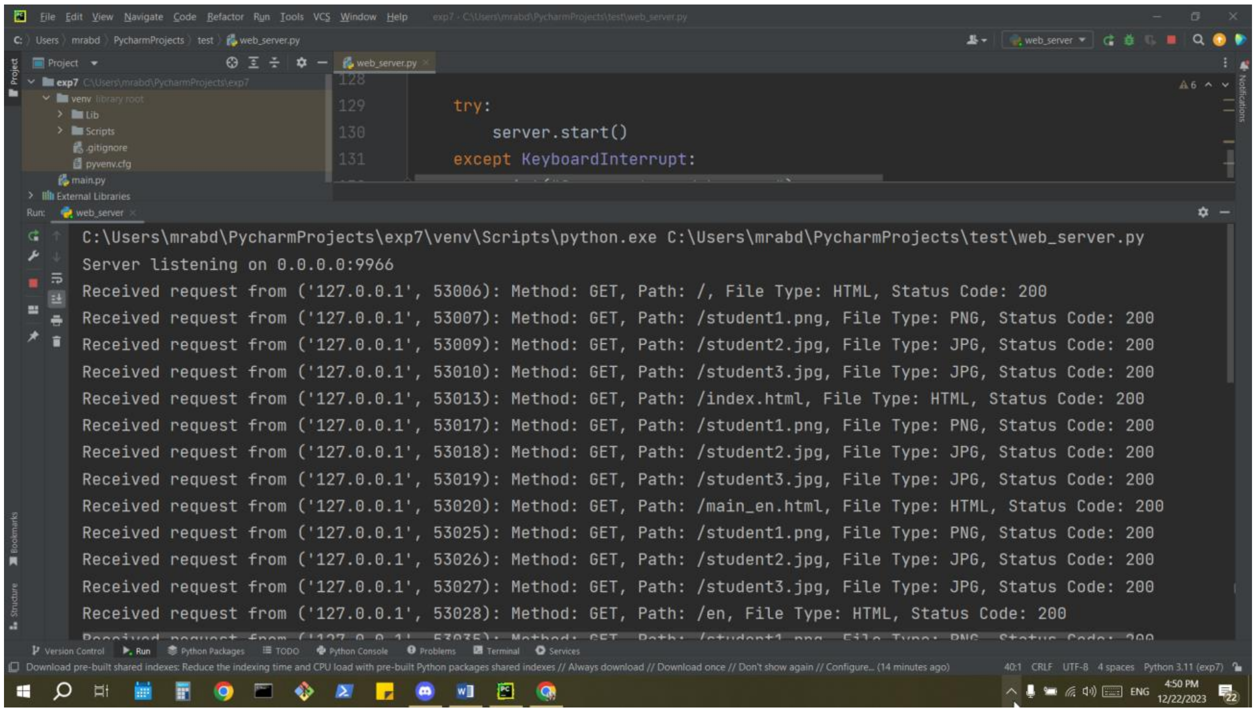
Client IP: localhost

Client Port: 9966

- If any file is not found then the server runs the file DNE.html.

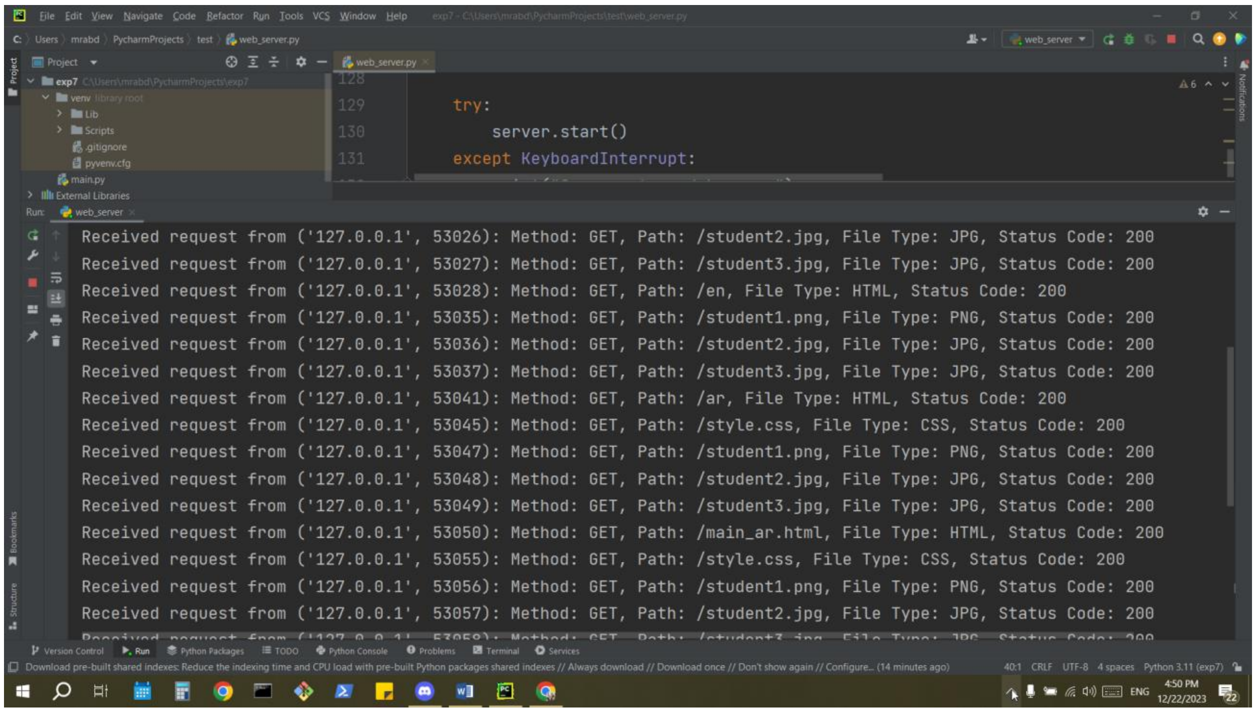
---

9. Command line window Result:



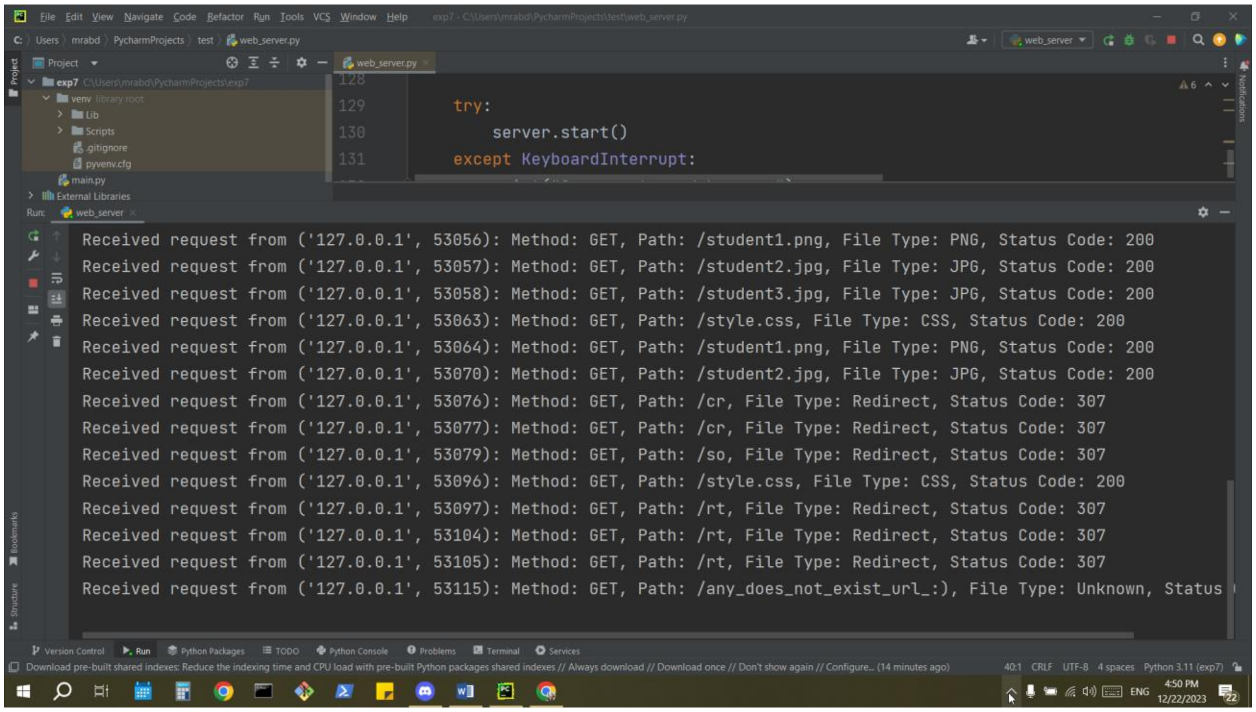
```
try:
    server.start()
except KeyboardInterrupt:
```

```
C:\Users\mrabd\PycharmProjects\exp7\venv\Scripts\python.exe C:\Users\mrabd\PycharmProjects\test\web_server.py
Server listening on 0.0.0.0:9966
Received request from ('127.0.0.1', 53006): Method: GET, Path: /, File Type: HTML, Status Code: 200
Received request from ('127.0.0.1', 53007): Method: GET, Path: /student1.png, File Type: PNG, Status Code: 200
Received request from ('127.0.0.1', 53009): Method: GET, Path: /student2.jpg, File Type: JPG, Status Code: 200
Received request from ('127.0.0.1', 53010): Method: GET, Path: /student3.jpg, File Type: JPG, Status Code: 200
Received request from ('127.0.0.1', 53013): Method: GET, Path: /index.html, File Type: HTML, Status Code: 200
Received request from ('127.0.0.1', 53017): Method: GET, Path: /student1.png, File Type: PNG, Status Code: 200
Received request from ('127.0.0.1', 53018): Method: GET, Path: /student2.jpg, File Type: JPG, Status Code: 200
Received request from ('127.0.0.1', 53019): Method: GET, Path: /student3.jpg, File Type: JPG, Status Code: 200
Received request from ('127.0.0.1', 53020): Method: GET, Path: /main_en.html, File Type: HTML, Status Code: 200
Received request from ('127.0.0.1', 53025): Method: GET, Path: /student1.png, File Type: PNG, Status Code: 200
Received request from ('127.0.0.1', 53026): Method: GET, Path: /student2.jpg, File Type: JPG, Status Code: 200
Received request from ('127.0.0.1', 53027): Method: GET, Path: /student3.jpg, File Type: JPG, Status Code: 200
Received request from ('127.0.0.1', 53028): Method: GET, Path: /en, File Type: HTML, Status Code: 200
```



```
    try:
        server.start()
    except KeyboardInterrupt:
```

```
Received request from ('127.0.0.1', 53026): Method: GET, Path: /student2.jpg, File Type: JPG, Status Code: 200
Received request from ('127.0.0.1', 53027): Method: GET, Path: /student3.jpg, File Type: JPG, Status Code: 200
Received request from ('127.0.0.1', 53028): Method: GET, Path: /en, File Type: HTML, Status Code: 200
Received request from ('127.0.0.1', 53035): Method: GET, Path: /student1.png, File Type: PNG, Status Code: 200
Received request from ('127.0.0.1', 53036): Method: GET, Path: /student2.jpg, File Type: JPG, Status Code: 200
Received request from ('127.0.0.1', 53037): Method: GET, Path: /student3.jpg, File Type: JPG, Status Code: 200
Received request from ('127.0.0.1', 53041): Method: GET, Path: /ar, File Type: HTML, Status Code: 200
Received request from ('127.0.0.1', 53045): Method: GET, Path: /style.css, File Type: CSS, Status Code: 200
Received request from ('127.0.0.1', 53047): Method: GET, Path: /student1.png, File Type: PNG, Status Code: 200
Received request from ('127.0.0.1', 53048): Method: GET, Path: /student2.jpg, File Type: JPG, Status Code: 200
Received request from ('127.0.0.1', 53049): Method: GET, Path: /student3.jpg, File Type: JPG, Status Code: 200
Received request from ('127.0.0.1', 53050): Method: GET, Path: /main_ar.html, File Type: HTML, Status Code: 200
Received request from ('127.0.0.1', 53055): Method: GET, Path: /style.css, File Type: CSS, Status Code: 200
Received request from ('127.0.0.1', 53056): Method: GET, Path: /student1.png, File Type: PNG, Status Code: 200
Received request from ('127.0.0.1', 53057): Method: GET, Path: /student2.jpg, File Type: JPG, Status Code: 200
```

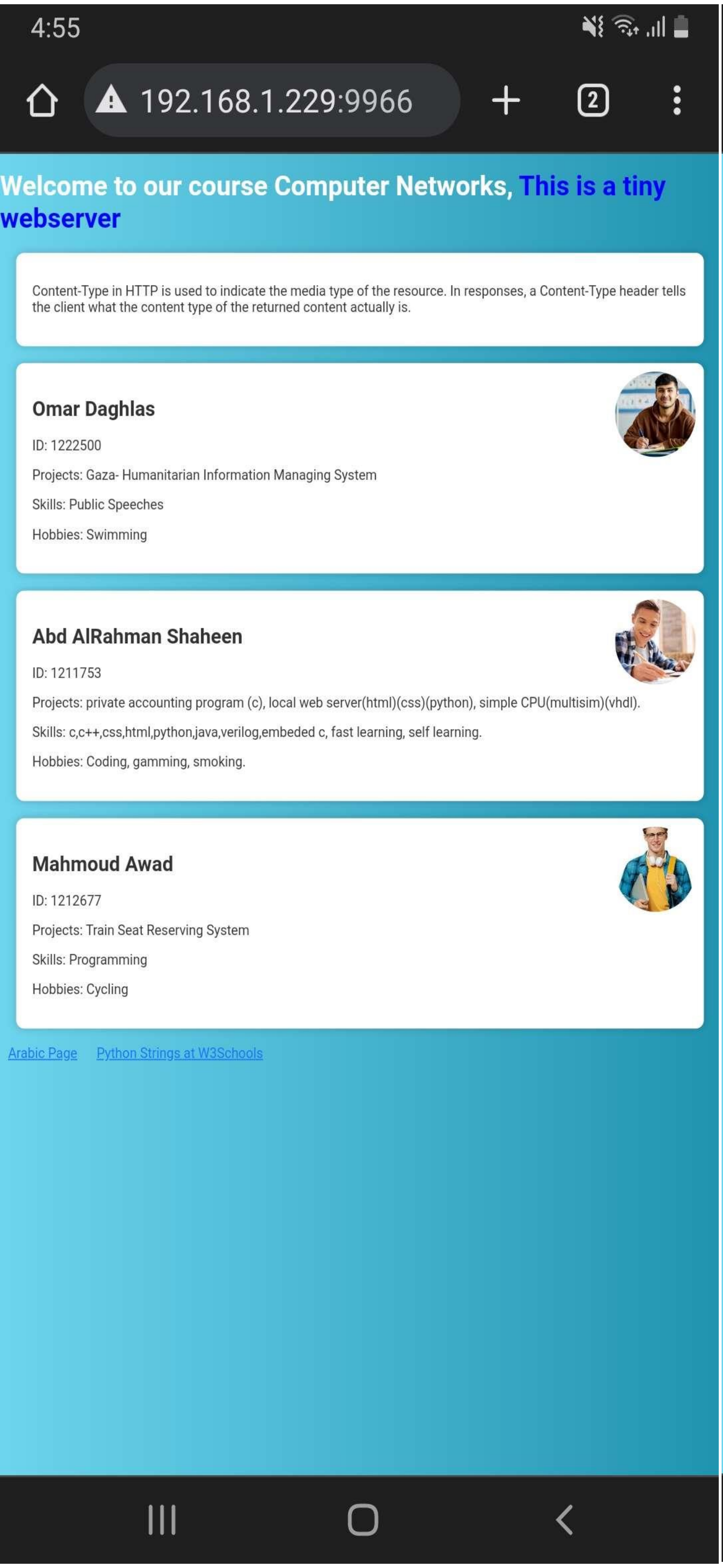


```
    try:
        server.start()
    except KeyboardInterrupt:
```

```
Received request from ('127.0.0.1', 53056): Method: GET, Path: /student1.png, File Type: PNG, Status Code: 200
Received request from ('127.0.0.1', 53057): Method: GET, Path: /student2.jpg, File Type: JPG, Status Code: 200
Received request from ('127.0.0.1', 53058): Method: GET, Path: /student3.jpg, File Type: JPG, Status Code: 200
Received request from ('127.0.0.1', 53063): Method: GET, Path: /style.css, File Type: CSS, Status Code: 200
Received request from ('127.0.0.1', 53064): Method: GET, Path: /student1.png, File Type: PNG, Status Code: 200
Received request from ('127.0.0.1', 53070): Method: GET, Path: /student2.jpg, File Type: JPG, Status Code: 200
Received request from ('127.0.0.1', 53076): Method: GET, Path: /cr, File Type: Redirect, Status Code: 307
Received request from ('127.0.0.1', 53077): Method: GET, Path: /cr, File Type: Redirect, Status Code: 307
Received request from ('127.0.0.1', 53079): Method: GET, Path: /so, File Type: Redirect, Status Code: 307
Received request from ('127.0.0.1', 53096): Method: GET, Path: /style.css, File Type: CSS, Status Code: 200
Received request from ('127.0.0.1', 53097): Method: GET, Path: /rt, File Type: Redirect, Status Code: 307
Received request from ('127.0.0.1', 53104): Method: GET, Path: /rt, File Type: Redirect, Status Code: 307
Received request from ('127.0.0.1', 53105): Method: GET, Path: /rt, File Type: Redirect, Status Code: 307
Received request from ('127.0.0.1', 53115): Method: GET, Path: /any_does_not_exist_url_:), File Type: Unknown, Status
```

- These are all the http requests for all images above.

➢ A device on the same Network:


4:55

192.168.1.229:9966

Welcome to our course Computer Networks, This is a tiny webserver

Content-Type in HTTP is used to indicate the media type of the resource. In responses, a Content-Type header tells the client what the content type of the returned content actually is.

Omar Daghlas

ID: 1222500

Projects: Gaza- Humanitarian Information Managing System

Skills: Public Speeches

Hobbies: Swimming

Abd AlRahman Shaheen

ID: 1211753

Projects: private accounting program (c), local web server(html)(css)(python), simple CPU(multisim)(vhdl).

Skills: c,c++,css,html,python,java,verilog,embeded c, fast learning, self learning.

Hobbies: Coding, gamming, smoking.

Mahmoud Awad

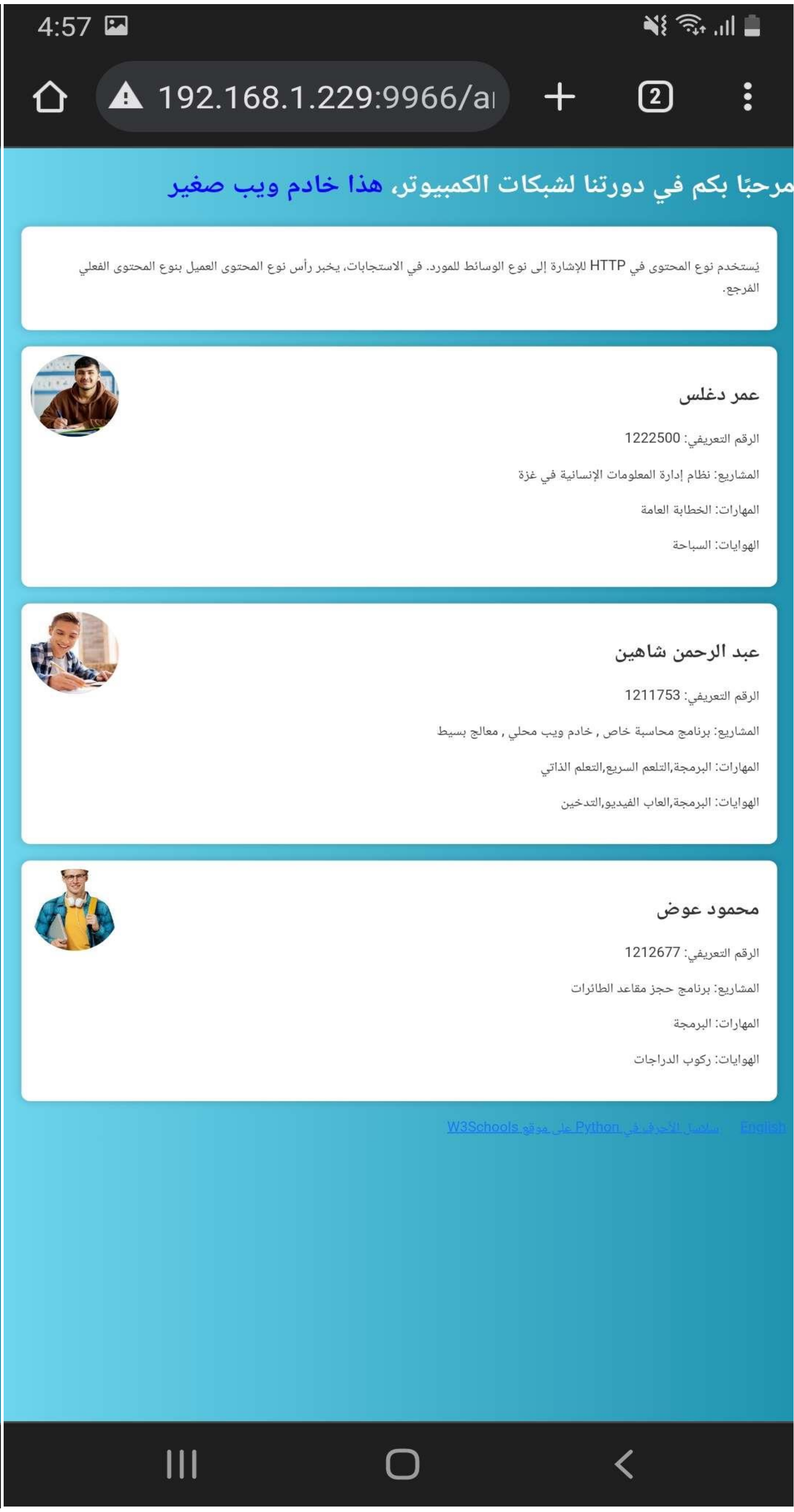
ID: 1212677

Projects: Train Seat Reserving System

Skills: Programming

Hobbies: Cycling

Arabic Page Python Strings at W3Schools


4:57

192.168.1.229:9966/a

مرحبًا بكم في دورتنا لشبكات الكمبيوتر، هذا خادم ويب صغير

يُستخدم نوع المحتوى في HTTP للإشارة إلى نوع الوسائط للمورد. في الاستجابات، يخبر رأس نوع المحتوى العميل بنوع المحتوى الفعلي المُرجع.

عمر دغلس

الرقم التعريفي: 1222500

المشاريع: نظام إدارة المعلومات الإنسانية في غزة

المهارات: الخطابة العامة

الهوايات: السباحة

عبد الرحمن شاهين

الرقم التعريفي: 1211753

المشاريع: برنامج محاسبة خاص , خادم ويب محلي , معالج بسيط

المهارات: البرمجة,التلعم السريع,التعلم الذاتي

الهوايات: البرمجة,العاب الفيديو,التدخين

محمود عوض

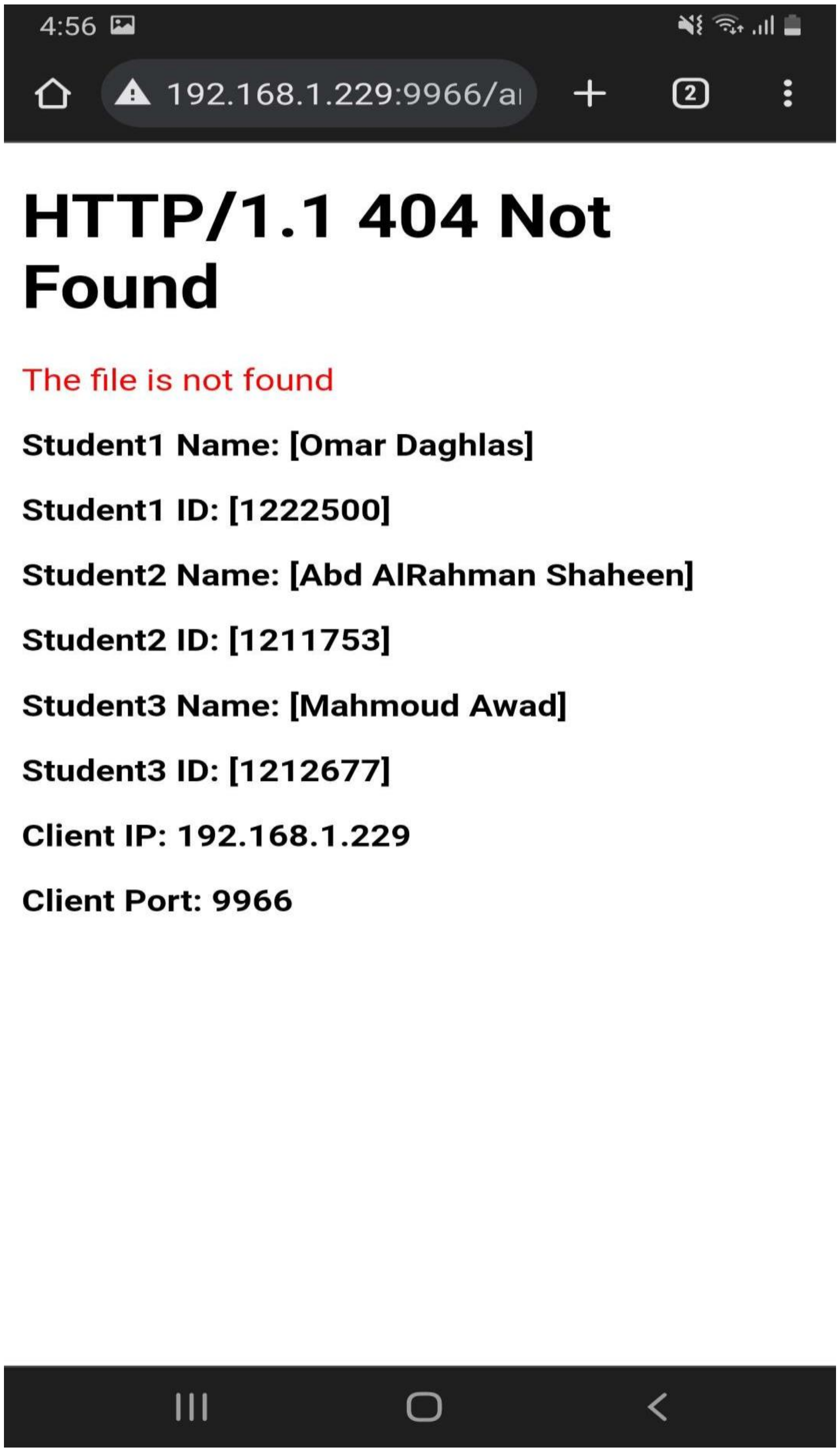
الرقم التعريفي: 1212677

المشاريع: برنامج حجز مقاعد الطائرات

المهارات: البرمجة

الهوايات: ركوب الدراجات

English سلاسل الأحرف في Python على موقع W3Schools


4:56

192.168.1.229:9966/a

HTTP/1.1 404 Not Found

The file is not found

Student1 Name: [Omar Daghlas]

Student1 ID: [1222500]

Student2 Name: [Abd AlRahman Shaheen]

Student2 ID: [1211753]

Student3 Name: [Mahmoud Awad]

Student3 ID: [1212677]

Client IP: 192.168.1.229

Client Port: 9966

- These are the runs of my phone on the same network.