Faculty of Engineering & Technology – Computer Systems Engineering Department

Second Semester 2023-2024

Computer Network ENCS3320

Project – 1-

Prepared by: Rasha Daoud - 1210382

: Nadia Thaer - 1210021

Section: 2

Instructor: Abdalkarim Awad

Date: 5th May 202

Table of Contant :

List of Figure

Part 1

1.1Ping

Ping is a network utility used to test host reachability on IP networks. It measures the round-trip time for messages sent from one host to another and back, derived from sonar terminology.

1.2Tracert

Tracert, or traceroute, maps the path data takes on the internet from its origin to its destination. It helps find connectivity problems and network slowdowns.

1.3.NSLookup

NSLookup, short for "Name Server Lookup," is a command-line tool used to query Domain Name System (DNS) servers to obtain domain name or IP address information.

1.4.Telnet

Telnet is a network protocol and command-line tool that allows users to establish a connection to a remote computer or server over a network. It provides a bidirectional interactive text-oriented communication channel between the local and remote systems.

Part one 2

2.1.Ping a device in the same network (from the laptop to the smartphone).

IPv4 Address (Laptop): 192.168.56.1
IPv4 Address (smartphone): 192.168.1.23

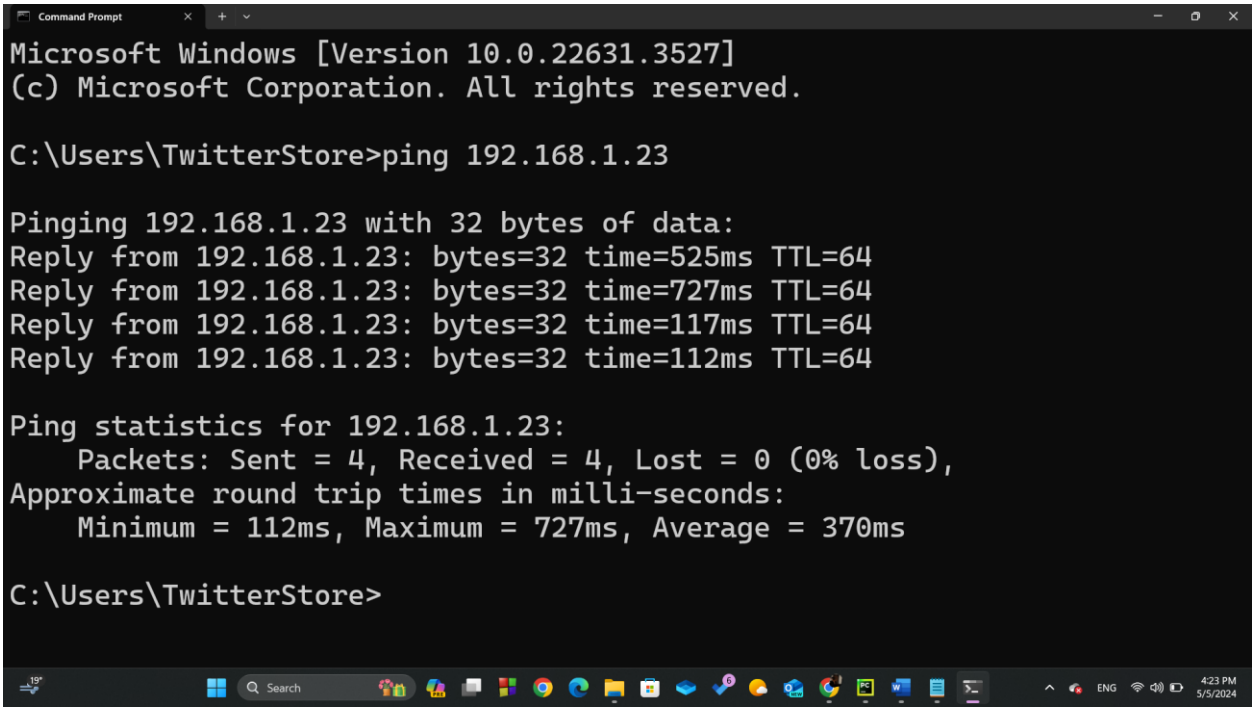


Figure 1:Ping a device in The Same Network from Labtop To Smart Phone.

- The ping command is sending ICMP echo requests to the IP address 192.168.1.23 with packets of size 32 bytes.
- Then a reply line show the reply from the IP address 192.168.1.23, indicating that the device at that address responded to the ping request. The time parameter indicates the round-trip time in, and TTL (Time To Live) represents the maximum number of hops the packet can traverse before being discarded.
- And in the next section provides statistics about the ping operation. It shows that 4 packets were sent, all 4 were received successfully (0% loss), and no packets were lost.

2.2.Ping www.stanford.edu

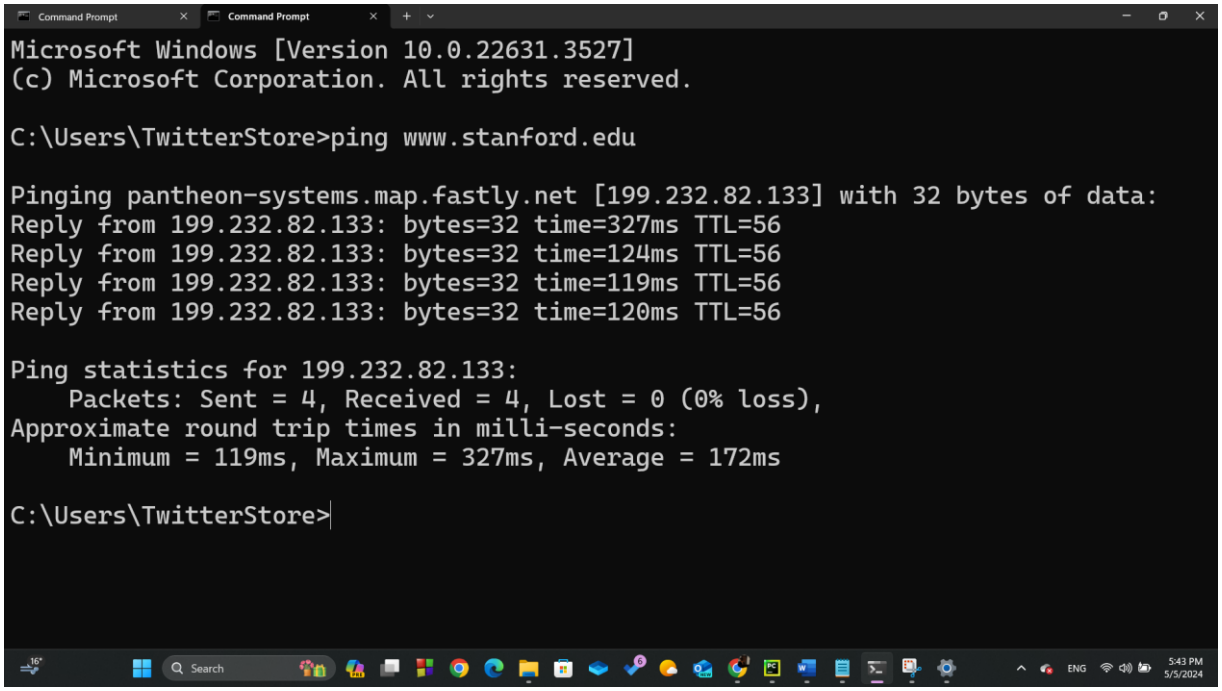


Figure 2:Ping www.stanford.

- The ping command is sending ICMP echo requests to the IP address 199.232.82.133, which corresponds to the domain www.stanford.edu. The IP address is resolved from the domain name using DNS (Domain Name System) resolution.

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

According to the information given in the ping results, the response is highly probable to be from the USA.

The domain www.stanford.edu is hosted on the Fastly content delivery network (CDN), as indicated by the hostname "pantheon-systems.map.fastly.net" in the ping results. Fastly is a globally distributed CDN service provider that accelerates content delivery by caching data closer to end-users worldwide.

Also, if we use a website that check the location of the IP address 199.232.82.133, we get the following details:

Hostname:	199.232.82.133
ASN:	54113
ISP:	Fastly Inc.
Services:	Datacenter
Assignment:	Likely Static IP
Country:	United States
State/Region:	California
City:	San Francisco

Thus, we can determine that this IP address is allocated in USA.

2.4.Tracert www.stanford.com

The traceroute shows the path taken by network packets from my computer to the destination server at IP address 3.33.152.147. along with the round-trip time (latency) for each hop.

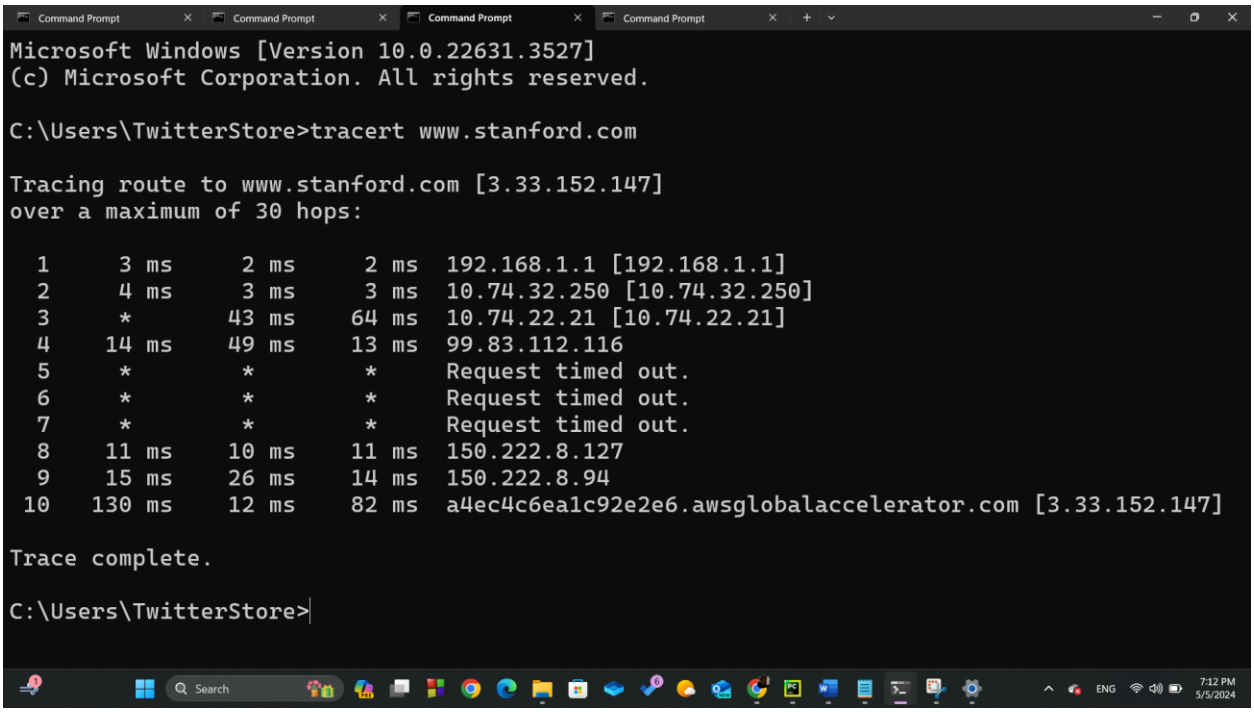


Figure 3:Tracert www.standford.com

In some cases, the traceroute may encounter hops that do not respond to the traceroute request, resulting in "Request timed out" messages. This could be due to network congestion, firewall configurations, or other reasons.

The last line of the traceroute output displays the IP address of the destination server (3.33.152.147) and its hostname (a4ec4c6ea1c92e2e6.awsglobalaccelerator.com), indicating that the traceroute reached its final destination.

2.5 nslookup www.stanford.com

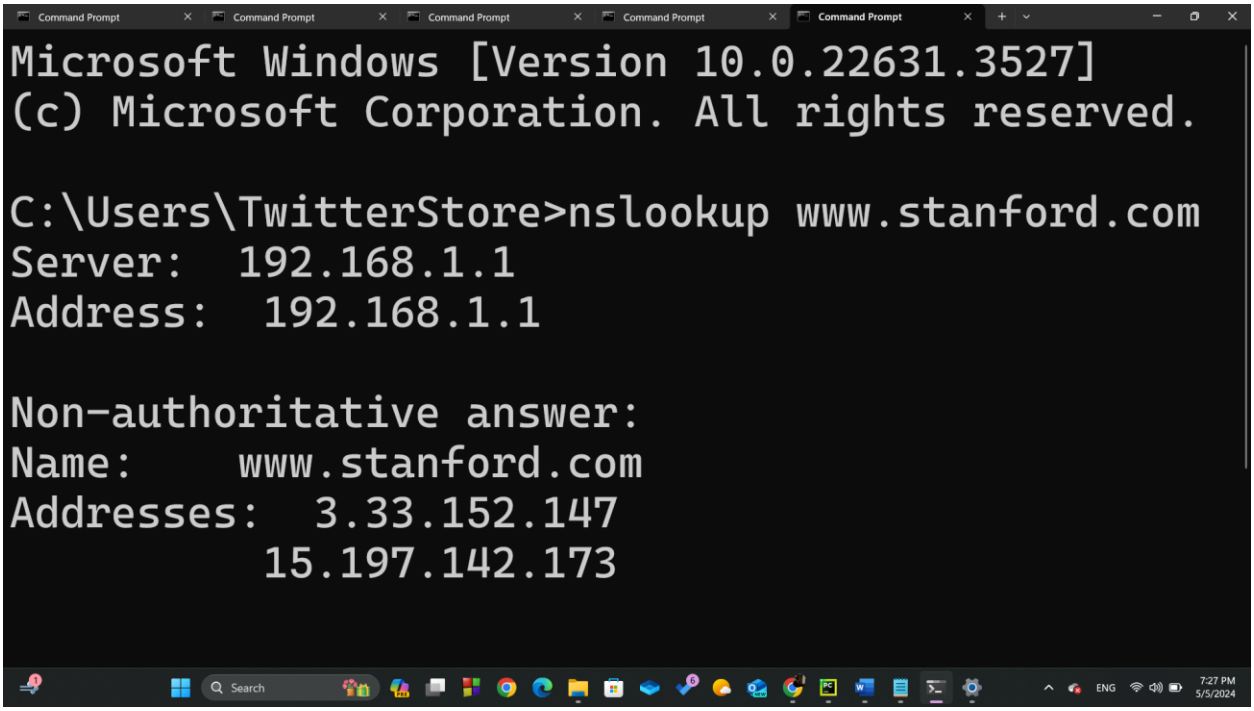


Figure 4:Nslookup www.standford.com

The nslookup command for www.stanford.com provides information about the domain name and its associated IP addresses.

Server: 192.168.1.1: This line indicates the DNS server used for the nslookup. And the DNS server at IP address 192.168.1.1 is providing the response.

Non-authoritative answer: This message indicates that the response comes from a DNS server other than the authoritative DNS server for the domain.

www.stanford.com resolves to two IP addresses: 3.33.152.147 and 15.197.142.173. These are the destinations that the computer would communicate with when accessing www.stanford.com over the internet.

Part one 3. use wireshark to capture some DNS messages

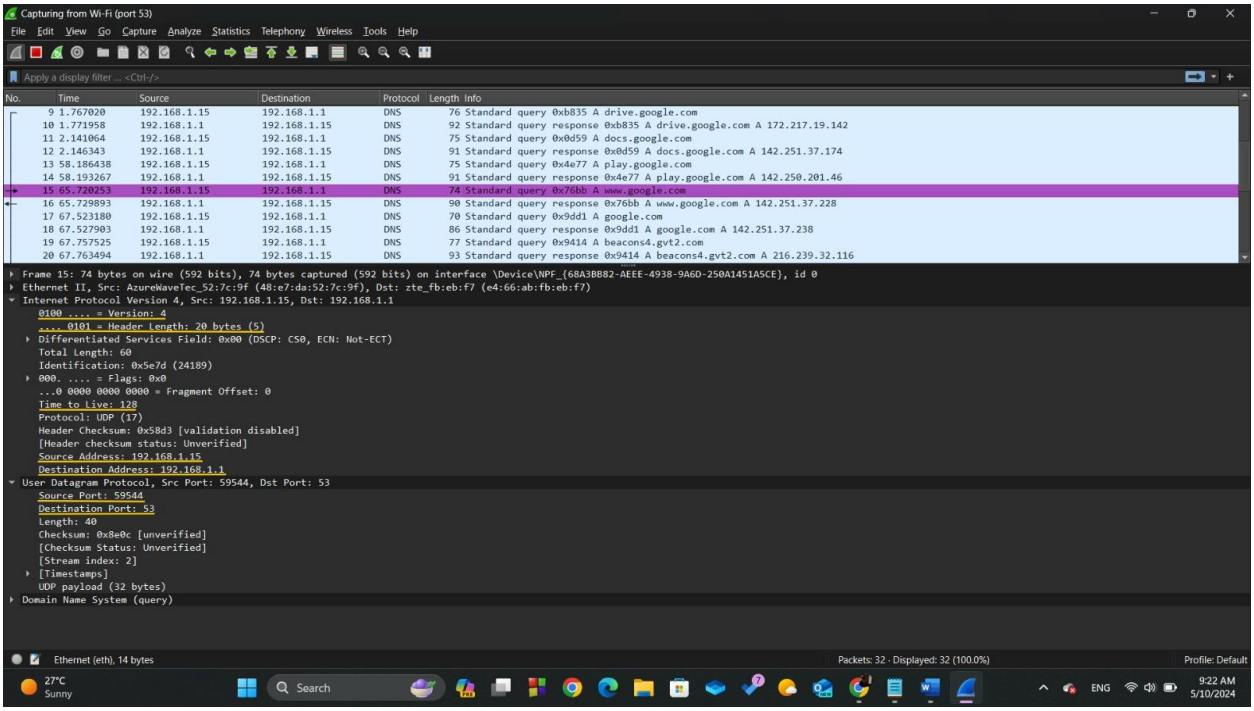
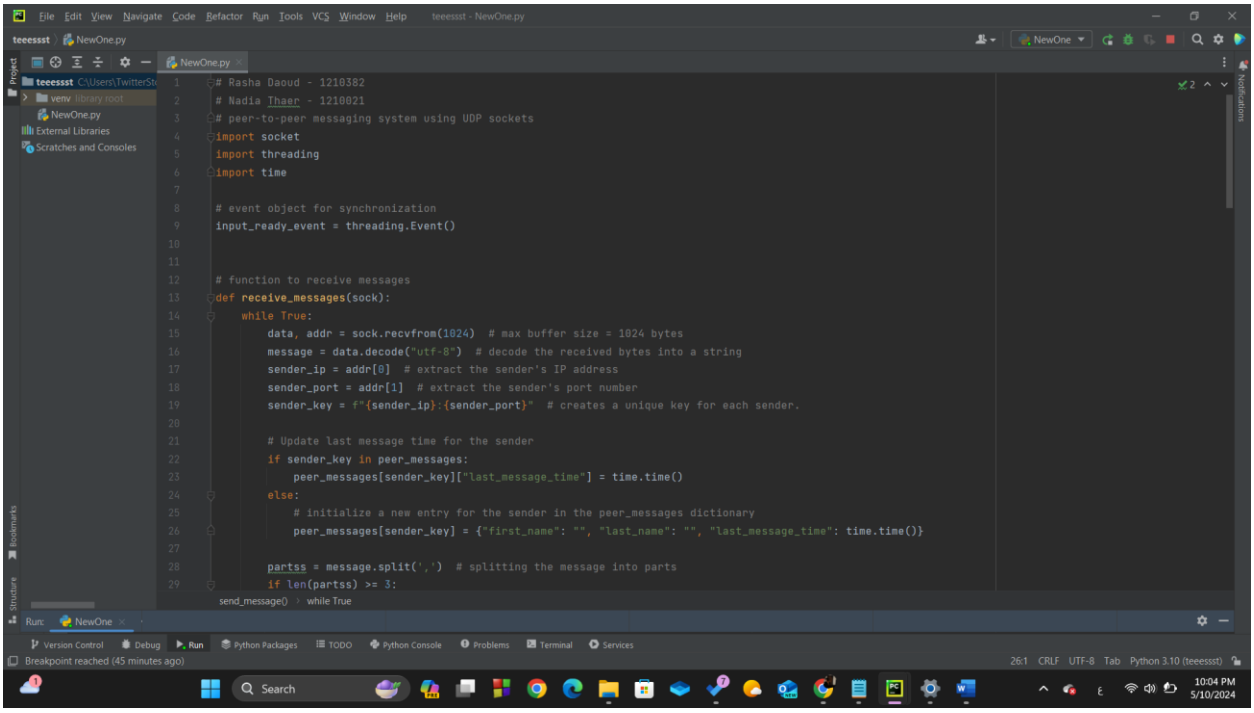


Figure 5:Result From Wireshark.

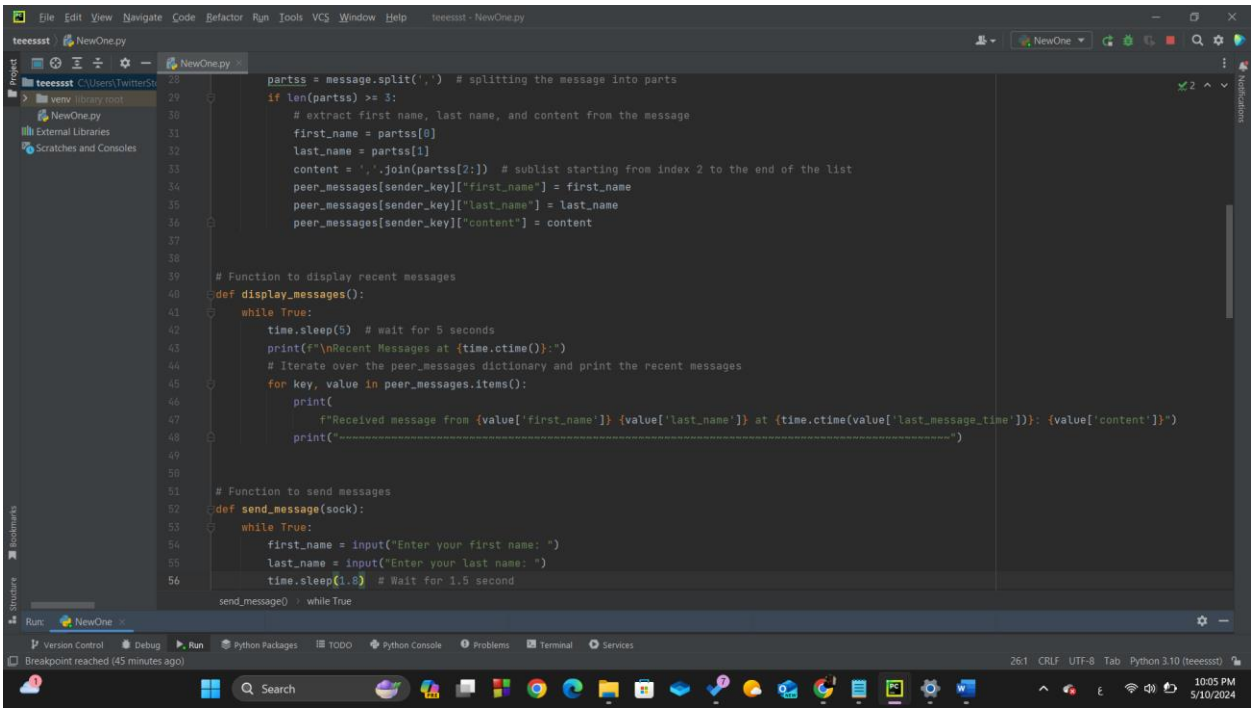
1. Version: the internet protocol addressing version is IPV4.
2. Header Length: the total length of the header is 20 bytes.
3. TTL: The packet can travel through 128 routers.
4. Source Address: the local IP address of my PC = 192.168.1.15.
5. Destination Address: the IP address of the destination = 192.168.1.1.
6. Source Port: the source port of the packets = 59544
7. Destination Port: the destination port of the packets = 53.

Part 2



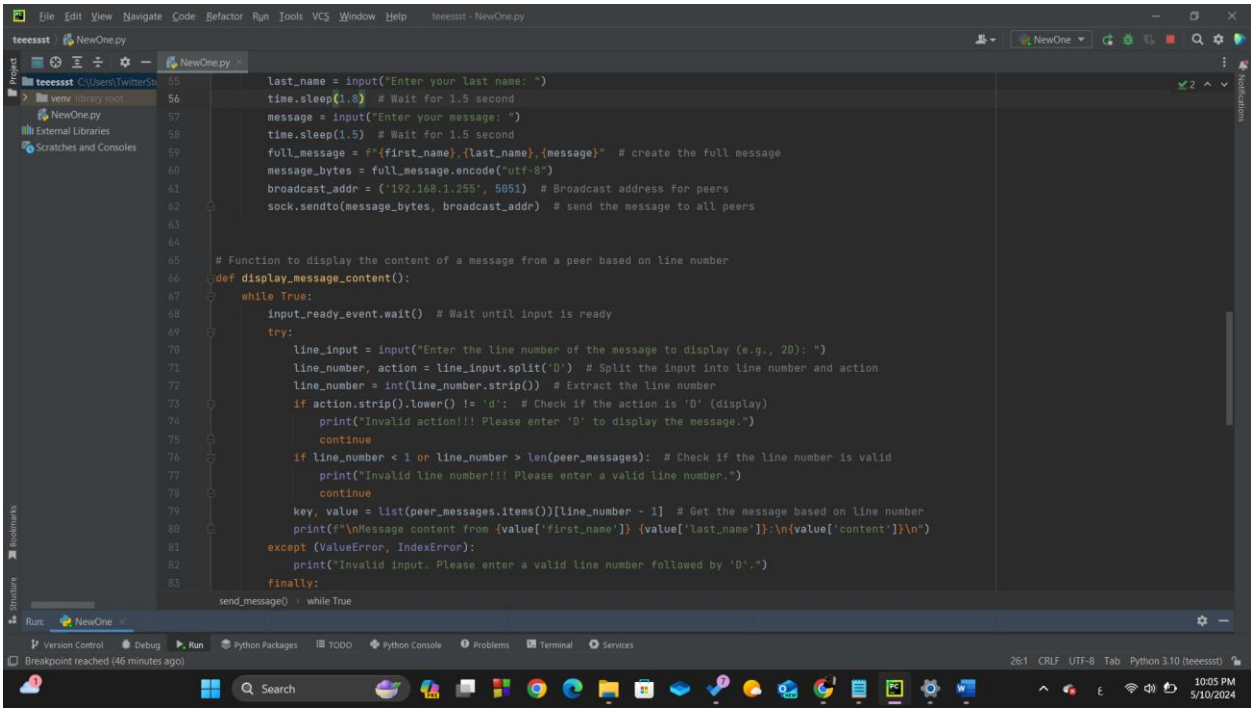
```
# Rasha Daoud - 1210382
# Nadia Thaer - 1210021
# peer-to-peer messaging system using UDP sockets
import socket
import threading
import time

# event object for synchronization
input_ready_event = threading.Event()


# function to receive messages
def receive_messages(sock):
    while True:
        data, addr = sock.recvfrom(1024)  # max buffer size = 1024 bytes
        message = data.decode("utf-8")  # decode the received bytes into a string
        sender_ip = addr[0]  # extract the sender's IP address
        sender_port = addr[1]  # extract the sender's port number
        sender_key = f"{sender_ip}:{sender_port}"  # creates a unique key for each sender.

        # Update last message time for the sender
        if sender_key in peer_messages:
            peer_messages[sender_key]["last_message_time"] = time.time()
        else:
            # initialize a new entry for the sender in the peer_messages dictionary
            peer_messages[sender_key] = {"first_name": "", "last_name": "", "last_message_time": time.time()}

        partss = message.split(',')  # splitting the message into parts
        if len(partss) >= 3:
```

Figure 6:part 2 Code .



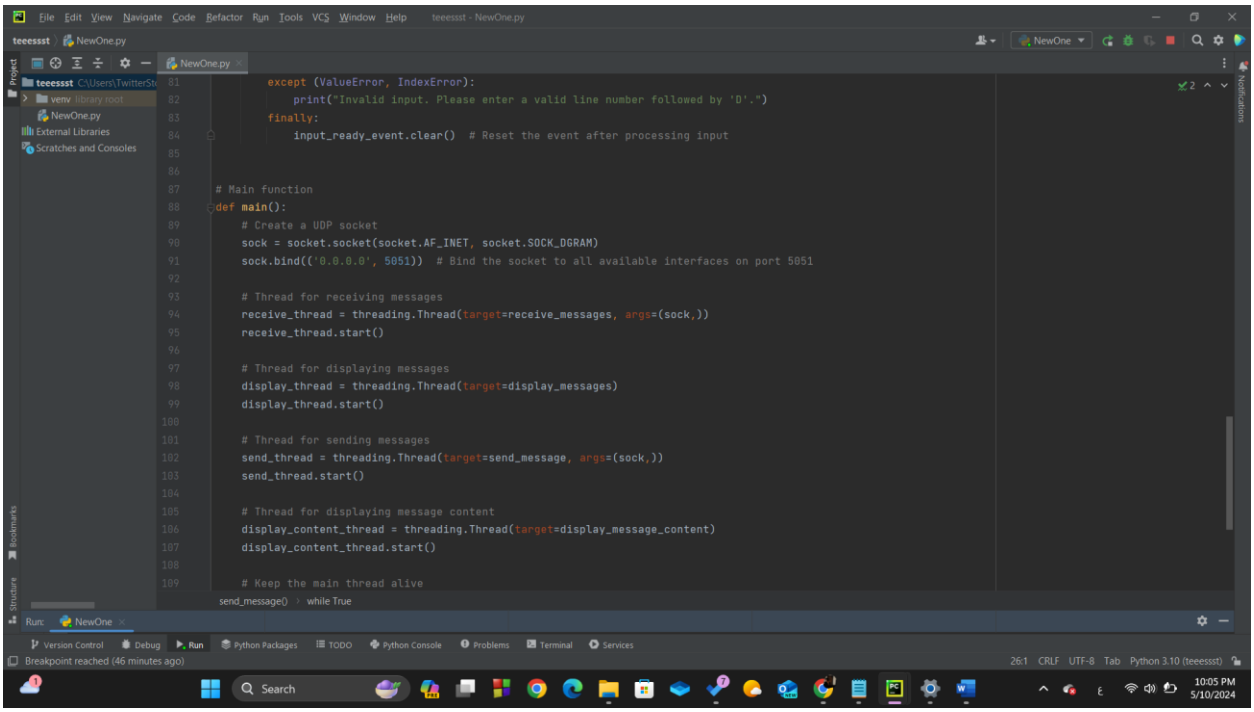
```
        partss = message.split(',')  # splitting the message into parts
        if len(partss) >= 3:
            # extract first name, last name, and content from the message
            first_name = partss[0]
            last_name = partss[1]
            content = ','.join(partss[2:])  # sublist starting from index 2 to the end of the list
            peer_messages[sender_key]["first_name"] = first_name
            peer_messages[sender_key]["last_name"] = last_name
            peer_messages[sender_key]["content"] = content


# Function to display recent messages
def display_messages():
    while True:
        time.sleep(5)  # wait for 5 seconds
        print(f"\nRecent Messages at {time.ctime()}:")
        # Iterate over the peer_messages dictionary and print the recent messages
        for key, value in peer_messages.items():
            print(
                f"Received message from {value['first_name']} {value['last_name']} at {time.ctime(value['last_message_time'])}: {value['content']}")
            print("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~")


# Function to send messages
def send_message(sock):
    while True:
        first_name = input("Enter your first name: ")
        last_name = input("Enter your last name: ")
        time.sleep(1.8)  # Wait for 1.5 second
```

Figure 7:part2 code



```
        last_name = input("Enter your last name: ")
        time.sleep(1.8)  # Wait for 1.5 second
        message = input("Enter your message: ")
        time.sleep(1.5)  # Wait for 1.5 second
        full_message = f"{first_name},{last_name},{message}"  # create the full message
        message_bytes = full_message.encode("utf-8")
        broadcast_addr = ('192.168.1.255', 5051)  # Broadcast address for peers
        sock.sendto(message_bytes, broadcast_addr)  # send the message to all peers


# Function to display the content of a message from a peer based on line number
def display_message_content():
    while True:
        input_ready_event.wait()  # Wait until input is ready
        try:
            line_input = input("Enter the line number of the message to display (e.g., 2D): ")
            line_number, action = line_input.split('D')  # Split the input into line number and action
            line_number = int(line_number.strip())  # Extract the line number
            if action.strip().lower() != 'd':  # Check if the action is 'D' (display)
                print("Invalid action!!! Please enter 'D' to display the message.")
                continue
            if line_number < 1 or line_number > len(peer_messages):  # Check if the line number is valid
                print("Invalid line number!!! Please enter a valid line number.")
                continue
            key, value = list(peer_messages.items())[line_number - 1]  # Get the message based on line number
            print(f"\nMessage content from {value['first_name']} {value['last_name']}:\n{value['content']}\n")
        except (ValueError, IndexError):
            print("Invalid input. Please enter a valid line number followed by 'D'.")
        finally:
```

Figure 8:part 2 Code.



```
        except (ValueError, IndexError):
            print("Invalid input. Please enter a valid line number followed by 'D'.")
        finally:
            input_ready_event.clear()  # Reset the event after processing input


# Main function
def main():
    # Create a UDP socket
    sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    sock.bind(('0.0.0.0', 5051))  # Bind the socket to all available interfaces on port 5051

    # Thread for receiving messages
    receive_thread = threading.Thread(target=receive_messages, args=(sock,))
    receive_thread.start()

    # Thread for displaying messages
    display_thread = threading.Thread(target=display_messages)
    display_thread.start()

    # Thread for sending messages
    send_thread = threading.Thread(target=send_message, args=(sock,))
    send_thread.start()

    # Thread for displaying message content
    display_content_thread = threading.Thread(target=display_message_content)
    display_content_thread.start()

    # Keep the main thread alive
```

Figure 9:part 2 code.

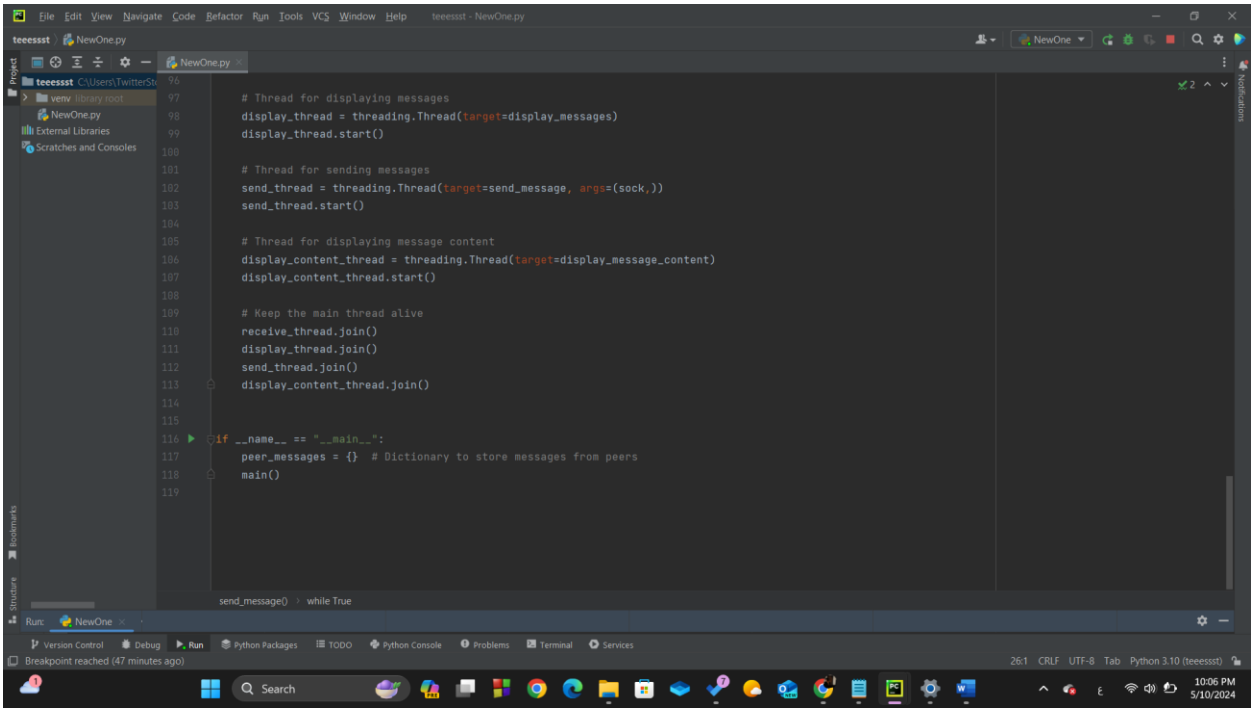


Figure 10:part 2 Code.

Explaination The Whole Code :

the provided code creates a simple peer-to-peer messaging system using UDP sockets in Python. It allows users to send and receive messages between multiple peers over a local network. The code consists of four main functions: receiving messages, displaying recent messages, sending messages, and displaying the content of a specific message. Each function runs in its own thread to handle concurrent operations. When a message is received, the sender's information and message content are stored in a dictionary. The program continuously displays recent messages and prompts users to send new messages or view specific message contents. Additionally, it binds to a specific IP address and port to listen for incoming messages and uses a broadcast address to send messages to all peers on the network.

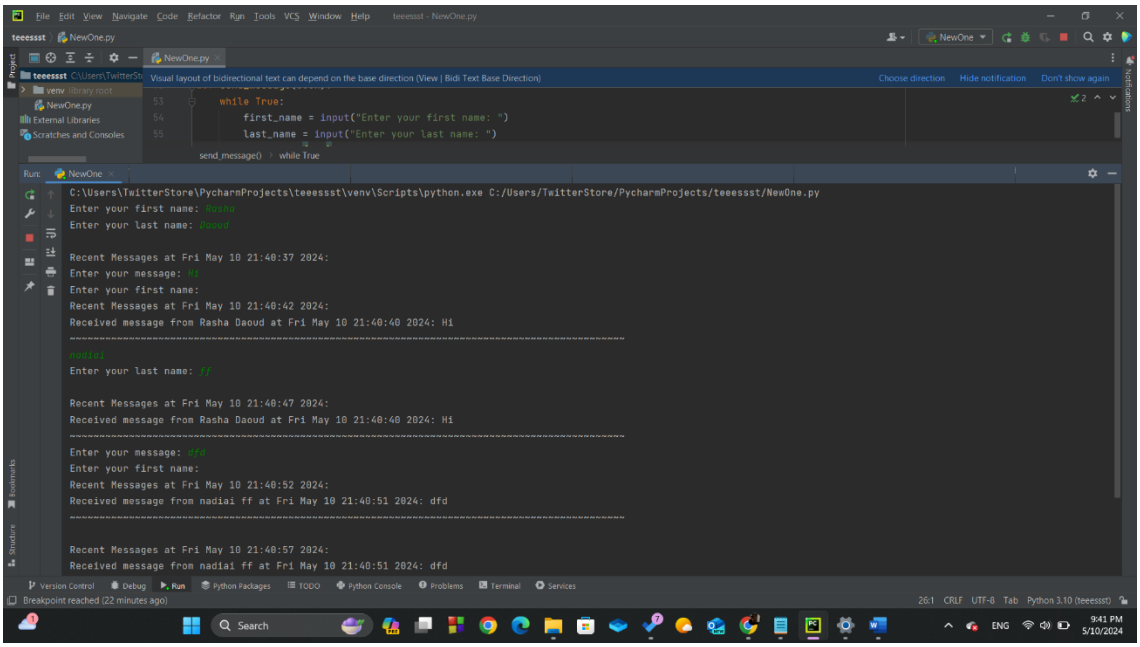


Figure 11: The output of part 2.

Part 3

3.0. Entity Tag Cache Vaildators in The HTTp protocol

Entity Tag (ETag) cache validators in the HTTP protocol are mechanisms used for cache validation. They provide a way for a client to check if its cached representation of a resource is still valid. ETags are opaque identifiers generated by the server to represent a specific version of a resource.

So According to This information , Entity Tag cache validators enhance the efficiency and reliability of caching mechanisms in the HTTP protocol, improving performance and reducing the load on servers by minimizing unnecessary data transfer.

3.1 Code and Explaination

3.1.1 The main.py

Code For the web Server → server process in Socket at server:

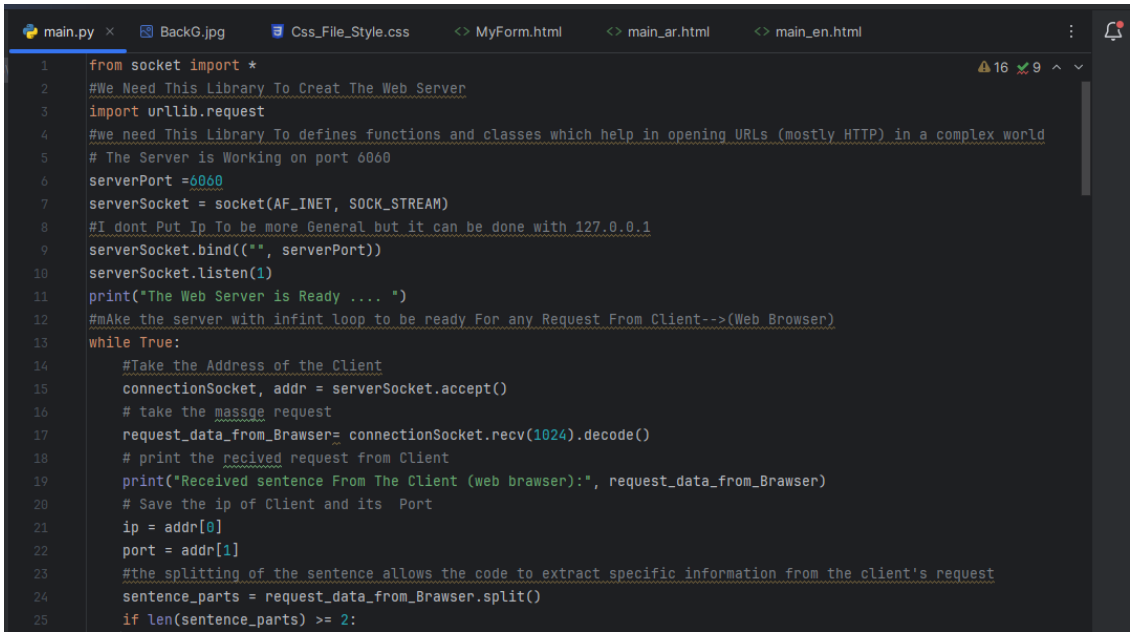


Figure 12:part(3)→Code of mainServer (1).

```
if len(sentence_parts) >= 2:
    #so the object will be -->/resourses-->the place where the server will give me the response
    object = sentence_parts[1]
    print("Requested object:", object)
```

Figure 13:part (3)→code of mainSer (2).

Notice : Each Line in The Code There is An Explanation .

```
30      if object == '/' or object == '/index.html' or object == '/main_en.html' or object == '/en':
31          connectionSocket.send("HTTP/1.1 200 OK \r\n".encode())
32          connectionSocket.send("Content-Type: text/html \r\n".encode())
33          connectionSocket.send("\r\n".encode())
34          file1 = open("main_en.html", "rb")
35          connectionSocket.send(file1.read())
36
37      elif object == '/ar' or object=='/main_ar.html':
38          connectionSocket.send("HTTP/1.1 200 OK \r\n".encode())
39          connectionSocket.send("Content-Type: text/html \r\n".encode())
40          connectionSocket.send("\r\n".encode())
41          file2 = open("main_ar.html", "rb")
42          connectionSocket.send(file2.read())
43
44      elif object.endswith('.html'):
45          connectionSocket.send("HTTP/1.1 200 OK \r\n".encode())
46          connectionSocket.send("Content-Type: text/html \r\n".encode())
47          connectionSocket.send("\r\n".encode())
48          file3 = open("main_en.html", "rb")
49          connectionSocket.send(file3.read())
```

Figure 14: part 3 →mainSer code (3).

- In the fiest if statement → the request is **/ or /index.html or /main_en.html or /en (for example localhost:6060/ or localhost:6060/en)** then the server should send **main_en.html** file with Content-Type: text/html.
- In the secound if statement→ if the request **is /ar or main_ar** then the server send the main_ar file
- In the thired if statement →if the request end with **.html then** the server will display the main_en.html this mean any request Ended with .html it will display the same file and this appear in our result if we request an **rasha.html** the file response is main_en.

```
50      elif object.endswith('.css'):
51          connectionSocket.send("HTTP/1.1 200 OK \r\n".encode())
52          connectionSocket.send("Content-Type: text/css \r\n".encode())
53          connectionSocket.send("\r\n".encode())
54          file4 = open("Css_File_Style.css", "rb")
55          connectionSocket.send(file4.read())
56
57      elif object.endswith('.png'):
58          url = "https://e7.pngegg.com/pngimages/423/224/png-clipart-desktop-high-definition-television-forest-1080p-
59          headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) (
60          requestFromClient = urllib.request.Request(url, headers=headers)
61          content=urllib.request.urlopen(requestFromClient).read()
62          connectionSocket.send("HTTP/1.1 200 OK \r\n".encode())
63          connectionSocket.send("Content-Type: image/png \r\n".encode())
64          connectionSocket.send("\r\n".encode())
65          connectionSocket.send(content)
```

Figure 15: part 3->the mainServer code (4)

- the fourth if statement → if the request end with **.css then** the server will send response the **Css_File_Style.css** this mean any request Ended with .css it will send the same response of file and this appear in our result if we request an **c.css** the opened file will be the **Css_File_Styel.css** in This Methode it will be the request By The Client (web brawser) be more General .
- The Fifth if statement → if the request end with **.png then** the server will send response the **image will take from the url website** this mean any request Ended with .png it will send the same response of image and this appear in our result if we request an **p.png** the opened image will be the **image from the url** in This Methode it will be the request By The Client (web brawser) be more General .
- **Explaine the Code of The Fifth if statement** :

This code fetches an image from a specified URL and sends it as a response to a client's request. It sets up a request to mimic a web browser, retrieves the image content, and sends it back to the client over a network connection i.

- **Notice** :In Our Code we need to use the set up the request to mimic web browser When I do it Like jpg , it Appear Error we Don't why This Error Happen .

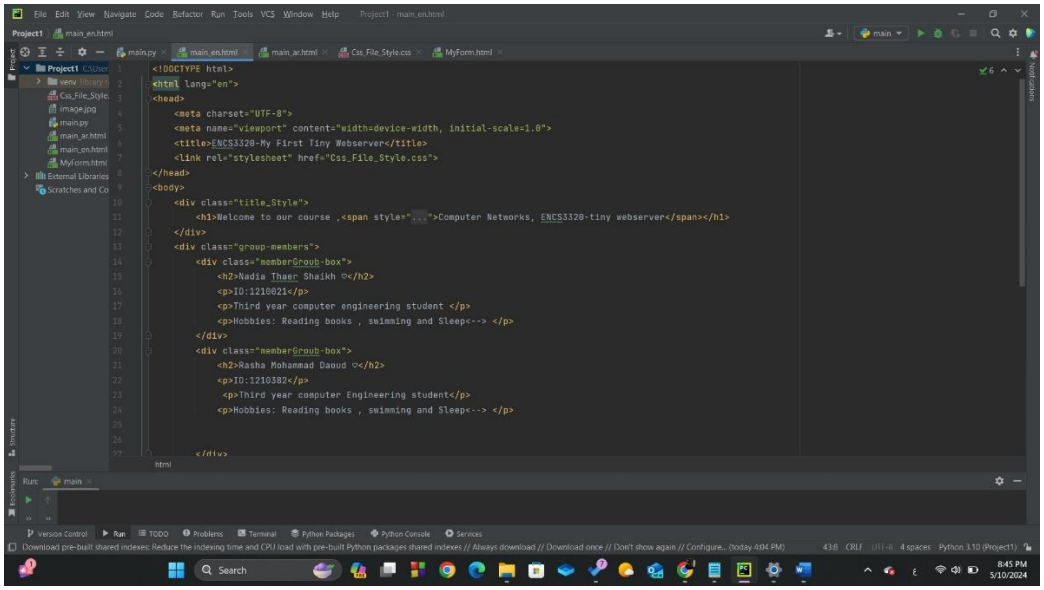
```
67          elif object.endswith('.jpg'):                                    ⚠ 16  ✓ 9
68              url = "https://www.w3schools.com/w3css/img_forest.jpg"
69              connectionSocket.send("HTTP/1.1 200 OK\r\n".encode())
70              connectionSocket.send("Content-Type: image/jpg \r\n".encode())
71              connectionSocket.send("\r\n".encode())
72              urllib.request.urlretrieve(url, filename: "image.jpg")
73              # Open the local image file and send its content
74              fileó = open("image.jpg", "rb")
75              connectionSocket.send(fileó.read())
76
77          elif object == '/so':
78              connectionSocket.send("HTTP/1.1 307 Temporary Redirect \r\n".encode())
79              connectionSocket.send("Content-Type: text/html \r\n".encode())
80              connectionSocket.send("Location: https://stackoverflow.com \r\n".encode())
81              connectionSocket.send("\r\n".encode())
82
83          elif object == '/itc':
84              connectionSocket.send("HTTP/1.1 307 Temporary Redirect \r\n".encode())
85              connectionSocket.send("Content-Type: text/html \r\n".encode())
86              connectionSocket.send("Location: https://ritaj.birzeit.edu// \r\n".encode())
87              connectionSocket.send("\r\n".encode())
```

Figure 16:part (3) -->mainSer Code (4)

- In the Sixth If Statement → if the request end with **.jpg then** the server will send response the **image will take from the url website** this mean any request Ended with .jpg it will send the same response of image and this appear in our result if we request an **j.jpg** the opened image will be Download locally **from the url** and Save it in File Called →**image** in This Methode it will be the request By The Client (web brawser) be more General .
- In the Seventh If Statement →it will Display an the Stackovelflow .com
- In Eight if Statement →it will Display The Ritage

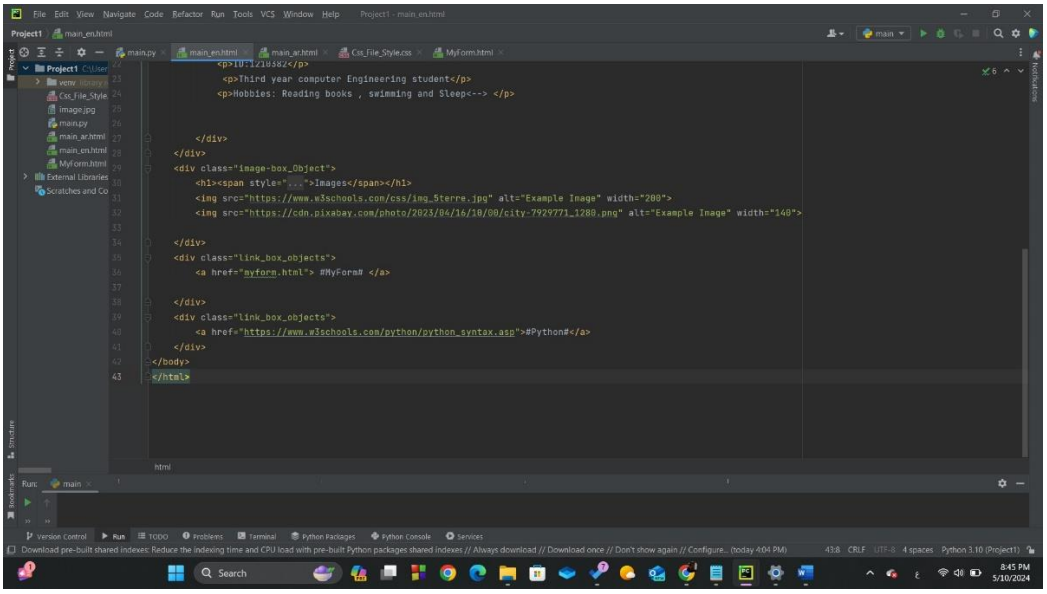
3.1.2 HTML and css Code

Maine_en HTML Code :



```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>ENCS3320-My First Tiny Webserver</title>
    <link rel="stylesheet" href="Css_File_Style.css">
</head>
<body>
    <div class="title_Style">
        <h1>Welcome to our course ,<span style="...">Computer Networks, ENCS3320-tiny webserver</span></h1>
    </div>
    <div class="group-members">
        <div class="memberGroub-box">
            <h2>Nadia Thaer Shaikh ♡</h2>
            <p>ID:1210021</p>
            <p>Third year computer engineering student </p>
            <p>Hobbies: Reading books , swimming and Sleep<--> </p>
        </div>
        <div class="memberGroub-box">
            <h2>Rasha Mohammad Daoud ♡</h2>
            <p>ID:1210382</p>
             <p>Third year computer Engineering student</p>
            <p>Hobbies: Reading books , swimming and Sleep<--> </p>
```

Figure 17:Main_en Code→> Part 1 .



```
            <p>ID:1210382</p>
             <p>Third year computer Engineering student</p>
            <p>Hobbies: Reading books , swimming and Sleep<--> </p>


        </div>
    </div>
    <div class="image-box_Object">
        <h1><span style="...">Images</span></h1>
        <img src="https://www.w3schools.com/css/img_5terre.jpg" alt="Example Image" width="200">
        <img src="https://cdn.pixabay.com/photo/2023/04/16/10/00/city-7929771_1280.png" alt="Example Image" width="140">

    </div>
    <div class="link_box_objects">
        <a href="myform.html"> #MyForm# </a>

    </div>
    <div class="link_box_objects">
        <a href="https://www.w3schools.com/python/python_syntax.asp">#Python#</a>
    </div>
</body>
</html>
```

Figure 18:Maine_en Cod -->Part 2

code is a basic HTML webpage that displays information about a course and its members. It includes elements like headings, paragraphs, and images. The title of the webpage is "ENCS3320-My First Tiny Webserver". It lists two group members with their names, IDs, and hobbies. There are also images displayed, along with links to other pages like "myform.html" and a Python syntax tutorial on W3Schools. The webpage uses simple styling with CSS to make it visually appealing. Overall, it's a

simple webpage showcasing information about a course and its members, with links to additional resources.

Maine _ar:

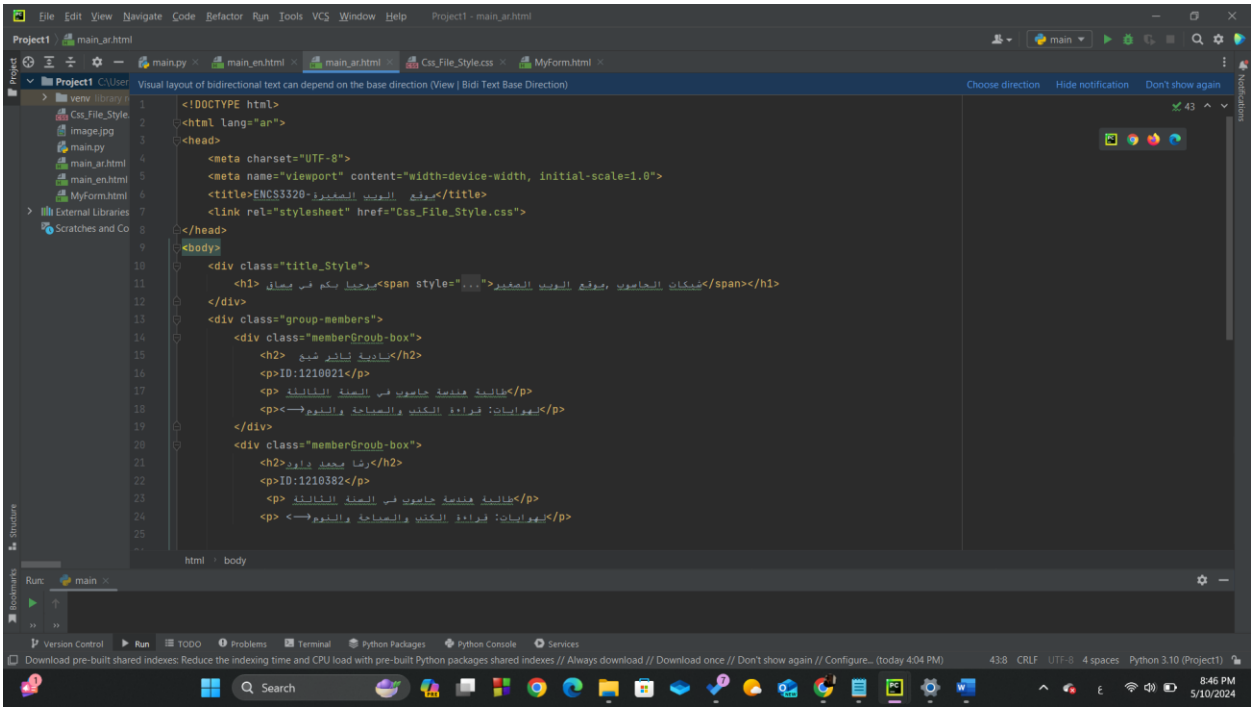


Figure 19:Main_ar.HTml Code 1

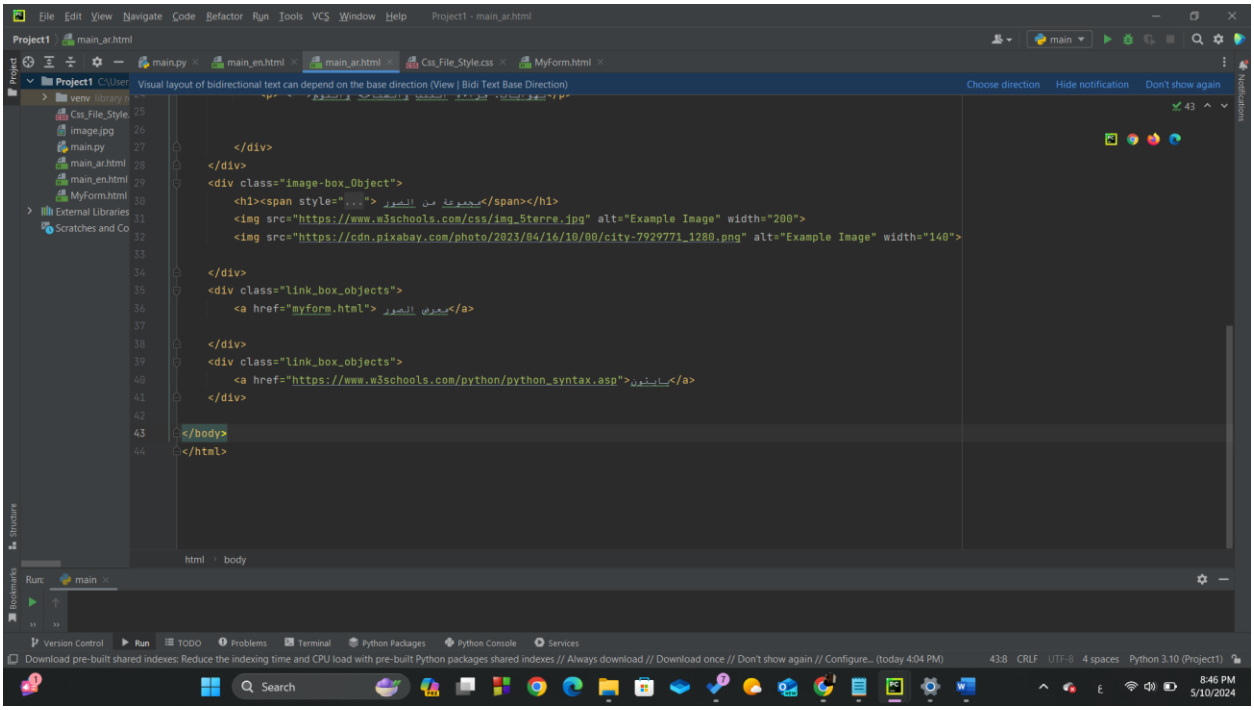


Figure 20:Main_ar.html code 2

Css Code :

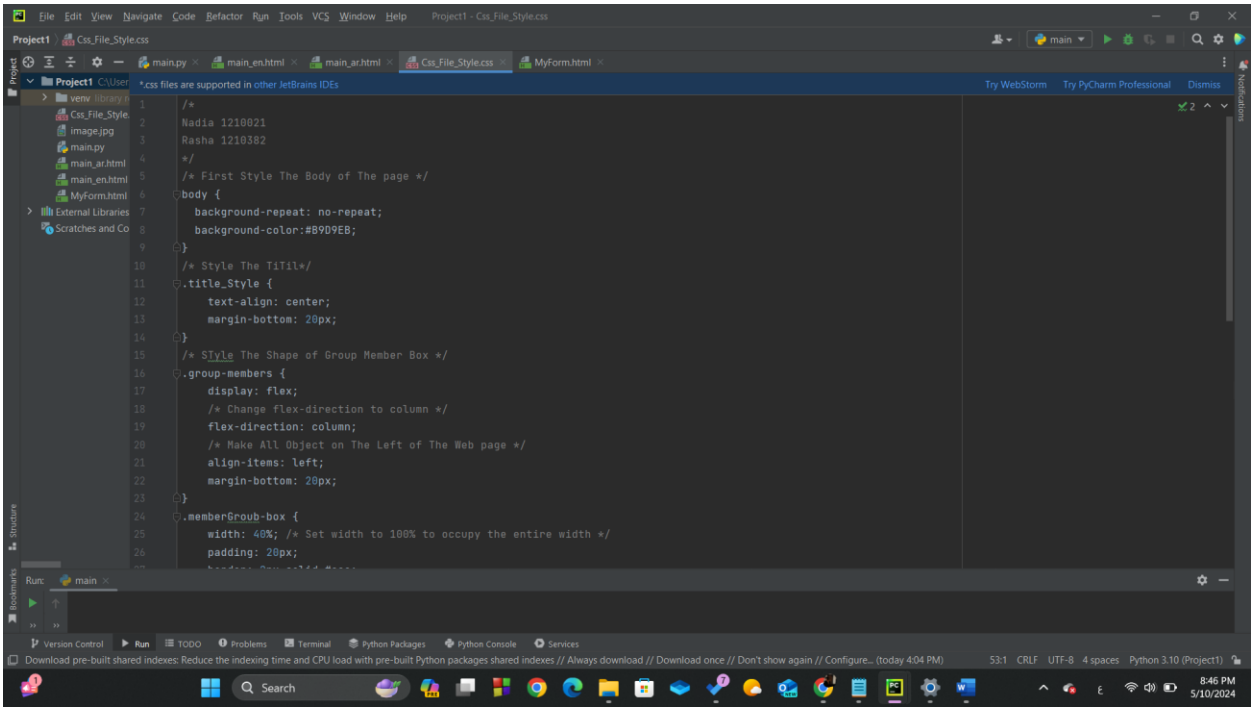


Figure 21:css Code 1

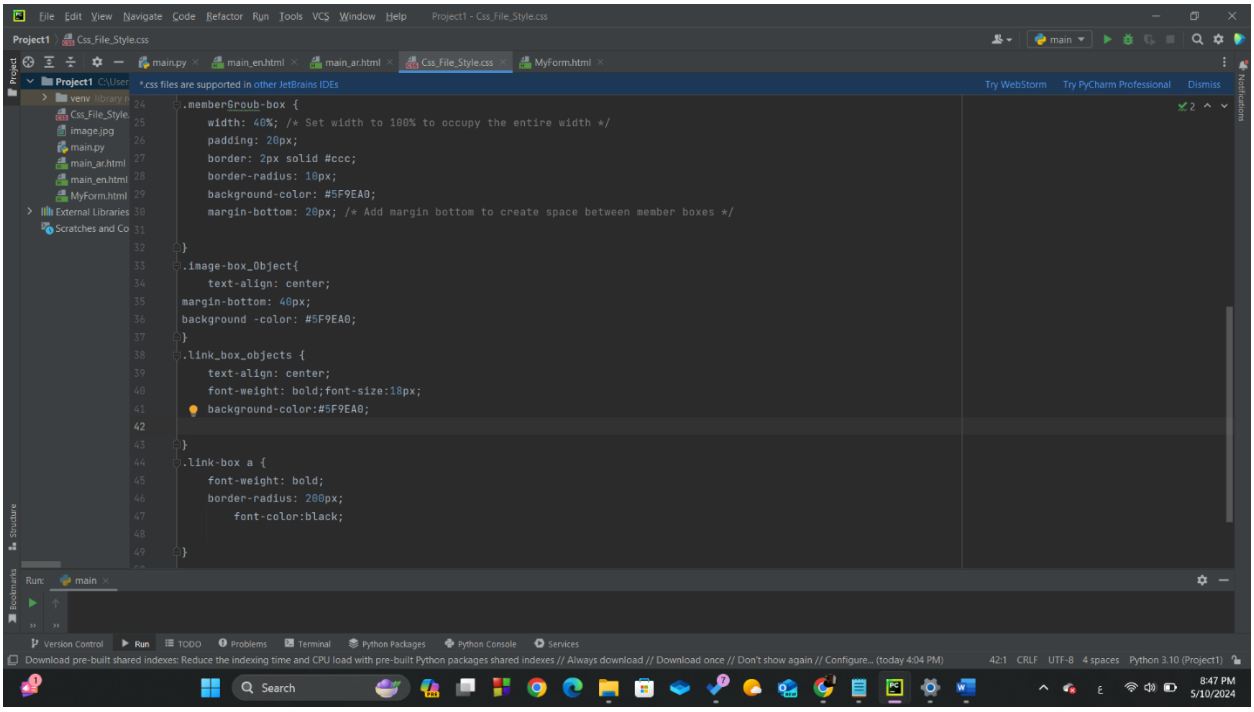


Figure 22css code 2:

My Form Code

main.py | BackG.jpg | Css_File_Style.css | MyForm.html | main_ar.html | main_en.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>MyForm</title>
    <style>
        body {
            background-color: #B9D9EB;
            width: 1000px;
            border-radius: 200px;
            display: flex;
            justify-content: center;
            align-items: center;
            height: 100vh;
            /* Center the container horizontally */
            margin: 0 auto;
        }
        form {
            text-align: center;
            width: 80%; /* Adjust the width of the form */
            max-width: 500px; /* Set maximum width */
        }
        /* Design The Box when i will put The Request */
        input[type="text"] {
            width: 100%;
```

Figure 23:My Form -->1

```
        /* Design The Display Button */
        input[type="submit"] {
            padding: 50px 50px;
            font-size: medium;
        }
    </style>
</head>
<body>
    <h2>Image Name </h2>
    <form action="/get_image" method="get">
        <label for="image">Image:</label><br>
<input type="text" id="image" name="image" required><br><br>
<input type="submit" value="Get Image">
    </form>
</body>
</html>
```

46:1 CRLF UTF-8 4 spaces Python 3.12

Links ENG 10:10 م

Figure 24:MyFor code-->2

3.2. if the request is / or /index.html or /main_en.html or /en (for example localhost:6060/ or localhost:6060/en)

3.2.1 If I request /index.html

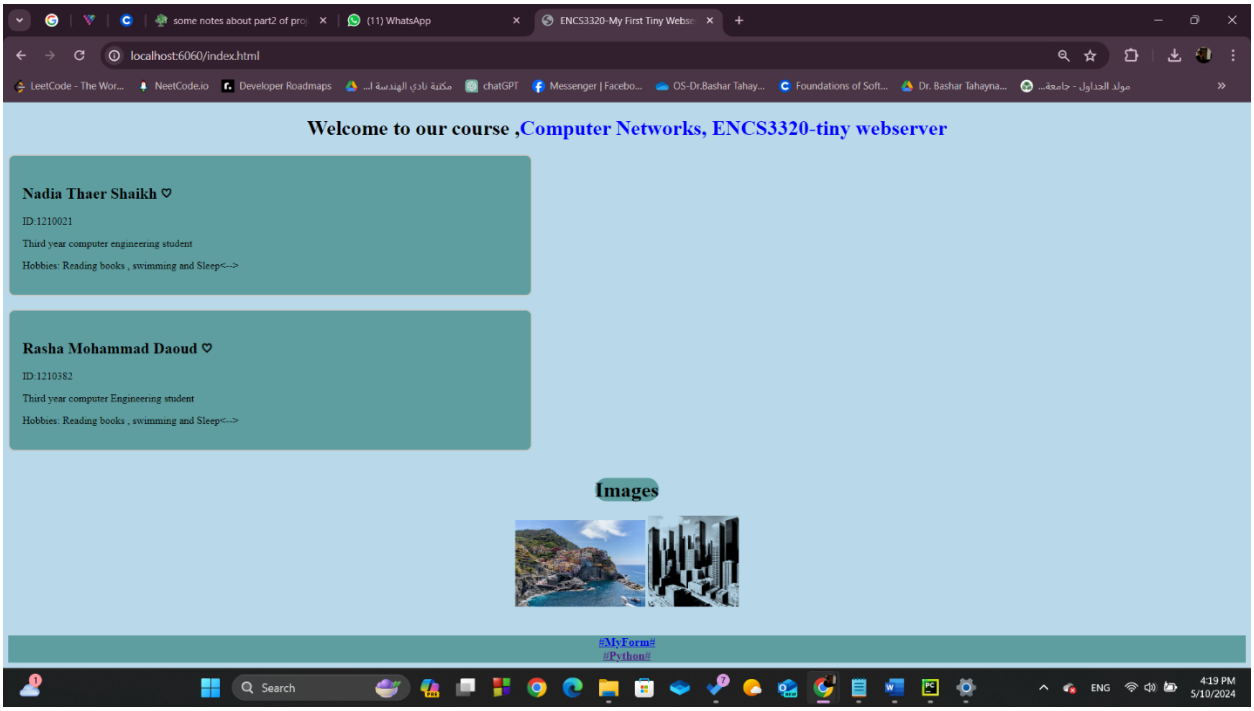


Figure 25:request /index.html Result.

The request massege :

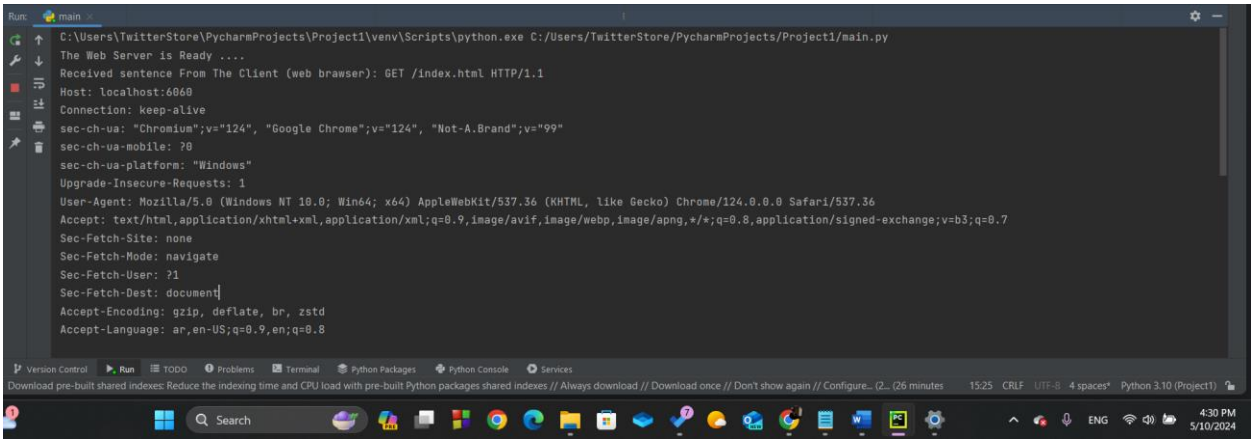


Figure 26:request /index.html Http request.

3.2.2 request / main_en.html

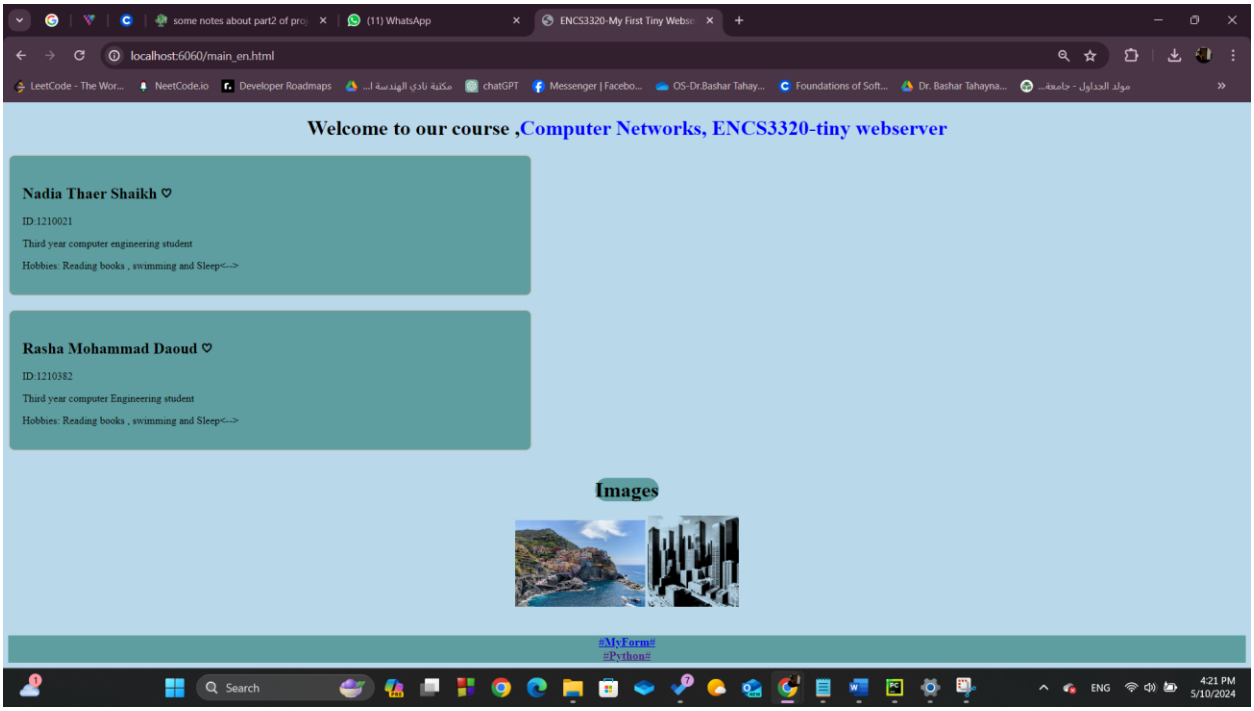


Figure 27:request / main_en.html Result.

The requasred massge HTTP:

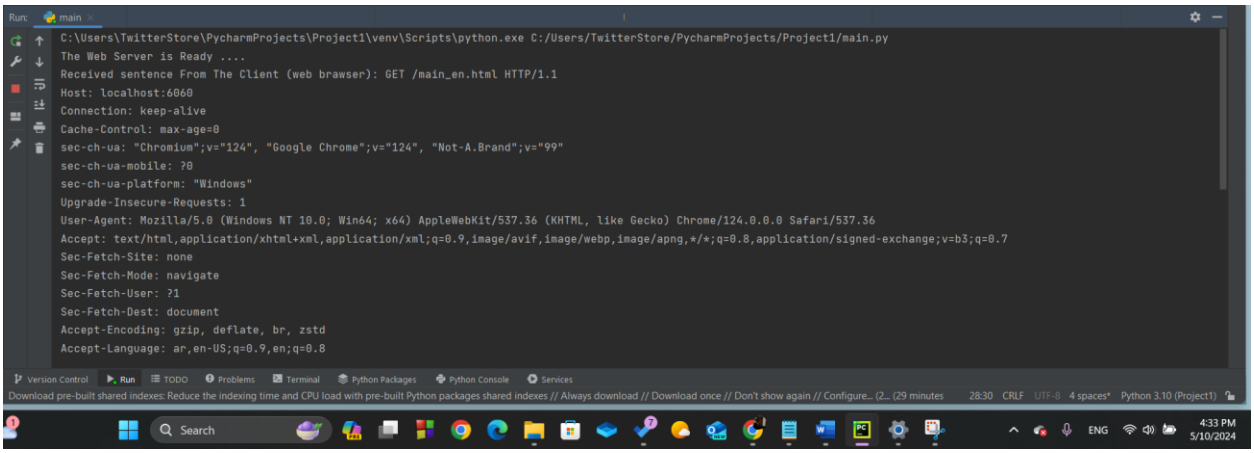


Figure 28:request / main_en.html Http request .

3.2.3 request /en

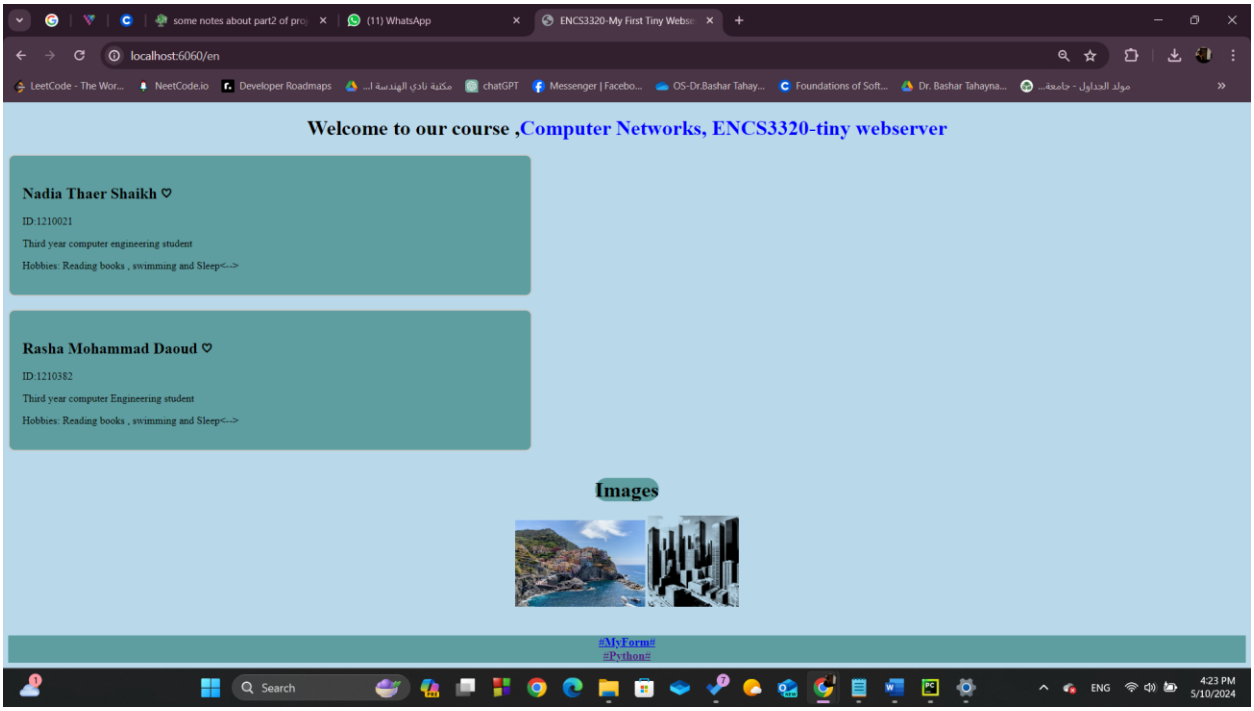


Figure 29:request /en Result .

The HTTP request :

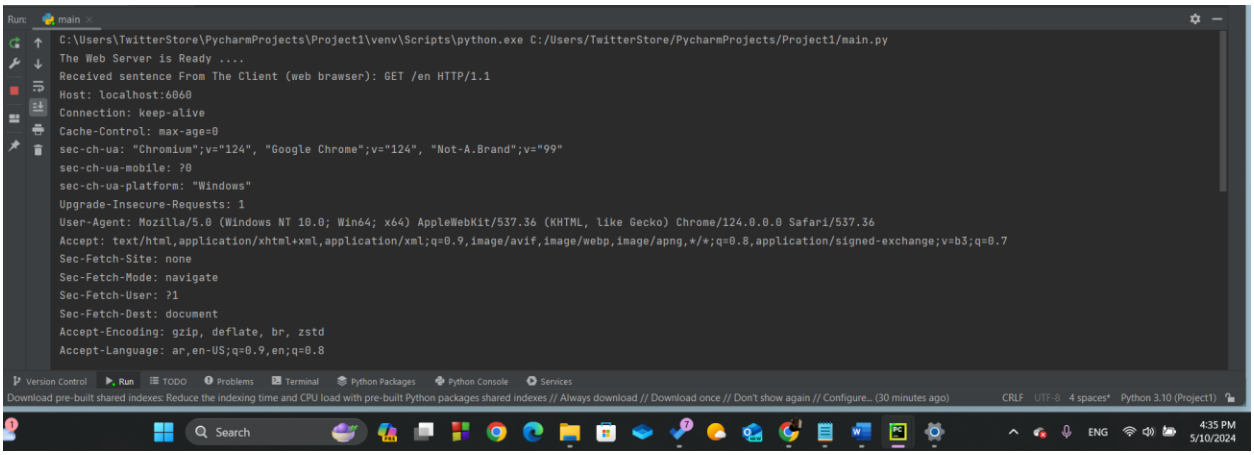


Figure 30:request /en Http Request Massge .

3.2.4 . request /

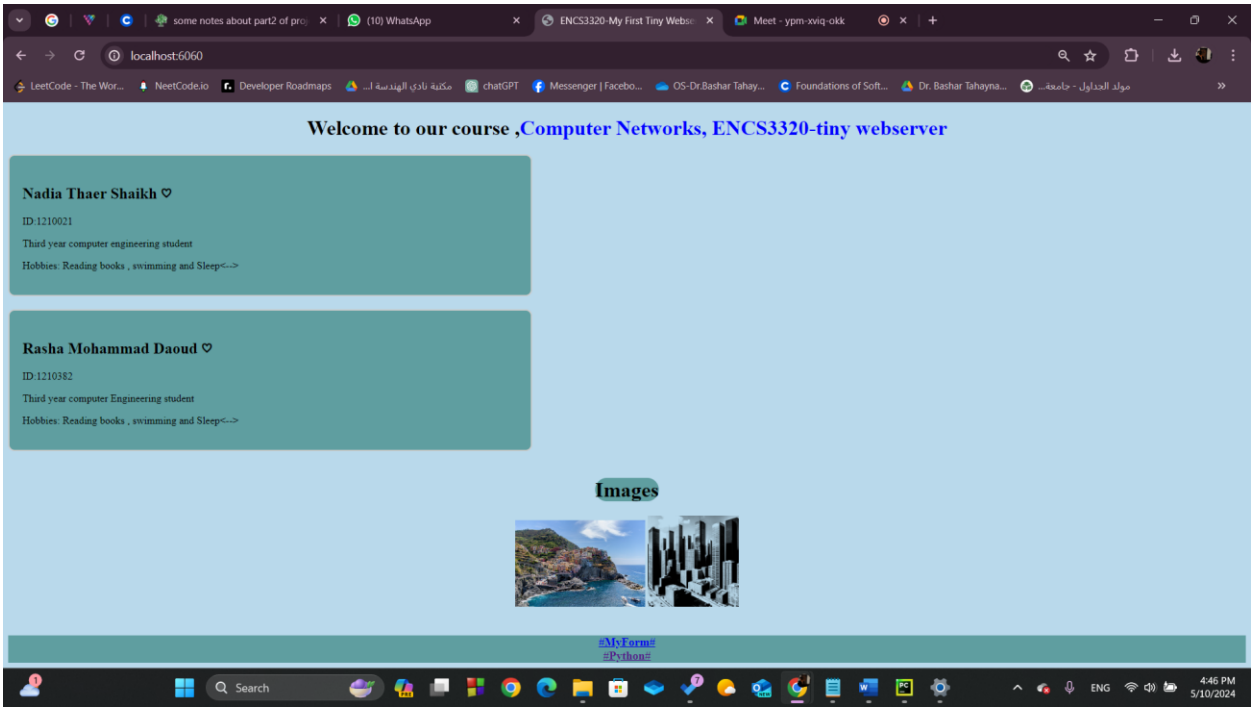


Figure 31:request / Result

The HTTP request :

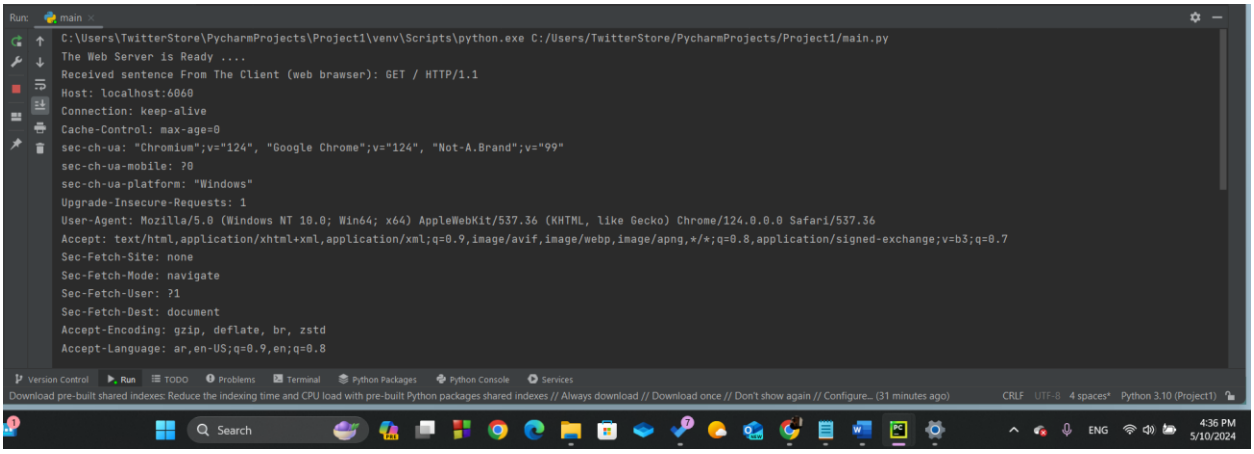


Figure 32:request / Http request massge .

3.3. Requsted /ar and main_ar

3.3.1 requeste the /ar

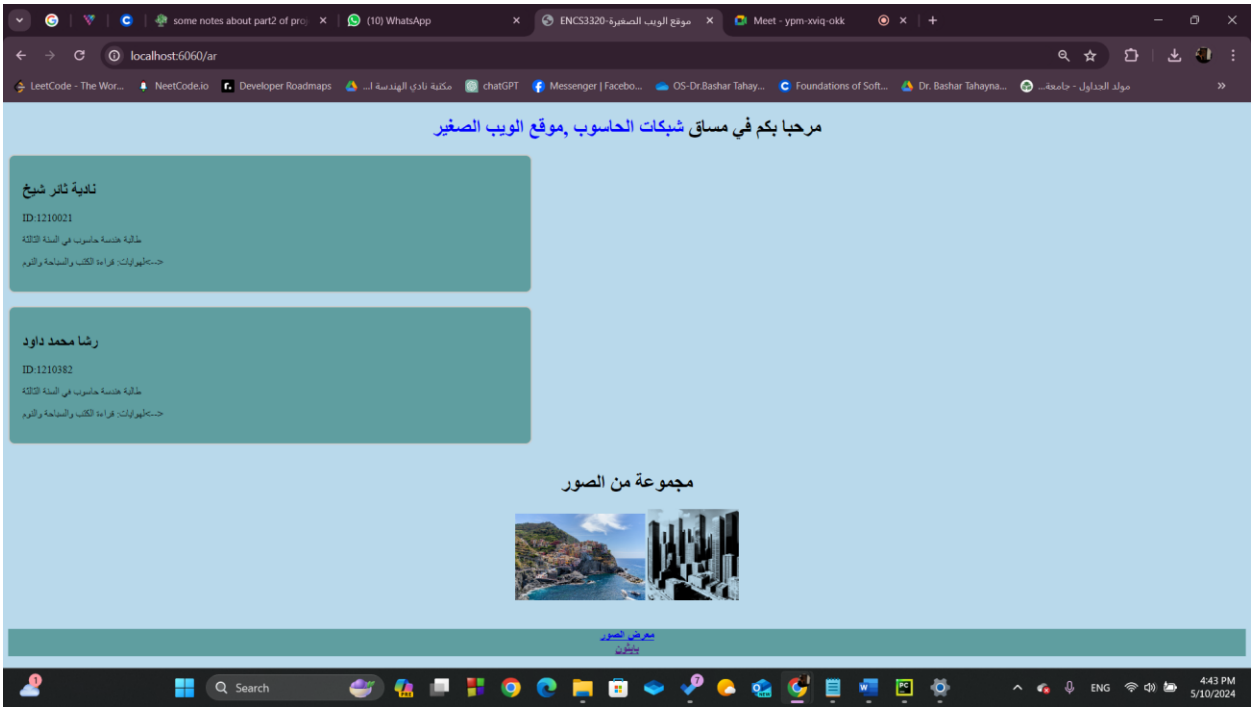


Figure 33:requeste the /ar Result .

The HTTP request:

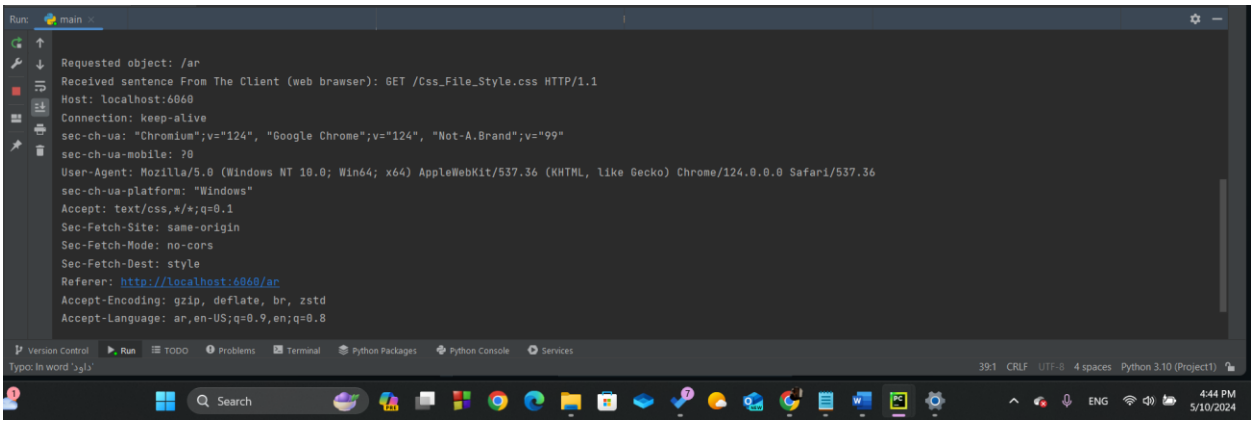


Figure 34:The Http request For -->requeste the /ar

3.3.2 request the /main_ar

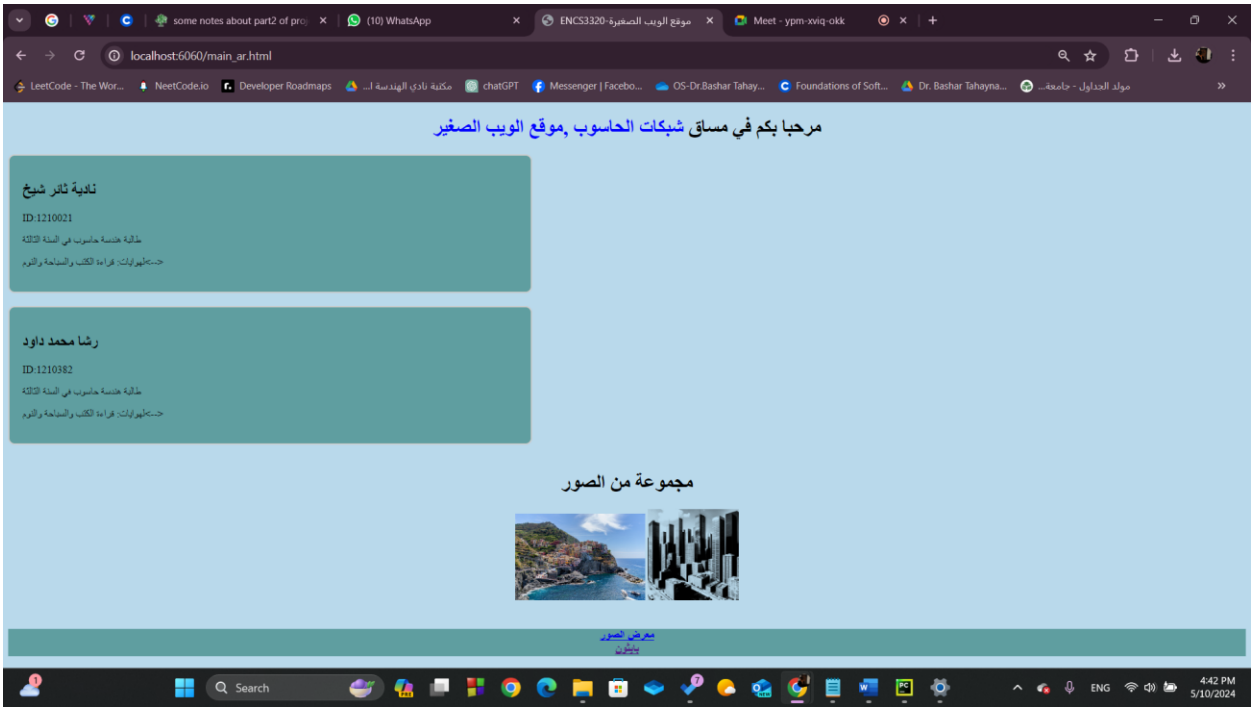


Figure 35: request the /main_ar Result.

The HTTP request

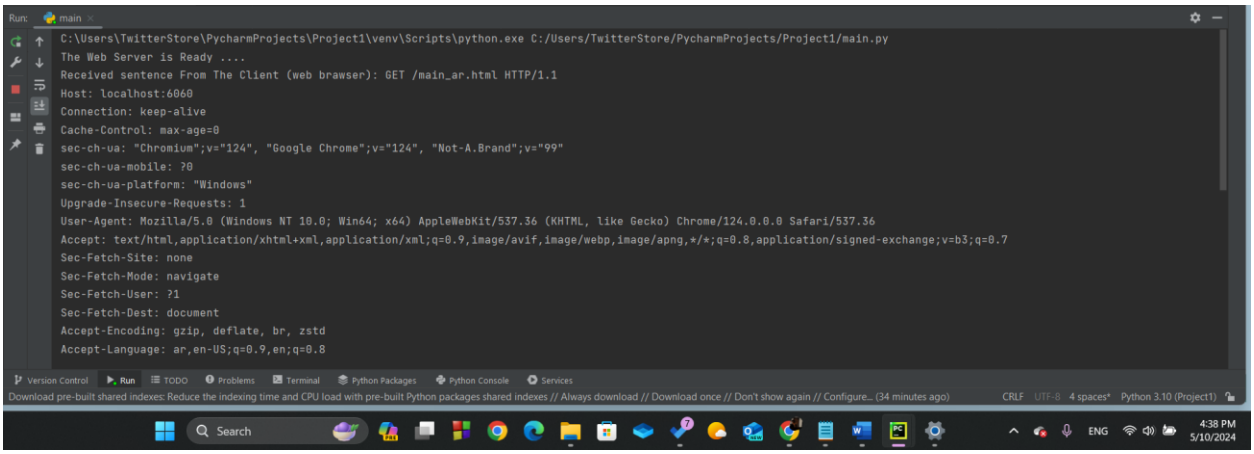


Figure 36: request the /main_ar Http request massge .

3.4 if the request Was rasha.html

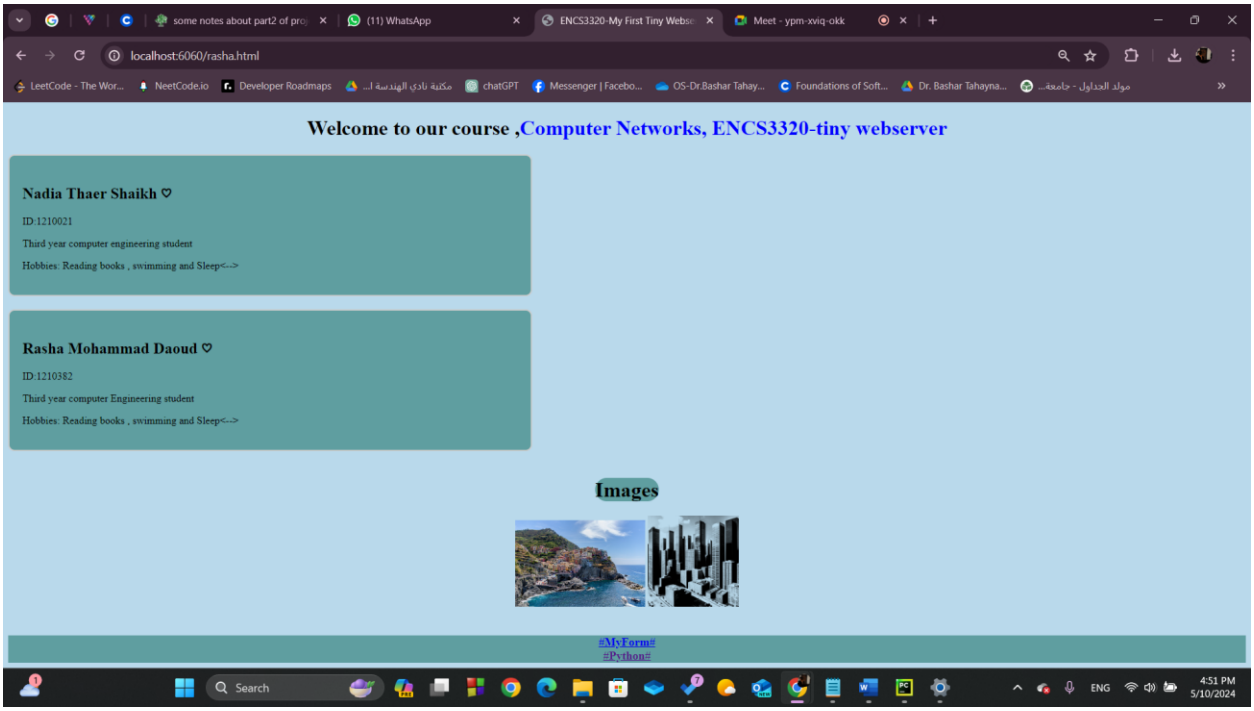


Figure 37:the request Was rasha.html Result .

The Http Request :

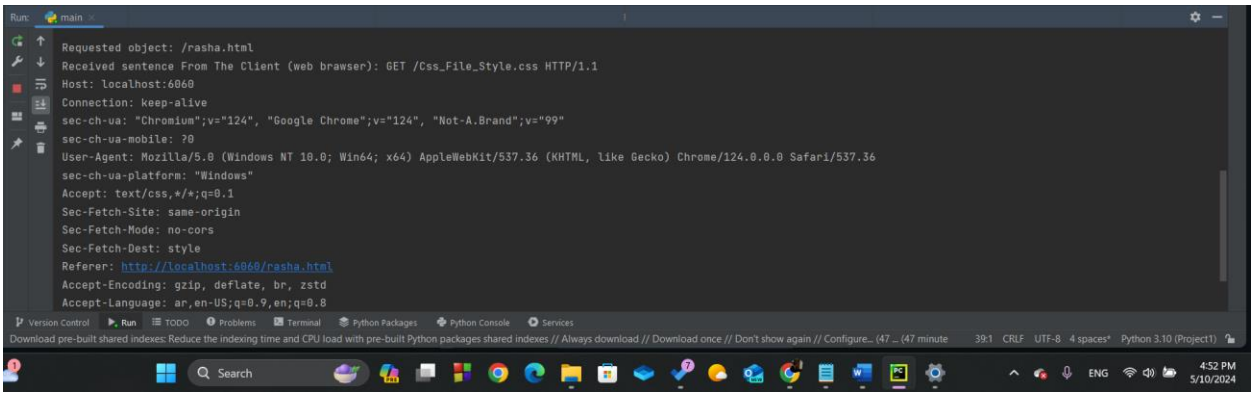


Figure 38:the request Was rasha.html Request Massge.

3.5 .if request c.css

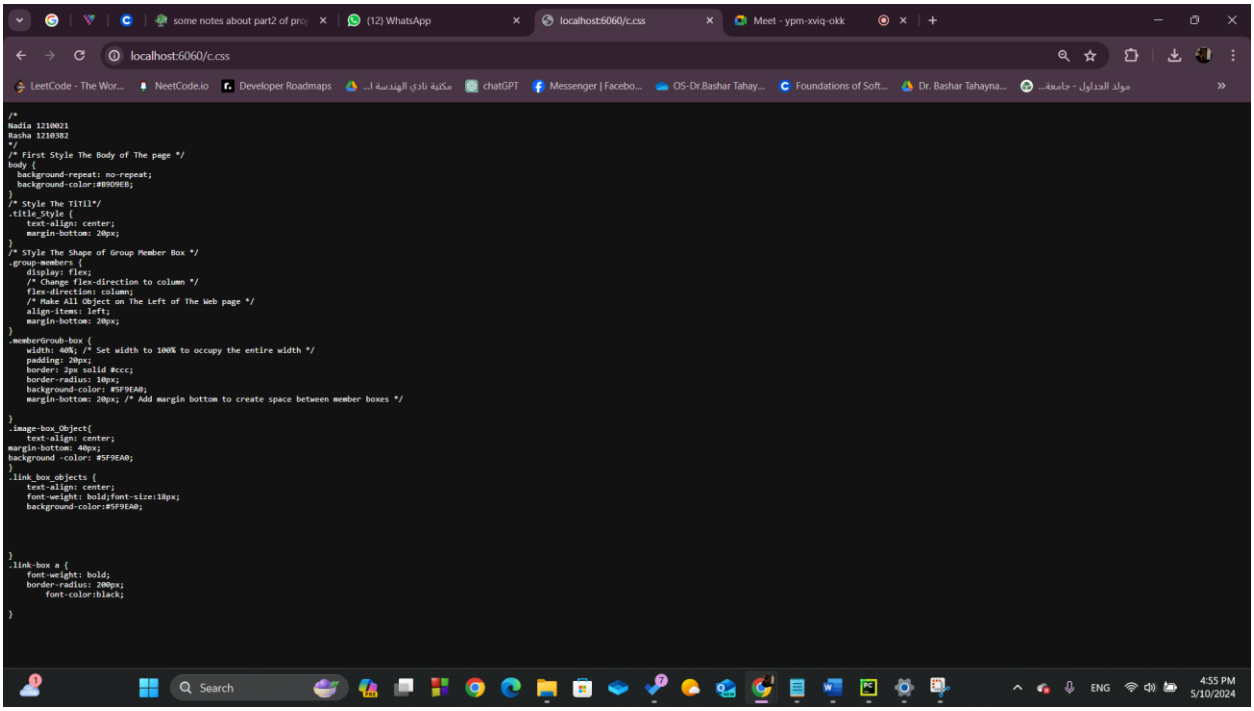


Figure 39:request c.css Result .

The Http request

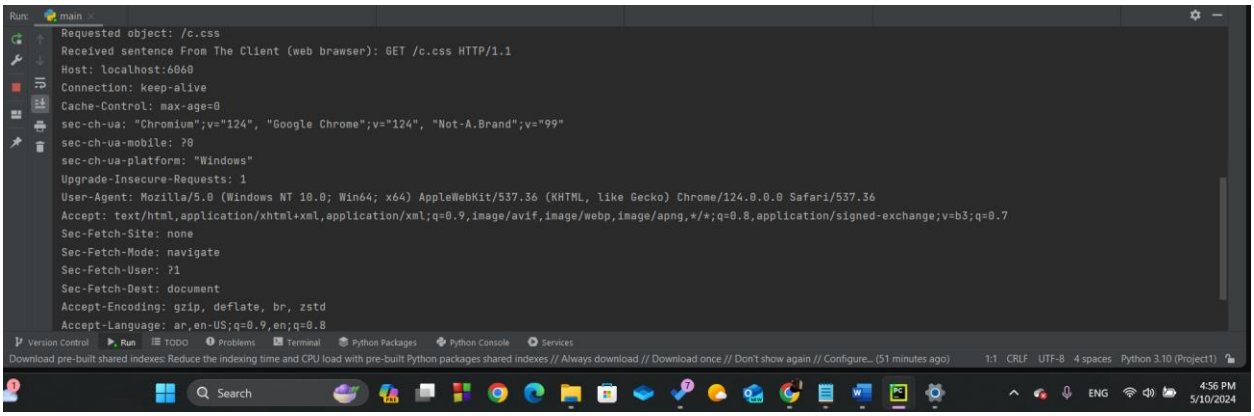


Figure 40:request c.css Http Massge Requested .

3.6 if I request p.png

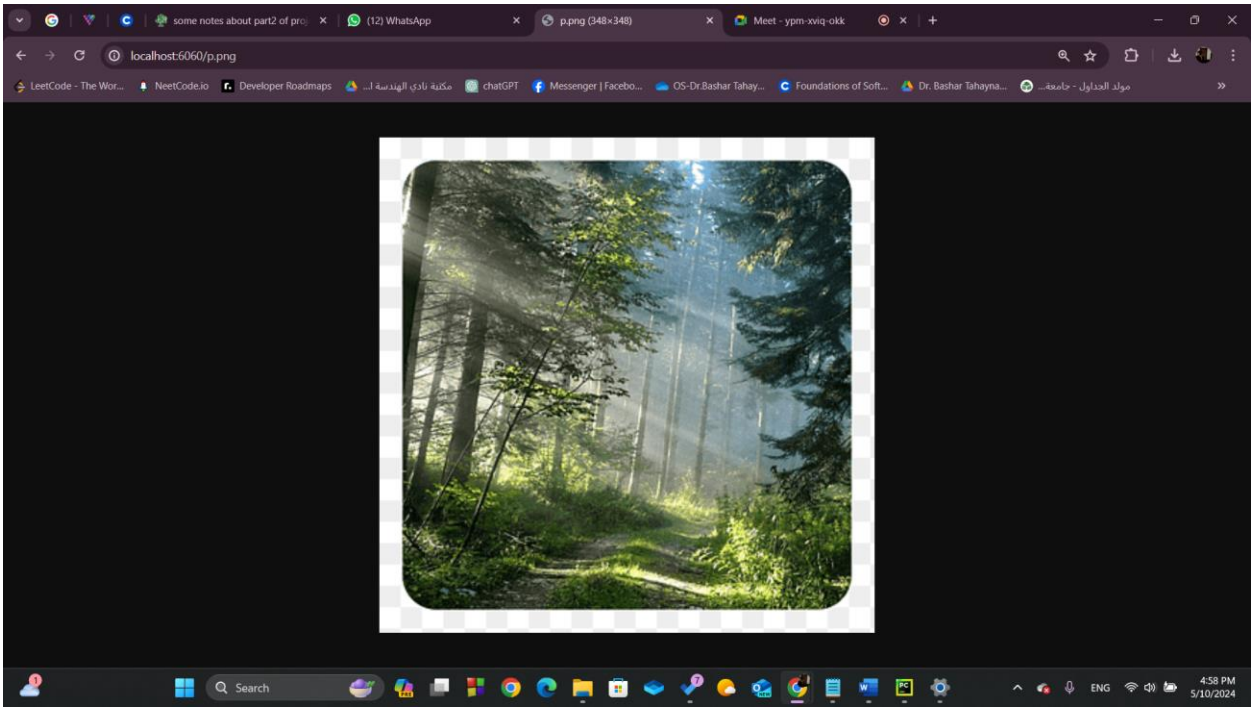


Figure 41:request p.png Result

The Http request :

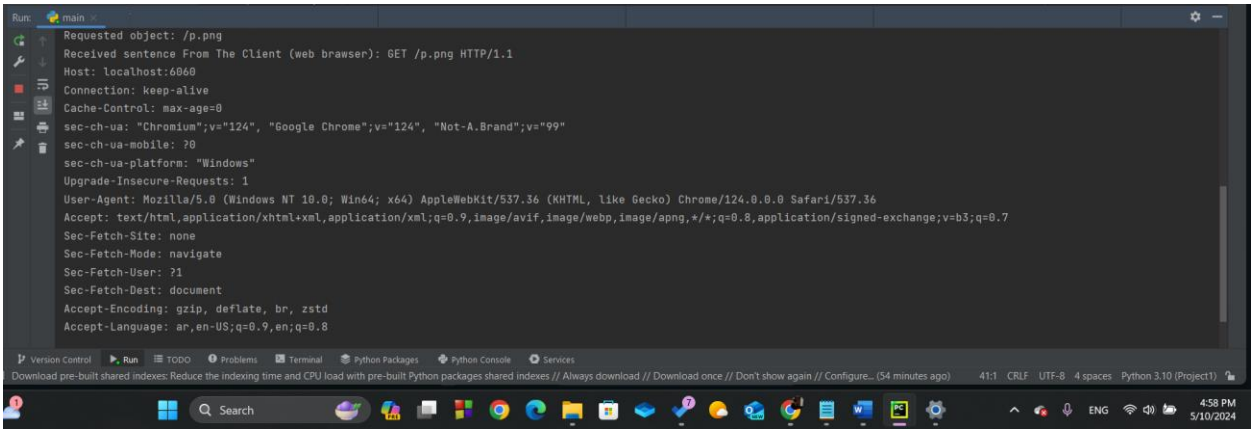


Figure 42:request p.png Massge Request .

3.7 if I request j.jpg

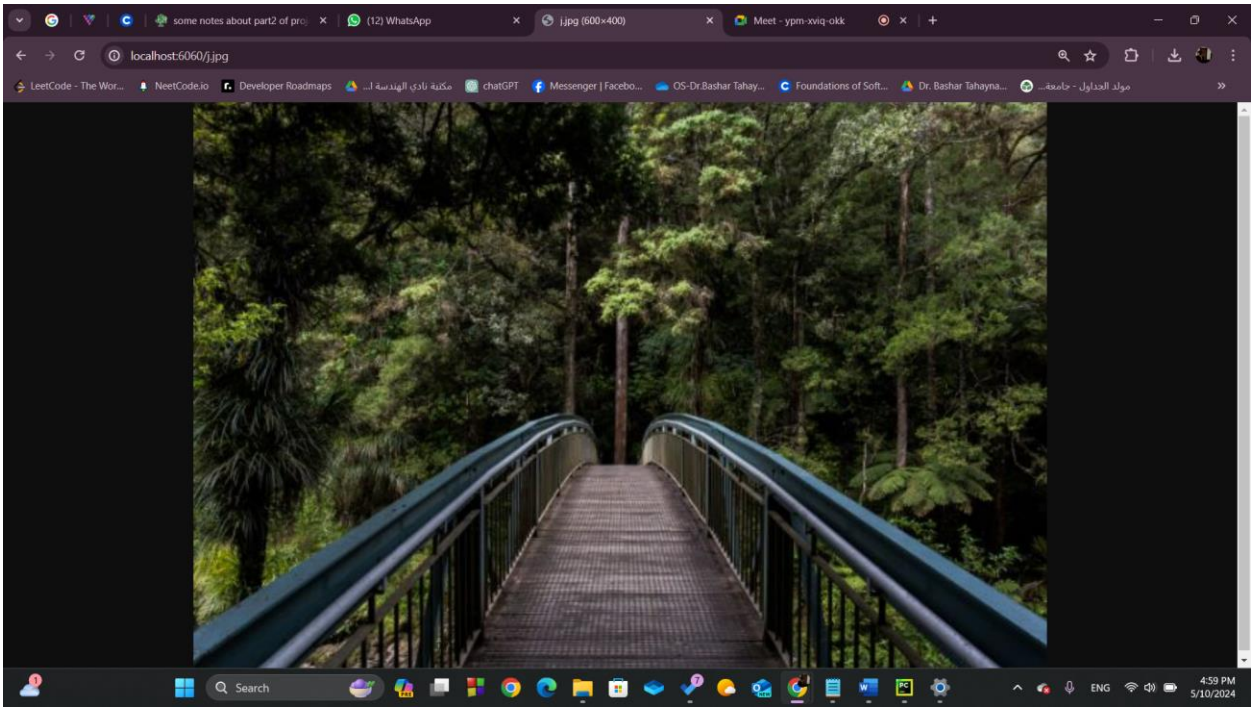


Figure 43:request j.jpg Result.

The Http request :

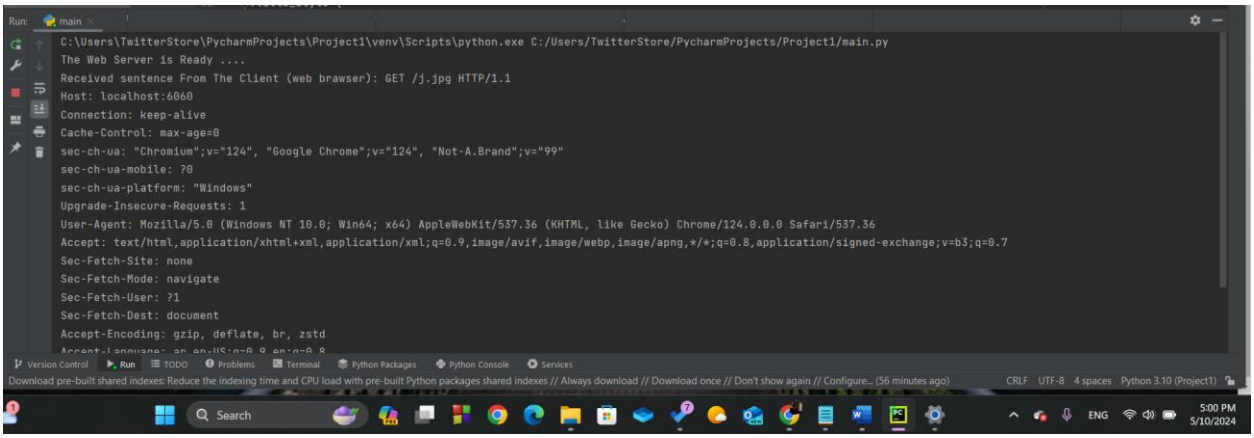


Figure 44:request j.jpg HTTP requested Massge.

3.8 MyForm

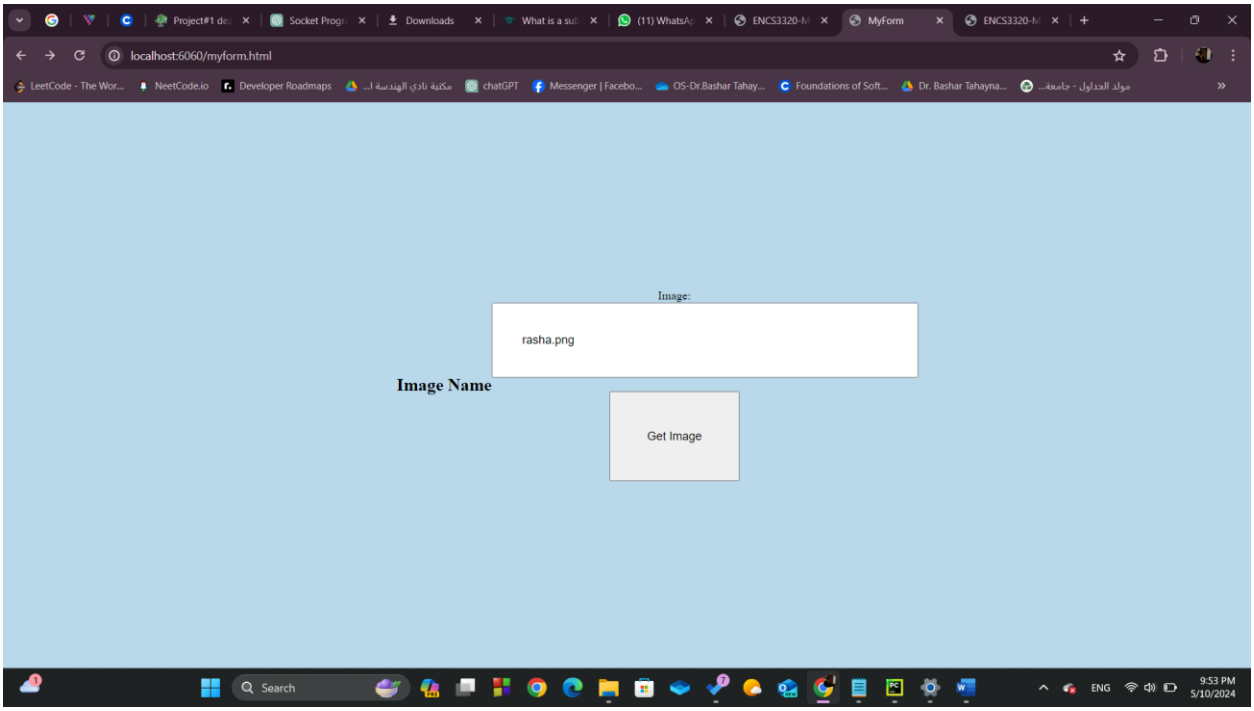


Figure 45:My Form Result

HTTP request:

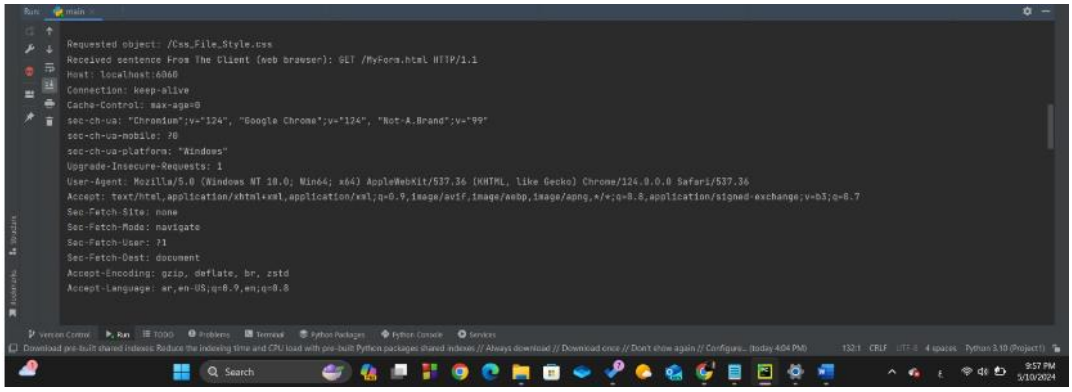
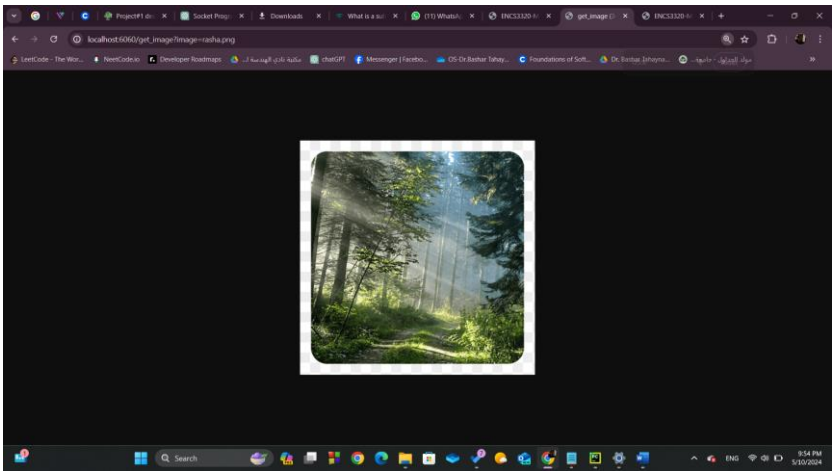


Figure 46:MyForm Result HTTP request.

Result of photo result:



3.9 if I request /so

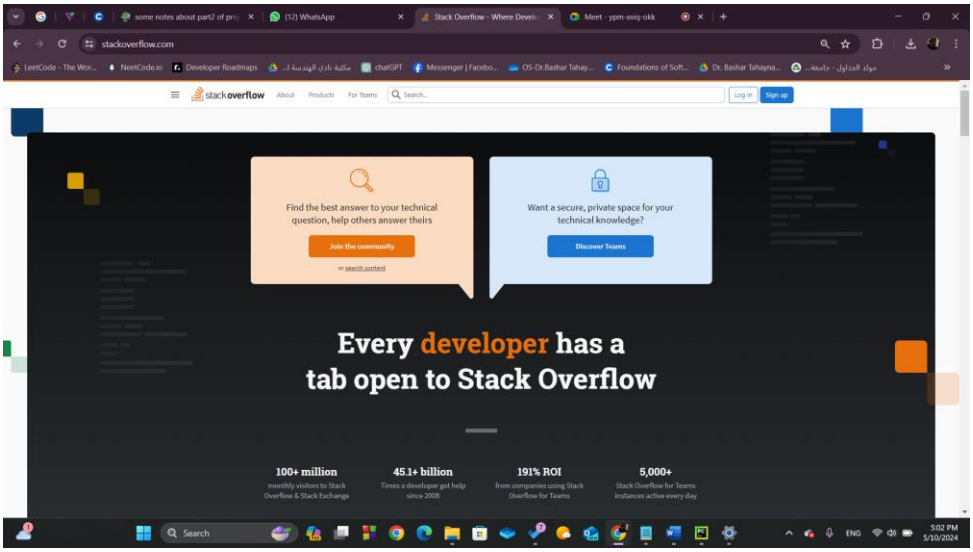


Figure 47:request /so Result.

The Http request :

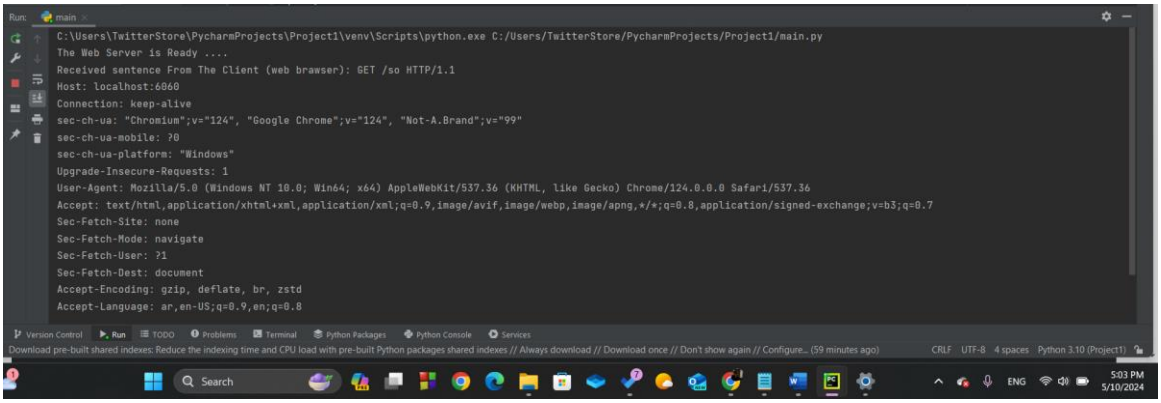


Figure 48:request /so HTTP requested Massge .

3.10.if I request /itc

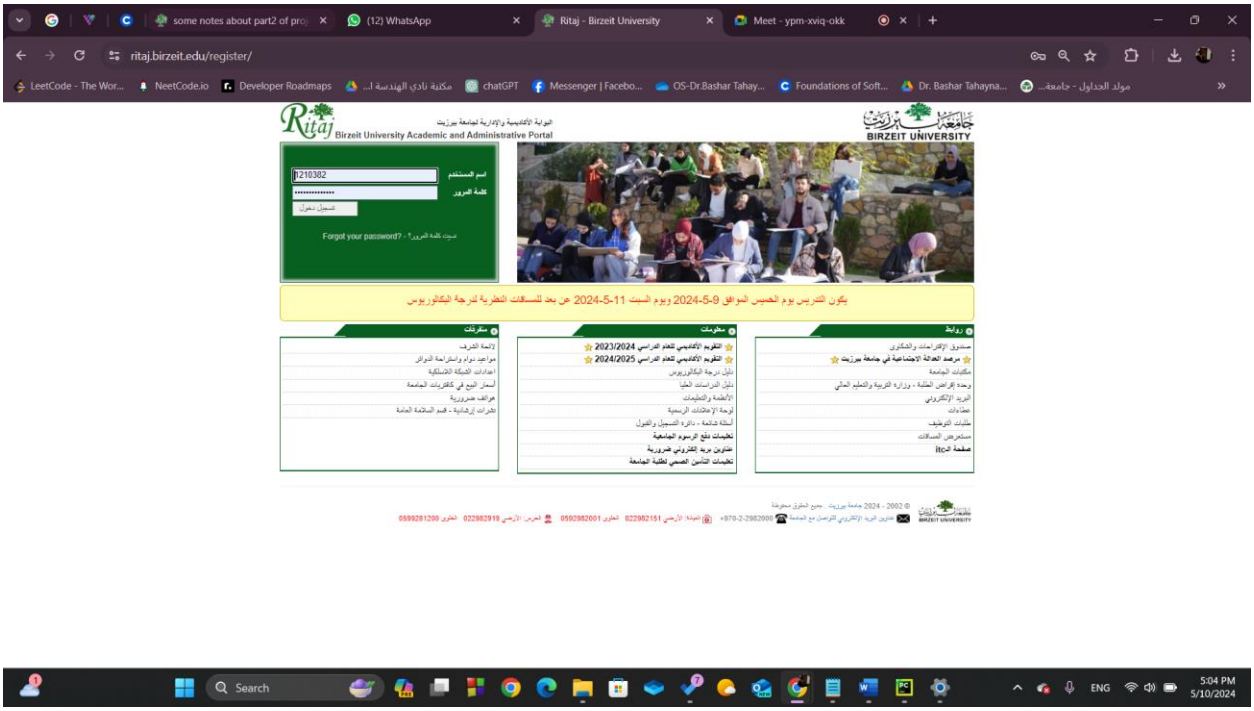


Figure 49:request /itc Result .

The Http request:

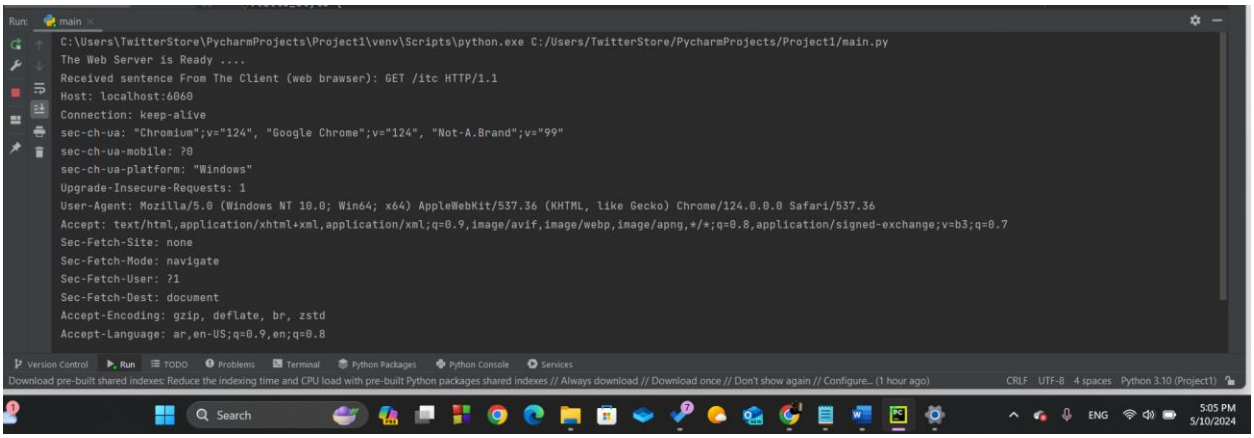


Figure 50:request /itc Requsted massge Http .

3.11. Request any File Does Not Contain in The Server

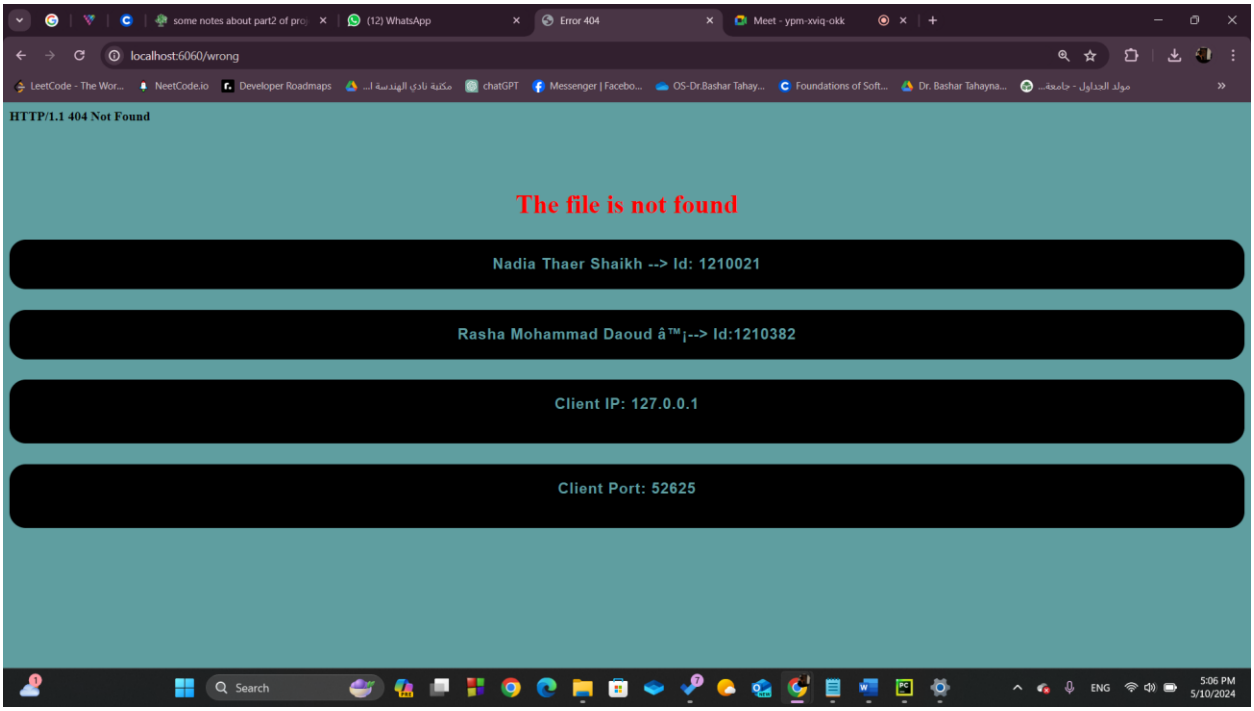


Figure 51:Request any File Does Not Contain in The Server Result.

The HTTP request:

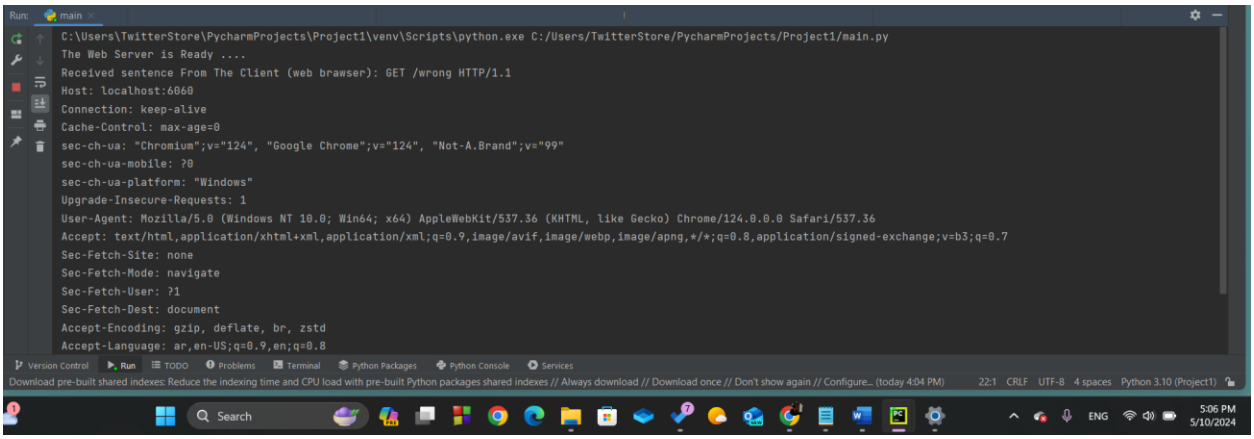


Figure 52:Request any File Does Not Contain in The Server Http requested massge .