Faculty of Engineering and Technology Electrical and Computer Engineering Department

Computer Networks ENCS3320

Project

Names:	Ids:
Adam Nassan	1202076
Hidaya Mustafa	1201910
Rana Odeh	1201750

Date: 1/6/2023

Contents

Part 1:- 3
1. Describe ping,tracert,nslookup and telnet 3
2. Run the following commands 3
Part 2:- 5
Part 3:- 7
1- Server code 7
2- CSS code 10
3- HTML code 11
a) ENG code 11
b) AR code 12
4- Code results 13
a. ENG page 13
b. AR page 13
c. Local page link 14
d. W3school link 14
e. Status code 307 Temporary Redirect 15
f. Try some requests 16
g. Test the project in smartphone 18

Table of figures:-

Figure 1: Ping same network 3
Figure 2: Ping Harvard.edu 3
Figure 3: Tracert Harvard.edu 4
Figure 4: Nslookup Harvard.edu 4
Figure 5: Server code 5
Figure 6: Client code 6
Figure 7:1st client name 6
Figure 8: 2nd client name 6
Figure 9: UDP results 6
Figure 10: Server code 1 7
Figure 11: Server code 2 7
Figure 12: Server code 3 8
Figure 13: Server code 4 8
Figure 14: Server code 5 9
Figure 15: CSS code 1 10
Figure 16: CSS code 2 10
Figure 17: ENG code 1 11
Figure 18: ENG code 2 11
Figure 19: AR code 1 12
Figure 20: AR code 2 12
Figure 21: ENG page 1 13
Figure 22: ENG page 2 13
Figure 23: AR page 13
Figure 24: Local page 14
Figure 25: W3School page 14
Figure 26: yt request 15
Figure 27: rt request 15
Figure 28: so request 15
Figure 29: .HTML request 16
Figure 30: .CSS request 16
Figure 31: .PNG request 17
Figure 32: .JPG request 17
Figure 33: Wrong request 18
Figure 34: Smartphone test 18

Part 1:-

1. Describe ping,tracert,nslookup and telnet

- Ping: is a command line tool that can track the status of different devices availability it help in identifying network connectivity and latency problems.
- Tracert: It is used to find out various information about the path that the packets take from your device to a specific place.
- NsLookUp: client that discover the IP address queries DNS records for a given domain name.
- Telnet: A network protocol used to physically access a computer, usually allowing a TCP/IP server to run and to provide a two-way, cooperative, text-based communication channel between two machines.

2. Run the following commands

- Ping a device in the same network

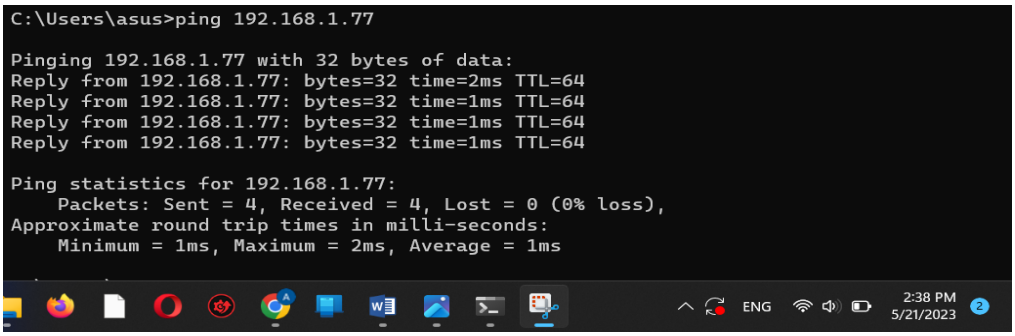


Figure 1: Ping same network

As we see in the figure 4 packets have been sent with the same TTL and different time delay with Minimum = 1ms, Maximum = 2ms, Average = 1ms.

- Ping www.harvard.edu

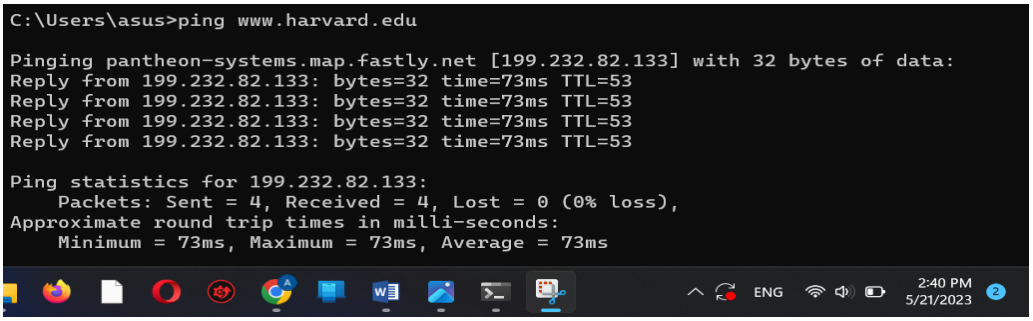


Figure 2: Ping Harvard.edu

As we see in the figure, 4 packets have been sent with the same TTL and different time delay with Minimum = 73ms, Maximum = 73ms, Average = 73ms. We notice that the time of the smartphone is less than the www.harvard.edu.

- Tracert www.harvard.edu

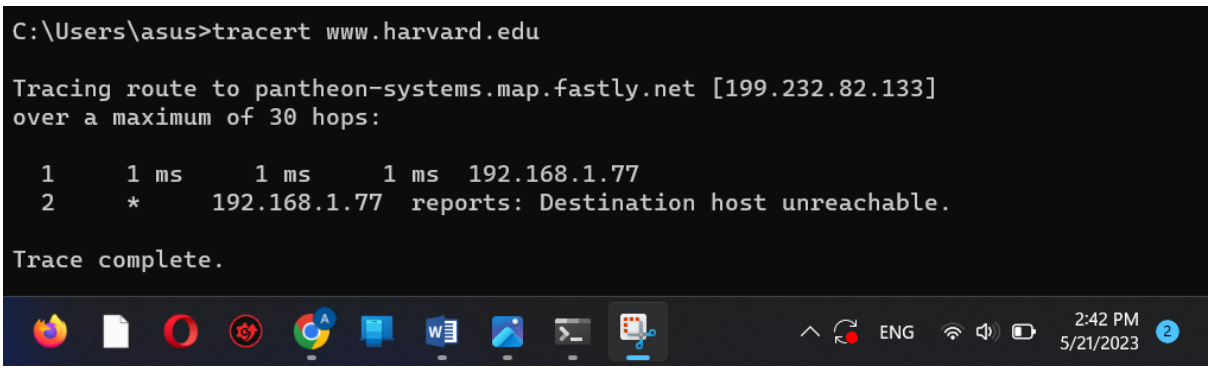


Figure 3: Tracert Harvard.edu

As we see in the figure, the trace of www.harvard.edu, where 199.232.82.133 is the website IP address, with maximum 30 hops, the packets were sent. The command awaits each router's response. Since the next router travels longer before reaching the website server, we can see that the delay increases there. The packet cannot reach the destination if the request timed out and the measurements are * because there is a problem there or the route is not correct.

- Nslookup www.harvard.edu

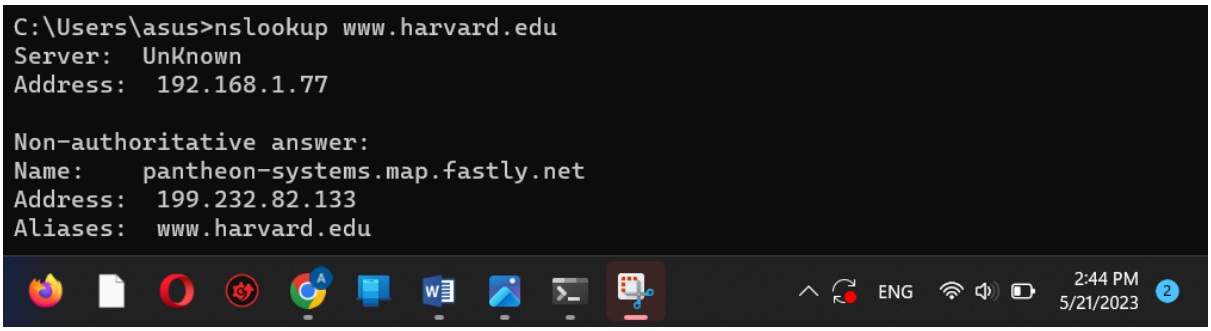


Figure 4: Nslookup Harvard.edu

As we can see in the figure, 192.168.1.77 is my IP address while 199.232.82.133 is Harvard IP address.

Part 2:-

Using socket programming, implement UDP client and server applications in go, python, java or C. The server should listen on port 8855. The client sends broadcast messages every 2 seconds. The message should contain the student's name. The server lists the last received message from a client.

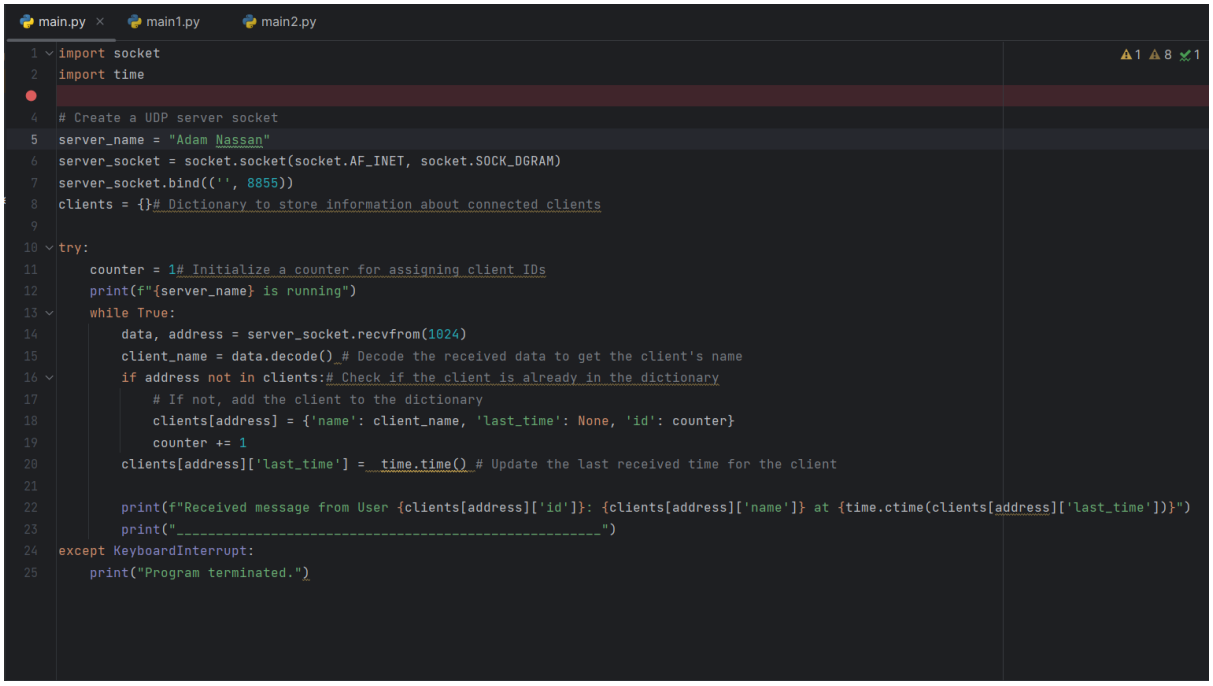


Figure 5: Server code

The server creates a socket and binds it to port 8855, indicating that it will listen for incoming messages on that port. It maintains a dictionary to store information about connected clients, including their IP address, name, and the time when the last message was received.

The server enters a main loop where it continuously receives data from clients. The data is received in the form of UDP packets, which contain the client's IP address, the message itself, and the message length. The message is decoded, and the client's name is extracted from it.

The server then prints the client's name and the time at which the message was received. The last received time for the client is updated in the dictionary.

This process repeats until the user interrupts the program indicating a keyboard interrupt.

The server demonstrates a basic implementation of UDP communication, where the server can receive messages from clients without establishing a connection beforehand.

main.py main1.py main2.py

```python
import time
from socket import *

serverName = "192.168.1.255"#define the server IP address
serverPort = 8855#define the port number
clientSocket = socket(AF_INET, SOCK_DGRAM)
clientSocket.setsockopt(SOL_SOCKET, SO_BROADCAST, 1)

student_name = input("Enter your name: ")

try:
    while True:
        message = student_name
        clientSocket.sendto(message.encode(), (serverName, serverPort))#send the encoded message to the server
        time.sleep(2)#delay of 2 seconds
except KeyboardInterrupt:
    print("Program terminated.")

clientSocket.close()
```

Figure 6: Client code

```
C:\Users\centrinoLaptop\PycharmProjects\pythonProject7\venv\Scripts\python.exe C:\Users\centrinoLaptop\PycharmProjects\pythonProject7\main2.py
Enter your name: Rana Odeh
```


Figure 7:1st client name

```
C:\Users\centrinoLaptop\PycharmProjects\pythonProject7\venv\Scripts\python.exe C:\Users\centrinoLaptop\PycharmProjects\pythonProject7\main1.py
Enter your name: Hidaya Mustafa
```


Figure 8: 2nd client name

The client program creates a UDP socket and binds it to the broadcast address 192.19.62.255 on port 8855. It then enters a loop where it sends the message "student_name" every 2 seconds to all devices on the network. This broadcast message allows the server to receive and process the message. The loop continues until the program is interrupted.

```
"C:\Users\Rana Odeh\PycharmProjects\pythonProject8\venv\Scripts\python.exe" "C:\Users\Rana Odeh\PycharmProjects\pythonProject8\main.py"
Adam Nassan is running
Received message from User 1: Rana Odeh at Thu Jun  1 21:29:29 2023
----------------------------------------------------
Received message from User 1: Rana Odeh at Thu Jun  1 21:29:31 2023
----------------------------------------------------
Received message from User 1: Rana Odeh at Thu Jun  1 21:29:33 2023
----------------------------------------------------
Received message from User 1: Rana Odeh at Thu Jun  1 21:29:35 2023
----------------------------------------------------
Received message from User 1: Rana Odeh at Thu Jun  1 21:29:37 2023
----------------------------------------------------
Received message from User 2: Hidaya mustafa at Thu Jun  1 21:29:38 2023
----------------------------------------------------
Received message from User 1: Rana Odeh at Thu Jun  1 21:29:39 2023
----------------------------------------------------
Received message from User 2: Hidaya mustafa at Thu Jun  1 21:29:40 2023
----------------------------------------------------
Received message from User 1: Rana Odeh at Thu Jun  1 21:29:41 2023
```

Figure 9: UDP results

Part 3:-

Using socket programming, implement a simple but a complete web server in go, python, java or C that is listening on port 9977.

1- Server code

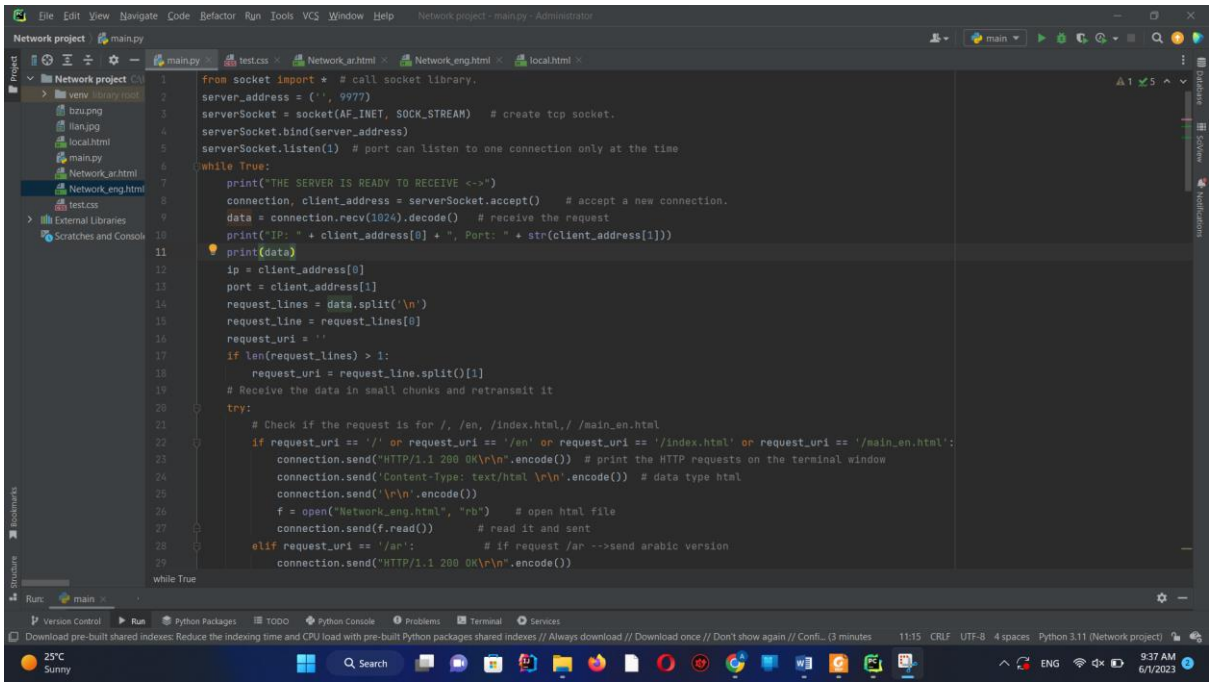


Figure 10: Server code 1

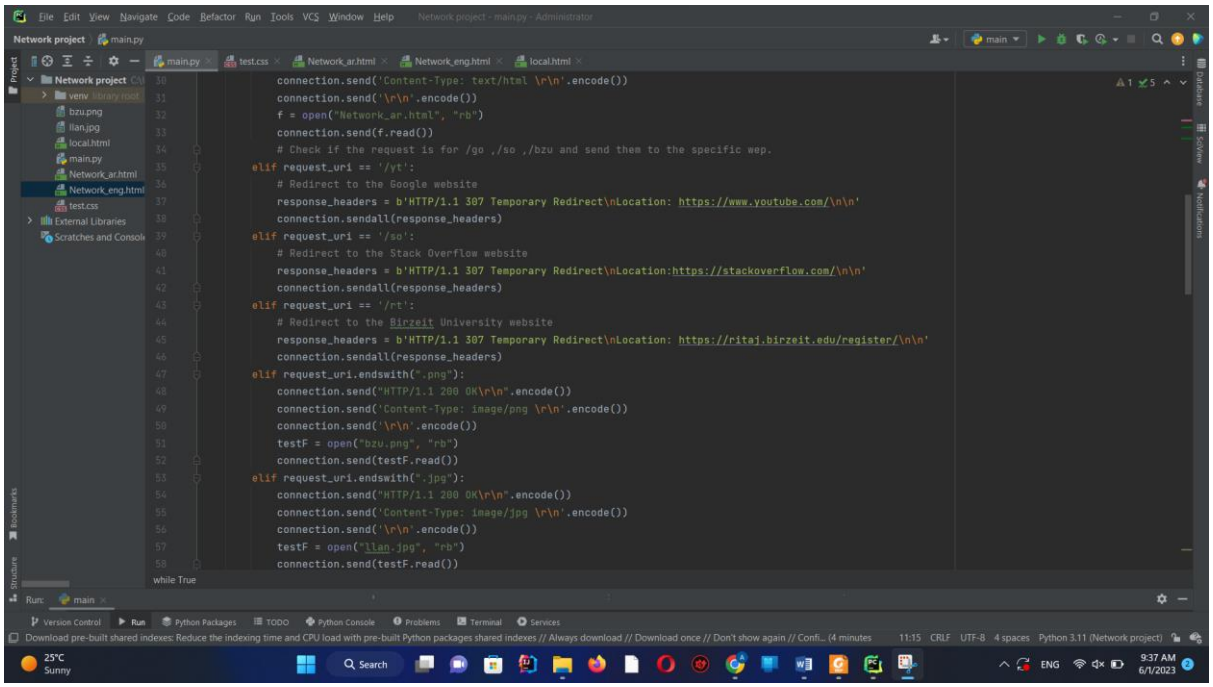


Figure 11: Server code 2

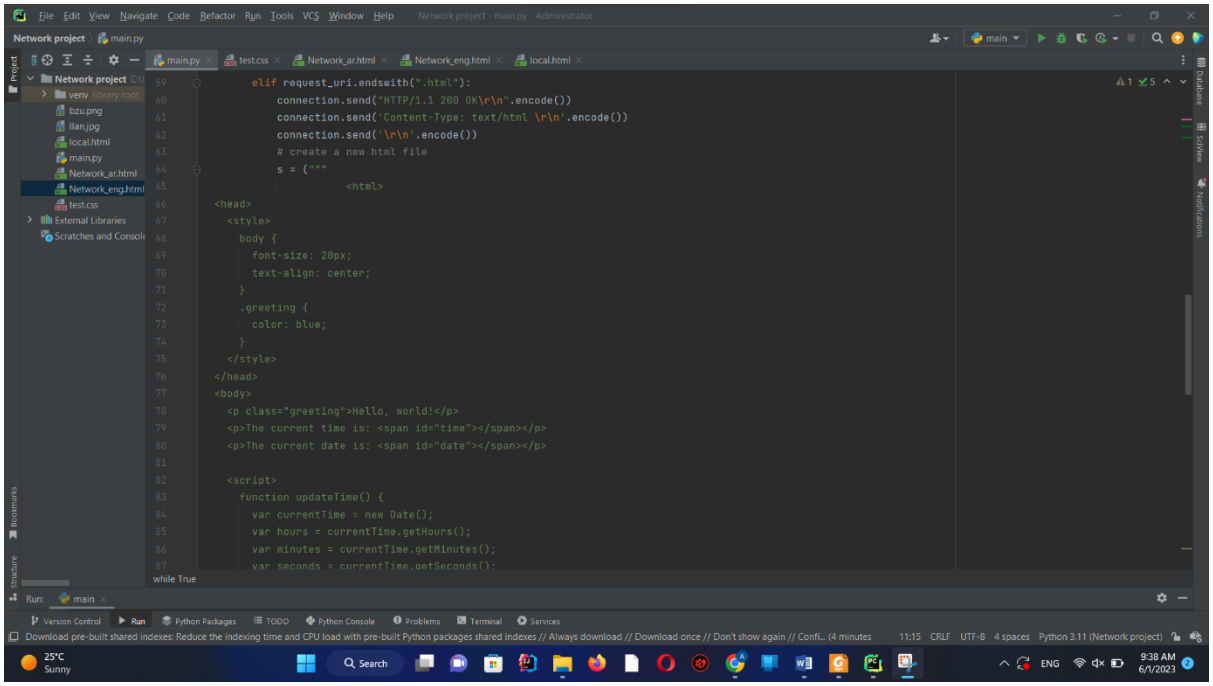


Figure 12: Server code 3

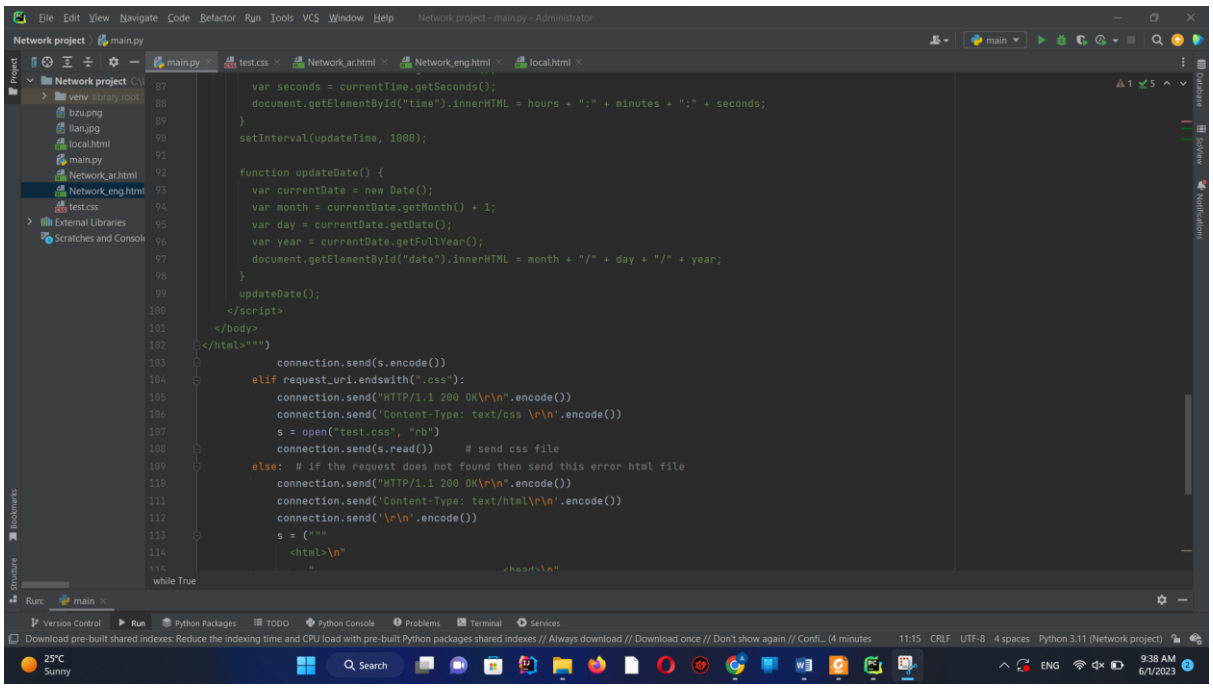


Figure 13: Server code 4

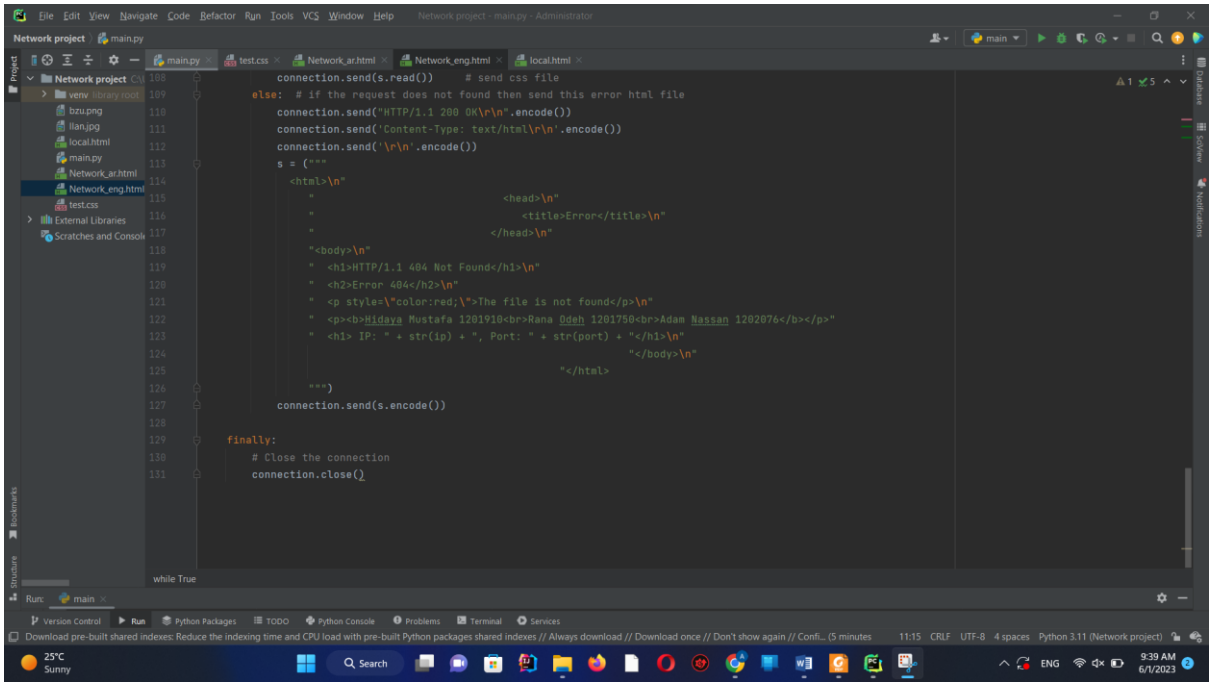
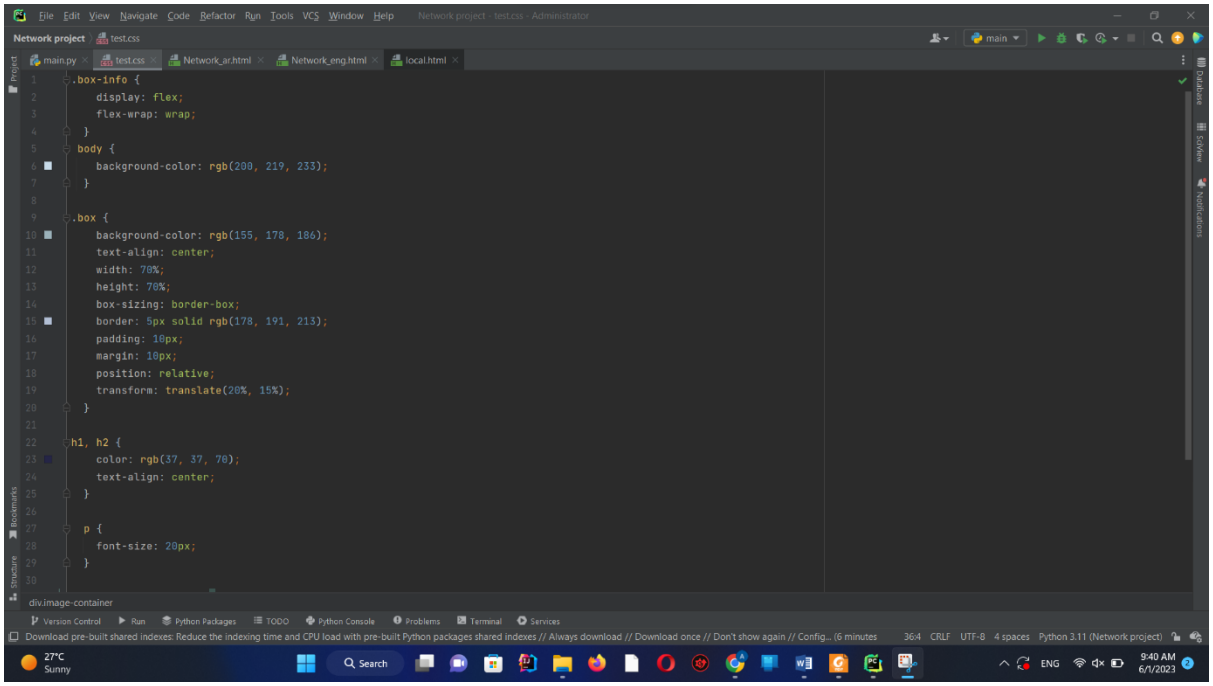


Figure 14: Server code 5

2- CSS code



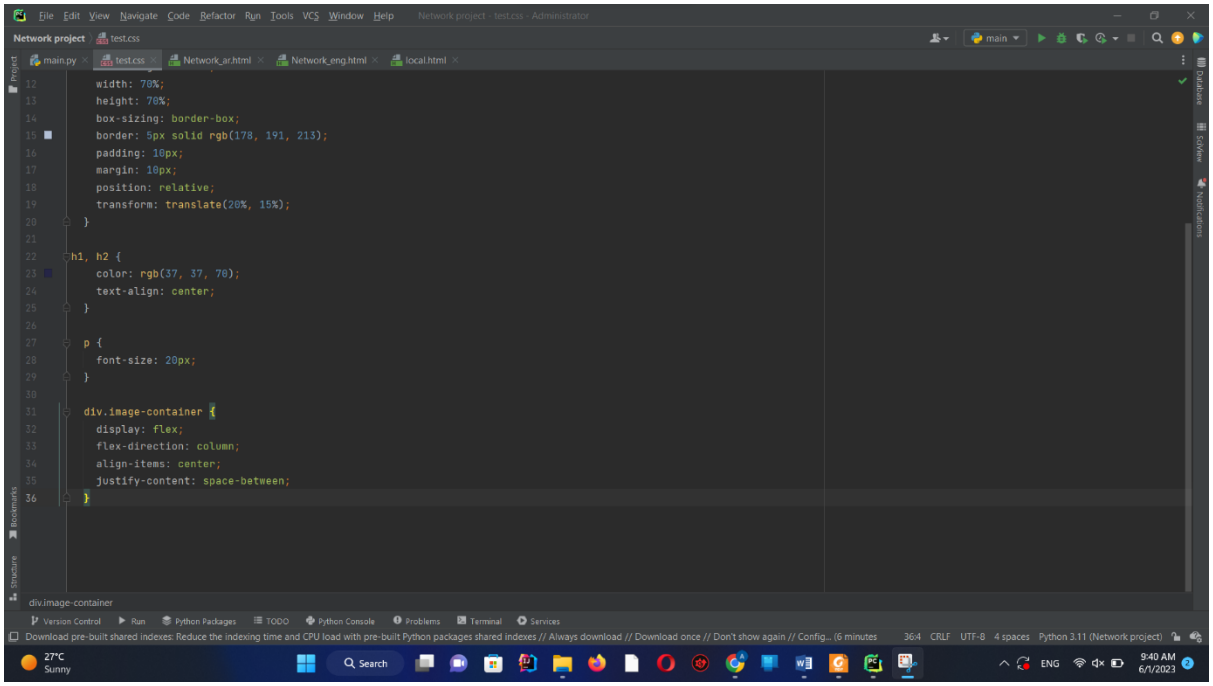
```css
.box-info {
    display: flex;
    flex-wrap: wrap;
}
body {
    background-color: rgb(200, 219, 233);
}

.box {
    background-color: rgb(155, 178, 186);
    text-align: center;
    width: 70%;
    height: 70%;
    box-sizing: border-box;
    border: 5px solid rgb(178, 191, 213);
    padding: 10px;
    margin: 10px;
    position: relative;
    transform: translate(20%, 15%);
}

h1, h2 {
    color: rgb(37, 37, 70);
    text-align: center;
}

p {
    font-size: 20px;
}
```

Figure 15: CSS code 1



```css
    width: 70%;
    height: 70%;
    box-sizing: border-box;
    border: 5px solid rgb(178, 191, 213);
    padding: 10px;
    margin: 10px;
    position: relative;
    transform: translate(20%, 15%);
}

h1, h2 {
    color: rgb(37, 37, 70);
    text-align: center;
}

p {
    font-size: 20px;
}

div.image-container {
    display: flex;
    flex-direction: column;
    align-items: center;
    justify-content: space-between;
}
```

Figure 16: CSS code 2

3- HTML code

a) ENG code

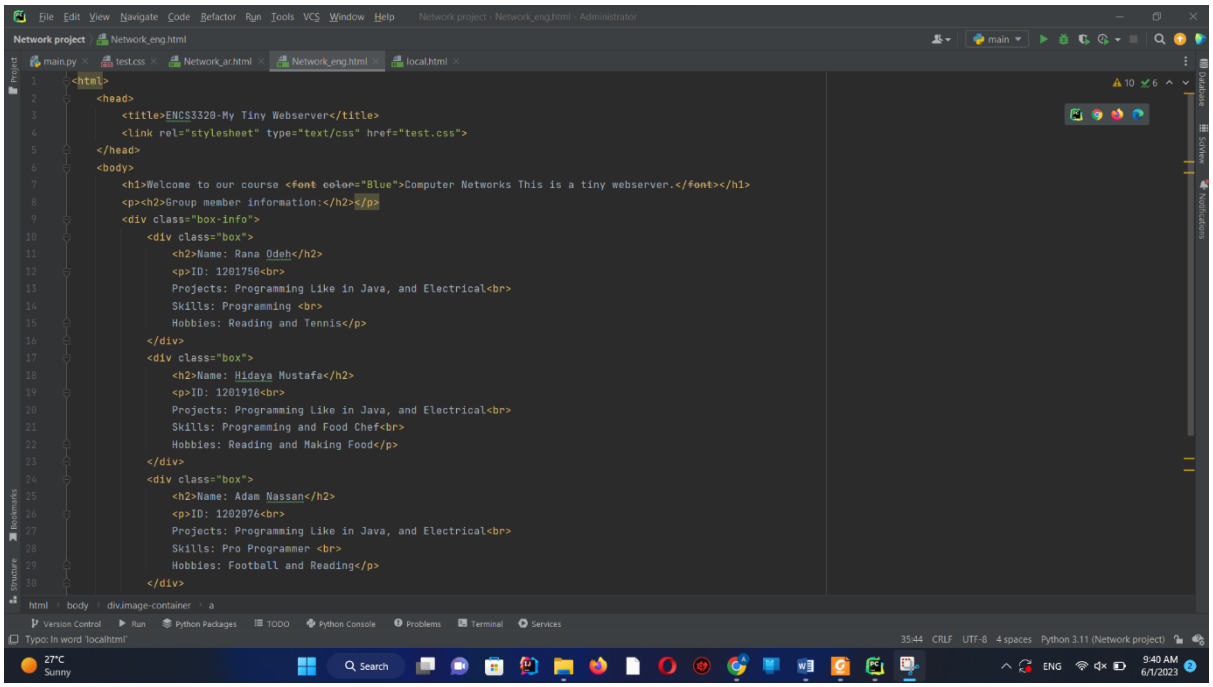


Figure 17: ENG code 1

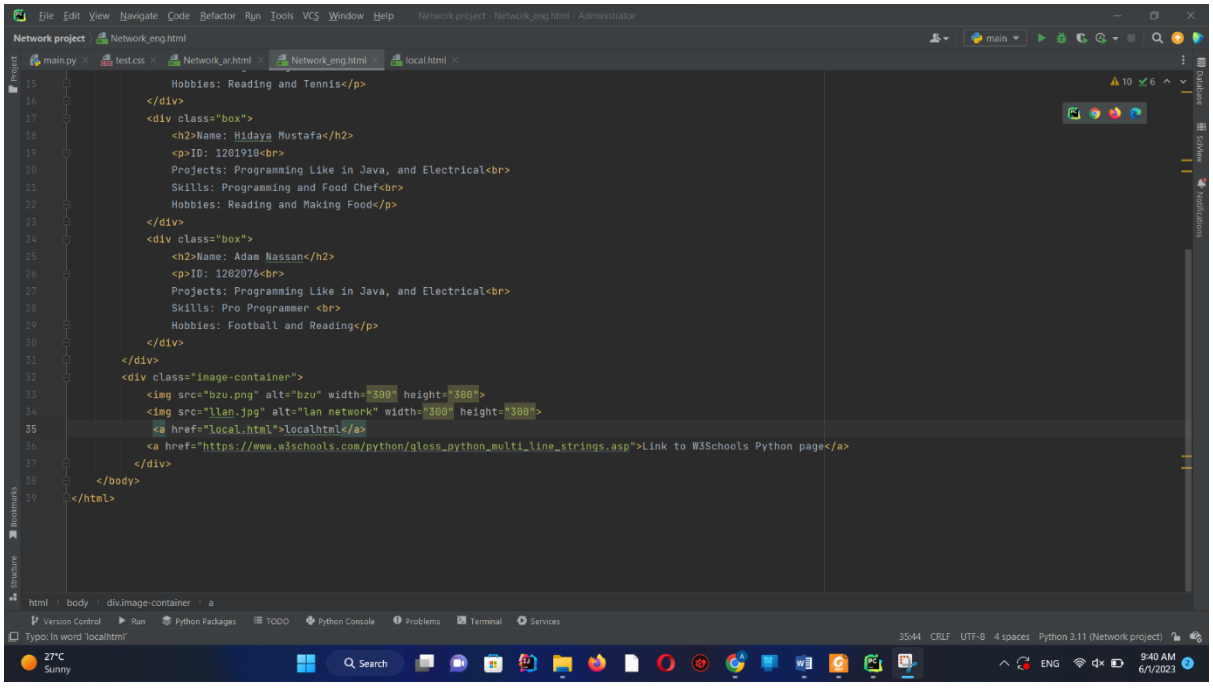


Figure 18: ENG code 2

b) AR code

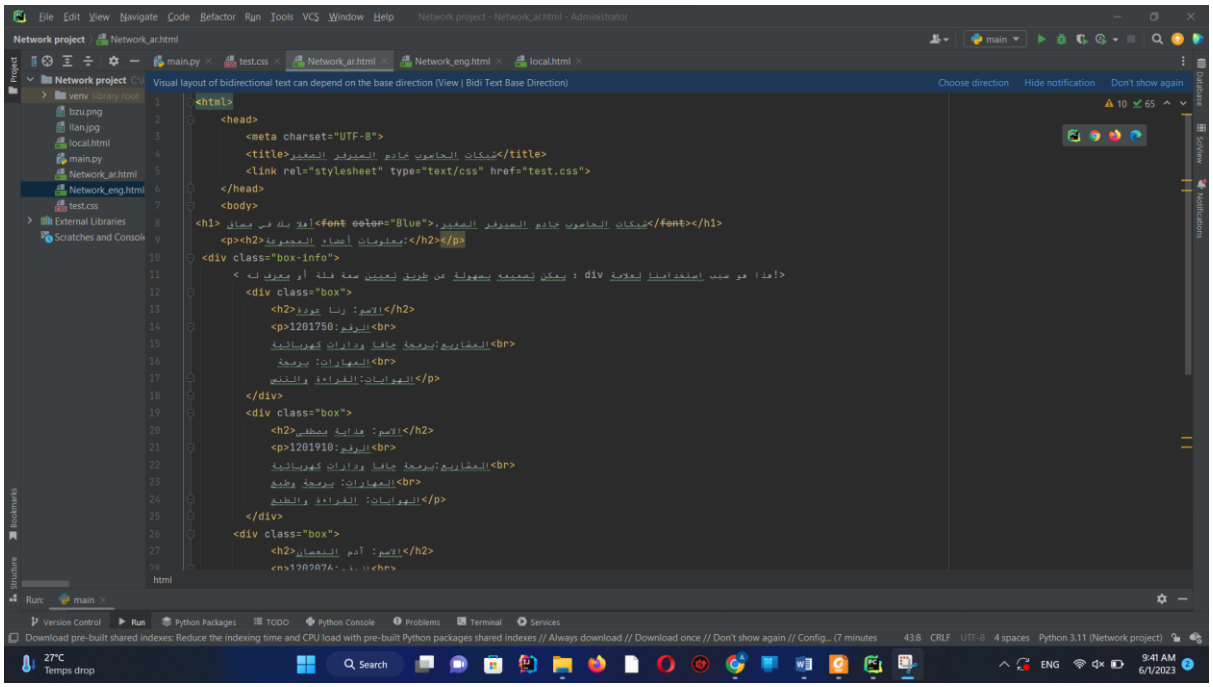


Figure 19: AR code 1

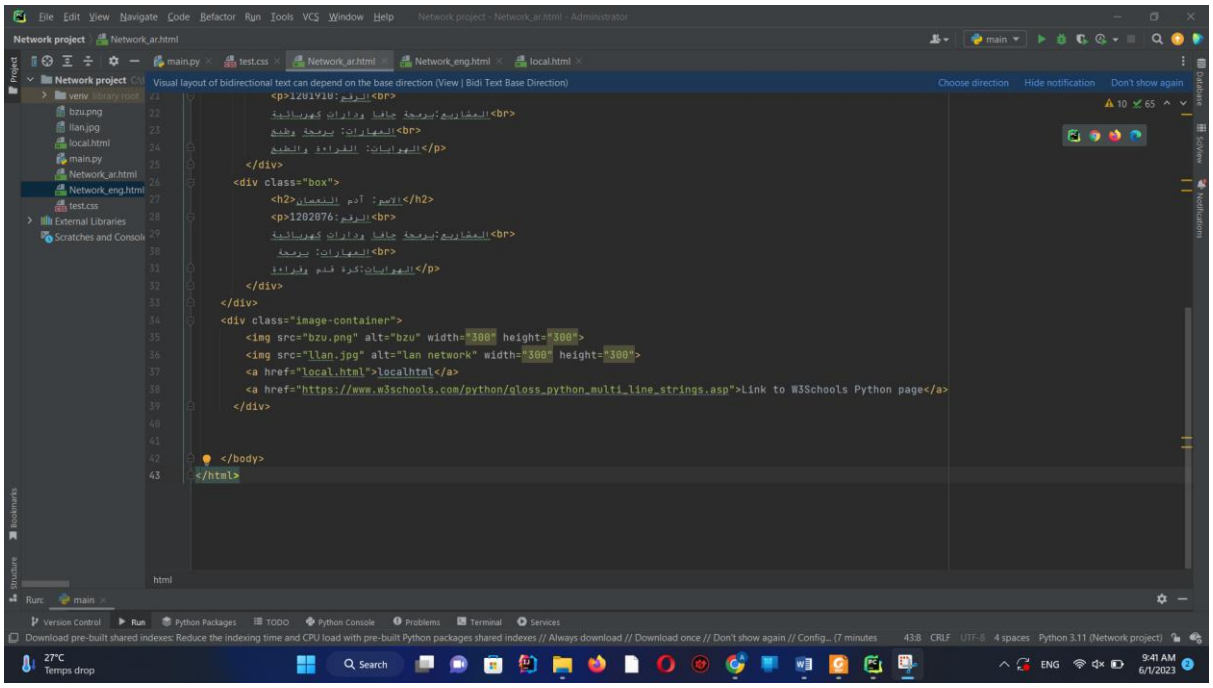


Figure 20: AR code 2

4- Code results

a. ENG page

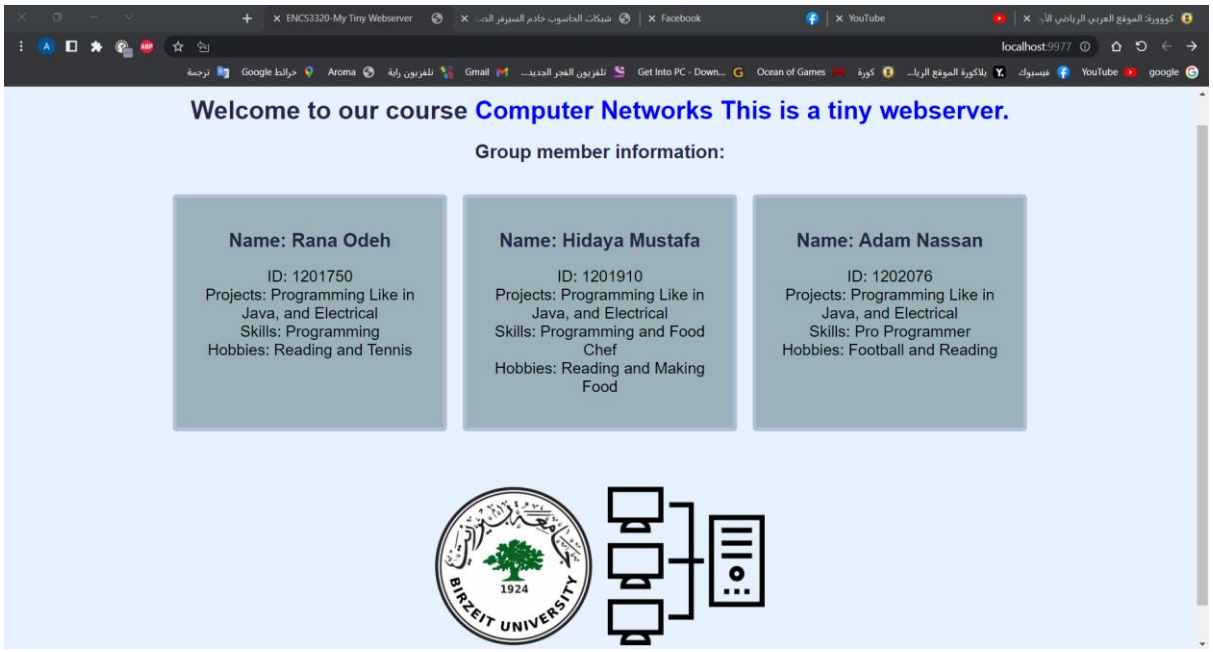


Figure 21: ENG page 1

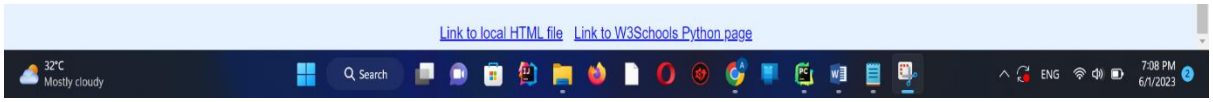


Figure 22: ENG page 2

b. AR page

Figure 23: AR page

c. Local page link

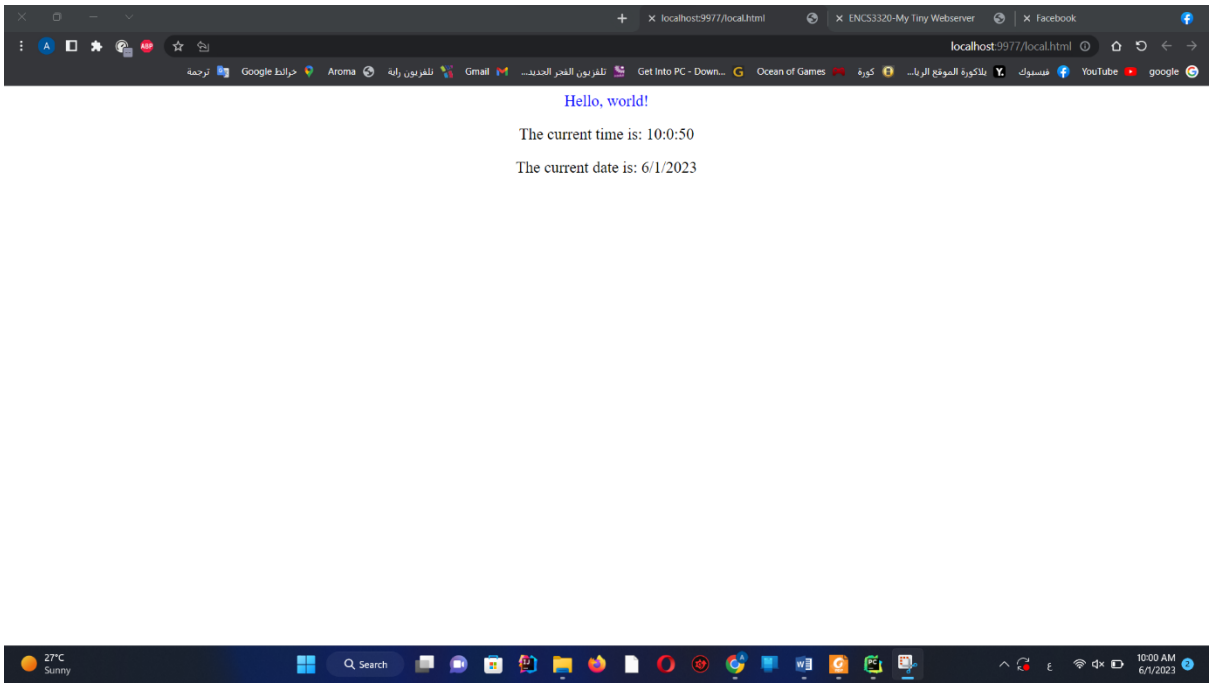


Figure 24: Local page

d. W3school link

+

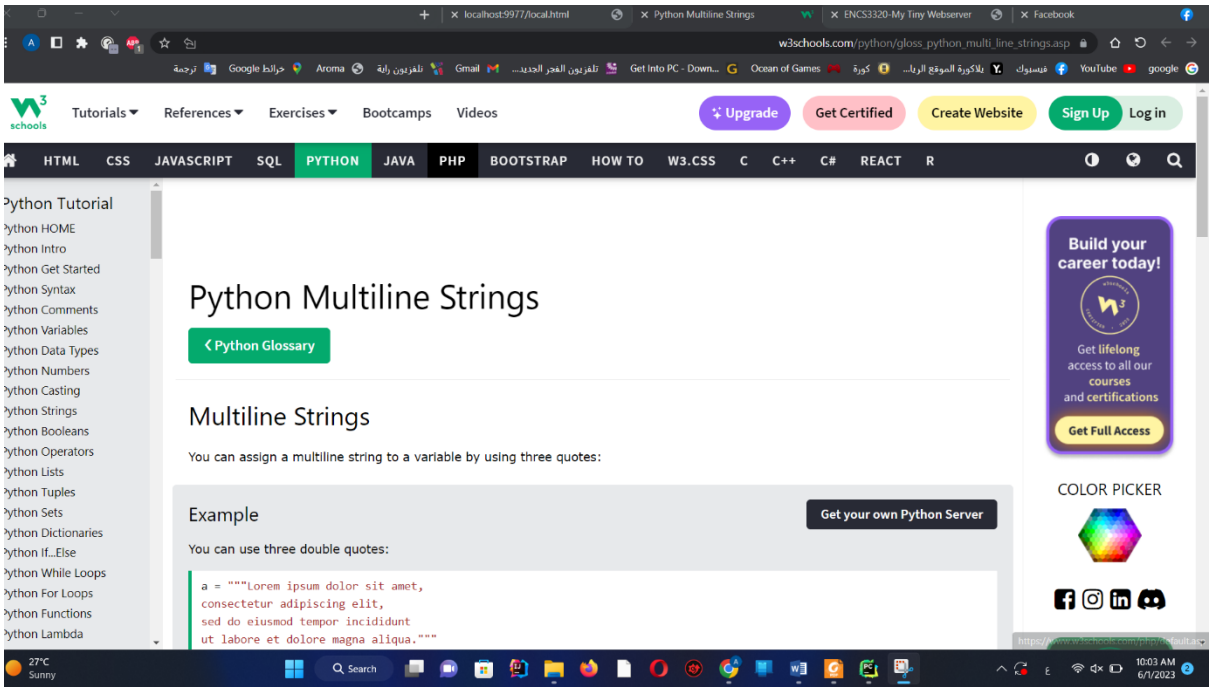


Figure 25: W3School page

e. Status code 307 Temporary Redirect

- If the request is /yt

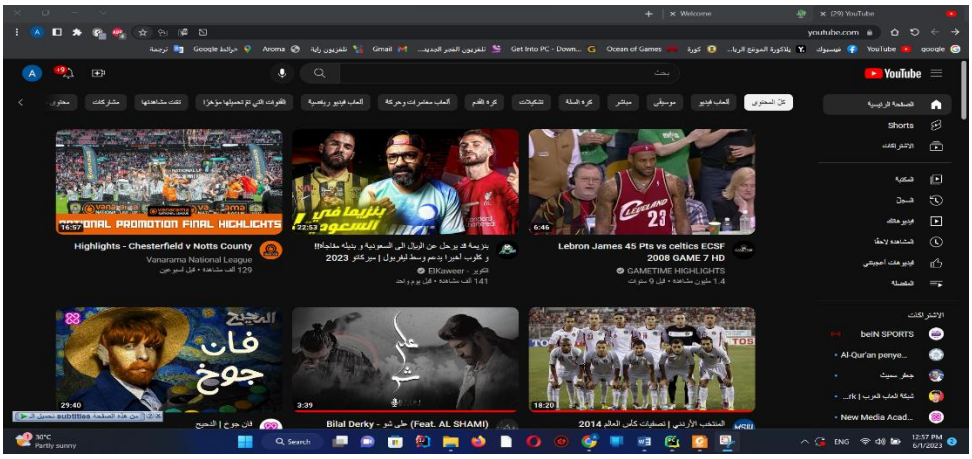


Figure 26: yt request

- If the request is /rt

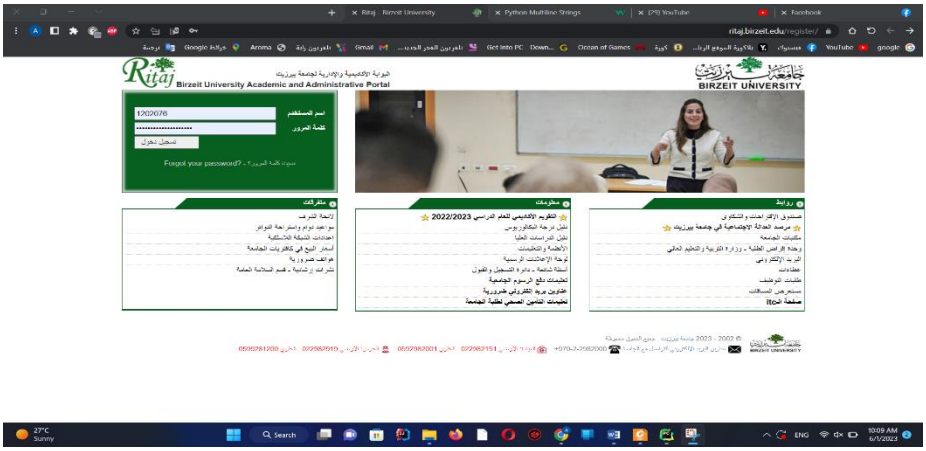


Figure 27: rt request

- If the request is /so

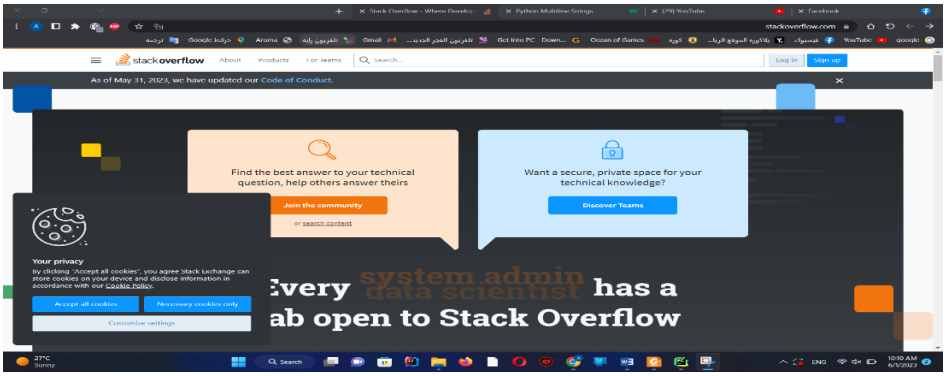


Figure 28: so request

f. Try some requests

- If the request is HTML

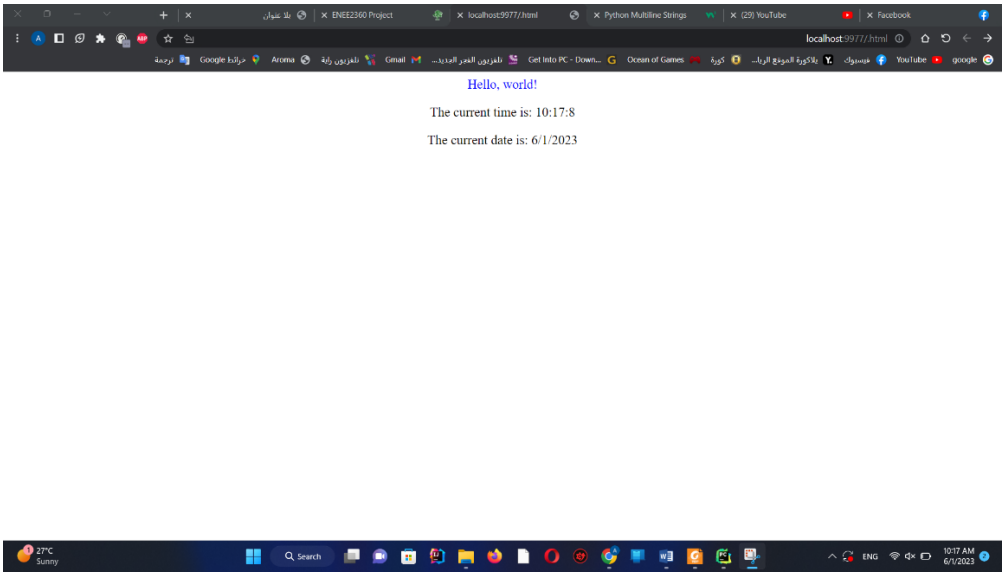


Figure 29: .HTML request

- If the request is .CSS

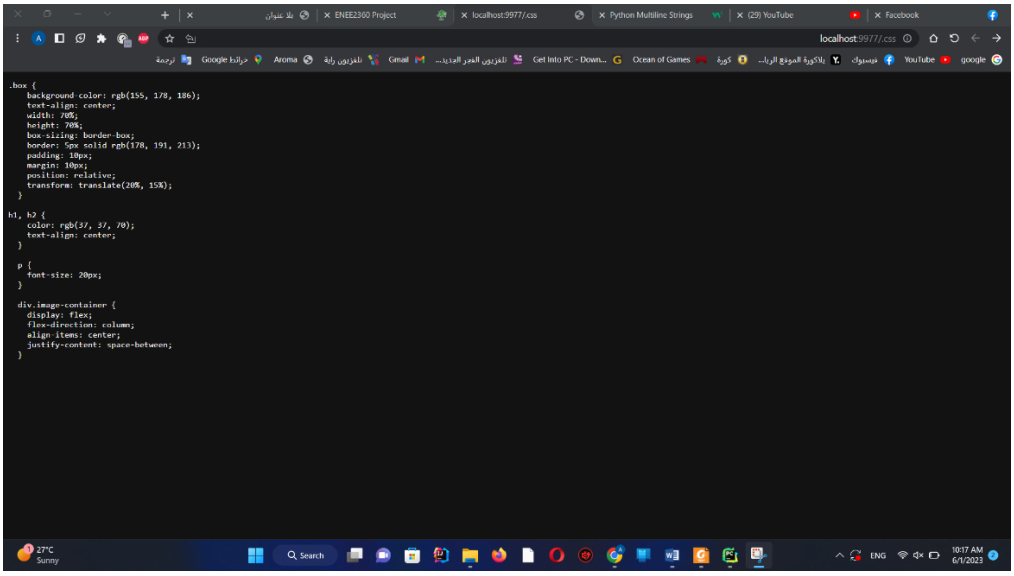


Figure 30: .CSS request

- If the request is .PNG

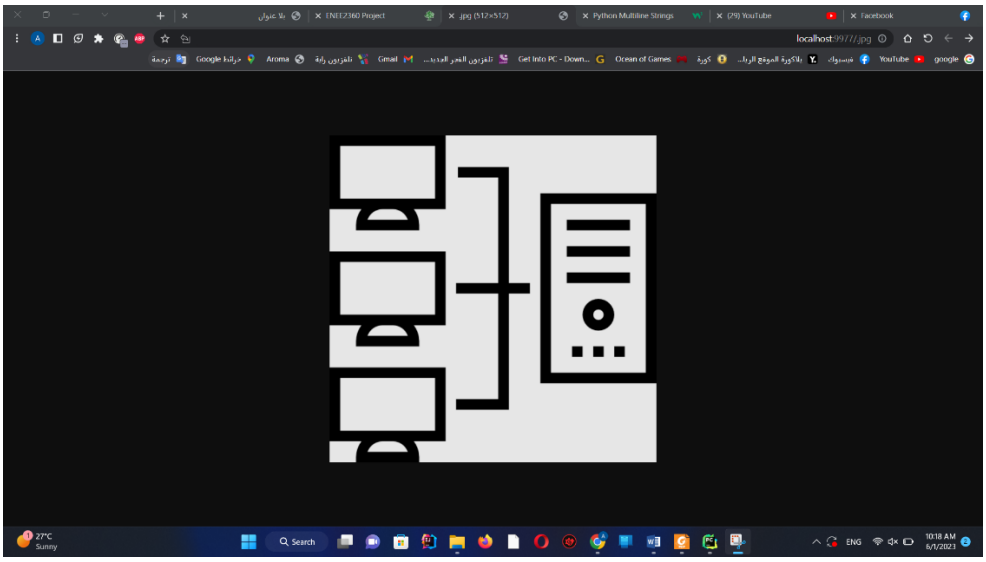


Figure 31: .PNG request

- If the request is .JPG

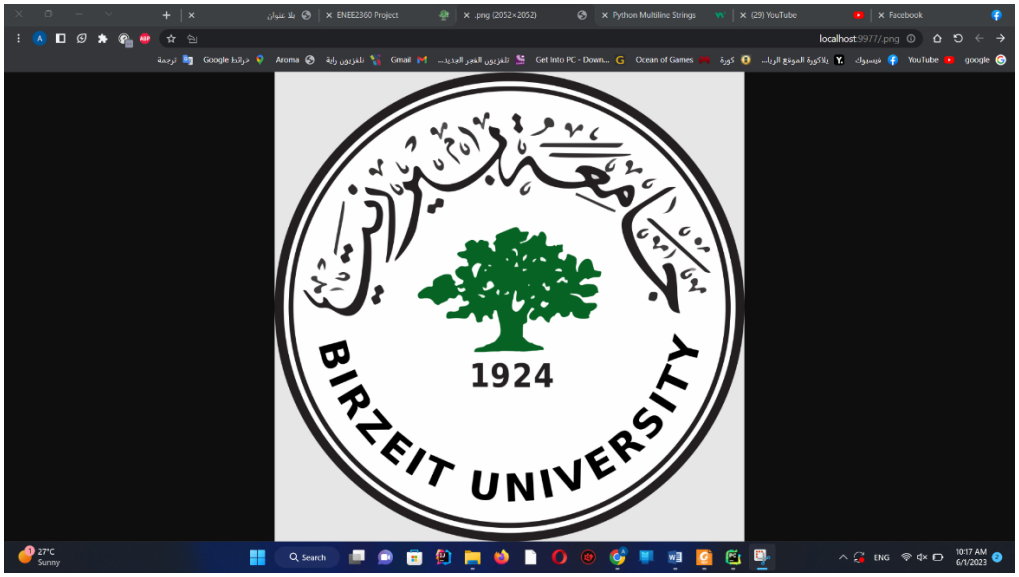


Figure 32: .JPG request

- In case of wrong request

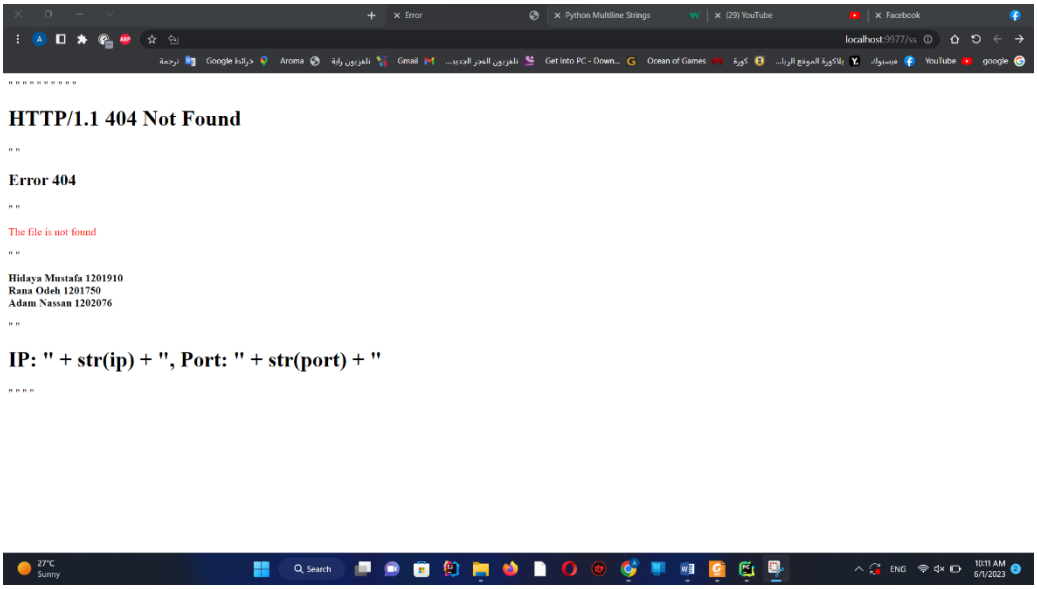


Figure 33: Wrong request

g. Test the project in smartphone

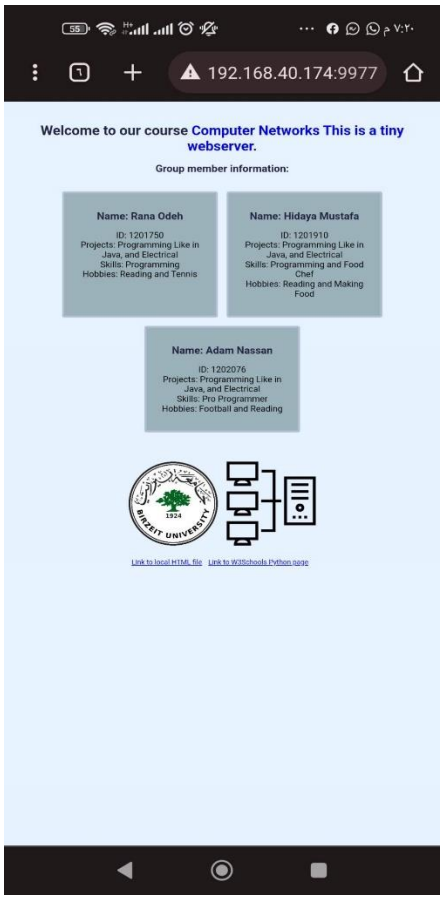


Figure 34: Smartphone test