



**Faculty of Engineering and Technology Electrical and Computer Engineering
Department**

Computer Networks ENCS3320

Project

Names:

Adam Nassan

Hidaya Mustafa

Rana Odeh

Ids:

1202076

1201910

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

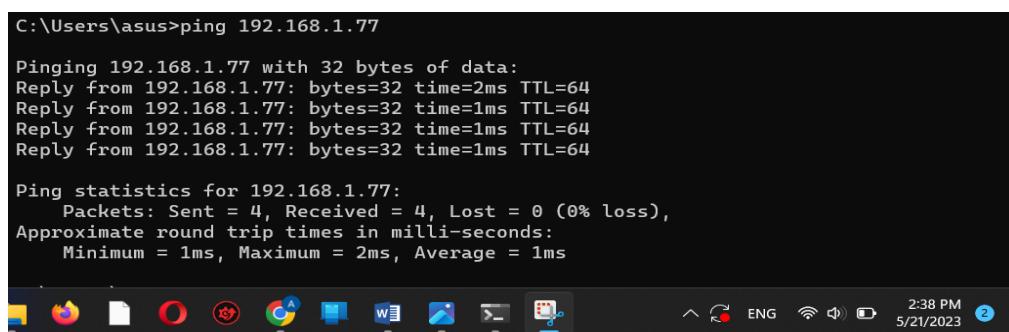
Part 1:-

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



```
C:\Users\asus>ping 192.168.1.77

Pinging 192.168.1.77 with 32 bytes of data:
Reply from 192.168.1.77: bytes=32 time=2ms TTL=64
Reply from 192.168.1.77: bytes=32 time=1ms TTL=64
Reply from 192.168.1.77: bytes=32 time=1ms TTL=64
Reply from 192.168.1.77: bytes=32 time=1ms TTL=64

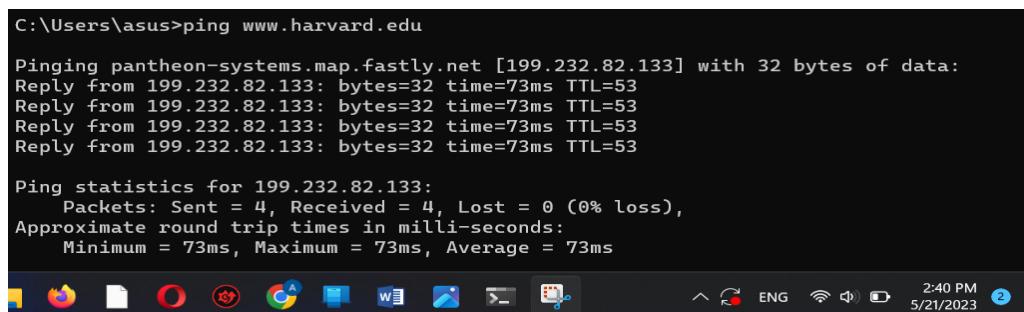
Ping statistics for 192.168.1.77:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 1ms, Maximum = 2ms, Average = 1ms

2:38 PM 5/21/2023
```

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



```
C:\Users\asus>ping www.harvard.edu

Pinging pantheon-systems.map.fastly.net [199.232.82.133] with 32 bytes of data:
Reply from 199.232.82.133: bytes=32 time=73ms TTL=53

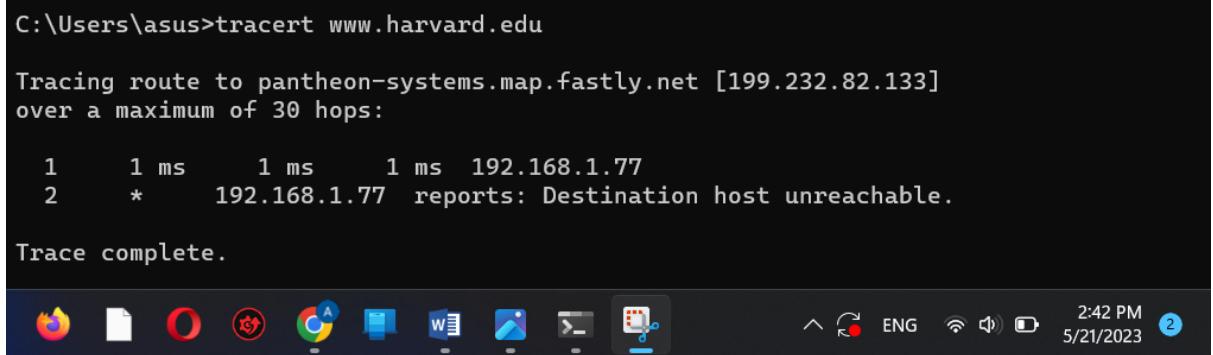
Ping statistics for 199.232.82.133:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 73ms, Maximum = 73ms, Average = 73ms

2:40 PM 5/21/2023
```

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



```
C:\Users\asus>tracert www.harvard.edu

Tracing route to pantheon-systems.map.fastly.net [199.232.82.133]
over a maximum of 30 hops:

 1      1 ms      1 ms      1 ms  192.168.1.77
 2      *      192.168.1.77  reports: Destination host unreachable.

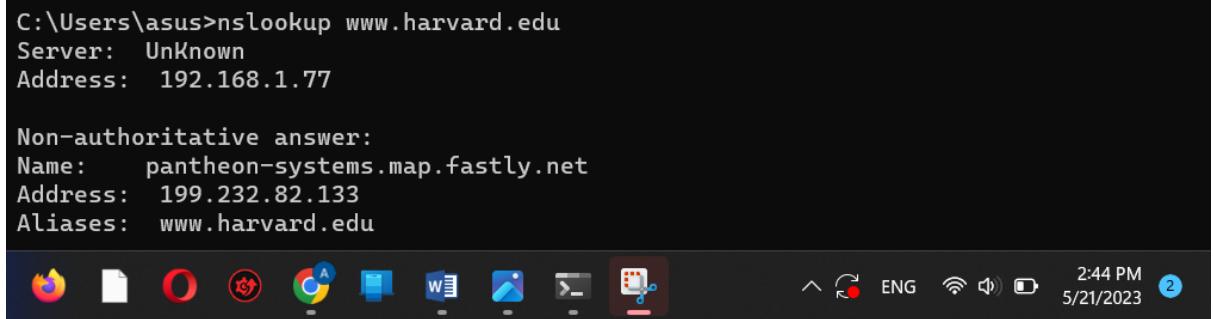
Trace complete.
```

The screenshot shows a Windows command prompt window. The title bar says "cmd.exe". The command "tracert www.harvard.edu" is entered. The output shows the tracing route to the website. The first two hops are successful (192.168.1.77), but the third hop is marked with an asterisk (*) and the message "reports: Destination host unreachable.", indicating a problem with the route or destination. The status bar at the bottom shows the date and time as 5/21/2023 2:42 PM.

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



```
C:\Users\asus>nslookup www.harvard.edu
Server:  Unknown
Address:  192.168.1.77

Non-authoritative answer:
Name:    pantheon-systems.map.fastly.net
Address:  199.232.82.133
Aliases:  www.harvard.edu
```

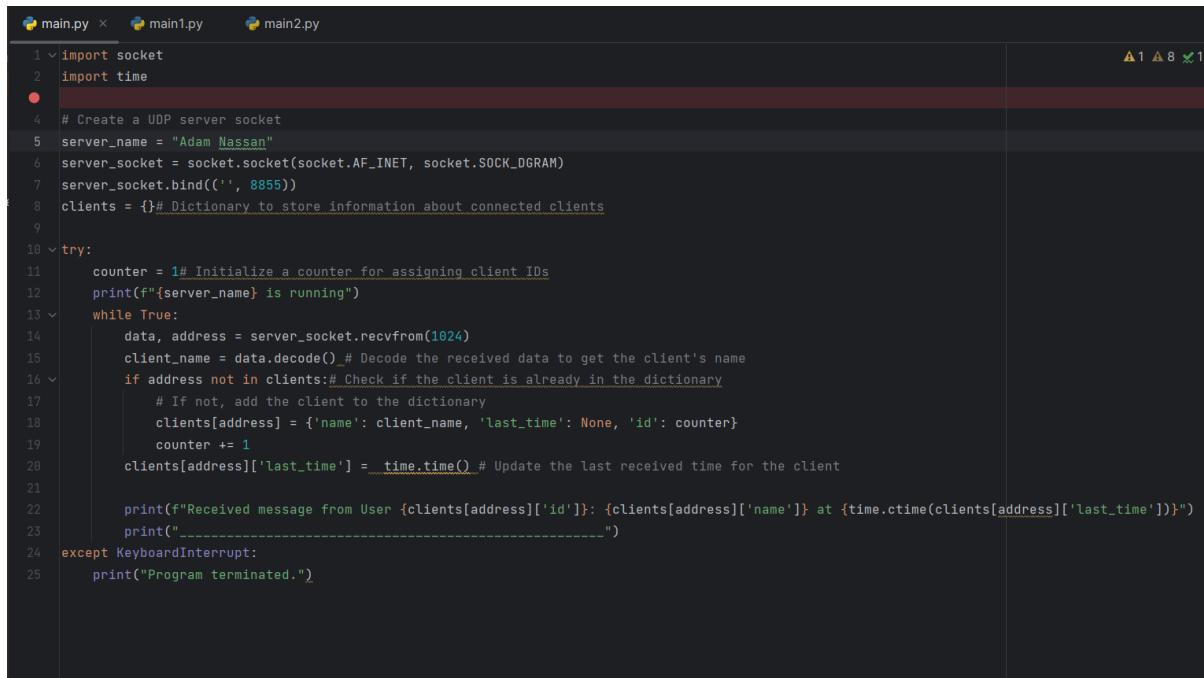
The screenshot shows a Windows command prompt window. The title bar says "cmd.exe". The command "nslookup www.harvard.edu" is entered. The output shows the non-authoritative answer. It lists the name "pantheon-systems.map.fastly.net" and its address "199.232.82.133", along with the alias "www.harvard.edu". The status bar at the bottom shows the date and time as 5/21/2023 2:44 PM.

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.



The screenshot shows a code editor with three tabs: main.py, main1.py, and main2.py. The main.py tab is active and contains the following Python code:

```
 1 import socket
 2 import time
 3
 4 # Create a UDP server socket
 5 server_name = "Adam Nassar"
 6 server_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
 7 server_socket.bind(('', 8855))
 8 clients = {}# Dictionary to store information about connected clients
 9
10 try:
11     counter = 1# Initialize a counter for assigning client IDs
12     print(f"{server_name} is running")
13     while True:
14         data, address = server_socket.recvfrom(1024)
15         client_name = data.decode() # Decode the received data to get the client's name
16         if address not in clients:# Check if the client is already in the dictionary
17             # If not, add the client to the dictionary
18             clients[address] = {'name': client_name, 'last_time': None, 'id': counter}
19             counter += 1
20         clients[address]['last_time'] = time.time() # Update the last received time for the client
21
22         print(f"Received message from User {clients[address]['id']}: {clients[address]['name']} at {time.ctime(clients[address]['last_time'])}")
23         print("-----")
24 except KeyboardInterrupt:
25     print("Program terminated.")
```

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.

```

1 import time
2 from socket import *
3
4 serverName = "192.168.1.255">#define the server IP address
5 serverPort = 8855#define the port number
6 clientSocket = socket(AF_INET, SOCK_DGRAM)
7 clientSocket.setsockopt(SOL_SOCKET, SO_BROADCAST, 1)

8
9 student_name = input("Enter your name: ")
10
11 try:
12     while True:
13         message = student_name
14         clientSocket.sendto(message.encode(), (serverName, serverPort))#send the encoded message to the server
15         time.sleep(2)#delay of 2 seconds
16 except KeyboardInterrupt:
17     print("Program terminated.")
18
19 clientSocket.close()
20

```

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
```

Activate Windows
Go to Settings to activate Windows

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
```

Activate Windows

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.

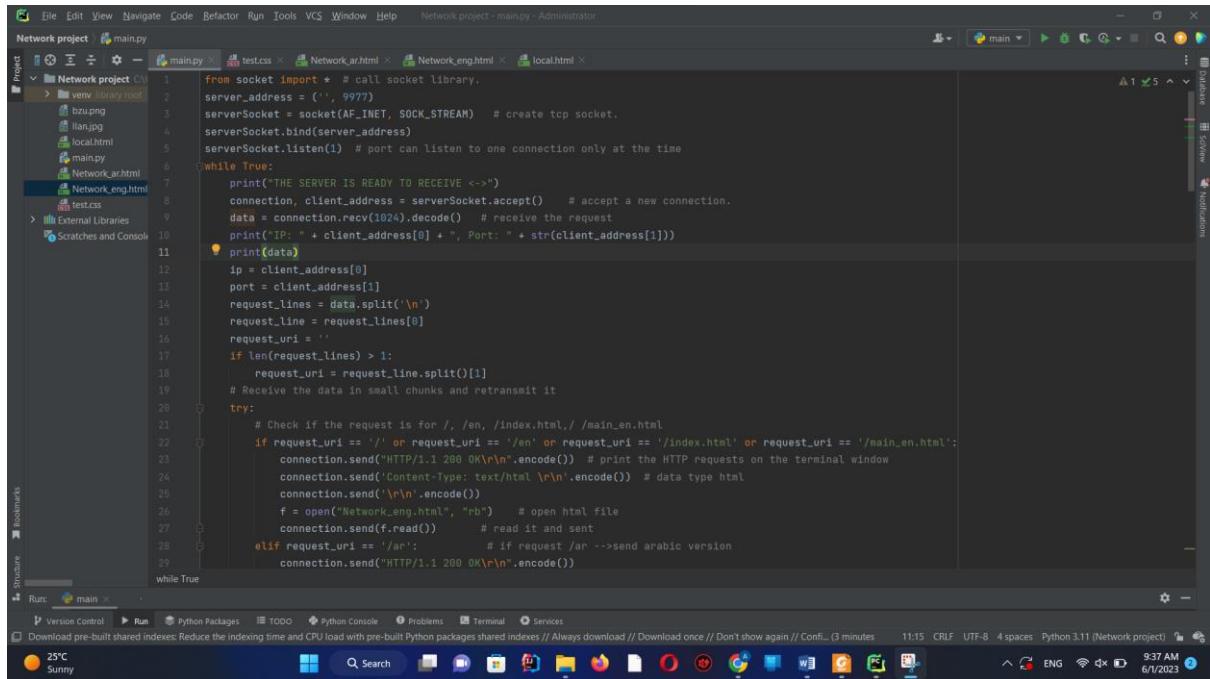
```
"C:\Users\Rana Odeh\PycharmProjects\pythonProject8\venv\Scripts\python.exe" "C:\Users\Rana Odeh\PycharmProjects\pythonProject8\main.py"
Adam Nassan is running
Received message from User 1: Rana Odeh at Thu Jun 1 21:29:29 2023
-----
Received message from User 1: Rana Odeh at Thu Jun 1 21:29:31 2023
-----
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:-

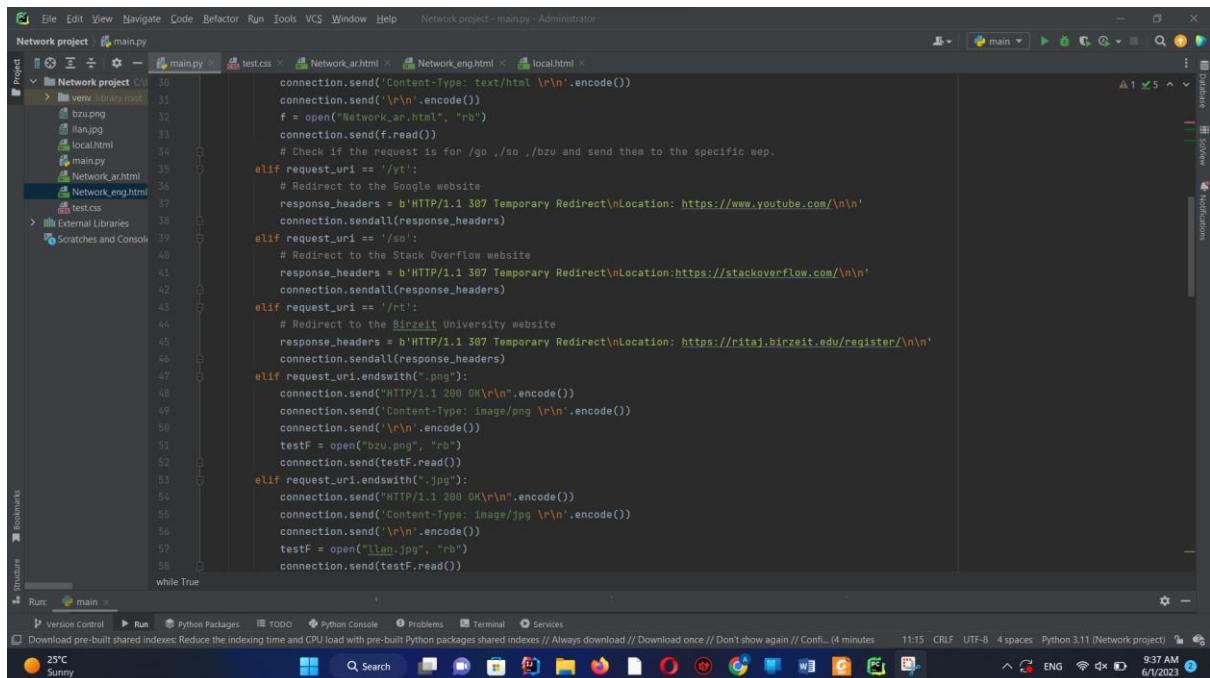
Using socket programming, implement a simple but a complete web server in go, python, java or C that is listening on port 9977.

1- Server code



```
from socket import * # call socket library.
server_address = ('', 9977)
serverSocket = socket(AF_INET, SOCK_STREAM) # create tcp socket.
serverSocket.bind(server_address)
serverSocket.listen(1) # port can listen to one connection only at the time
while True:
    print("THE SERVER IS READY TO RECEIVE <->")
    connection, client_address = serverSocket.accept() # accept a new connection.
    data = connection.recv(1024).decode() # receive the request
    print("IP: " + client_address[0] + ", Port: " + str(client_address[1]))
    print(data)
    ip = client_address[0]
    port = client_address[1]
    request_lines = data.split('\n')
    request_line = request_lines[0]
    request_uri = ''
    if len(request_lines) > 1:
        request_uri = request_line.split()[1]
    # Receive the data in small chunks and retransmit it
    try:
        # Check if the request is for /, /en, /index.html, /main_en.html
        if request_uri == '' or request_uri == '/en' or request_uri == '/index.html' or request_uri == '/main_en.html':
            connection.send("HTTP/1.1 200 OK\r\n".encode()) # print the HTTP requests on the terminal window
            connection.send("Content-type: text/html \r\n".encode()) # data type html
            connection.send("\r\n".encode())
            f = open("Network_eng.html", "rb") # open html file
            connection.send(f.read()) # read it and send
        elif request_uri == '/ar':
            # if request /ar -->send arabic version
            connection.send("HTTP/1.1 200 OK\r\n".encode())
    except:
        pass
while True:
```

Figure 10: Server code 1



```
connection.send("Content-Type: text/html \r\n".encode())
connection.send("\r\n".encode())
f = open("Network_ar.html", "rb")
connection.send(f.read())
# Check if the request is for /go ,so ./bzu and send them to the specific web.
elif request_uri == '/yt':
    # Redirect to the Google website
    response_headers = b'HTTP/1.1 307 Temporary Redirect\r\nLocation: https://www.youtube.com/\r\n'
    connection.sendall(response_headers)
elif request_uri == '/so':
    # Redirect to the Stack Overflow website
    response_headers = b'HTTP/1.1 307 Temporary Redirect\r\nLocation:https://stackoverflow.com/\r\n'
    connection.sendall(response_headers)
elif request_uri == '/rt':
    # Redirect to the Birzeit University website
    response_headers = b'HTTP/1.1 307 Temporary Redirect\r\nLocation: https://ritaj.birzeit.edu/register/\r\n'
    connection.sendall(response_headers)
elif request_uri.endswith(".png"):
    connection.send("HTTP/1.1 200 OK\r\n".encode())
    connection.send("Content-type: image/png \r\n".encode())
    connection.send("\r\n".encode())
    testF = open("bzu.png", "rb")
    connection.send(testF.read())
elif request_uri.endswith(".jpg"):
    connection.send("HTTP/1.1 200 OK\r\n".encode())
    connection.send("Content-type: image/jpg \r\n".encode())
    connection.send("\r\n".encode())
    testF = open("ilan.jpg", "rb")
    connection.send(testF.read())
while True:
```

Figure 11: Server code 2

The screenshot shows a code editor interface with a Python file named `main.py` open. The code handles HTTP requests for files like `index.html`, `Network_ar.html`, and `Network_eng.html`. It generates dynamic HTML responses with a greeting message and the current time. It also includes a JavaScript function to update the page's date and time every second.

```
elif request_uri.endswith(".html"):
    connection.send("HTTP/1.1 200 OK\r\n".encode())
    connection.send("Content-Type: text/html \r\n".encode())
    connection.send("\r\n".encode())
    # create a new html file
    s = """
        <html>
            <head>
                <style>
                    body {
                        font-size: 20px;
                        text-align: center;
                    }
                    .greeting {
                        color: blue;
                    }
                </style>
            </head>
            <body>
                <p class="greeting">Hello, world!</p>
                <p>The current time is: <span id="time"></span></p>
                <p>The current date is: <span id="date"></span></p>
            </body>
        </html>
    """
    connection.send(s.encode())
    connection.send("\r\n".encode())
while True
```

Figure 12: Server code 3

The screenshot shows a code editor interface with a Python file named `main.py` open. The code handles requests for CSS files like `test.css`. It reads the CSS content from a file and sends it back to the client. If a requested file is not found, it serves a default error HTML page.

```
var seconds = currentTime.getSeconds();
document.getElementById("time").innerHTML = hours + ":" + minutes + ":" + seconds;
} setInterval(updateTime, 1000);

function updateDate() {
    var currentDate = new Date();
    var month = currentDate.getMonth() + 1;
    var day = currentDate.getDate();
    var year = currentDate.getFullYear();
    document.getElementById("date").innerHTML = month + "/" + day + "/" + year;
}
updateDate();
</script>
</body>
</html>"})
connection.send(s.encode())
elif request_uri.endswith(".css"):
    connection.send("HTTP/1.1 200 OK\r\n".encode())
    connection.send("Content-Type: text/css \r\n".encode())
    s = open("test.css", "rb")
    connection.send(s.read()) # send css file
else: # if the request does not found then send this error html file
    connection.send("HTTP/1.1 200 OK\r\n".encode())
    connection.send("Content-Type: text/html \r\n".encode())
    connection.send("\r\n".encode())
    s = """
        <html>\r
            <head>\r
                <title>404 Error</title>\r
            </head>\r
            <body>\r
                <h1>404 - Page Not Found</h1>\r
            </body>\r
        </html>\r
    """
    connection.send(s.encode())
    connection.send("\r\n".encode())
while True
```

Figure 13: Server code 4

The screenshot shows the PyCharm IDE interface with the following details:

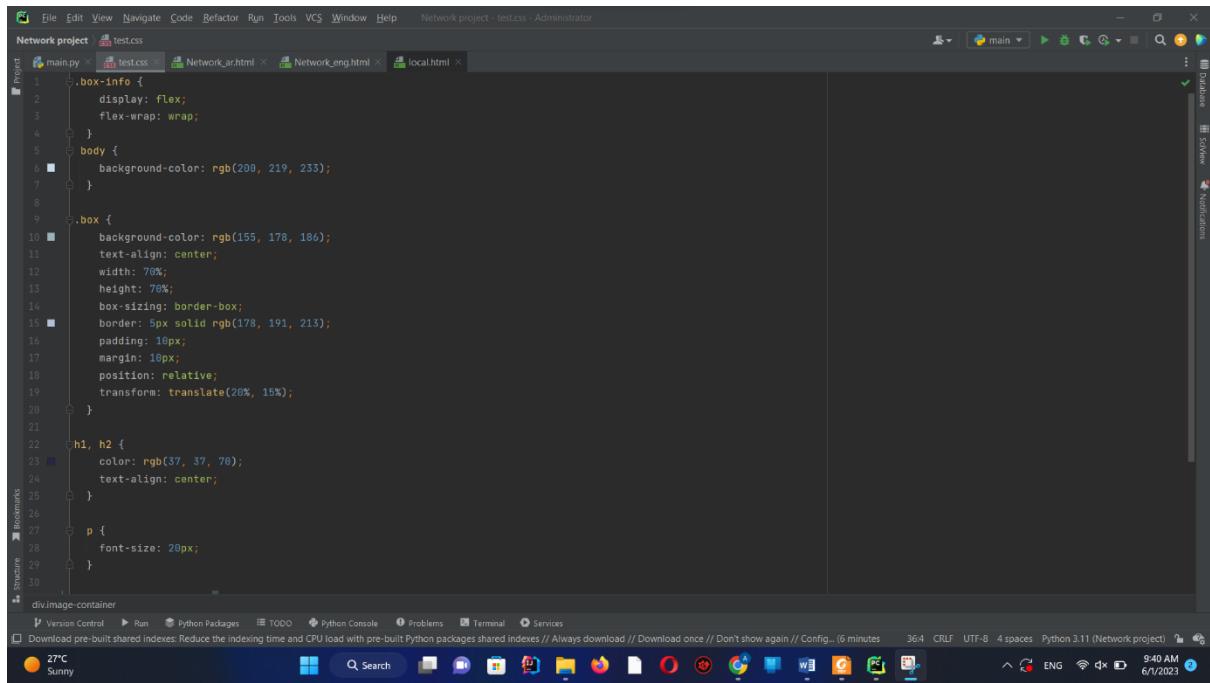
- Project:** Network project
- Code Editor:** main.py (selected tab)
- Code Content:** Python code for a simple web server. The code handles file requests and returns 404 errors with custom HTML. It includes imports for socket, os, and re, and uses a while True loop.

```
File Edit View Navigate Code Befactor Run Tools VCS Window Help Network project - main.py - Administrator
Network project main.py
Project > Network project > venv\library root 108
else: # if the request does not found then send this error html file
    connection.send("HTTP/1.1 200 OK\r\n".encode())
    connection.send('Content-Type: text/html\r\n'.encode())
    connection.send('\r\n'.encode())
    s = """
<html>\n"
    "      <head>\n"
    "          <title>Error</title>\n"
    "      </head>\n"
    "<body>\n"
    "      <h1>HTTP/1.1 404 Not Found</h1>\n"
    "      <h2>Error 404</h2>\n"
    "      <p style='color:red;'>The file is not found</p>\n"
    "      <p><b>Hidayah Mustafa 1201910<br>Rana Odeh 1201750<br>Adam Hassan 1202076</b></p>\n"
    "      <h1> IP: " + str(ip) + ", Port: " + str(port) + "</h1>\n"
    "      </body>\n"
"</html>
"""
    connection.send(s.encode())
finally:
    # Close the connection
    connection.close()
while True
```

- Toolbars and Status Bar:** Version Control, Run, Python Packages, TODO, Python Console, Problems, Terminal, Services. Status bar shows: Download pre-built shared indexes: Reduce the indexing time and CPU load with pre-built Python packages shared indexes // Always download // Download once // Don't show again // Configuration (5 minutes), 11:15, CRLF, UTF-8, 4 spaces, Python 3.11 (Network project).
- Bottom Icons:** Weather (25°C, Sunny), Search, Services, and various system icons.

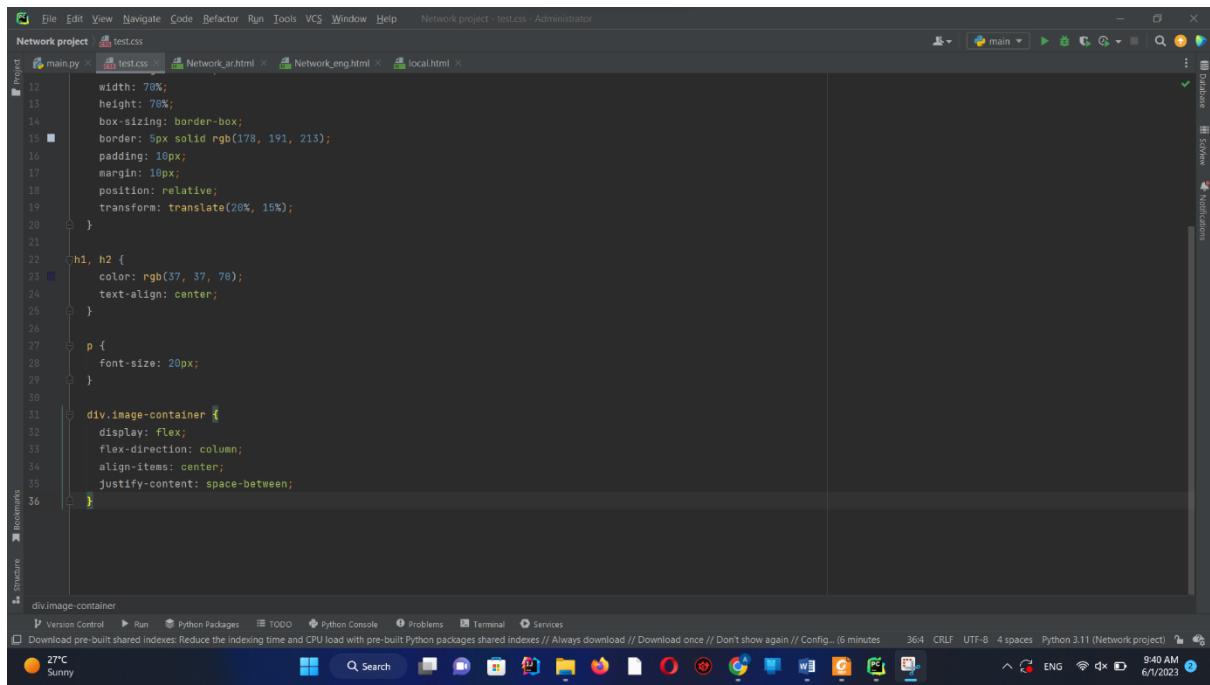
Figure 14: Server code 5

2- CSS code



```
.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

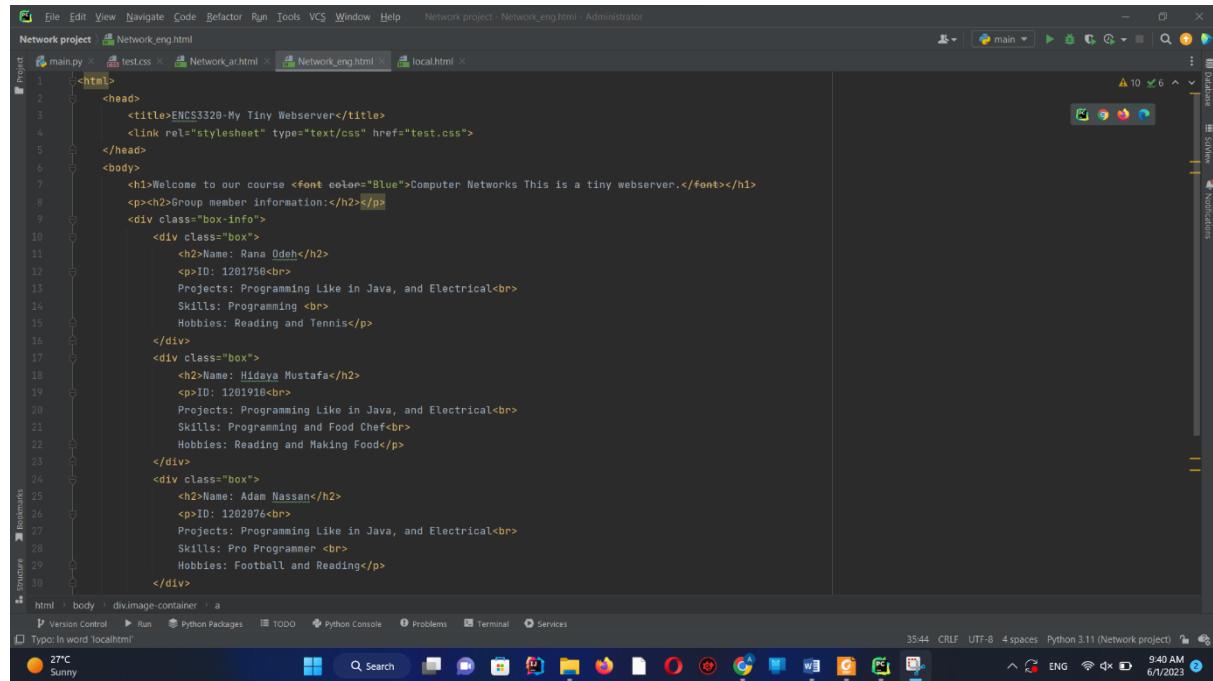


```
.width: 70%;
.height: 70%;
.box-sizing: border-box;
border: 5px solid rgb(178, 191, 213);
padding: 10px;
margin: 10px;
position: relative;
transform: translate(20%, 15%);
}
h1, h2 {
    color: rgb(37, 37, 70);
    text-align: center;
}
p {
    font-size: 20px;
}
div.image-container {
    display: flex;
    flex-direction: column;
    align-items: center;
    justify-content: space-between;
}
```

Figure 16: CSS code 2

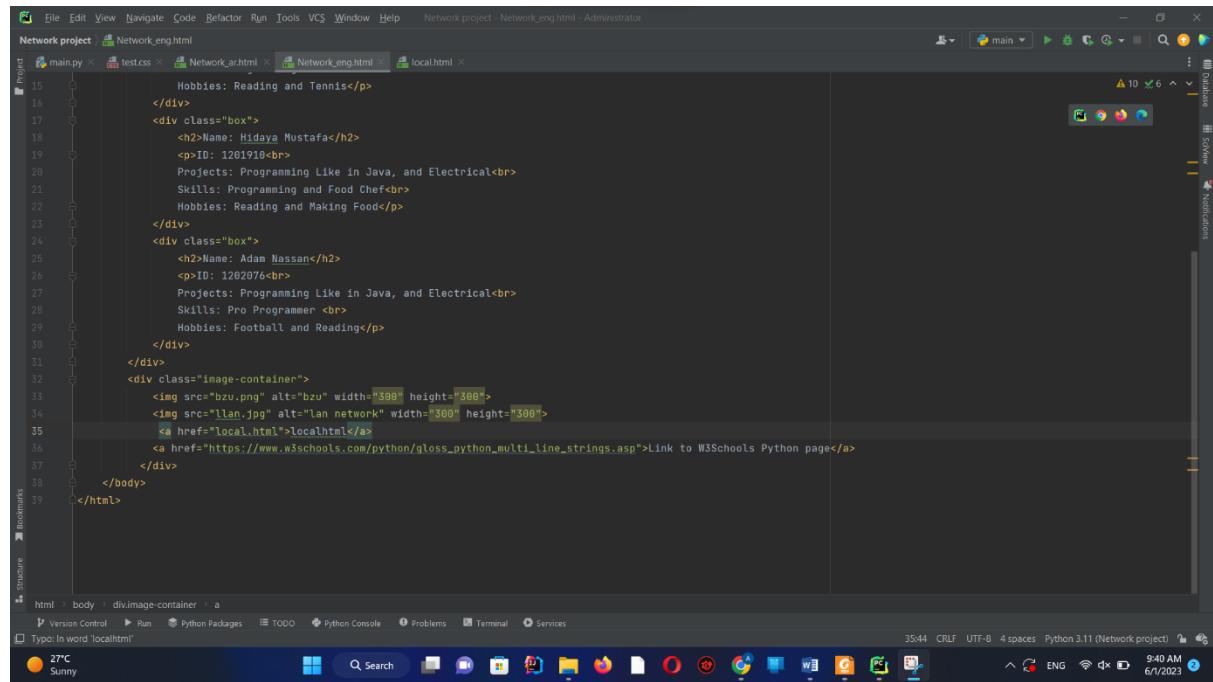
3- HTML code

a) ENG code



```
<html>
    <head>
        <title>ENCS3320- My Tiny Webserver</title>
        <link rel="stylesheet" type="text/css" href="test.css">
    </head>
    <body>
        <h1>Welcome to our course <font color="Blue">Computer Networks This is a tiny webserver.</font></h1>
        <p><h2>Group member information:</h2></p>
        <div class="box-info">
            <div class="box">
                <h2>Name: Rana Odeh</h2>
                <p>ID: 1201756<br>
                    Projects: Programming Like in Java, and Electrical<br>
                    Skills: Programming <br>
                    Hobbies: Reading and Tennis</p>
            </div>
            <div class="box">
                <h2>Name: Hidaya Mustafa</h2>
                <p>ID: 1201910<br>
                    Projects: Programming Like in Java, and Electrical<br>
                    Skills: Programming and Food Chef<br>
                    Hobbies: Reading and Making Food</p>
            </div>
            <div class="box">
                <h2>Name: Adam Nassan</h2>
                <p>ID: 1202076<br>
                    Projects: Programming Like in Java, and Electrical<br>
                    Skills: Pro Programmer <br>
                    Hobbies: Football and Reading</p>
            </div>
        </div>
        <div class="image-container">
            
            
            <a href="local.html">localhtml</a>
            <a href="https://www.w3schools.com/python/gloss_python_multi_line_strings.asp">Link to W3Schools Python page</a>
        </div>
    </body>
</html>
```

Figure 17: ENG code 1



```
<html>
    <head>
        <title>ENCS3320- My Tiny Webserver</title>
        <link rel="stylesheet" type="text/css" href="test.css">
    </head>
    <body>
        <h1>Welcome to our course <font color="Blue">Computer Networks This is a tiny webserver.</font></h1>
        <p><h2>Group member information:</h2></p>
        <div class="box-info">
            <div class="box">
                <h2>Name: Rana Odeh</h2>
                <p>ID: 1201756<br>
                    Projects: Programming Like in Java, and Electrical<br>
                    Skills: Programming <br>
                    Hobbies: Reading and Tennis</p>
            </div>
            <div class="box">
                <h2>Name: Hidaya Mustafa</h2>
                <p>ID: 1201910<br>
                    Projects: Programming Like in Java, and Electrical<br>
                    Skills: Programming and Food Chef<br>
                    Hobbies: Reading and Making Food</p>
            </div>
            <div class="box">
                <h2>Name: Adam Nassan</h2>
                <p>ID: 1202076<br>
                    Projects: Programming Like in Java, and Electrical<br>
                    Skills: Pro Programmer <br>
                    Hobbies: Football and Reading</p>
            </div>
        </div>
        <div class="image-container">
            
            
            <a href="local.html">localhtml</a>
            <a href="https://www.w3schools.com/python/gloss_python_multi_line_strings.asp">Link to W3Schools Python page</a>
        </div>
    </body>
</html>
```

Figure 18: ENG code 2

b) AR code

The screenshot shows the PyCharm IDE interface with the following details:

- File Menu:** File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, Help.
- Project Bar:** Network project - Network_ar.html - Administrator.
- Toolbars:** Choose direction, Hide notification, Don't show again.
- External Libraries:** venv library root, bzip2ng, libraqm, local.html, main.py, Network_ar.html, Network_eng.html, test.css.
- Scratches and Console:** External Libraries, Scratches and Console.
- Code Editor:** Visual layout of bidirectional text can depend on the base direction (View | Bidi Text Base Direction). The code is in Arabic and uses the "Arabic" font color. It includes comments in English and Arabic, such as "هذه هي صيغة استجابة لـ" and "يمكن تصميمه بسهولة عن طريق تعين صفة ملئ أو عرض له".
- Bottom Navigation:** Run, main, Version Control, Run, Python Packages, TODO, Python Console, Problems, Terminal, Services.
- Status Bar:** Download pre-built shared indexes: Reduce the indexing time and CPU load with pre-built Python packages shared indexes // Always download // Download once // Don't show again // Config... (7 minutes), 43.8 CRLF, UTF-8, 4 spaces, Python 3.11 (Network project), 6/1/2023, 9:41 AM, ENG, Wi-Fi, Battery level: 27°C, Temp drop.

Figure 19: AR code 1

Figure 20: AR code 2

4- Code results

a. ENG page

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

Group member information:

Name	ID	Projects	Skills	Hobbies
Rana Odeh	1201750	Programming Like in Java, and Electrical	Programming	Reading and Tennis
Hidayah Mustafa	1201910	Programming Like in Java, and Electrical	Programming and Food Chef	Reading and Making Food
Adam Nassan	1202076	Programming Like in Java, and Electrical	Pro Programmer	Football and Reading

BIRZEIT UNIVERSITY

Figure 21: ENG page 1



Figure 22: ENG page 2

b. AR page

أهلا بك في مساق شبكات الحاسوب خادم الميرور الصغير.

معلومات أعضاء المجموعة:

الاسم	الرقم	المشاريع	المهارات	الهوايات
رنا عودة	1201750	برمجة جافا و دارات كهربائية	برمجة	قراءة وتنفس
هداية مصطفى	1201910	برمجة جافا و دارات كهربائية	برمجة وطبخ	قراءة و الطبخ
آدم النصان	1202076	برمجة جافا و دارات كهربائية	برمجة	كرة قدم و القراءة

BIRZEIT UNIVERSITY

Figure 23: AR page

c. Local page link

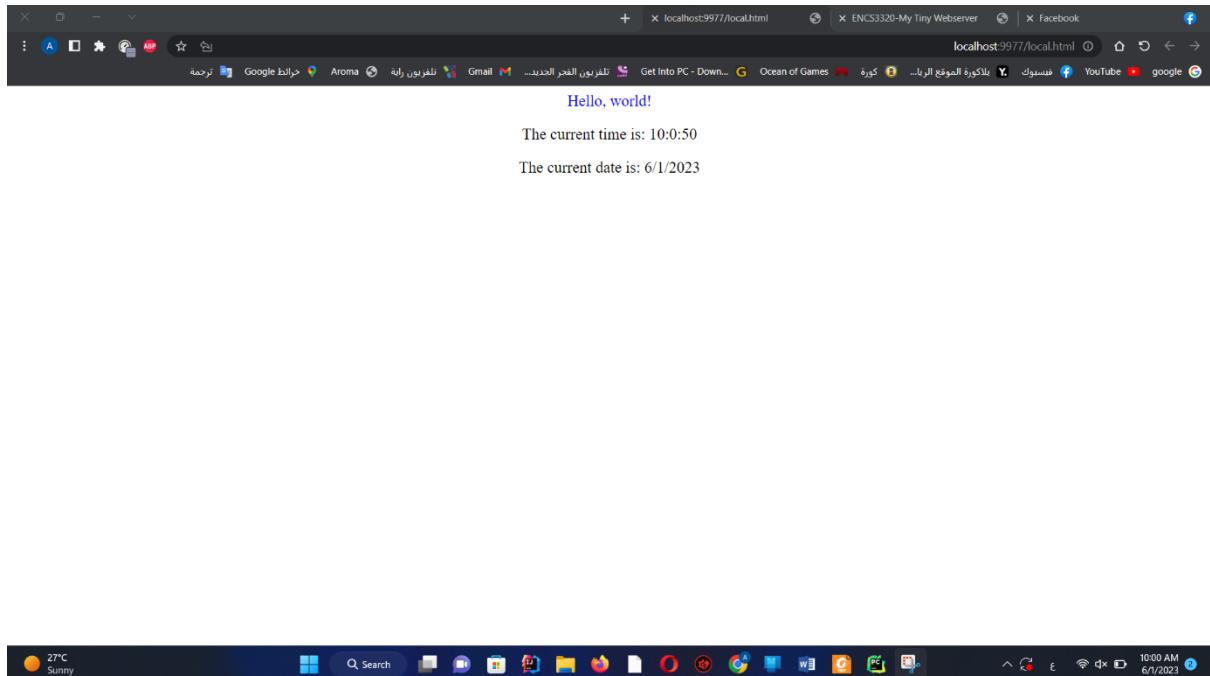


Figure 24: Local page

d. W3school link

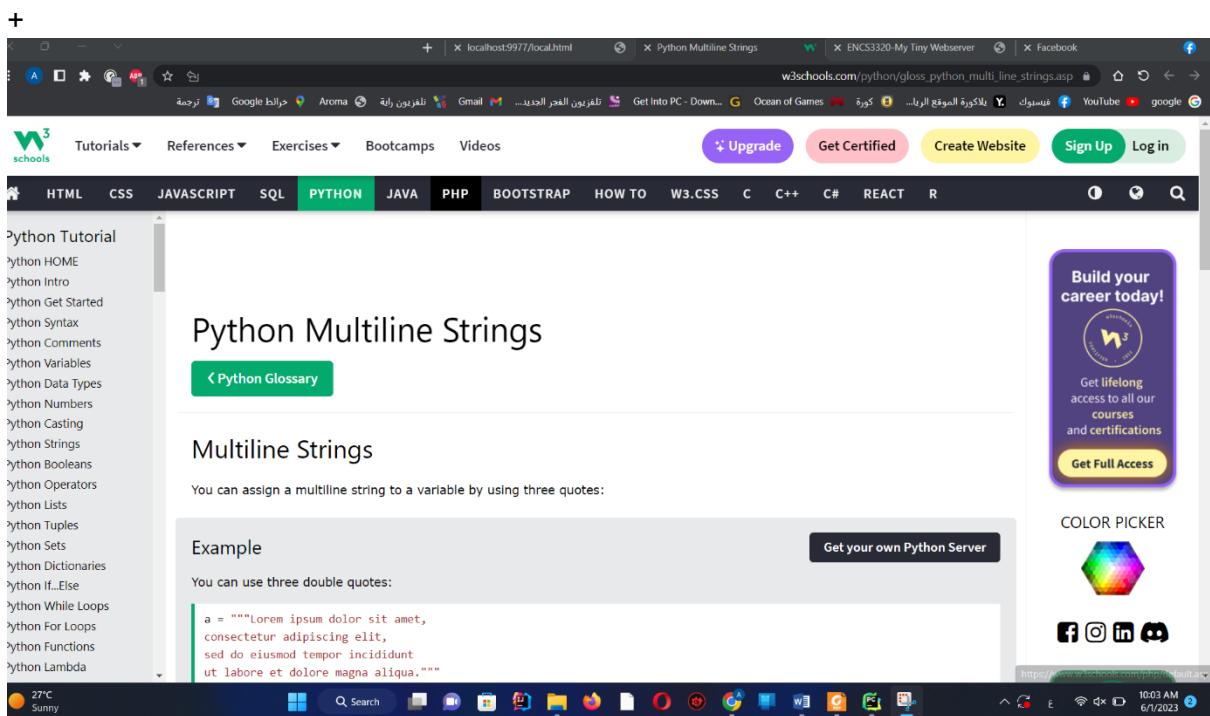


Figure 25: W3School page

e. Status code 307 Temporary Redirect

- If the request is /yt

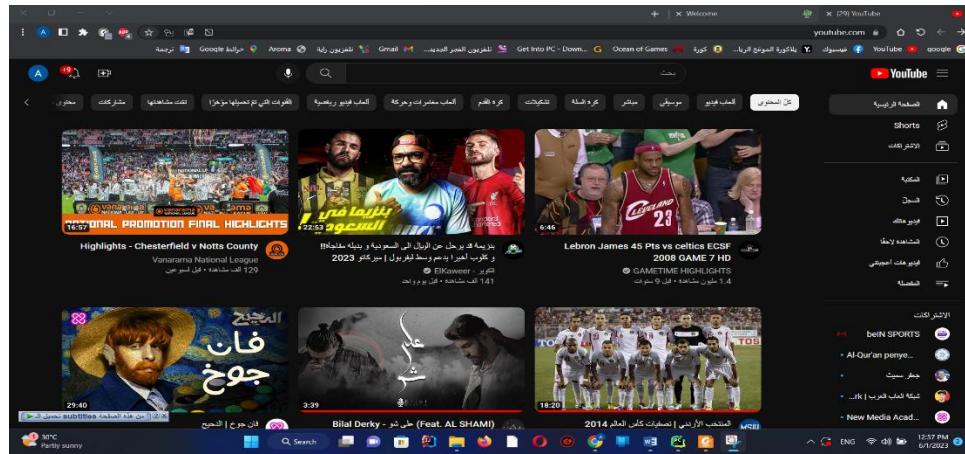


Figure 26: yt request

- If the request is /rt

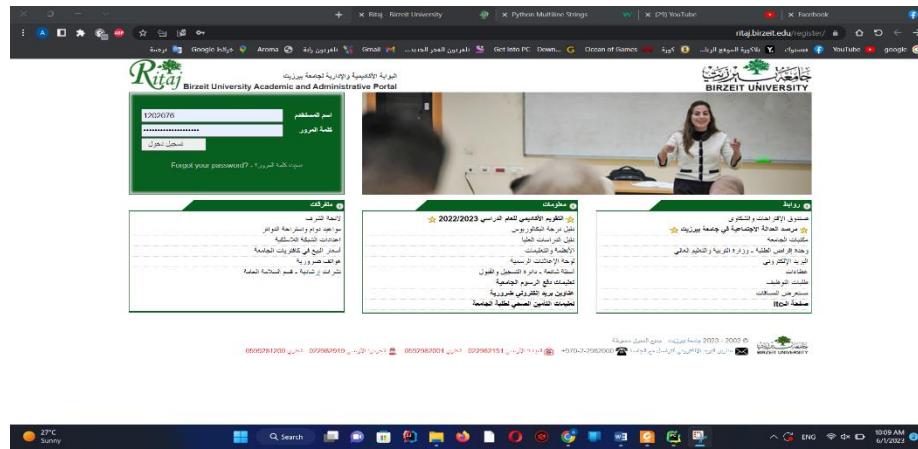


Figure 27: rt request

- If the request is /so

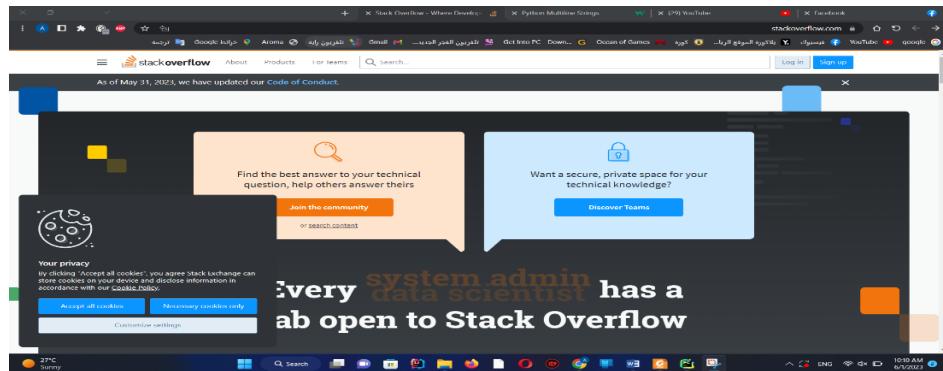


Figure 28: so request

f. Try some requests

- If the request is HTML

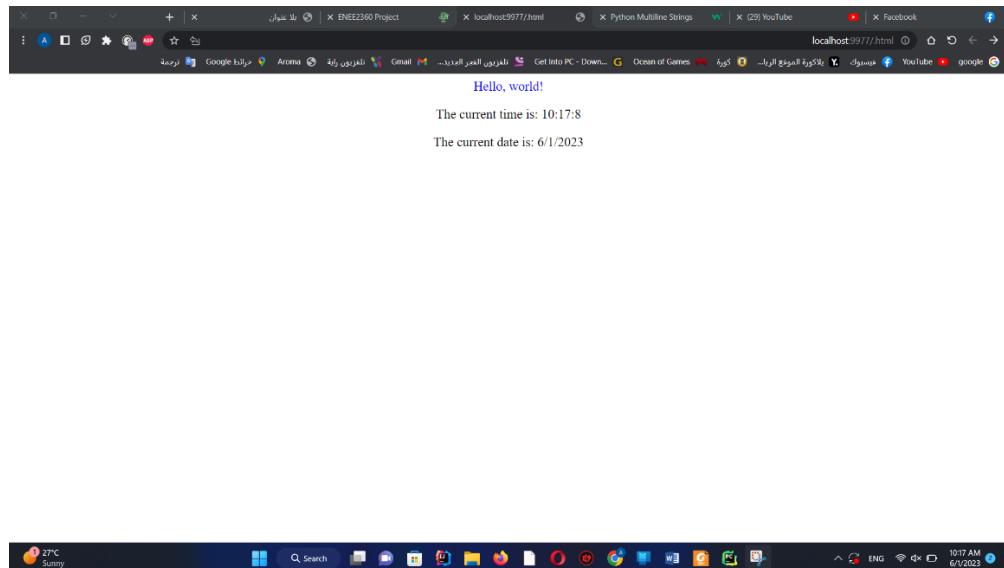


Figure 29: .HTML request

- If the request is .CSS

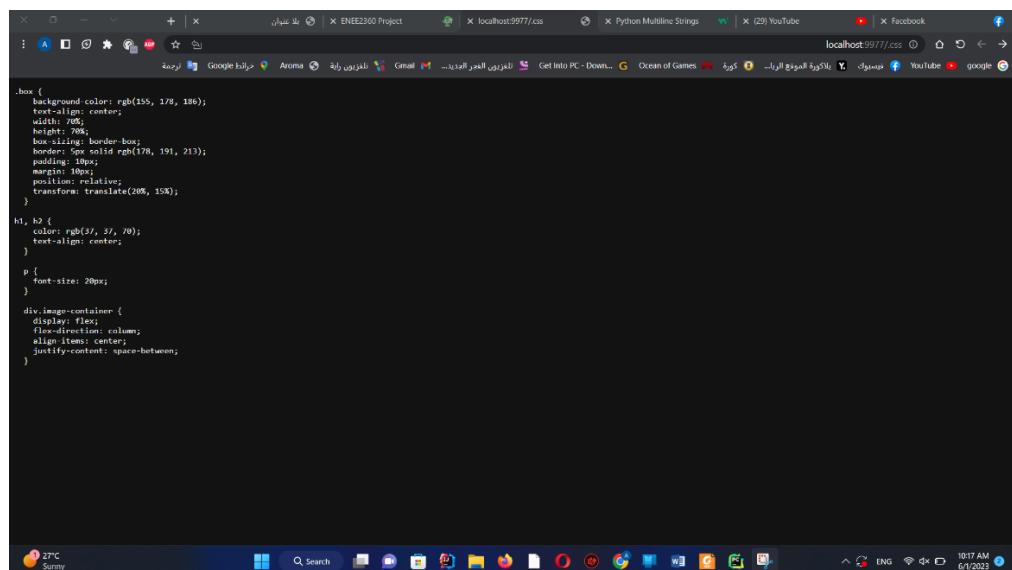


Figure 30: .CSS request

- If the request is .PNG

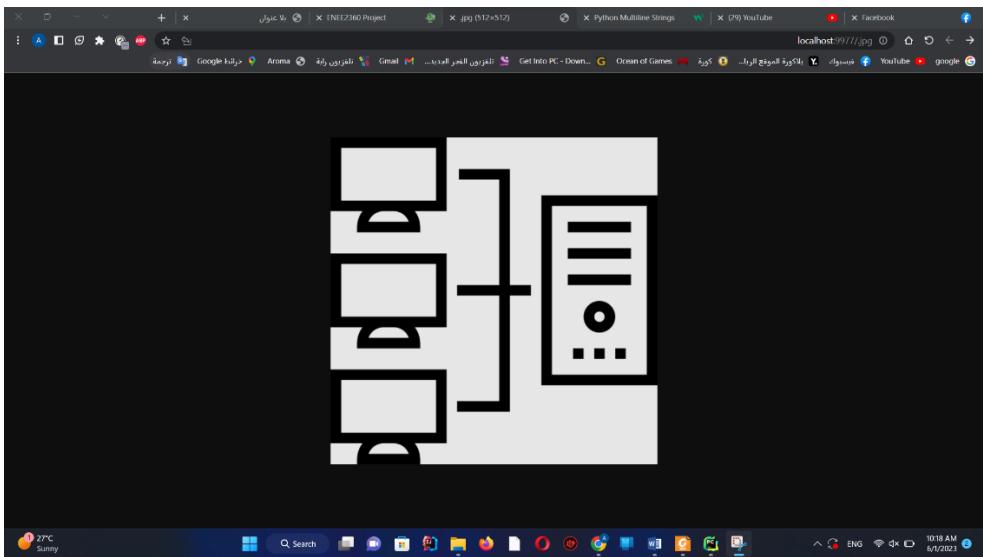


Figure 31: .PNG request

- If the request is .JPG

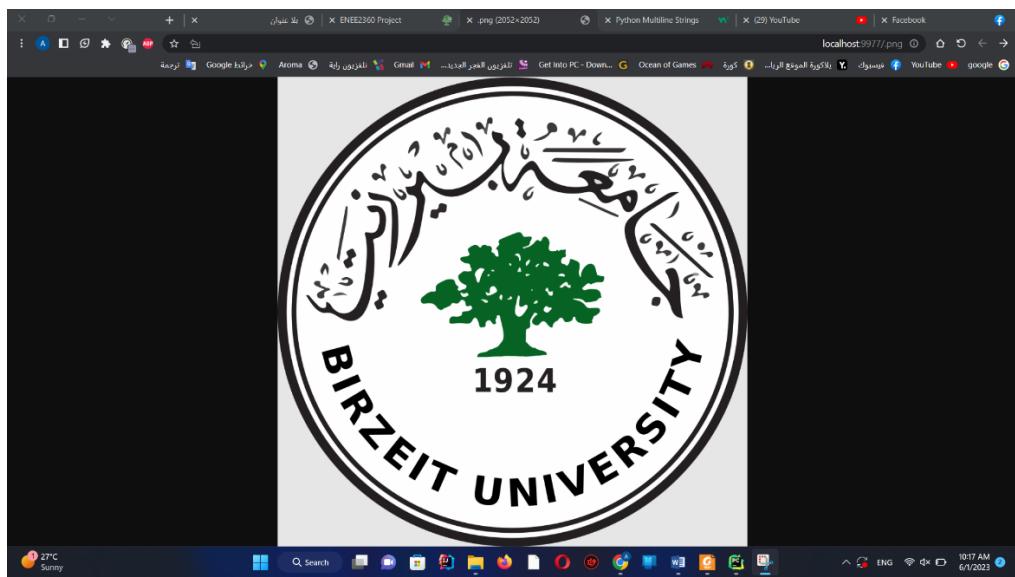


Figure 32: .JPG request

- In case of wrong request

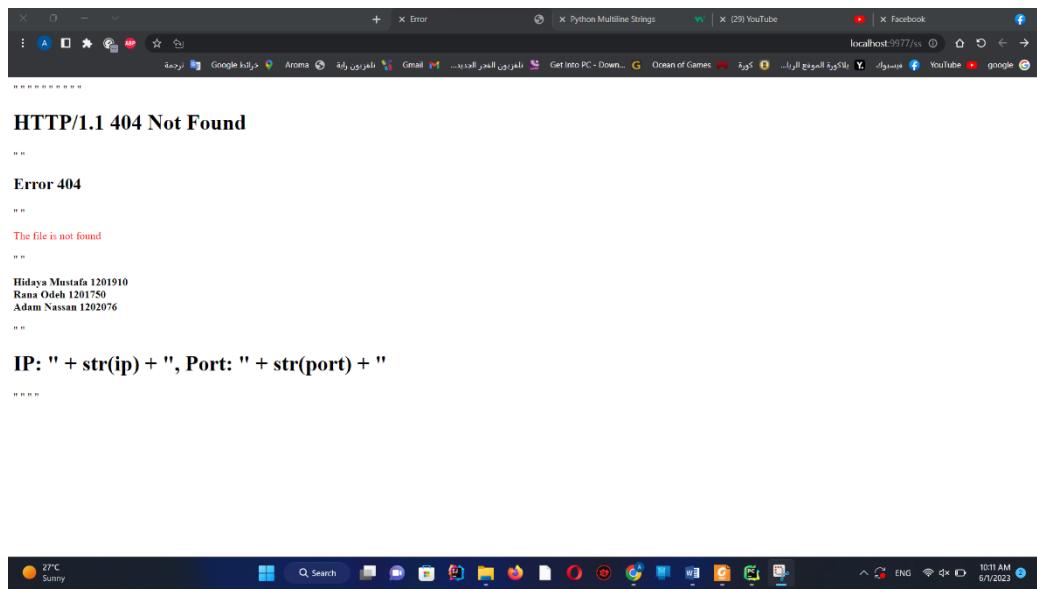


Figure 33: Wrong request

g. Test the project in smartphone

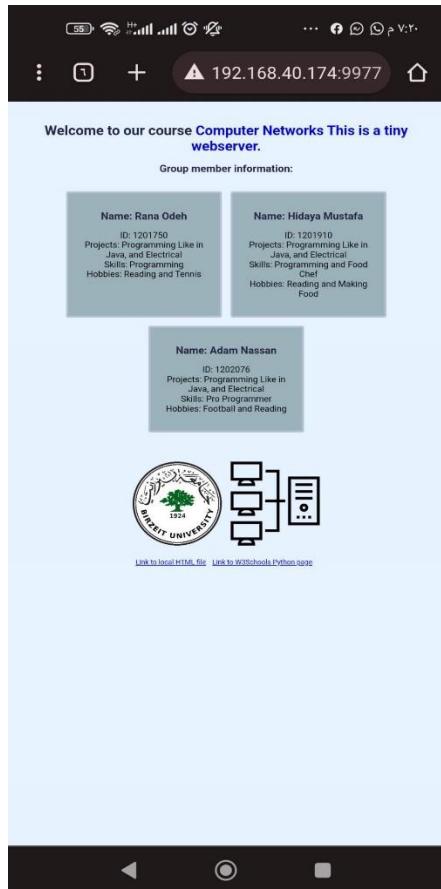


Figure 34: Smartphone test