Electrical and Computer Engineering Department

Computer Networks – ENCS3320

Project #1

Student name: Kareem Qutob ID:1211756

Partner name: Ghandy Ghanem ID: 1212386

Partner name: Qais Ayyash ID:1213205

Instructor name: Dr. Mohammed Jubran

Section: 2

Abstract

This group project explores fundamental networking concepts through hands-on socket programming. Divided into three parts, it encompasses understanding network utilities, implementing a TCP client-server application for screen locking, and creating a web server capable of serving dynamic content. Using Python, the team executes commands like ping, tracert, and nslookup, capturing DNS messages with Wireshark. The TCP application responds differently based on student IDs, initiating a screen lock for group members. The web server handles diverse content types, implements redirects, and displays custom error pages.

Content table :

List of figures:

Part 1:

1) What are ping, tracert , nslookup and telnet

ping: a command line utility that tests if a networked device is connected / responsive by measuring the RTT time of a packet sent to it

tracert: a command line utility that traces the route for a data packet from a source to a specified destination

nslookup: a command line utility to retrieve domain names, IP address

telnet: a command line protocol to establish a text-based connection between two devices over a network

2) Commands screenshot

A) Ping a device in the same network

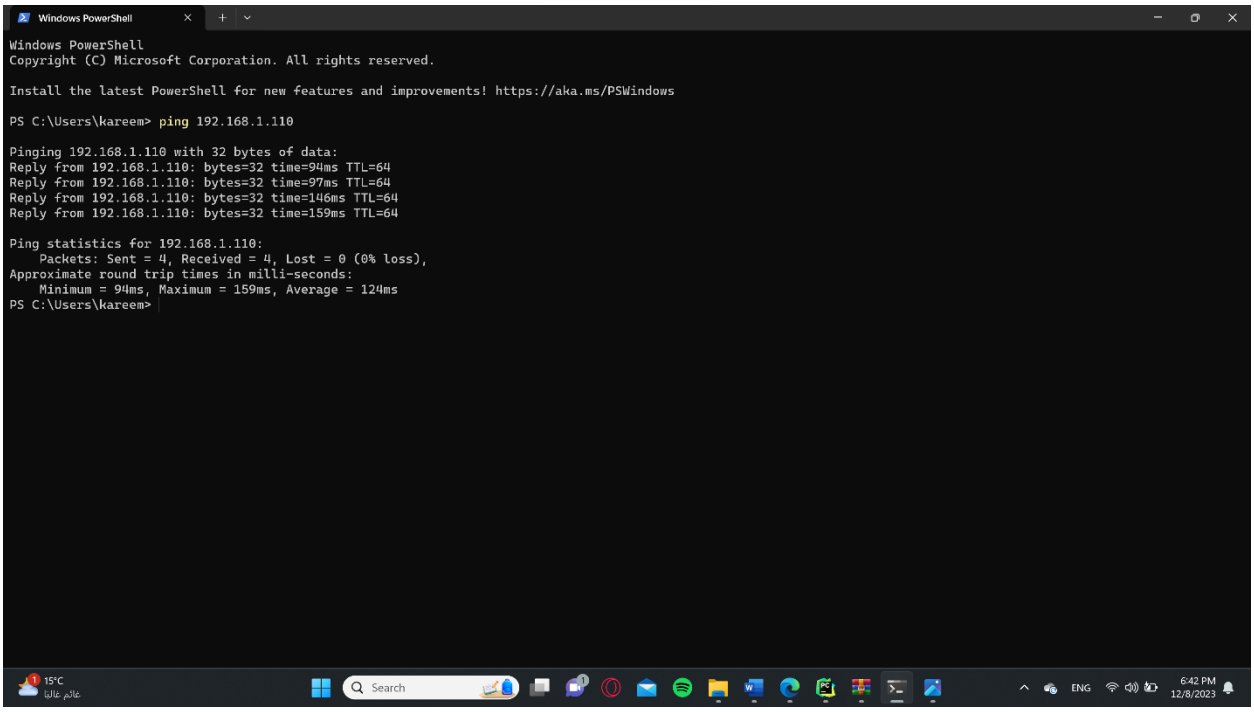


Figure 1: pinging a device on the same network

we pinged a phone with IP 192.168.1.110 on the same network as the computer that we run the command in, and a successful reply was received, four packets were received with an average RTT time of 124ms. %0 percent lost indicates a stable and responsive connection which is expected because we are working on the same network.

B) ping www.cornell.edu

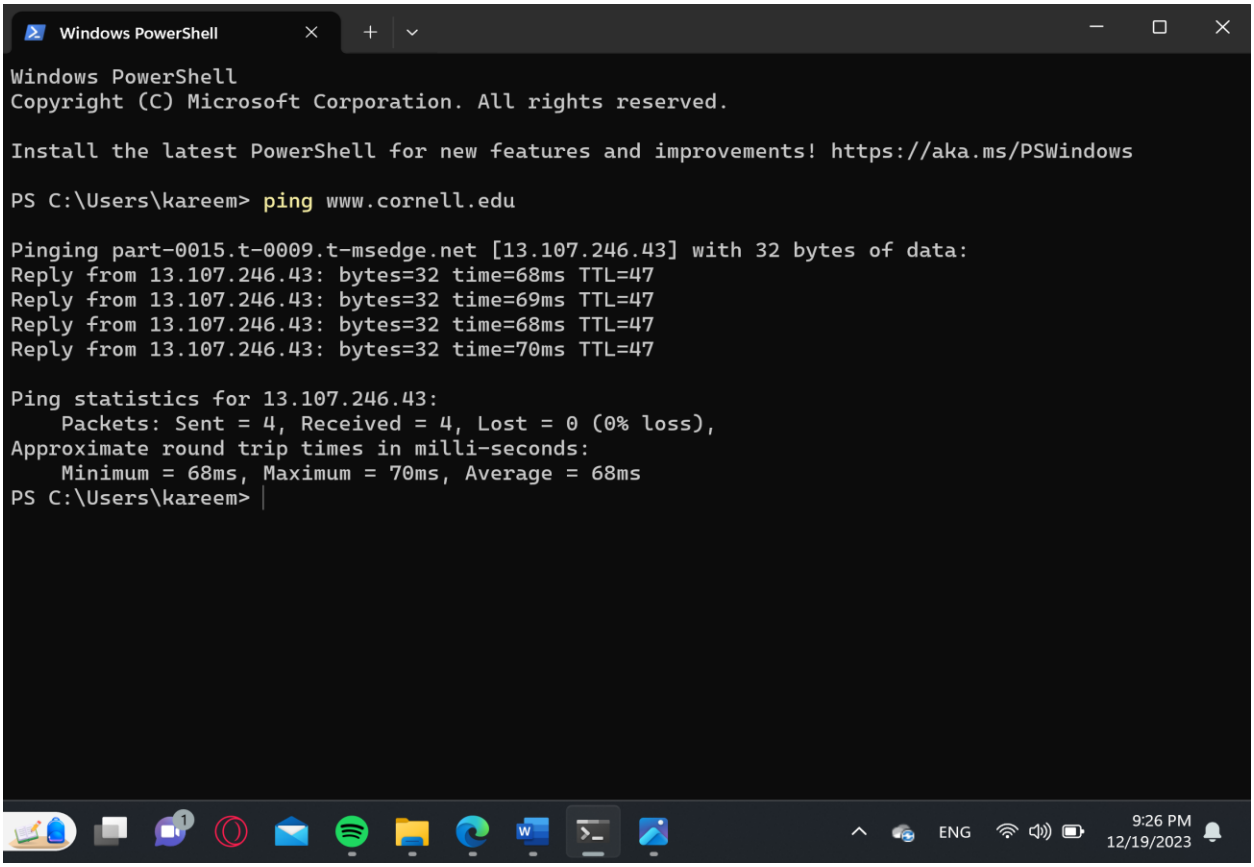


Figure 2:ping www.cornell.edu

The device used successfully communicated with the server that has IP 13.107.246.43, which is associated with the domain www.cornell.com, the 4 packets were received successfully with an average delay of 68ms The "0% loss" suggests a stable connection, and all hardware such as cables, the network card, the router, and the modem are all working correctly, including the internet service provider.

Based on the ping results it is hard to tell if the response is from the USA the response times are relatively low which could indicate that the server is close demographically the internet's routing and infrastructure complexities make it difficult to pinpoint the exact location. The IP address 13.107.213.43 after searching is associated with Microsoft Edge servers and Microsoft uses CDNs to distribute content globally which means that server placements are in a place nearer than the USA but that does not mean that the website is not based in the USA.

C) tracert www.cornell.edu

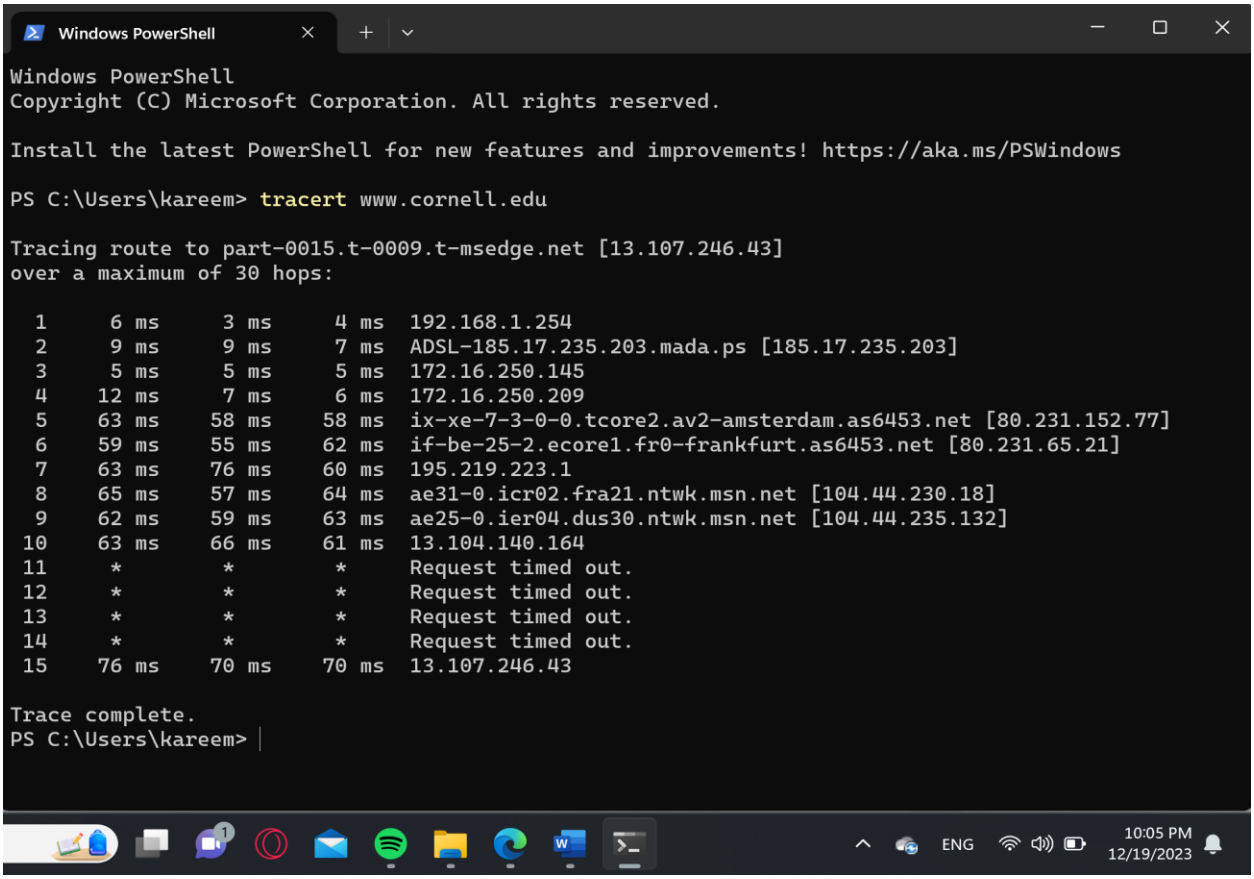


Figure 3: tracert www.cornell.edu

The tracert command is used to trace the route of data packets from our device to a specified destination in this case www.cronell.edu showing IP addresses of hops and routers with their RTT, as shown in Figure 3 there are 15 hops with the first one being the local router then the second one is an ISP , the * means that the request timed out which mean the hops did not respond to the tracert request

D) nslookup www.cornell.edu

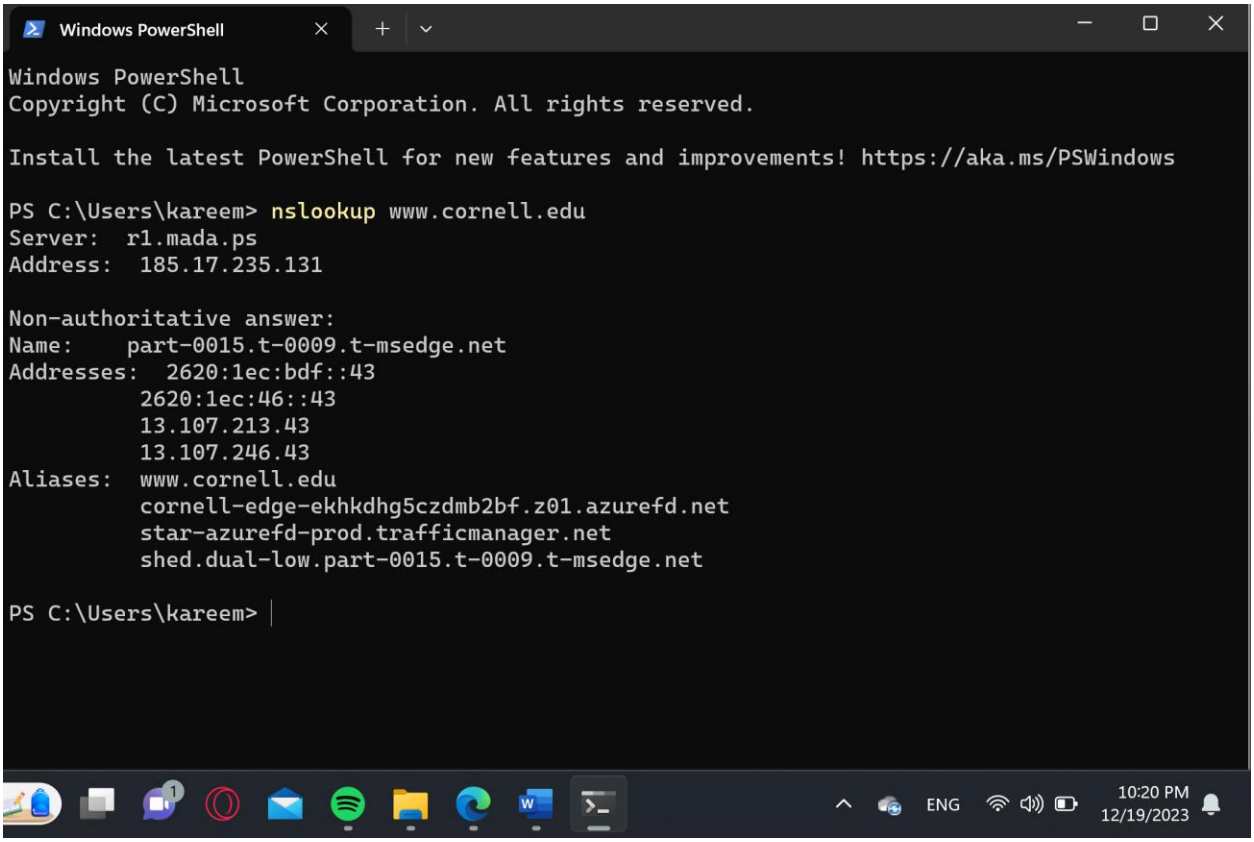


Figure 4: nslookup www.cornell.edu

nslookup command reveal DNS information of www.cornell.com , a list of all IPs associated with the domain part, and alternative names with the same IP address

3) wireshark to capture some DNS messages.

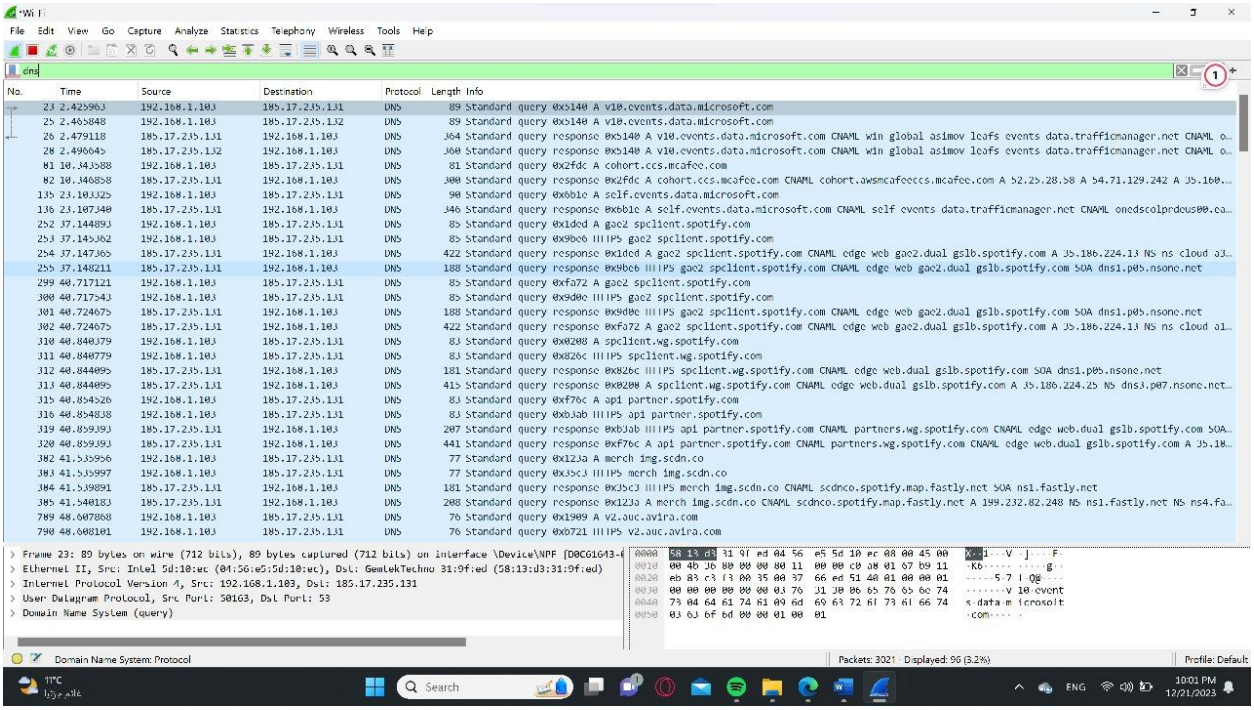


Figure 5: wireshark DNS captured packets

The capture paints a picture of internet activity, with a focus on DNS resolution and potential Spotify usage(which I was using at that time).and many other dns queries that suggest the computer was actively resolving domain names for various online activities.

Part 2: Implementation of a server-client applications

Using socket programming in python we implemented a simple TCP server-client application that listen on port 9955, the server will wait message from client if message has one of our group IDs it will display a message that the OS will close the lock screen in 10 seconds, otherwise it will display an error message.

Server code:

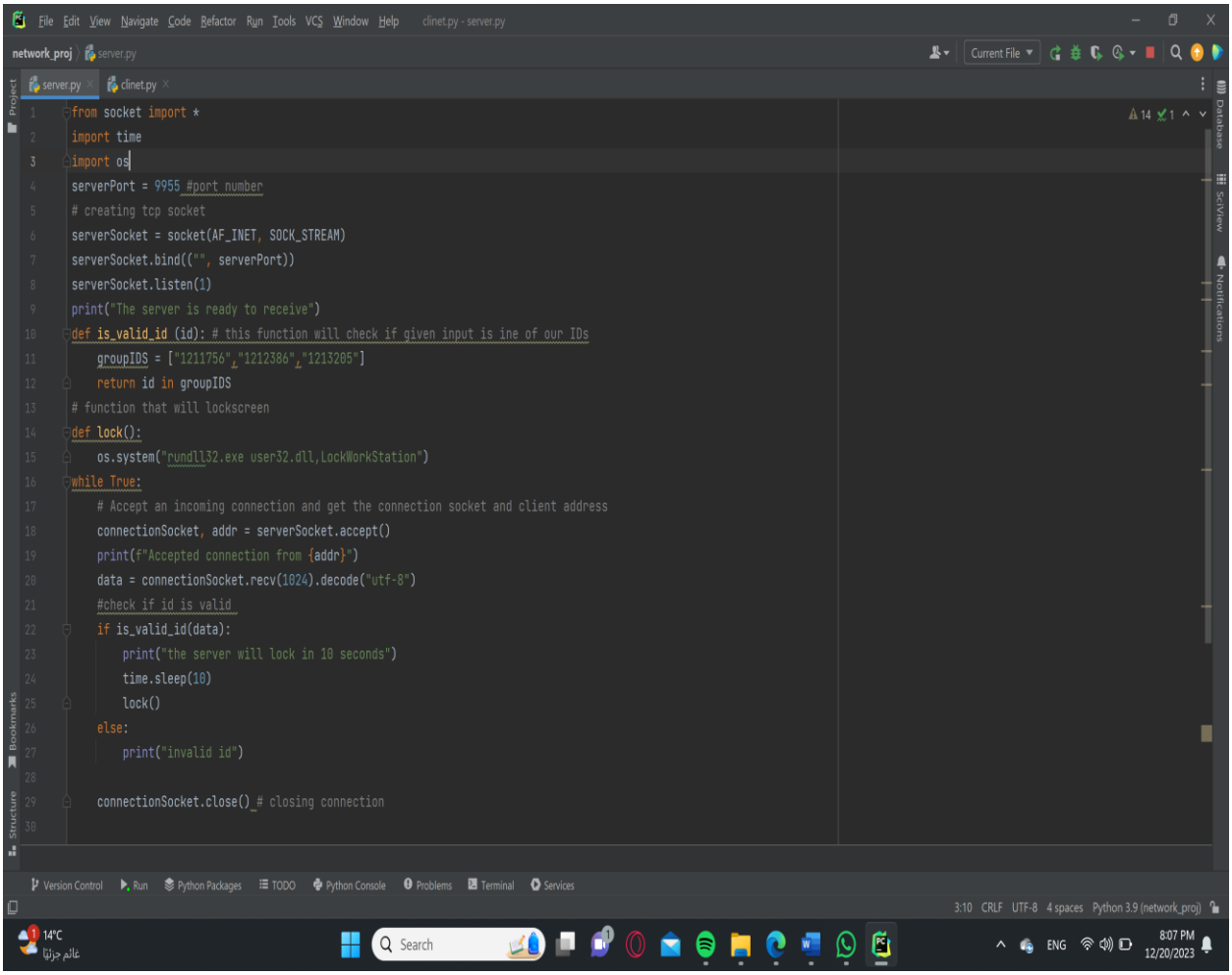


Figure 6:TCP server code

A simple server code that create a TCP connection and wait for a client to send input to it

Client code:

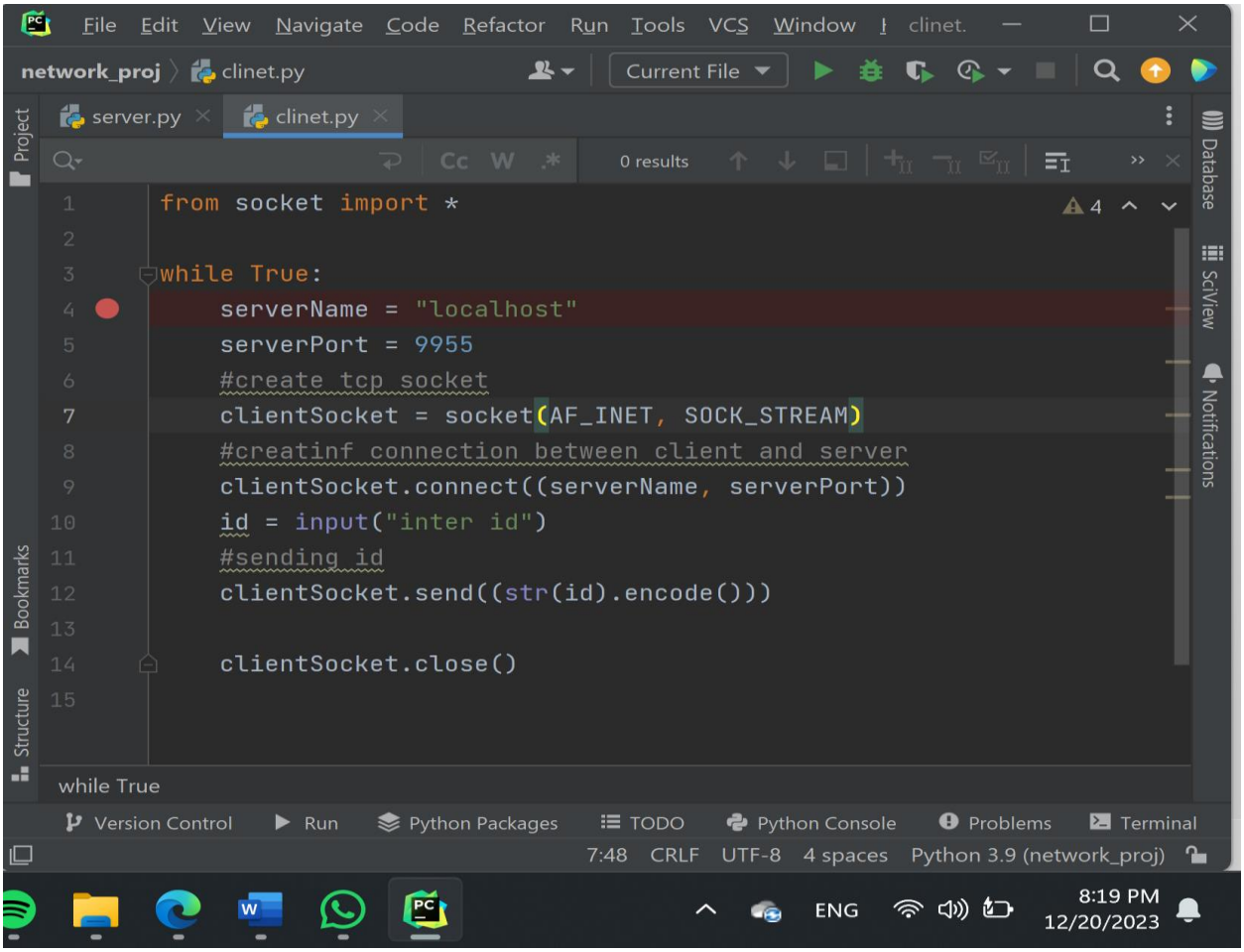


Figure 7:Client code

A simple client code that will create a TCP connection and send input for the server to process

Code run:

For testing we will send two inputs the first input will be a random string and the second will be a an ID for one of group members

Figure 8:client-server testcases

Code result:

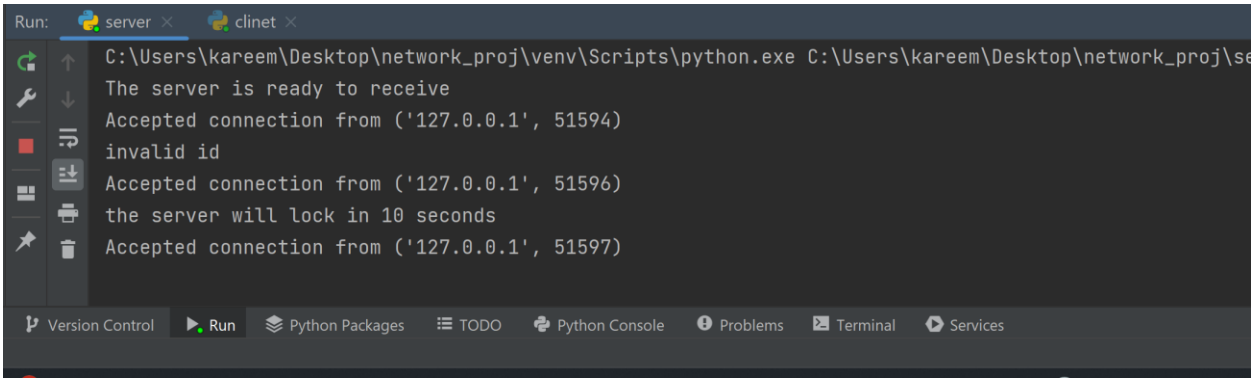


Figure 9:server-clinet code run

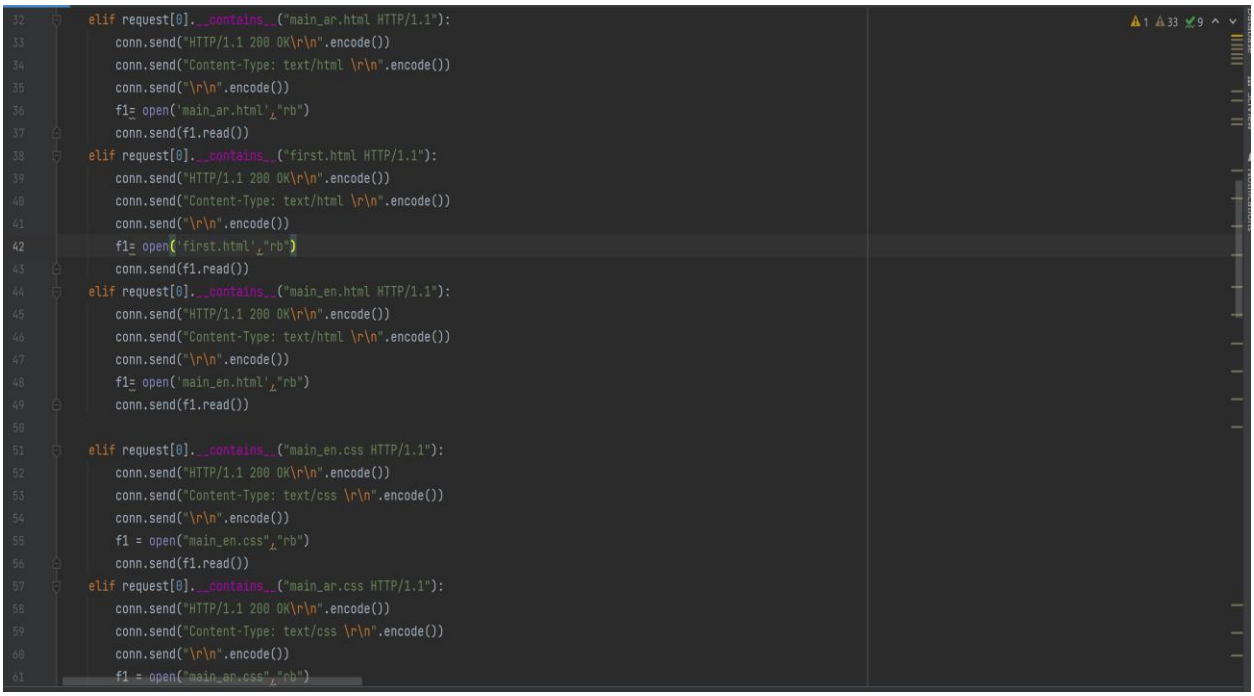
In the two test cases, the server accepted the connection, the first input was decalerd invalid, the second activated the lock screen after 10 seconds .

Part 3: Web server application

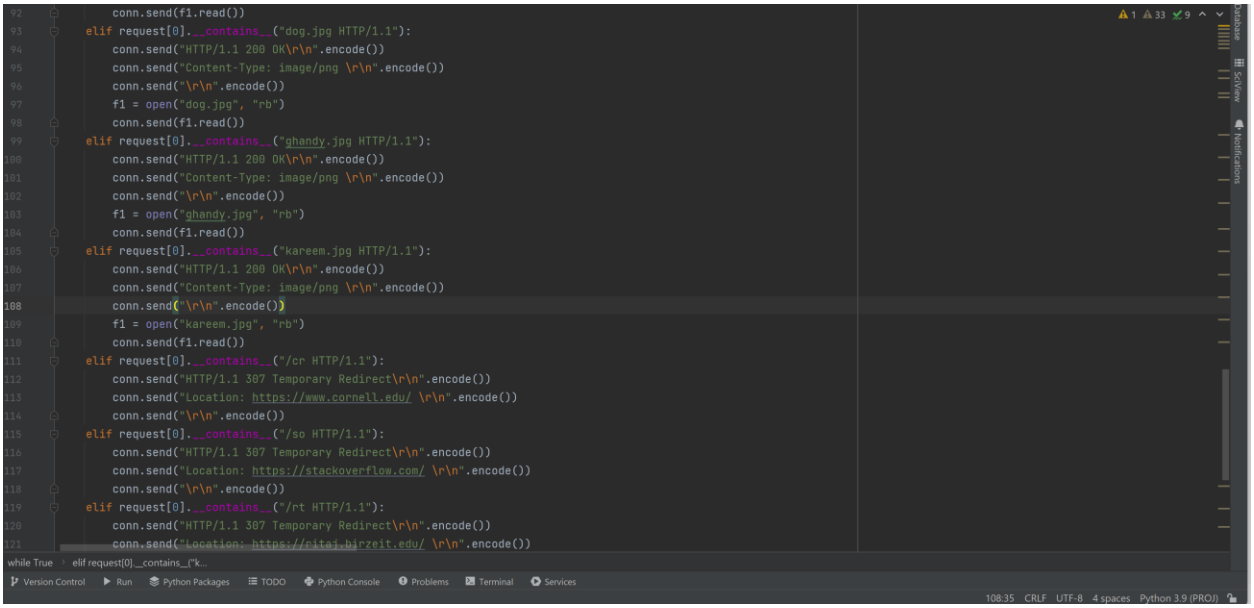
In this part we will build a simple but a complete web server using python listening on port 9966

Server Code:

```
import socket
from socket import *# importing socket programming librariy
serverPort=9966# used port number
serverSocket = socket(AF_INET,SOCK_STREAM)# creating tcp socket
serverSocket.bind(("",serverPort))
serverSocket.listen(1)# server begin listining to TCP request
print("The server is ready to receive")
while True:
    conn, addr = serverSocket.accept()
    sentence = conn.recv(2048).decode()
    print(addr)
    print(sentence)
    #spliting the request and taking the first line
    request = sentence.split("\n")
    #taking ip and address
    ip = addr[0]
    port = addr[1]

    #cases for diffrent requests
    if request[0].__contains__("GET /index.html HTTP/1.1") or request[0].__contains__("GET /main_in.html HTTP/1.1")or (request[0].__contains__("GET /en HTTP/1.1")or request[0].
        conn.send("HTTP/1.1 200 OK\r\n".encode())
        conn.send("Content-Type: text/html \r\n".encode())
        conn.send("\r\n".encode())
        f1 = open('main_en.html', "rb")
        conn.send(f1.read())
    elif request[0].__contains__("/ar HTTP/1.1"):
        conn.send("HTTP/1.1 200 OK\r\n".encode())
        conn.send("Content-Type: text/html \r\n".encode())
        conn.send("\r\n".encode())
        f1 =open('main_ar.html',"rb")
```



```
    elif request[0].__contains__("main_ar.html HTTP/1.1"):
        conn.send("HTTP/1.1 200 OK\r\n".encode())
        conn.send("Content-Type: text/html \r\n".encode())
        conn.send("\r\n".encode())
        f1= open('main_ar.html',"rb")
        conn.send(f1.read())
    elif request[0].__contains__("first.html HTTP/1.1"):
        conn.send("HTTP/1.1 200 OK\r\n".encode())
        conn.send("Content-Type: text/html \r\n".encode())
        conn.send("\r\n".encode())
        f1= open('first.html',"rb")
        conn.send(f1.read())
    elif request[0].__contains__("main_en.html HTTP/1.1"):
        conn.send("HTTP/1.1 200 OK\r\n".encode())
        conn.send("Content-Type: text/html \r\n".encode())
        conn.send("\r\n".encode())
        f1= open('main_en.html',"rb")
        conn.send(f1.read())

    elif request[0].__contains__("main_en.css HTTP/1.1"):
        conn.send("HTTP/1.1 200 OK\r\n".encode())
        conn.send("Content-Type: text/css \r\n".encode())
        conn.send("\r\n".encode())
        f1 = open("main_en.css","rb")
        conn.send(f1.read())
    elif request[0].__contains__("main_ar.css HTTP/1.1"):
        conn.send("HTTP/1.1 200 OK\r\n".encode())
        conn.send("Content-Type: text/css \r\n".encode())
        conn.send("\r\n".encode())
        f1 = open("main_ar.css","rb")
```



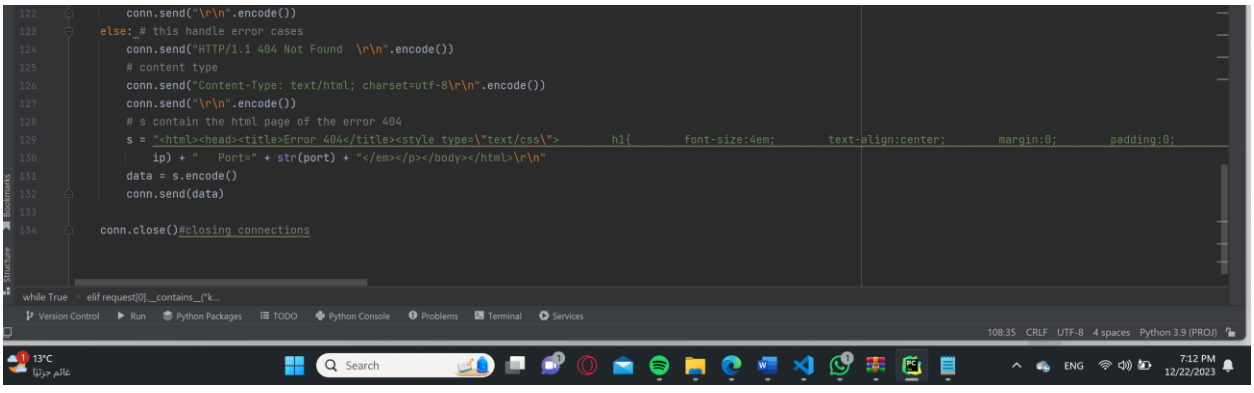
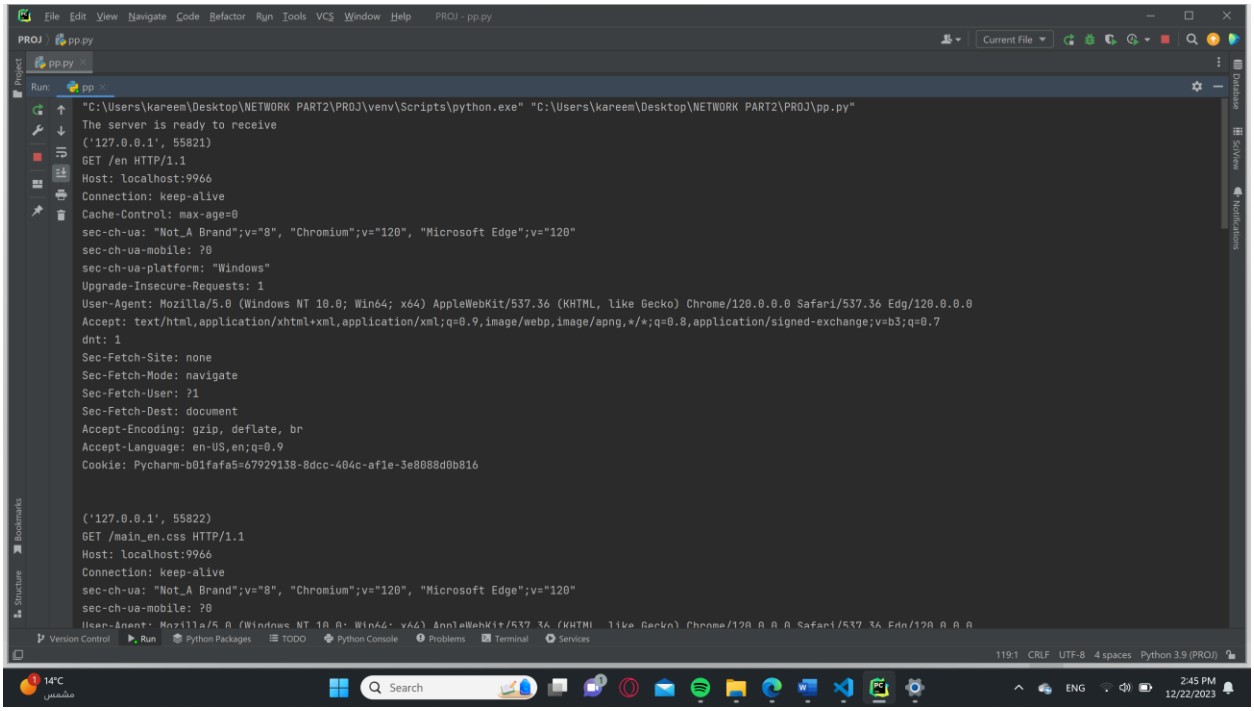


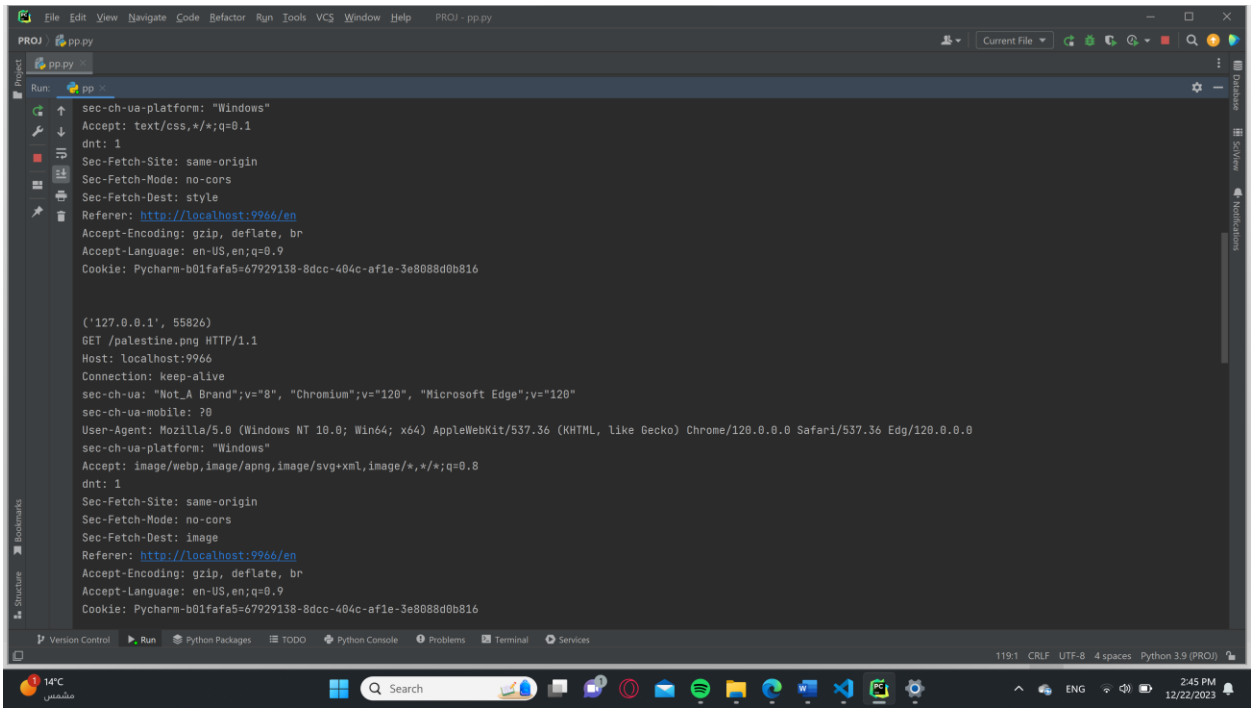
Figure 10:Web server code

Testing on the same computer:

1- if the request is **/ or /index.html or /main_en.html or /en (for example localhost:9966/ or localhost:9966/en)** then the server should send main_en.html file with Content-Type: text/html.

Main_en on command line:





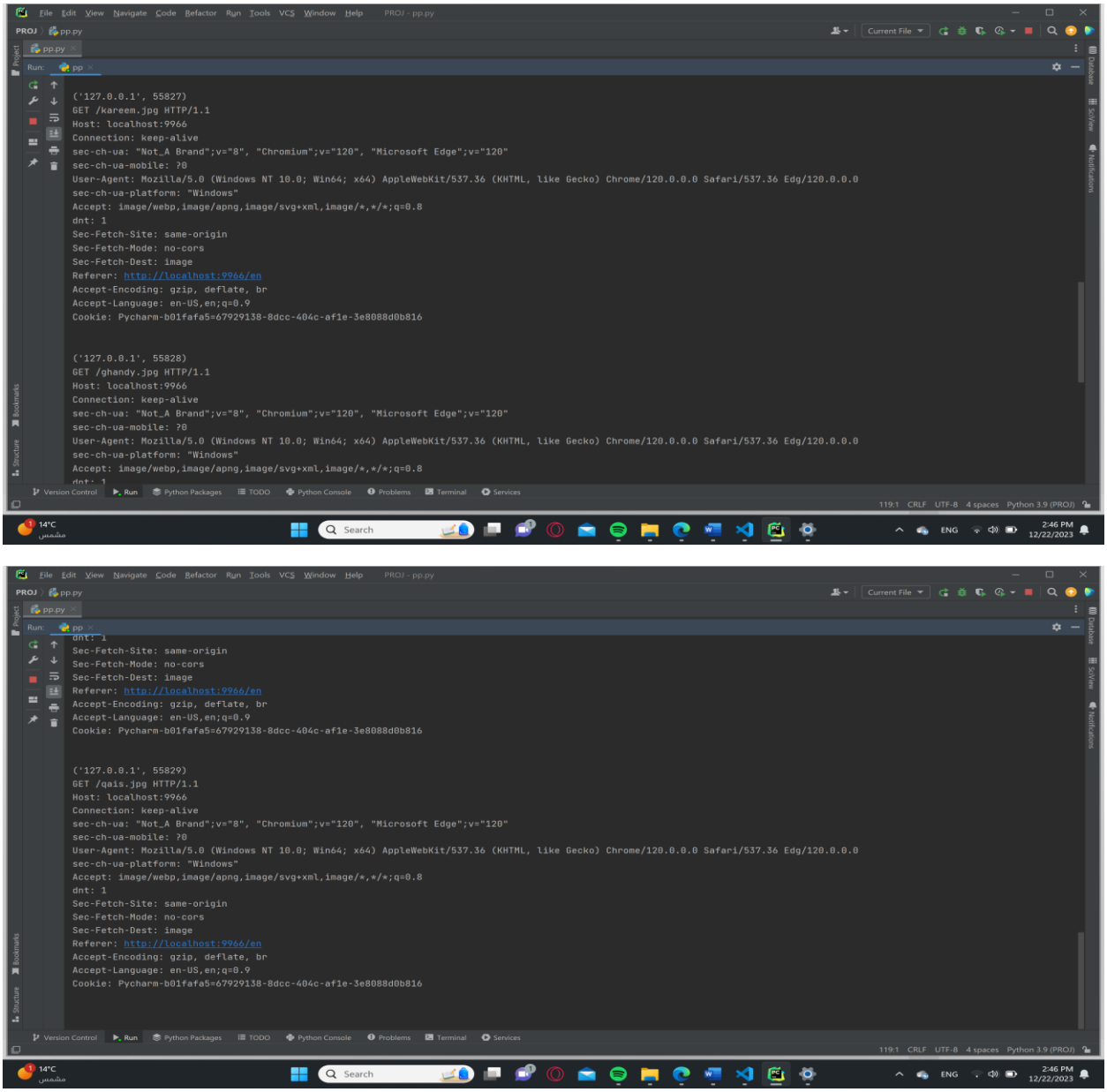


Figure 11: reguest messege for / or /en or /index.html or /main_en.html

note that the html request will request other objects as pngs and jpgs and css file , “keep alive” is shown I request which mean it is a presistent connection and all other obejcts will have on RRT request time.

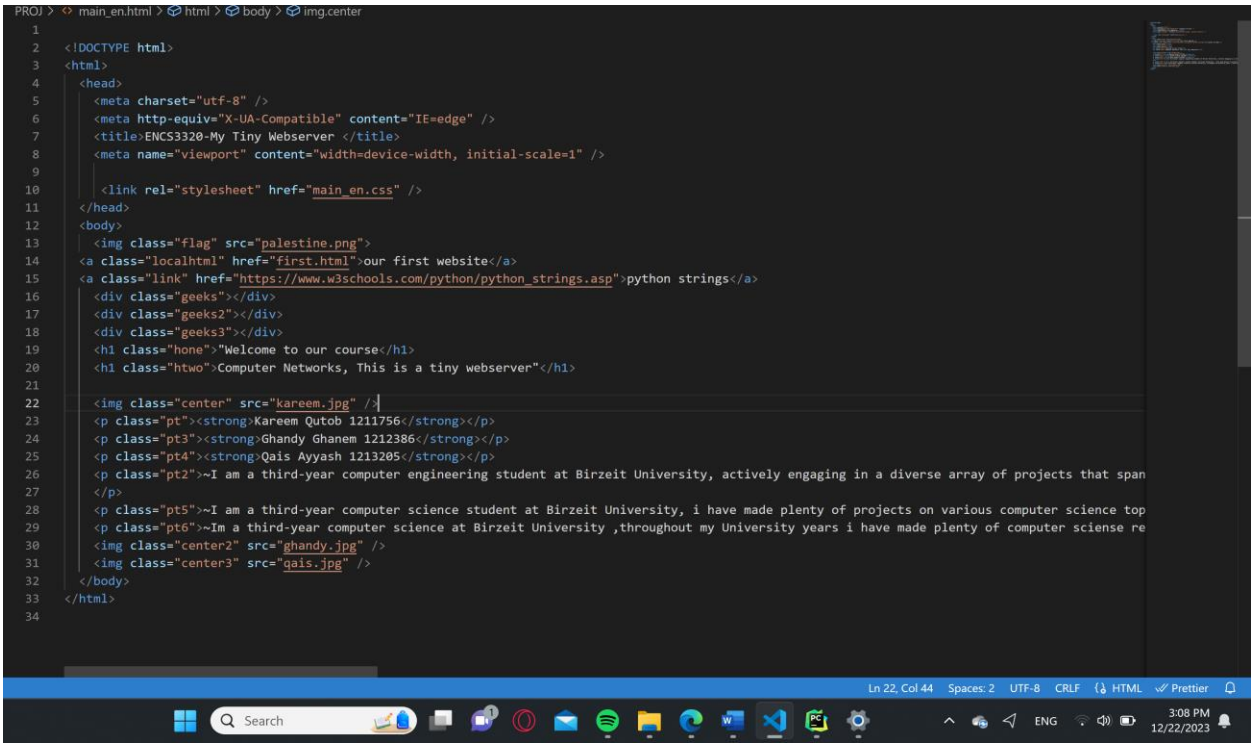
Main_en on browser:

When requesting any main_en.html or any equivalent request like / this html page will appear on the browser

Figure 12: main_en.html on browser

This web page check all the boxes requested for page design using html and css for styling , with brief information about us.

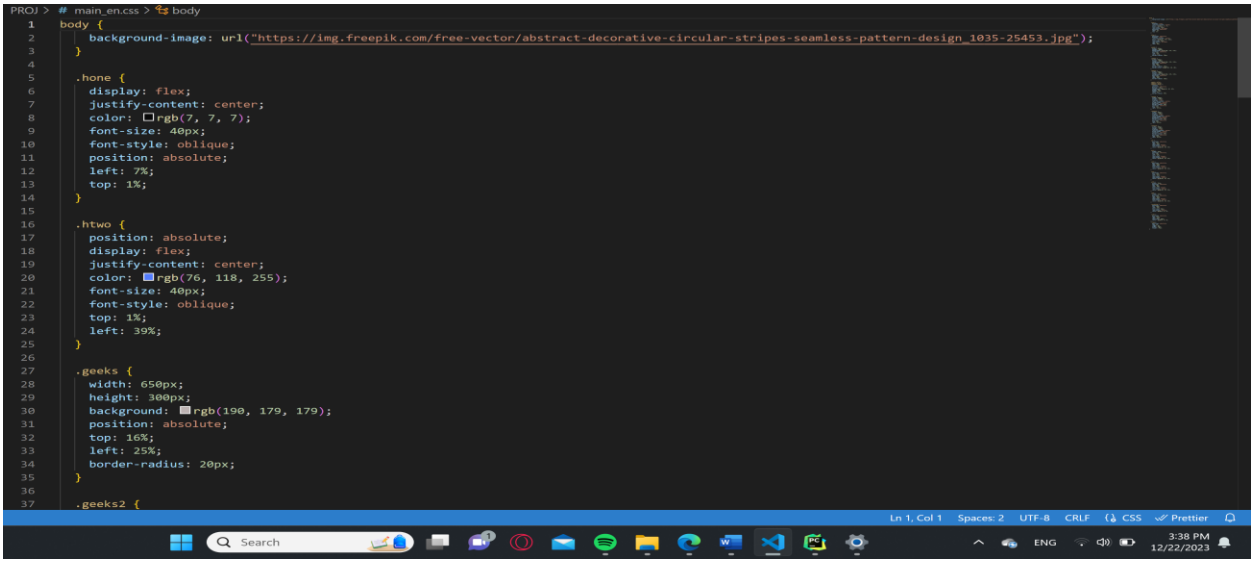
main_en html code:



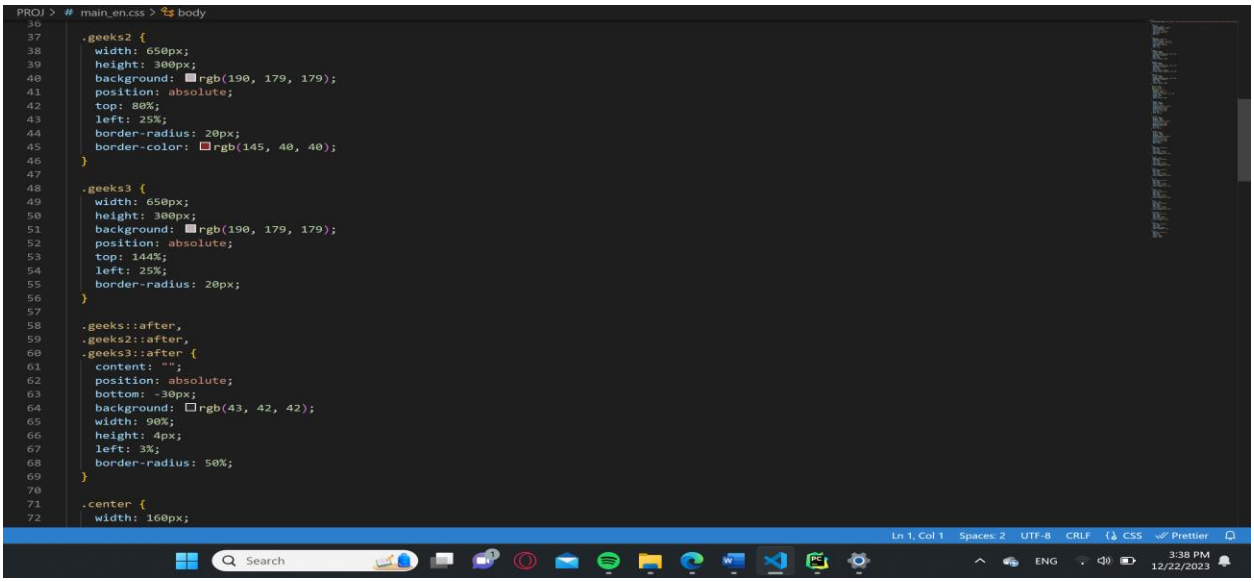
```
PROJ > main_en.html > html > body > img.center
1
2   <!DOCTYPE html>
3   <html>
4     <head>
5       <meta charset="utf-8" />
6       <meta http-equiv="X-UA-Compatible" content="IE=edge" />
7       <title>ENCS3320-My Tiny Webserver </title>
8       <meta name="viewport" content="width=device-width, initial-scale=1" />
9
10      <link rel="stylesheet" href="main_en.css" />
11    </head>
12    <body>
13      <img class="flag" src="palestine.png">
14    <a class="localhtml" href="first.html">our first website</a>
15    <a class="link" href="https://www.w3schools.com/python/python_strings.asp">python strings</a>
16      <div class="geeks"></div>
17      <div class="geeks2"></div>
18      <div class="geeks3"></div>
19      <h1 class="hone">"Welcome to our course</h1>
20      <h1 class="htwo">Computer Networks, This is a tiny webserver"</h1>
21
22      <img class="center" src="kareem.jpg" />
23      <p class="pt"><strong>Kareem Qutob 1211756</strong></p>
24      <p class="pt3"><strong>Ghandy Ghanem 1212386</strong></p>
25      <p class="pt4"><strong>Qais Ayyash 1213205</strong></p>
26      <p class="pt2">~I am a third-year computer engineering student at Birzeit University, actively engaging in a diverse array of projects that span
27      </p>
28      <p class="pt5">~I am a third-year computer science student at Birzeit University, i have made plenty of projects on various computer science top
29      <p class="pt6">~Im a third-year computer science at Birzeit University ,throughout my University years i have made plenty of computer sciense re
30      <img class="center2" src="ghandy.jpg" />
31      <img class="center3" src="qais.jpg" />
32    </body>
33  </html>
34
```

Figure 13:main_en html code

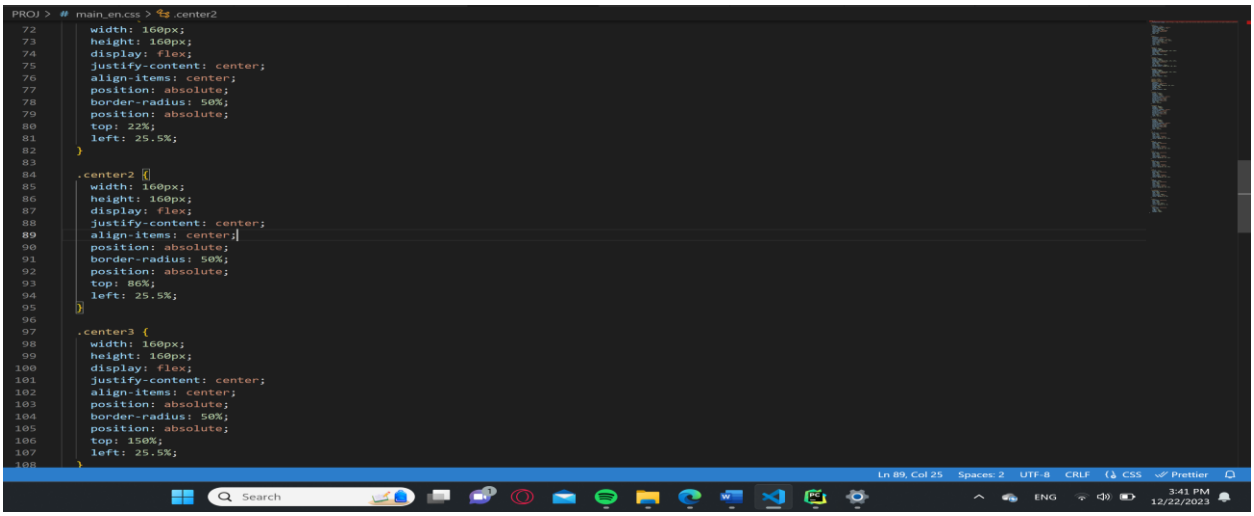
Main_en CSS code:



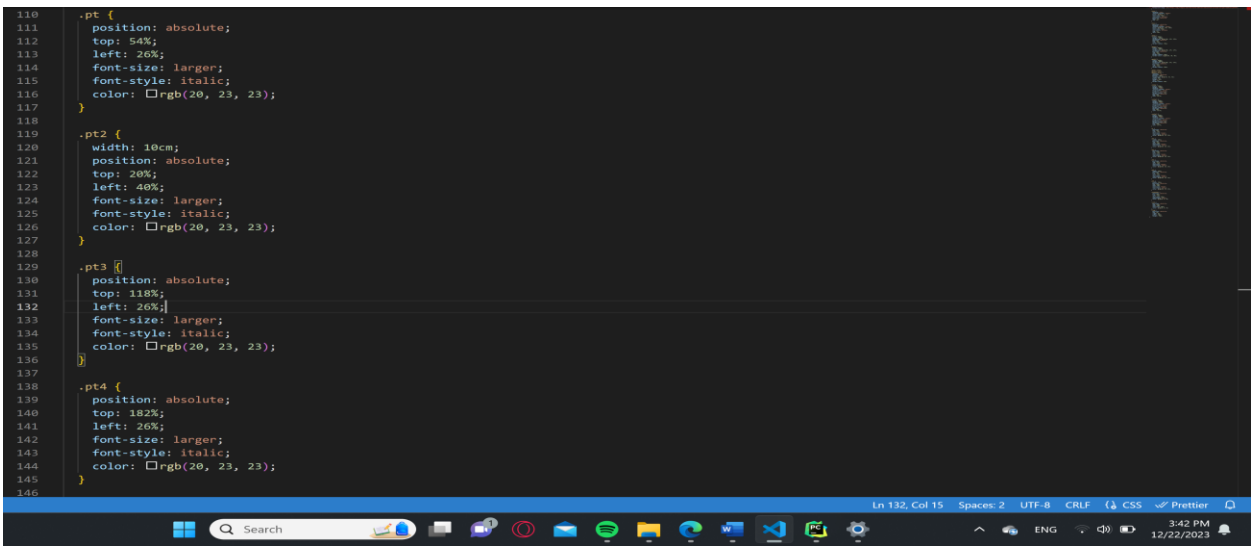
```
PROJ > # main_en.css > body
1   body {
2     background-image: url("https://img.freepik.com/free-vector/abstract-decorative-circular-stripes-seamless-pattern-design_1035-25453.jpg");
3   }
4
5   .hone {
6     display: flex;
7     justify-content: center;
8     color: rgb(7, 7, 7);
9     font-size: 40px;
10    font-style: oblique;
11    position: absolute;
12    left: 7%;
13    top: 1%;
14  }
15
16  .htwo {
17    position: absolute;
18    display: flex;
19    justify-content: center;
20    color: rgb(76, 118, 255);
21    font-size: 40px;
22    font-style: oblique;
23    top: 1%;
24    left: 39%;
25  }
26
27  .geeks {
28    width: 650px;
29    height: 300px;
30    background: rgb(190, 179, 179);
31    position: absolute;
32    top: 16%;
33    left: 25%;
34    border-radius: 20px;
35  }
36
37  .geeks2 {
```



```
PROJ > # main_en.css > body

.geeks2 {
  width: 650px;
  height: 300px;
  background: rgb(190, 179, 179);
  position: absolute;
  top: 80%;
  left: 25%;
  border-radius: 20px;
  border-color: rgb(145, 40, 40);
}

.geeks3 {
  width: 650px;
  height: 300px;
  background: rgb(190, 179, 179);
  position: absolute;
  top: 144%;
  left: 25%;
  border-radius: 20px;
}

.geeks::after,
.geeks2::after,
.geeks3::after {
  content: "";
  position: absolute;
  bottom: -30px;
  background: rgb(43, 42, 42);
  width: 90%;
  height: 4px;
  left: 3%;
  border-radius: 50%;
}

.center {
  width: 160px;
```



```
PROJ > # main_en.css > .center2

  width: 160px;
  height: 160px;
  display: flex;
  justify-content: center;
  align-items: center;
  position: absolute;
  border-radius: 50%;
  position: absolute;
  top: 22%;
  left: 25.5%;
}

.center2 {
  width: 160px;
  height: 160px;
  display: flex;
  justify-content: center;
  align-items: center;
  position: absolute;
  border-radius: 50%;
  position: absolute;
  top: 86%;
  left: 25.5%;
}

.center3 {
  width: 160px;
  height: 160px;
  display: flex;
  justify-content: center;
  align-items: center;
  position: absolute;
  border-radius: 50%;
  position: absolute;
  top: 150%;
  left: 25.5%;
}
```



```
.pt {
  position: absolute;
  top: 54%;
  left: 26%;
  font-size: larger;
  font-style: italic;
  color: rgb(20, 23, 23);
}

.pt2 {
  width: 10cm;
  position: absolute;
  top: 20%;
  left: 40%;
  font-size: larger;
  font-style: italic;
  color: rgb(20, 23, 23);
}

.pt3 {
  position: absolute;
  top: 118%;
  left: 26%;
  font-size: larger;
  font-style: italic;
  color: rgb(20, 23, 23);
}

.pt4 {
  position: absolute;
  top: 182%;
  left: 26%;
  font-size: larger;
  font-style: italic;
  color: rgb(20, 23, 23);
}
```

Figure 14: main_en css code

Clicking “our first website” link:

As asked in the main_en design there are a link that will redirect to local html page of our choice

Figure 15: local html file link

When clicked thid page will appear:

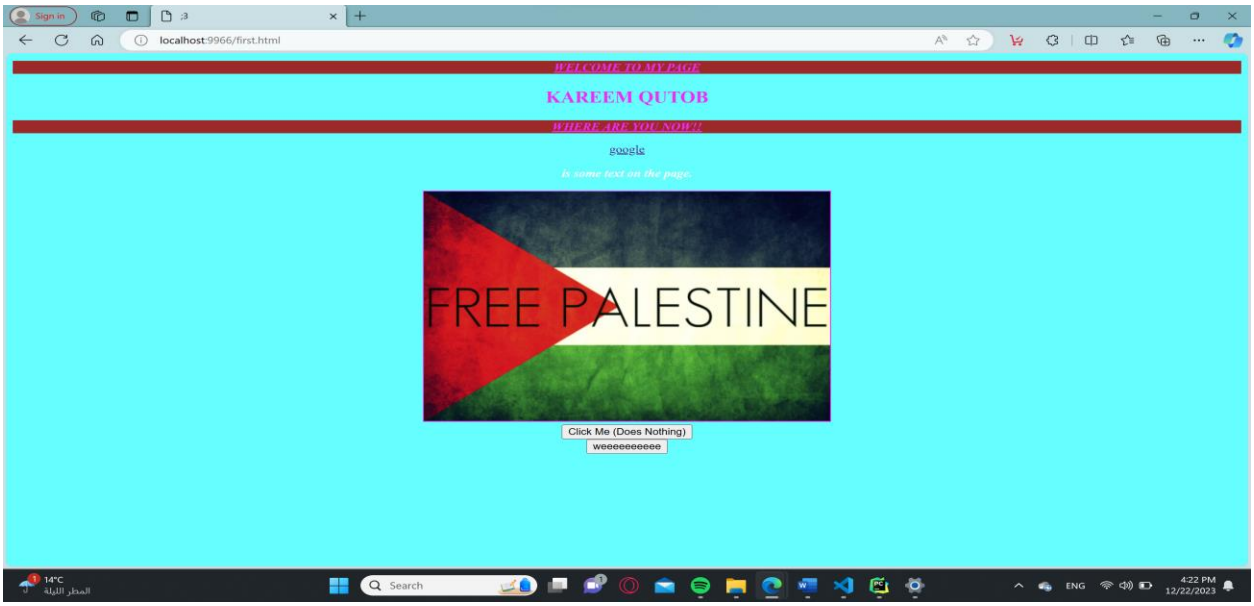
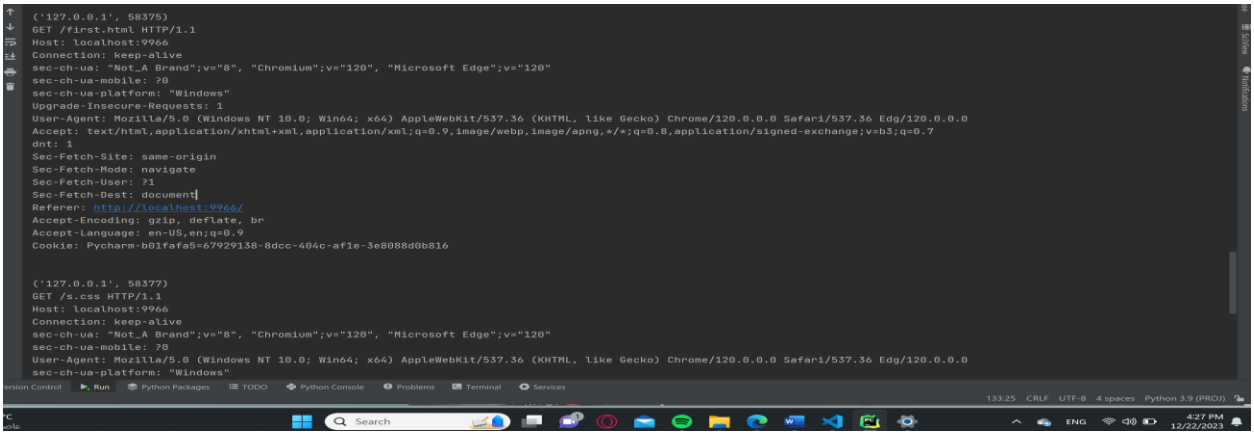


Figure 16:local html page

This our first attempt of HTML/CSS coding so it was only an attempt of trying different objects and styles

Local Html page request on command line:



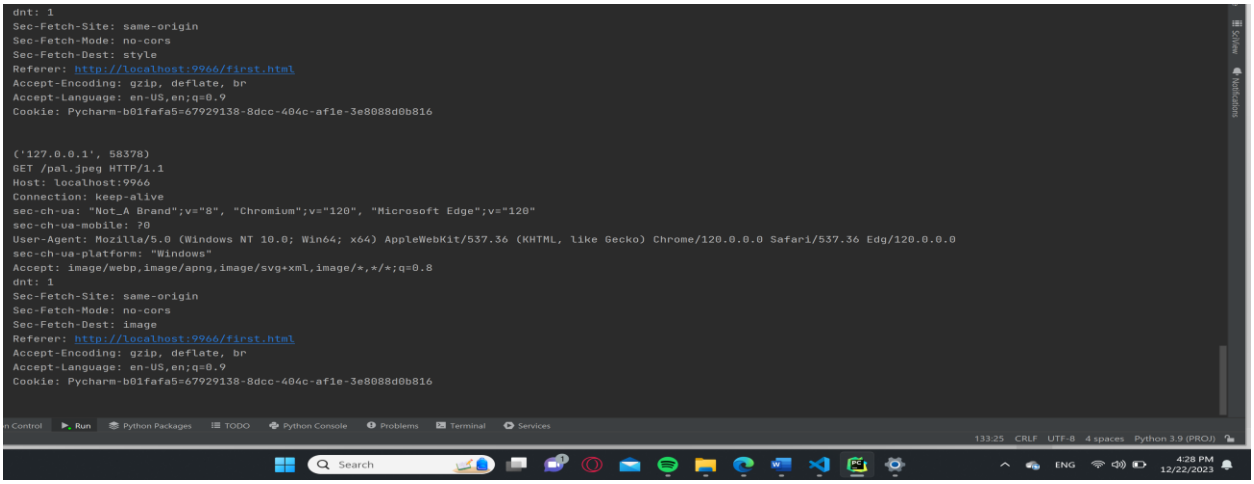


Figure 17:local html file request

the request of html file will include the jpg and css code attached to the website

Local html file code:

Figure 18: local html file code

Local html file css code:

```
PROJ > # s.css > img
1  img{
2
3      border: solid;
4      cursor: pointer;
5      box-shadow: inset;
6  }
7  p{
8      color: rgb(255, 255, 255);
9  }
10 body{
11     color: magenta;
12     text-align: center;
13     background-color: aqua;
14 }
15 .n{
16     background-color: brown;
17     font-style: italic;
18     text-decoration: underline;
19     text-transform: uppercase;
20     font-weight: bold;
21     font-size: medium;
22 }
```

Figure 19: local html file css code

Clicking python string link:

As asked in the main_en design there are a link that will redirect to w3chools page about python strings

Figure 20:python strings link

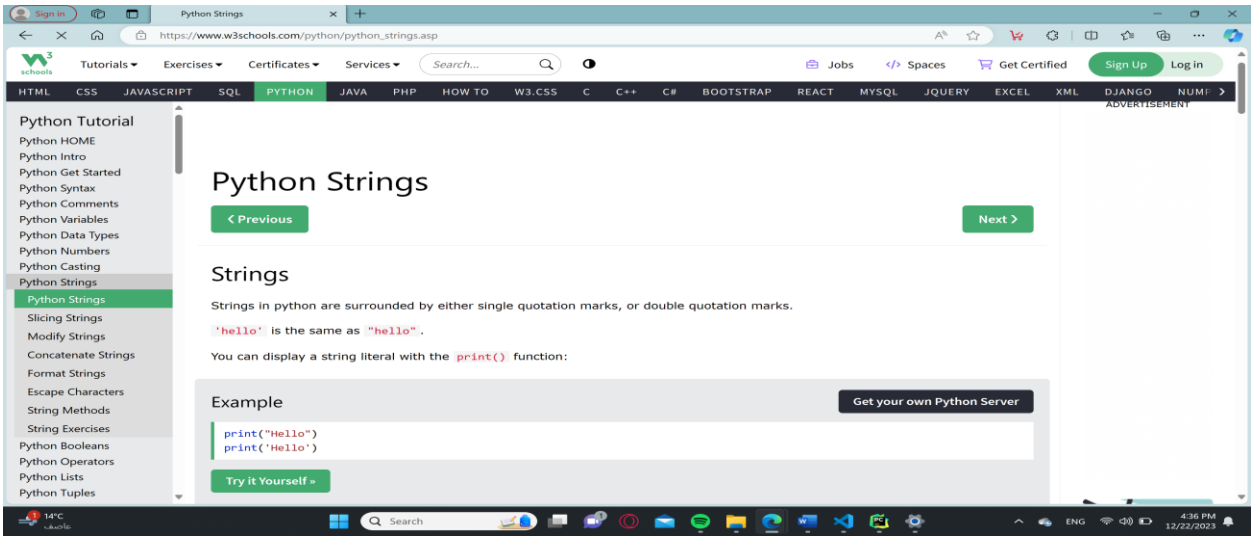
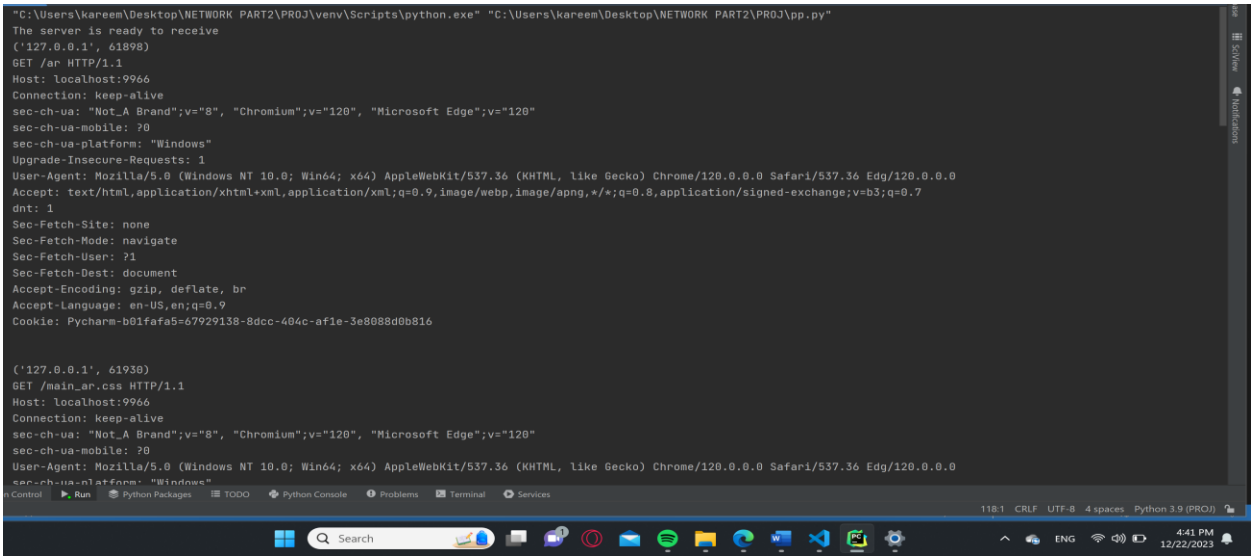


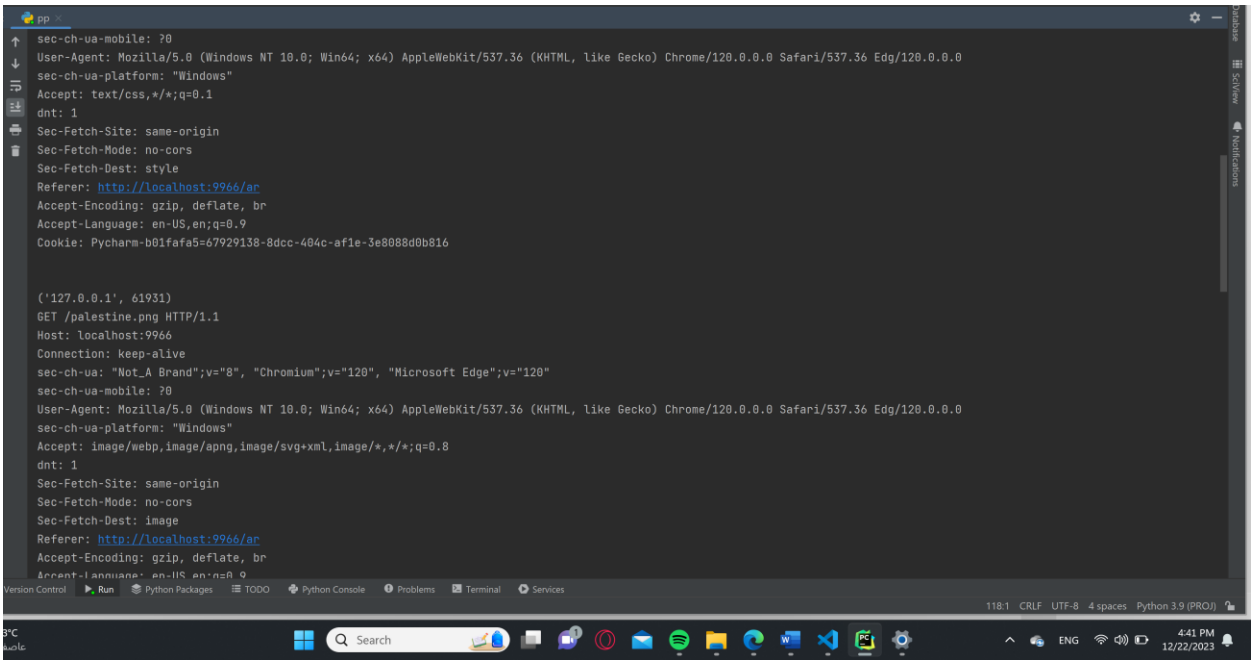
Figure 21: w3school python strings tutorial

Main_ar request:

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

Main_ar request on command line:





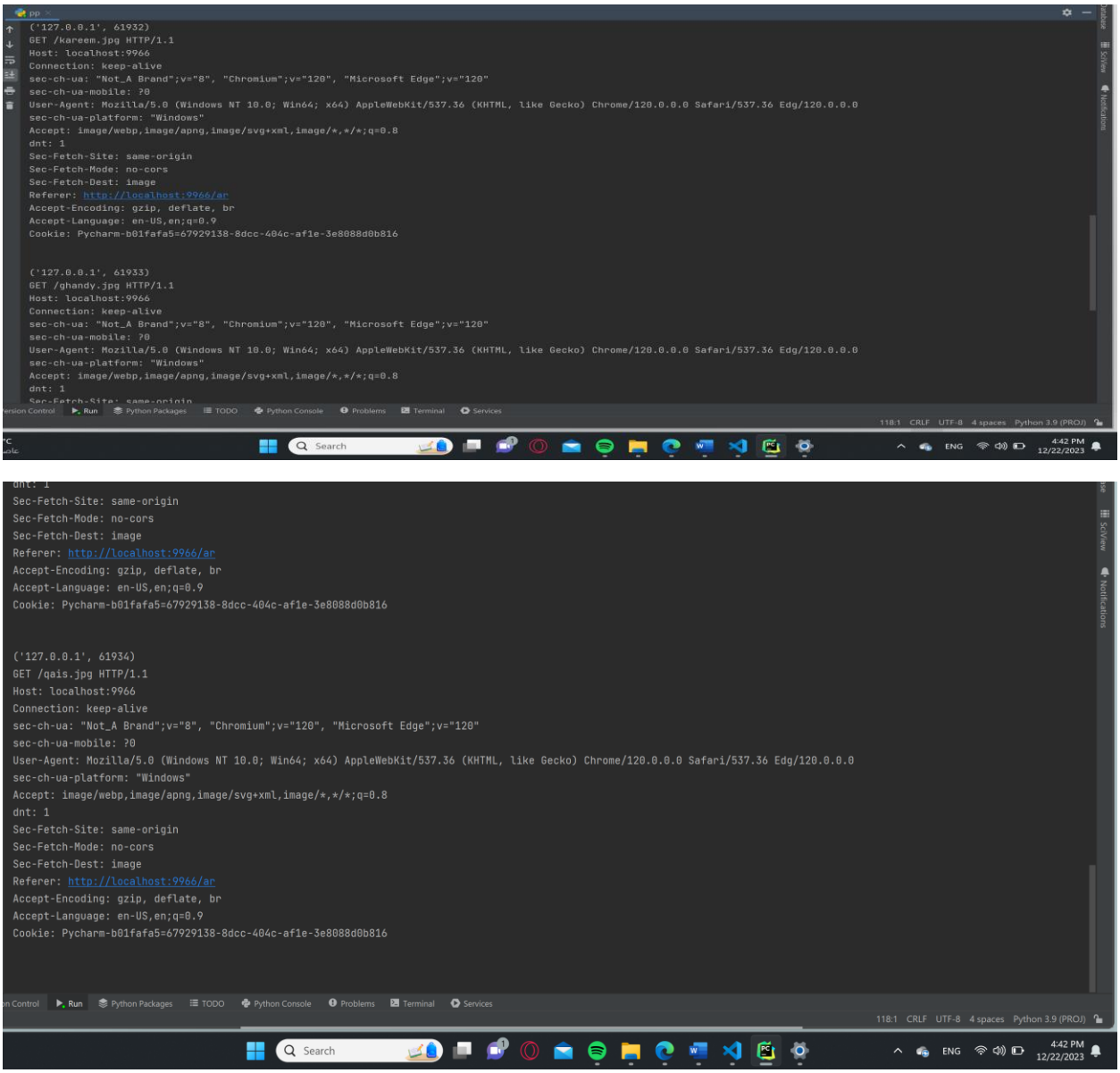


Figure 22:/ar request message

Similar to main_en request, a presistent tcp connection that request an html file with objects like pictures and css code

Main_ar on browser:

Figure 23:main_ar on browser

An Arabic version of main_en website

Main_ar html code:

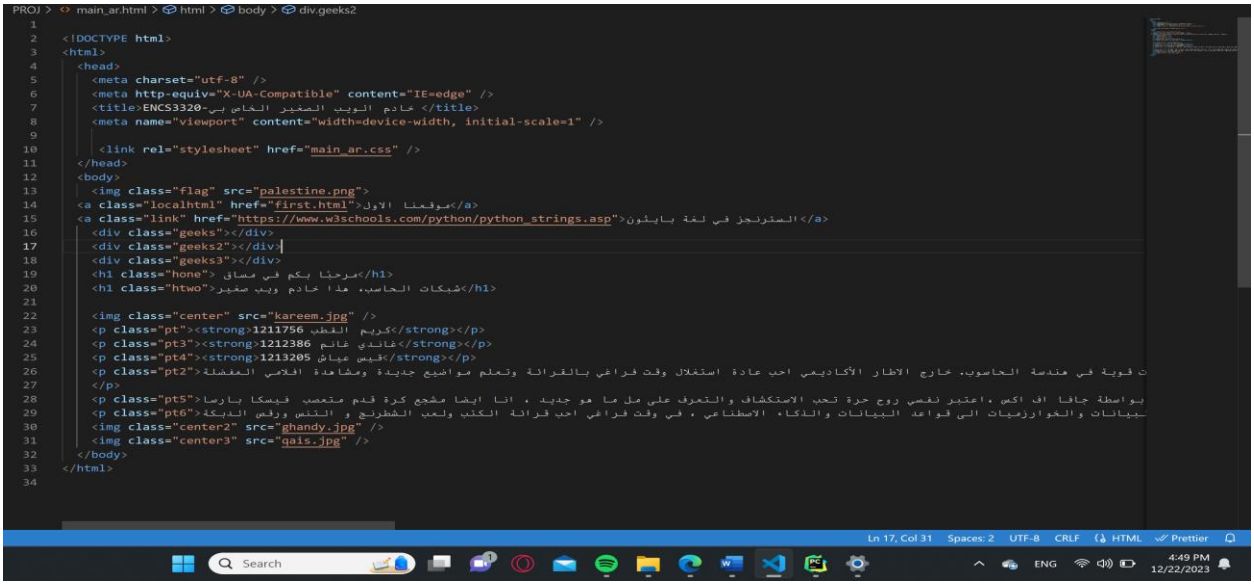
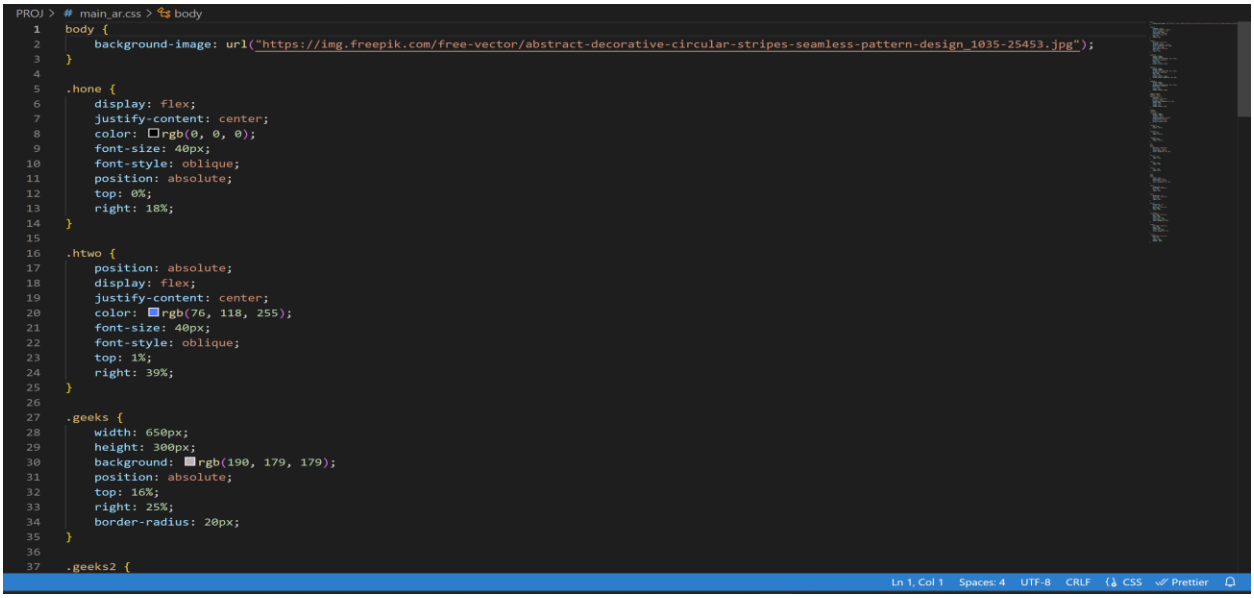
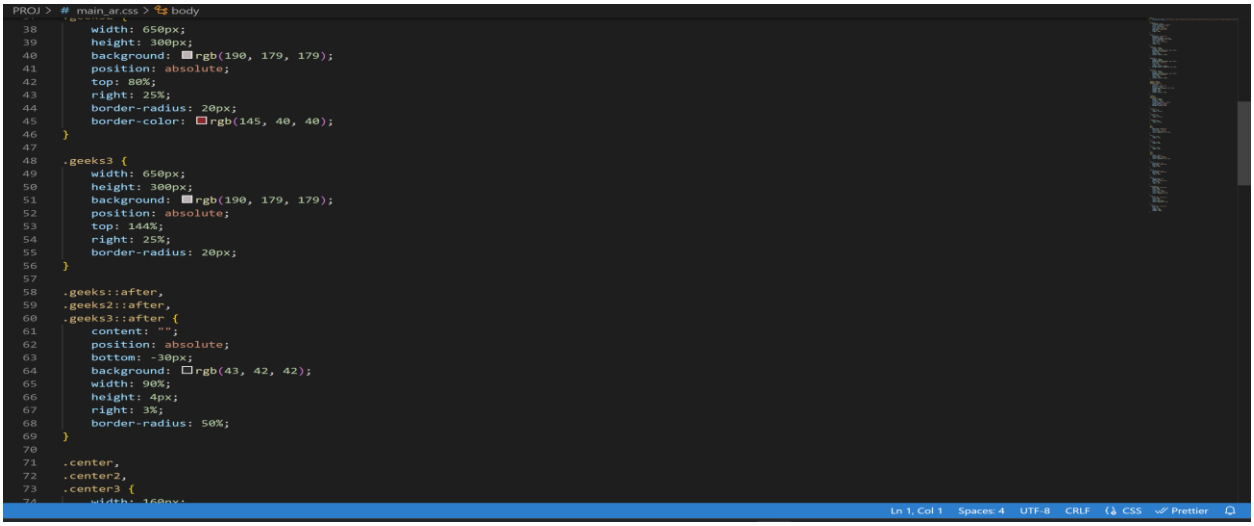


Figure 24:main_ar html code

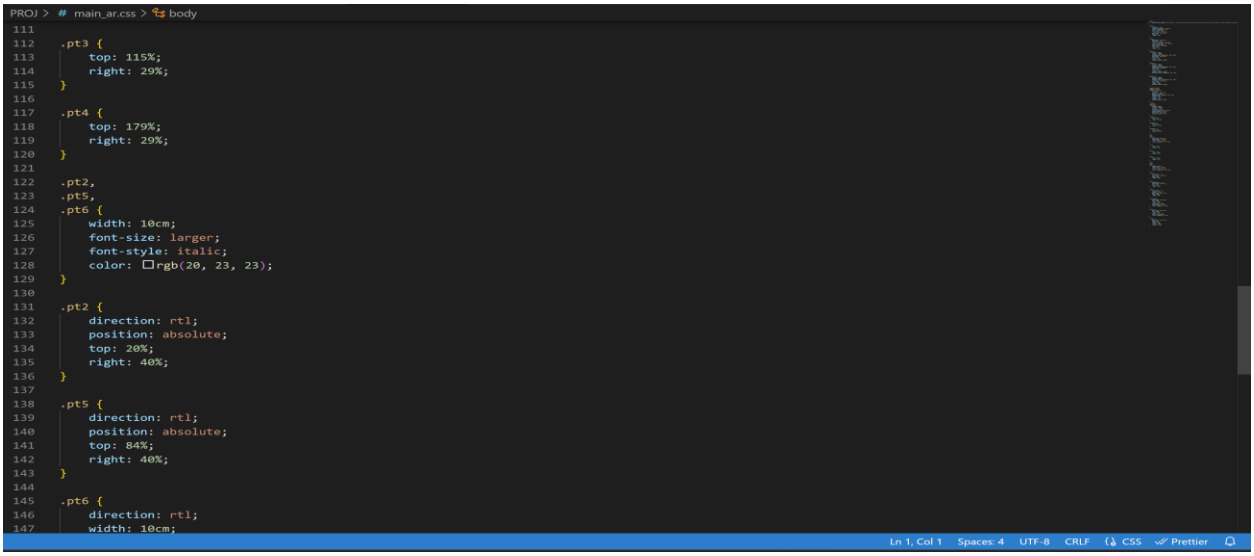
Main_ar css code:





```
PROJ > # main_ar.css > body
    width: 650px;
    height: 300px;
    background: rgb(190, 179, 179);
    position: absolute;
    top: 80%;
    right: 25%;
    border-radius: 20px;
    border-color: rgb(145, 40, 40);
}

.geeks3 {
    width: 650px;
    height: 300px;
    background: rgb(190, 179, 179);
    position: absolute;
    top: 144%;
    right: 25%;
    border-radius: 20px;
}

.geeks::after,
.geeks2::after,
.geeks3::after {
    content: "";
    position: absolute;
    bottom: -30px;
    background: rgb(43, 42, 42);
    width: 90%;
    height: 4px;
    right: 3%;
    border-radius: 50%;
}

.center,
.center2,
.center3 {
    width: 160px;
```

```
PROJ > # main_ar.css > body
    width: 160px;
    height: 160px;
    display: flex;
    justify-content: center;
    align-items: center;
    position: absolute;
    border-radius: 50%;
}

.center {
    top: 22%;
    right: 25.5%;
}

.center2 {
    top: 86%;
    right: 25.5%;
}

.center3 {
    top: 150%;
    right: 25.5%;
}

.pt,
.pt3,
.pt4 {
    position: absolute;
    font-size: larger;
    font-style: italic;
    color: rgb(20, 23, 23);
}

.pt {
    top: 51%;
    right: 29%;
}
```



```
PROJ > # main_ar.css > body

.pt3 {
    top: 115%;
    right: 29%;
}

.pt4 {
    top: 179%;
    right: 29%;
}

.pt2,
.pt5,
.pt6 {
    width: 10cm;
    font-size: larger;
    font-style: italic;
    color: rgb(20, 23, 23);
}

.pt2 {
    direction: rtl;
    position: absolute;
    top: 20%;
    right: 40%;
}

.pt5 {
    direction: rtl;
    position: absolute;
    top: 84%;
    right: 40%;
}

.pt6 {
    direction: rtl;
    width: 10cm;
```

```css
.pt6 {
    direction: rtl;
    width: 10cm;
    position: absolute;
    top: 148%;
    right: 40%;
}

.localhtml {
    position: absolute;
    top: 208%;
    right: 40%;
    font-size: 40px;
    font-style: italic;
    color: rgb(0, 0, 0);
}

.link {
    position: absolute;
    top: 220%;
    right: 40%;
    font-size: 40px;
    color: rgb(0, 0, 0);
}

.flag {
    position: absolute;
    top: 2%;
    right: 4%;
    width: 70px;
    height: 70px;
}
```

Figure 25:main_ar css code

.html request:

3- if the request is an .**html** file then the server should send the requested html file with Content-Type: text/html. You can use any html file.

.html request on command line:

```
"C:\Users\kareem\Desktop\NETWORK PART2\PROJ\venv\Scripts\python.exe" "C:\Users\kareem\Desktop\NETWORK PART2\PROJ\pp.py"
The server is ready to receive
('127.0.0.1', 65159)
GET /first.html HTTP/1.1
Host: localhost:9966
Connection: keep-alive
sec-ch-ua: "Not_A Brand";v="8", "Chromium";v="120", "Microsoft Edge";v="120"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Windows"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.0.0 Safari/537.36 Edg/120.0.0.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
dnt: 1
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Cookie: Pycharm-b01fafa5=67929138-8dcc-404c-af1e-3e8088d0b816
```

Figure 26:.html request

For convenience I reused an html file from previous parts

.Html request on browser

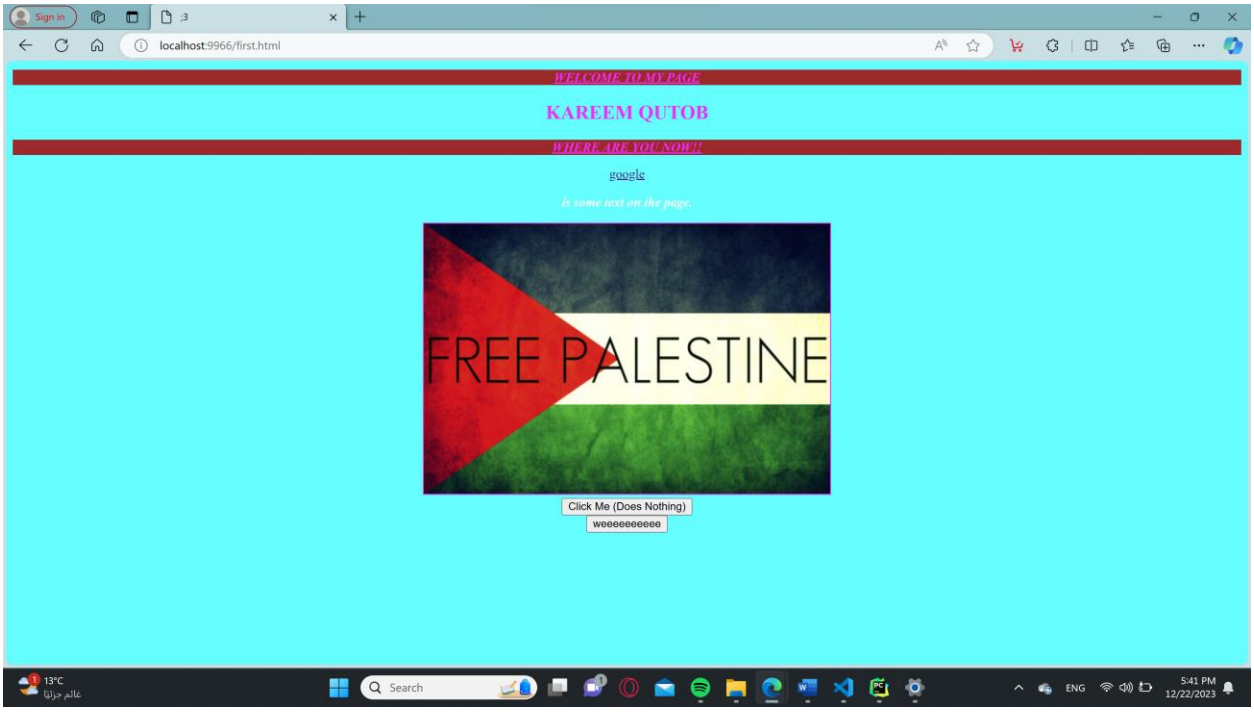


Figure 27: /.html request on browser

Our code will return the html file for any request with \filename.html request

.CSS request:

if the request is a **.css** file then the server should send the requested css file with Content-Type: text/css. You can use any CSS file

.css request on command line:

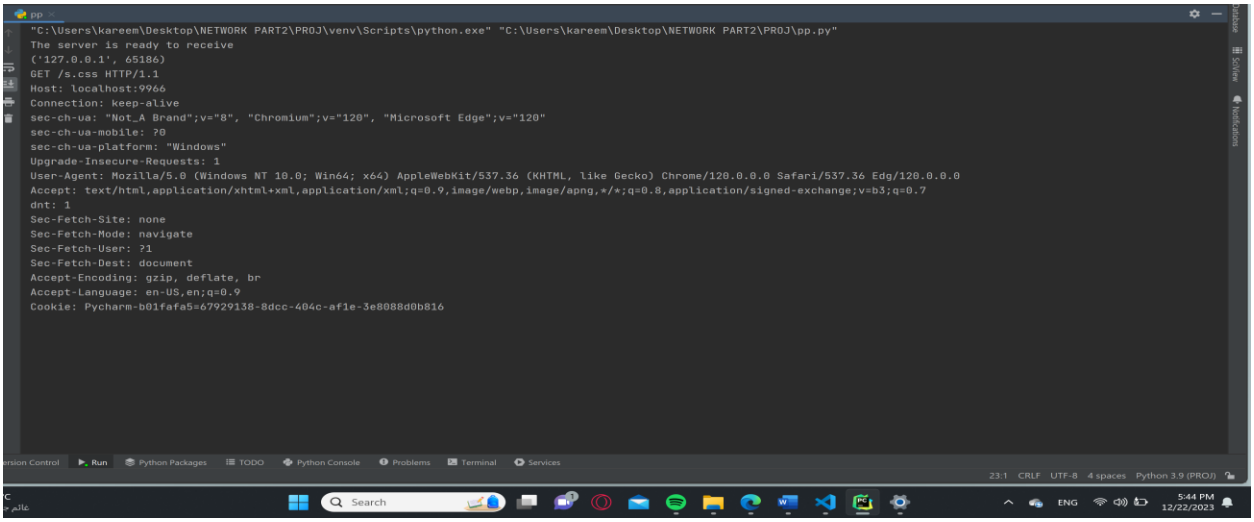


Figure 28: .css request message

.css request on browser:

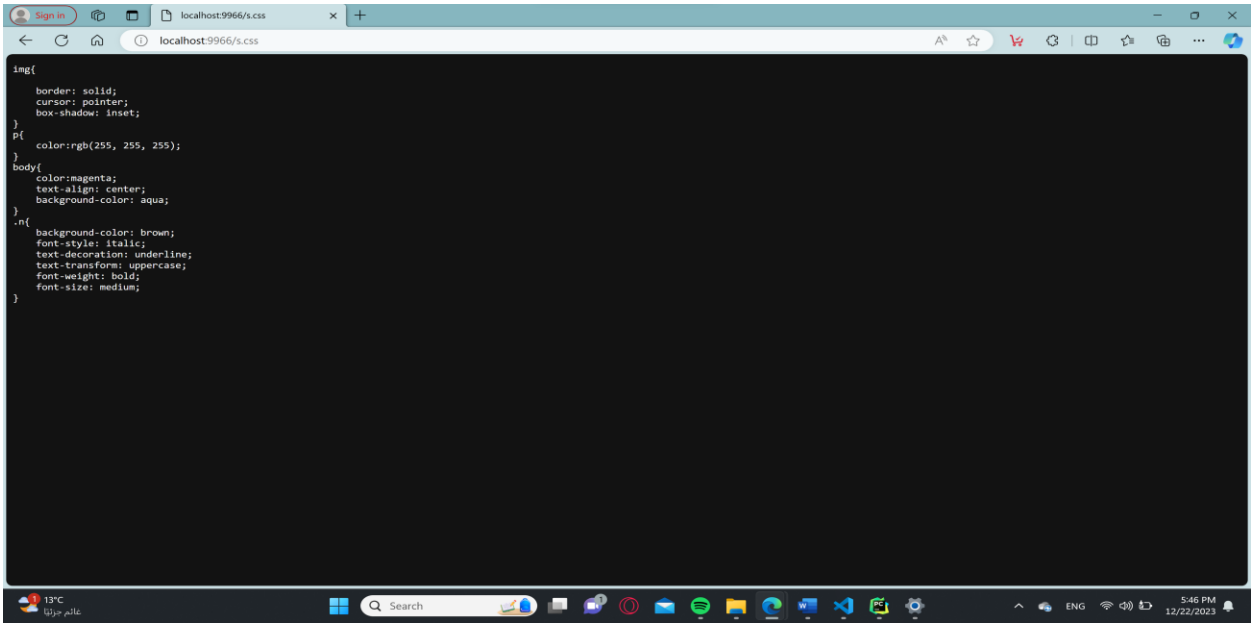


Figure 29:.css request on browser

PNG request:

if the request is a .png then the server should send the png image with Content-Type: image/png. You can use any image.

PNG request on command line:

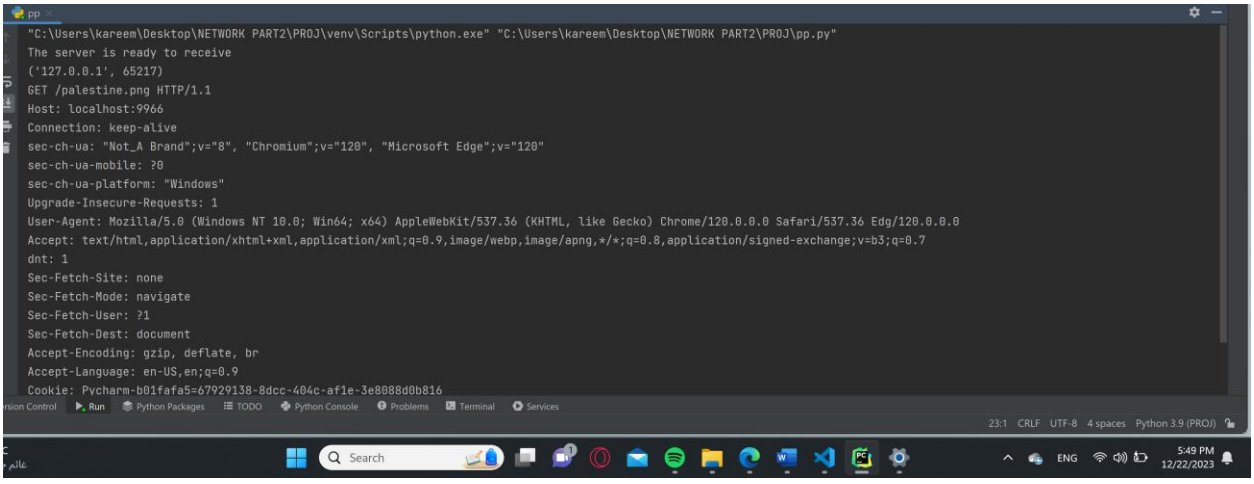


Figure 30: png request message

.PNG request on browser:

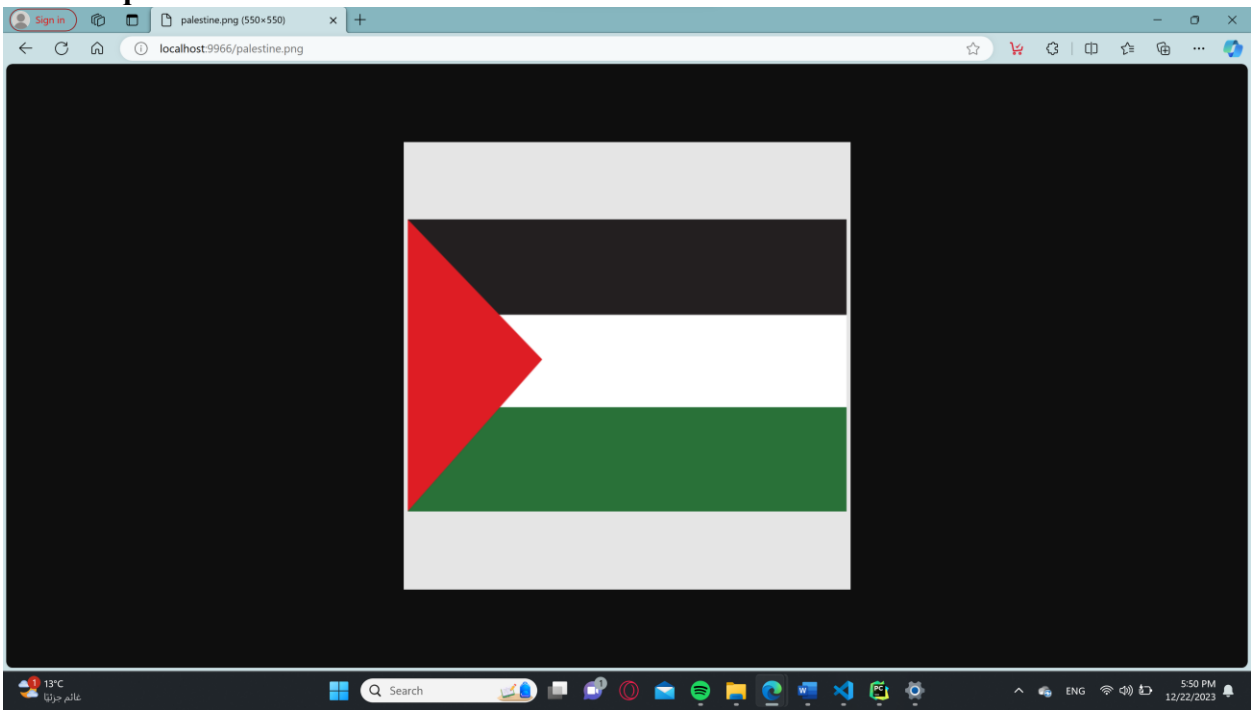


Figure 31:>PNG on browser

.JPG request:

if the request is a .jpg then the server should send the jpg image with Content-Type: image/jpeg. You can use any image.

.JPG request on command line:

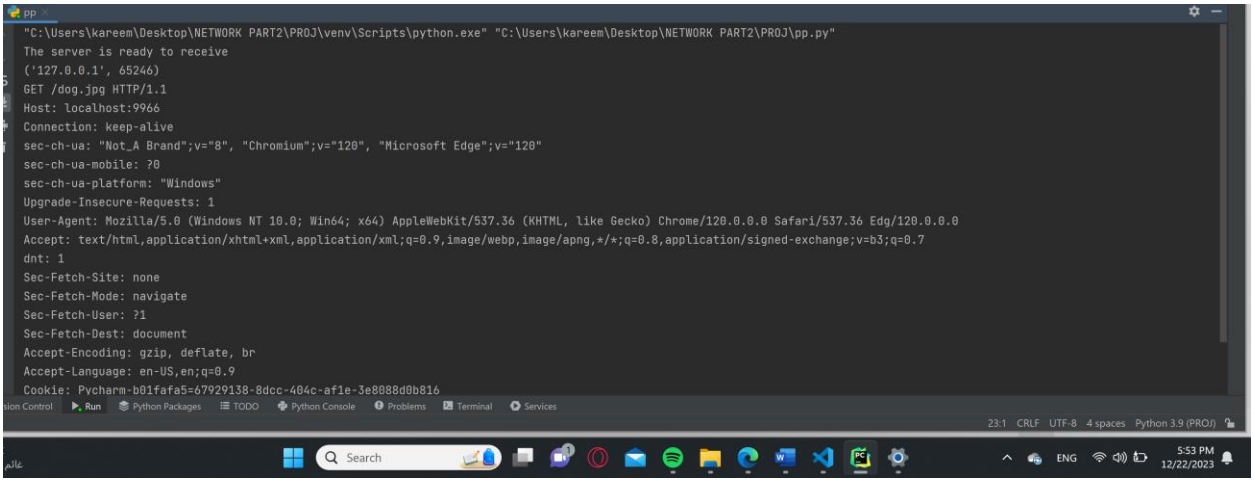


Figure 32:JPG request message

.JPG on browser:

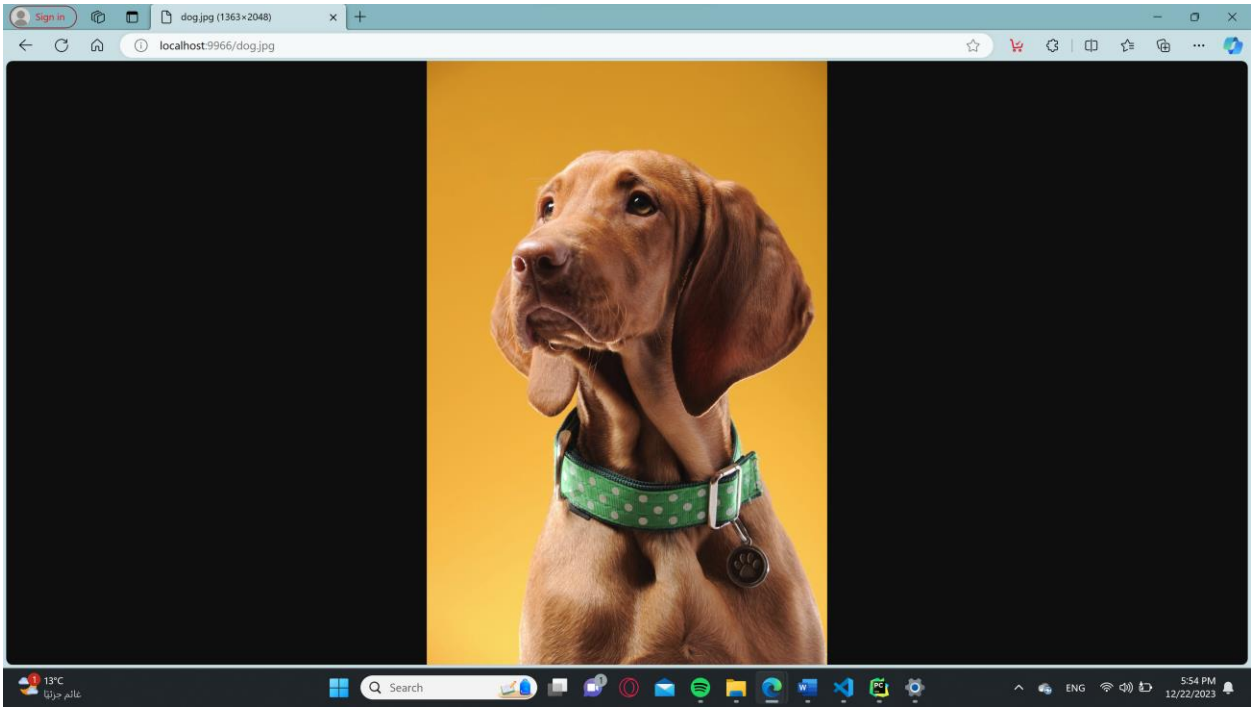


Figure 33:JPG on browser

Page redirecting:

Use the status code 307 Temporary Redirect to redirect the following

a. If the request is /cr then redirect to cornell.edu website
b. If the request is /so then redirect to stackoverflow.com website
c. If the request is /rt then redirect to ritaj website

/cr on command line:

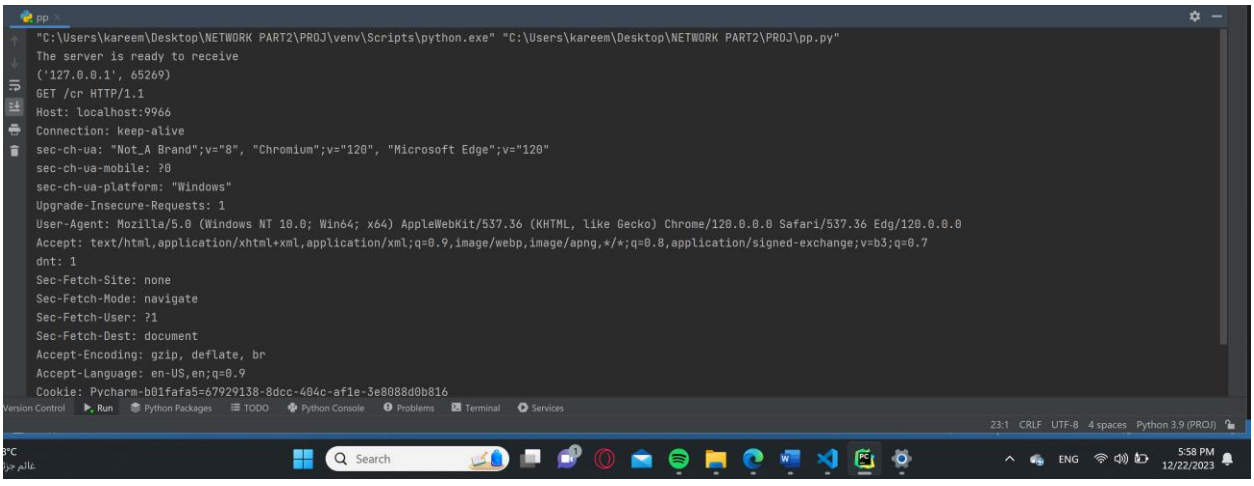


Figure 34:/cr on command line

/cr on browser:

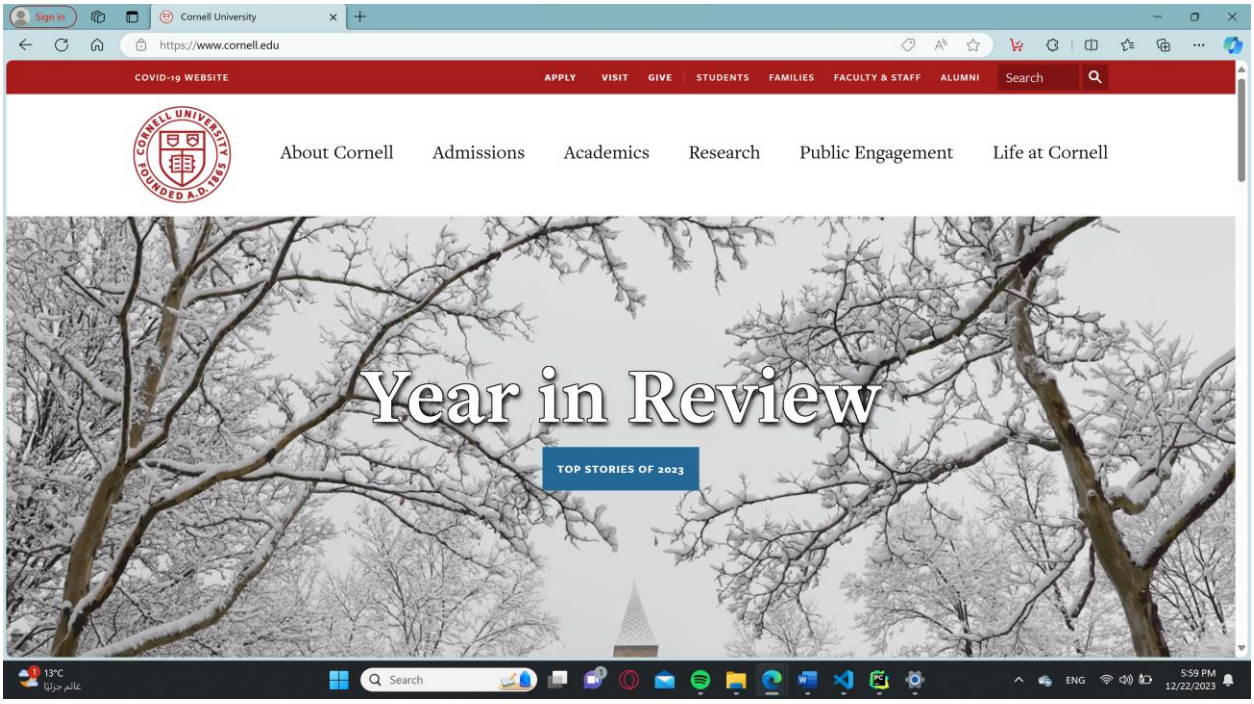


Figure 35:/cr on browser

It will redirect to www.cornell.edu

/rt on command line:

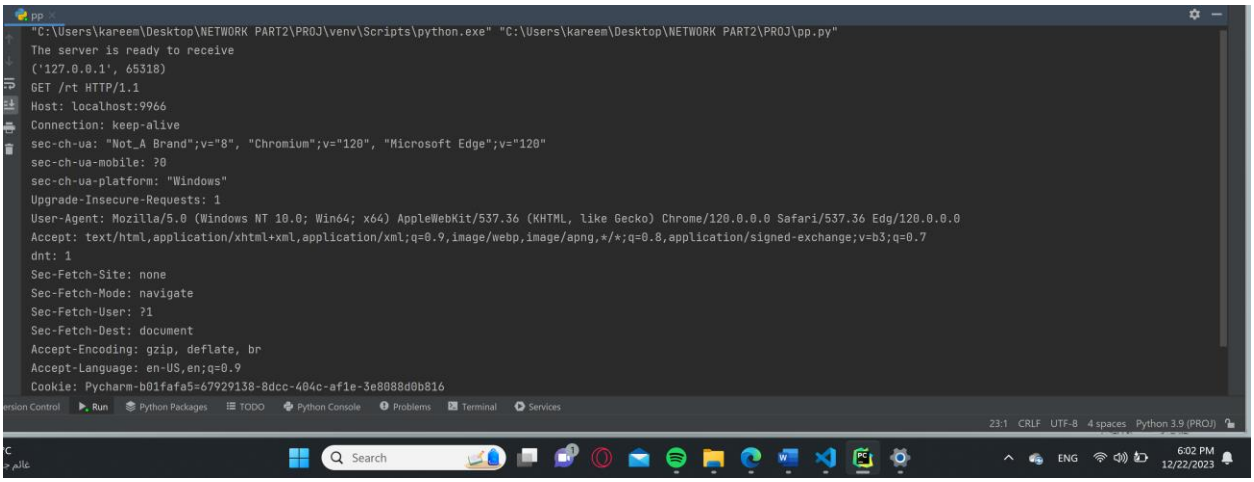


Figure 36: /rt request message

/rt on browser :

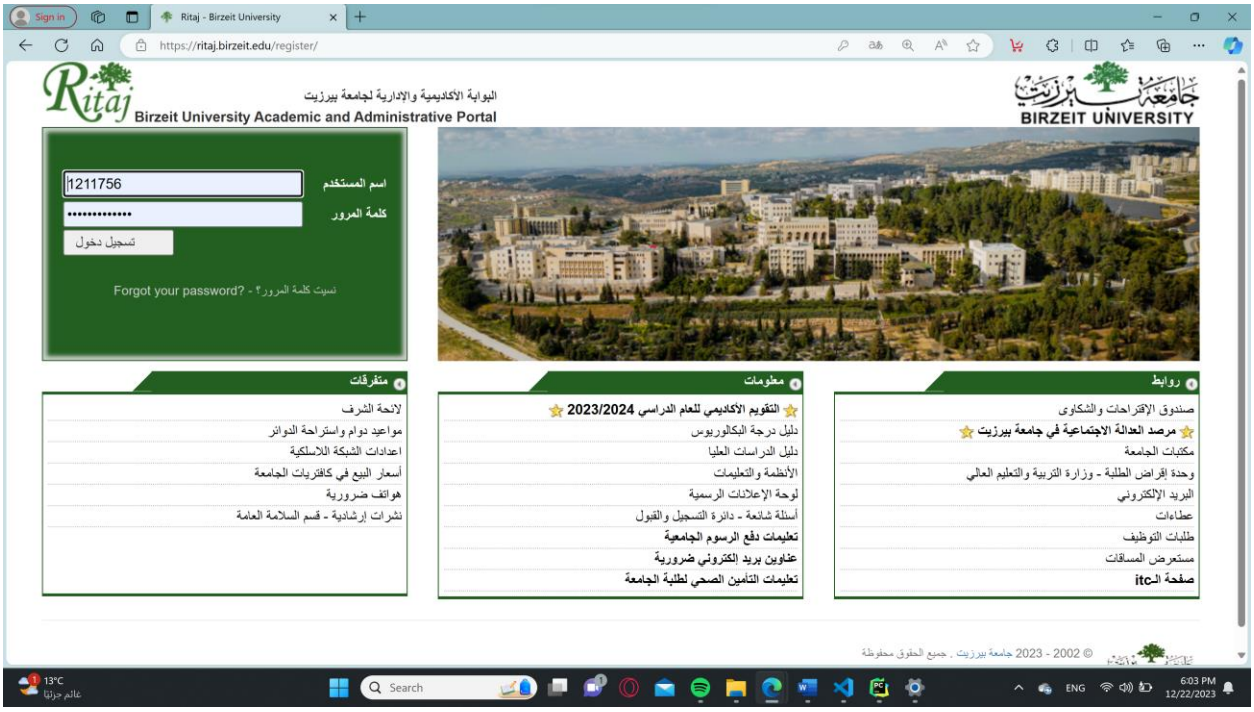


Figure 37:/rt on browser

/so on command line:

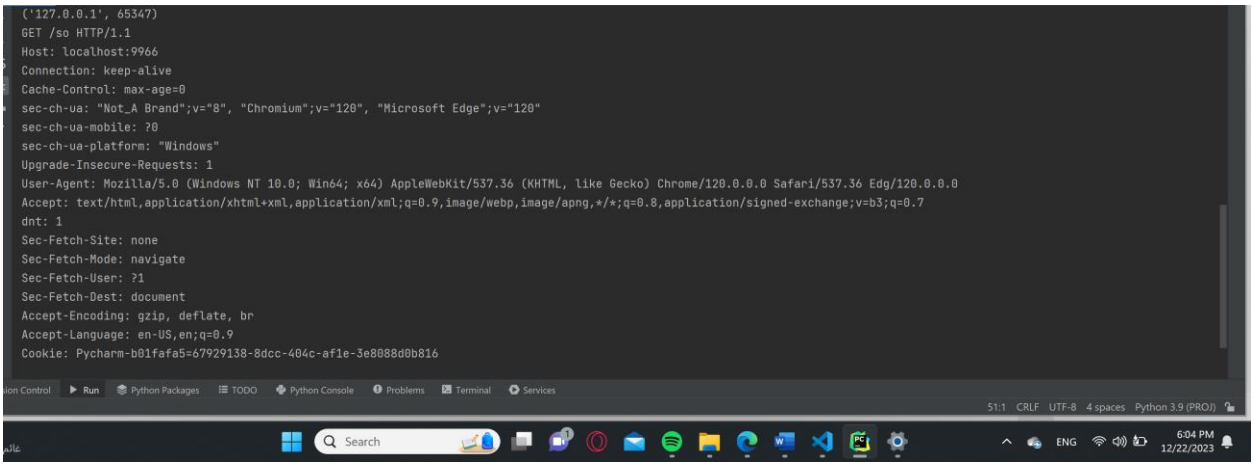


Figure 38:/so message request

/so on browser:

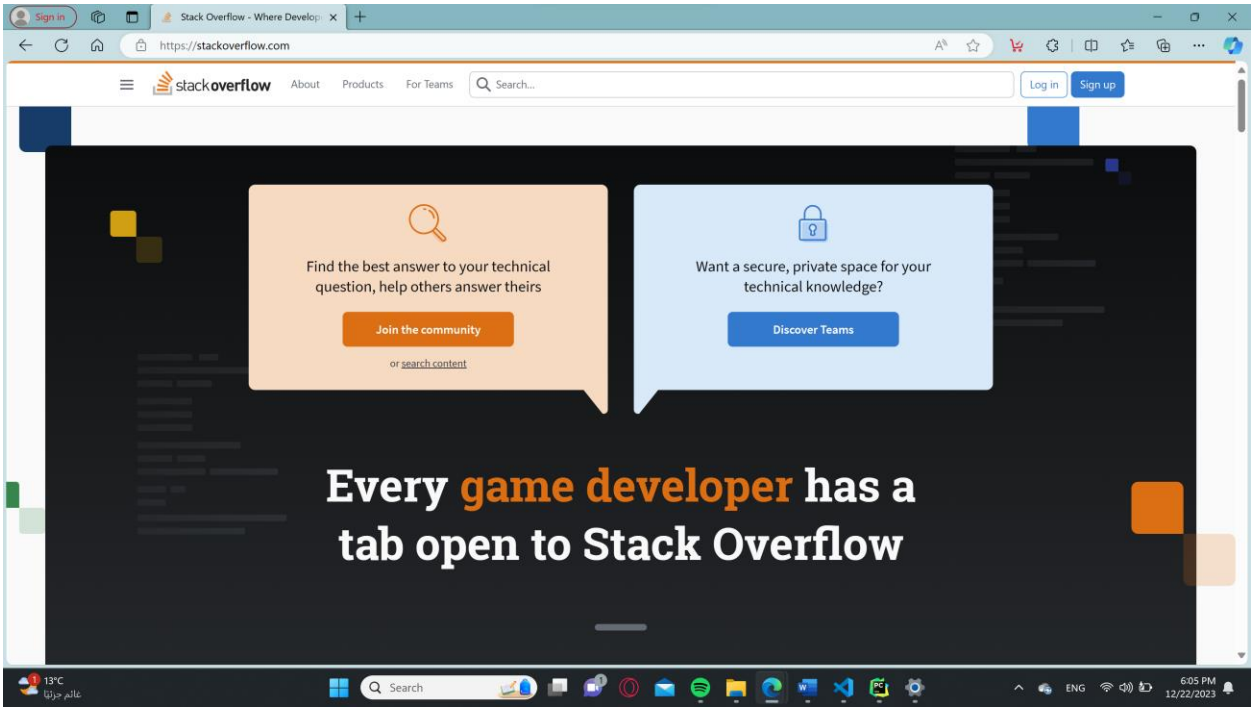


Figure 39:/so on browser

ERROR webpage:

If the request is wrong or the file doesn't exist the server should return a simple HTML webpage that contains (Content-Type: text/html)

1- "HTTP/1.1 404 Not Found" in the response status
2- "Error 404" in the title
3- "The file is not found" in the body in **red**
4- Your names and IDs in **Bold**
5- The IP and port number of the client

Error on command line:

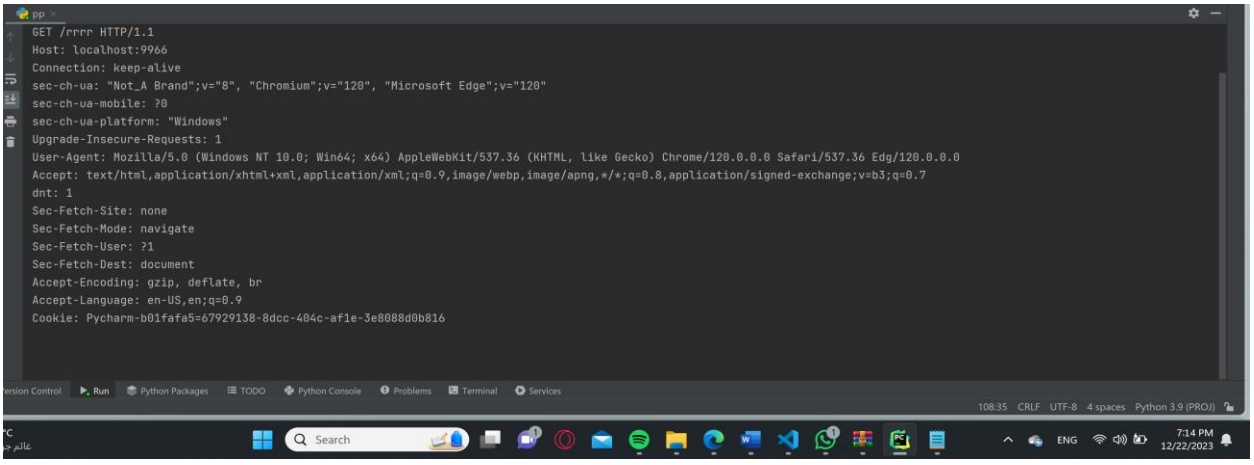


Figure 40: error on command line

Error page on browser:

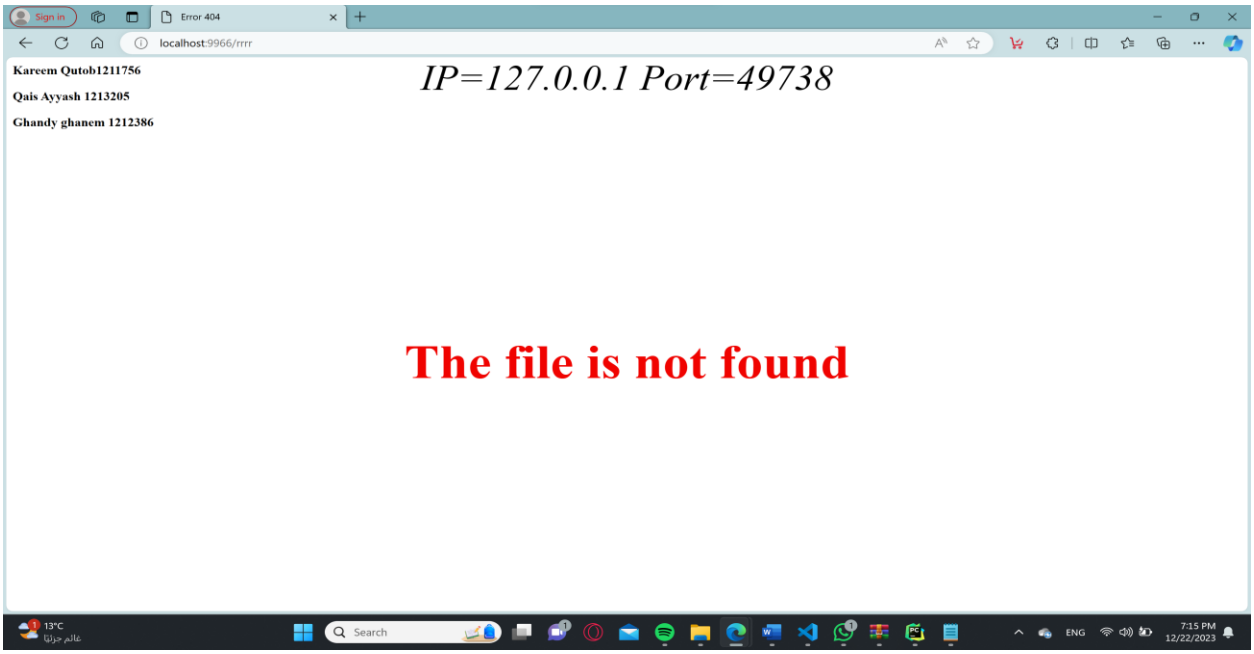


Figure 41:error on browser

In this part the html code was part of python code so we can add the id and port number for every client

Error code segment:

```
else: # this handle error cases
    conn.send("HTTP/1.1 404 Not Found  \r\n".encode())
    # content type
    conn.send("Content-Type: text/html; charset=utf-8\r\n".encode())
    conn.send("\r\n".encode())
    # s contain the html page of the error 404
    s = "<html><head><title>Error 404</title><style type=\"text/css\">       h1{       font-size:4em;       text-align:center;       margin:0;       padding:0;
       ip) + "   Port=" + str(port) + "</em></p></body></html>\r\n"
    data = s.encode()
    conn.send(data)
```

Figure 42:error code

The html file test will be stored in variable sentence so we can open different pages based on ip and port number

Test from another device:

From another laptop:

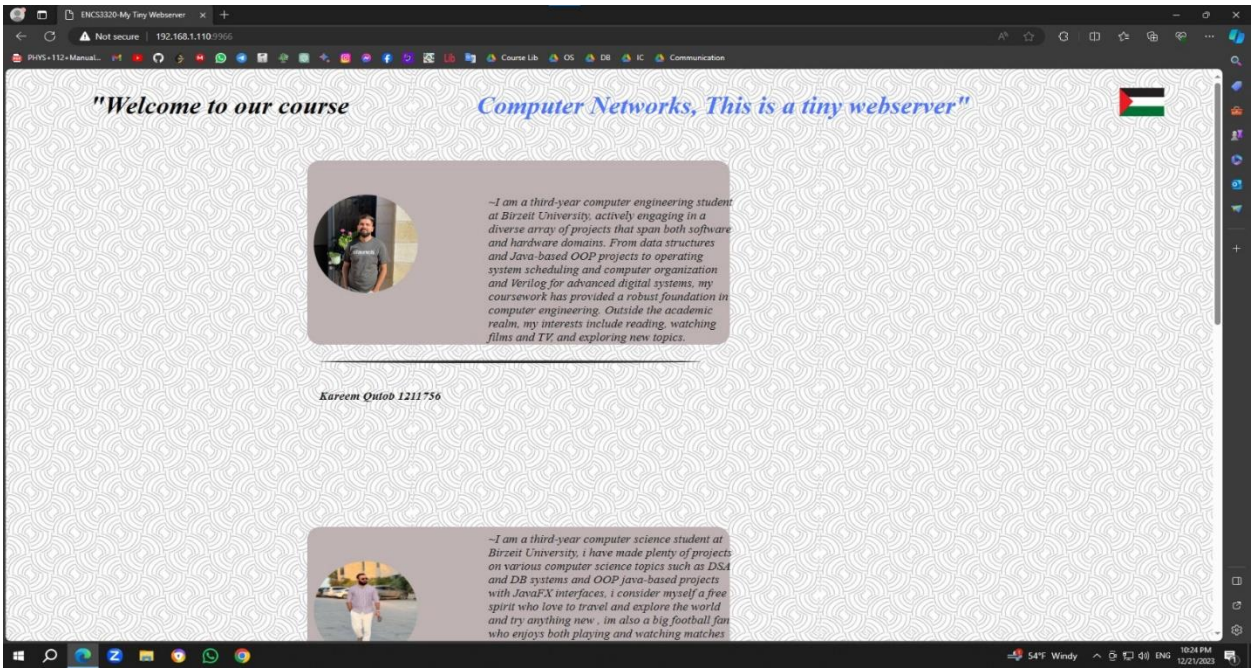


Figure 43:test from another laptob

From a phone:

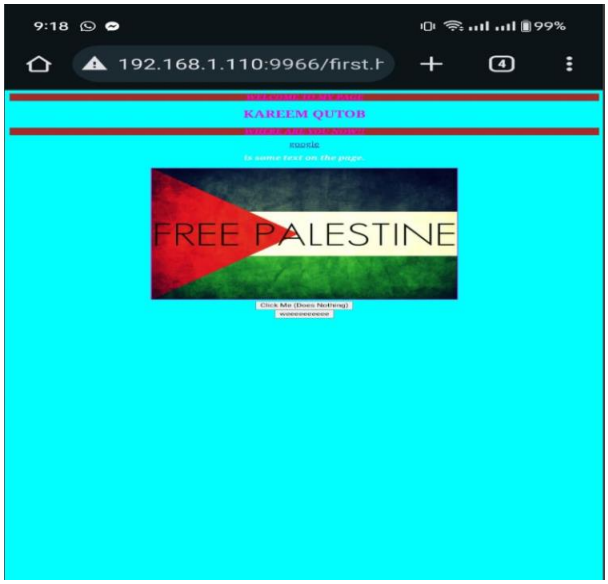


Figure 44:test from a phone

Conclusion:

In conclusion, this comprehensive computer networks and socket programming project requires us to demonstrate our understanding and practical application of network utilities, TCP client-server communication, and web server development. The project, divided into three parts, covers executing network commands, implementing a TCP client-server application with screen-locking functionality based on messages, and creating a web server capable of handling diverse HTTP requests, redirects, and error responses. Emphasizing the use of socket programming and forbidding external libraries. Successfully completing the project involves a holistic grasp of networking concepts, programming proficiency, and rigorous testing to validate the functionality of implemented solutions.