Faculty of Engineering and Technology
Electrical and Computer Engineering Department
Computer Networks -ENCS3320

Report of Project 1

Socket Programming

Group Members		
Name	**ID**	**Section**
Francis Miadi	**1210100**	**2**
Husain Abughosh	**1210338**	**1**
Mayar Masalmeh	**1211246**	**1**

Friday, 10 May 2024

Table of contents

Table of figures

Part 1:

Definitions:

Ping:

Abbreviated from “Packet Internet or Inter-Network Groper”. This command allows the user to verify whether an IP address actually exists and can accept requests, it also shows us how many seconds it takes for the response.

Tracert:

This command clarifies the actual path taken by a packet to travel from source to destination on an IP network. This command also records the time it takes for each hop during its travel from source to destination. And it includes the stops it makes.

NsLookup:

Abbreviated from “name server lookup”. This command finds the IP addresses of a certain domain name or the opposite. This command relies on the DNS server only

Telnet:

This is a client server protocol that allows the user to log into a remote host. Which means it will let the user feel as if they were communicating with the actual server directly. It can be generally used to see whether a server is responsive or not.

Running the commands

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

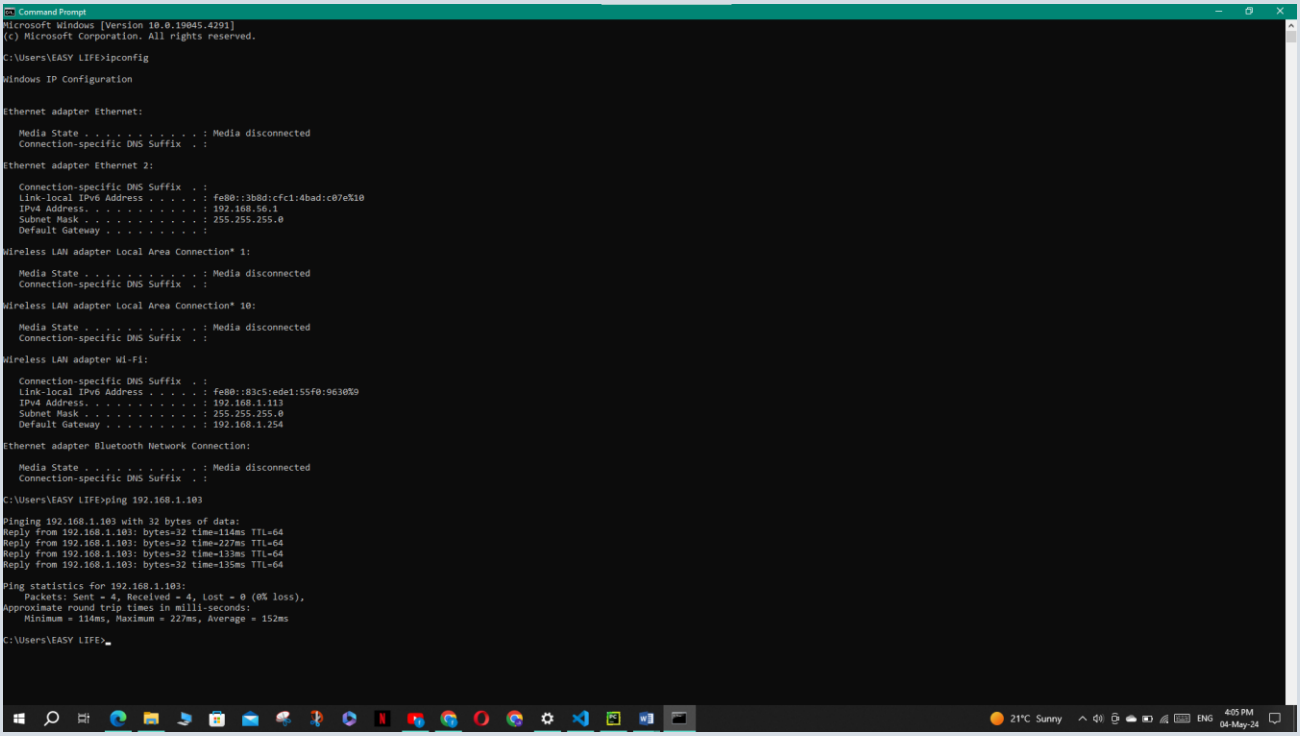


Figure 1-pinging from laptop to a smartphone

Firstly, on the cmd the ip-config of my device was requested, by obtaining the ip address of the router of the current network. after that, by going to an iphone connected to the network, and checking for its ip from the wifi section, then type ping (the ip address of the mobile). The average time for completing this command is 152 ms as shown from the ping results.

b) ping www.stanford.edu

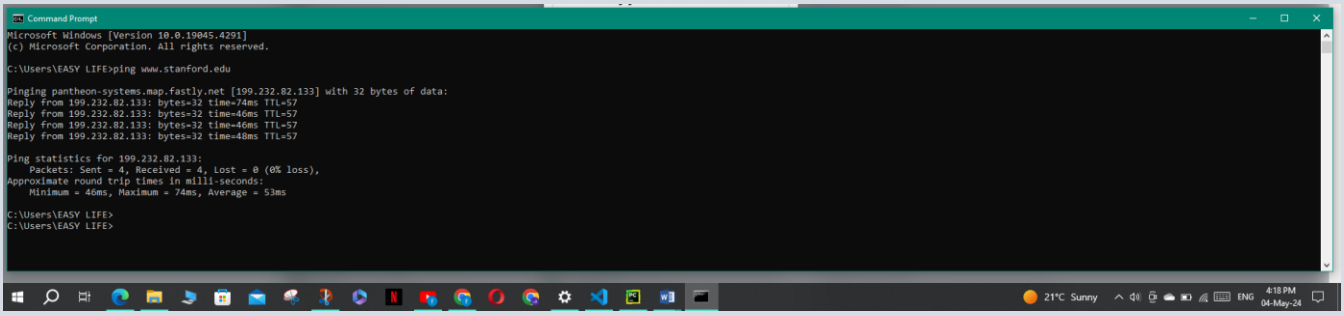


Figure 2- ping www.stanford.edu

c) From the ping results, do you think the response you got is from USA?

This results demonstrates how long it took the Stanford website to respond to the request types in the cmd. The average time was 53 ms with 4 packets sent and none of them was lost on its way. By taking the IP address from the cmd prompt window, and searching in some websites related to the ip address location approximate, for example iplocation.net it did show that the results can be driven from the USA because this IP 199.232.82.133, belongs to either the USA more specifically California or France more specifically Marseille. However, the exact source can not be verified for sure from the results obtained by the ping.

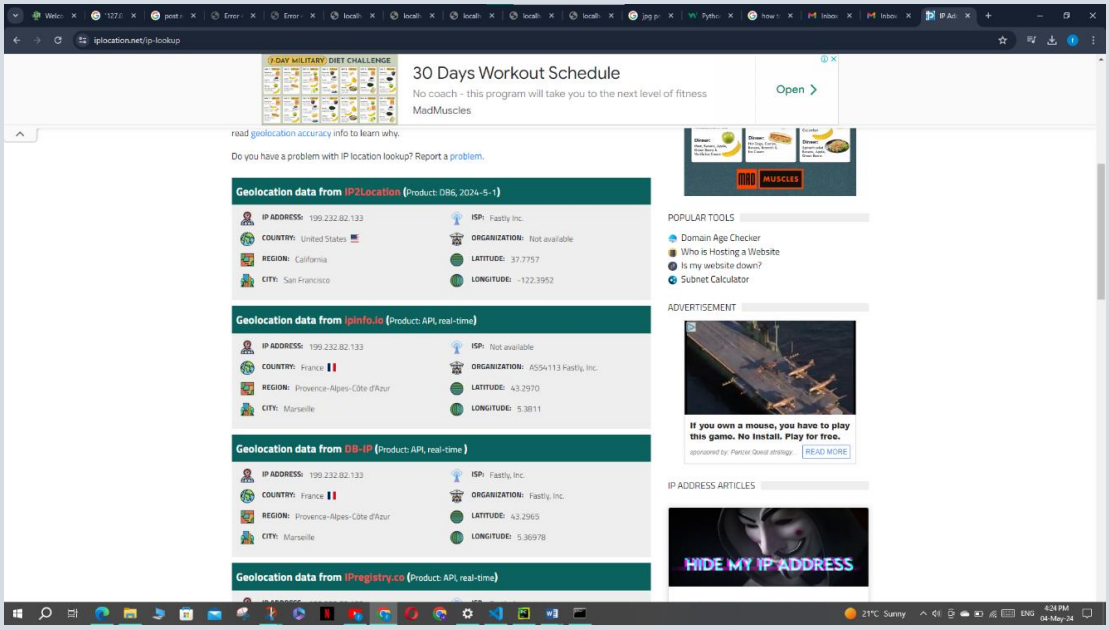


Figure 3- location of the ip address

d) tracert www.stanford.edu

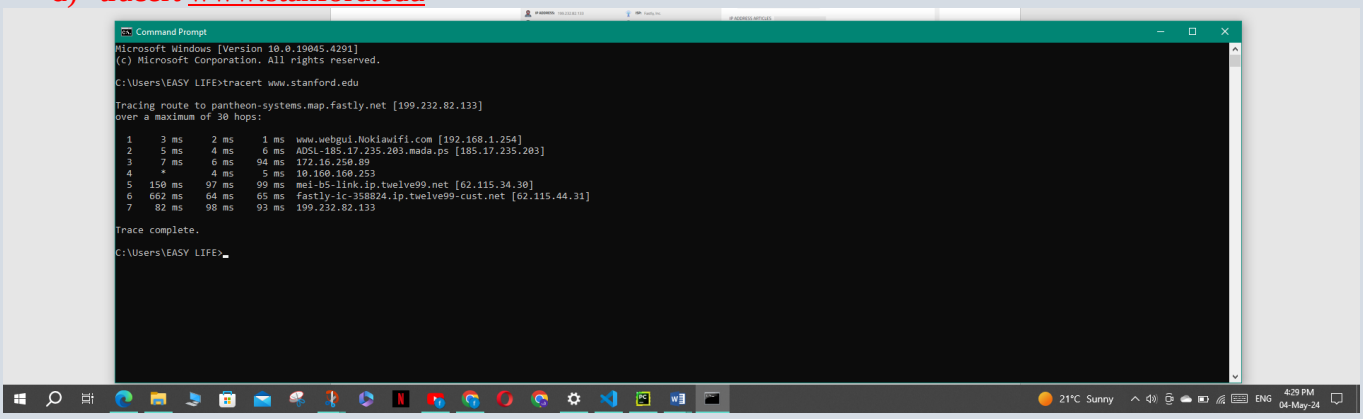


Figure 4- tracert www.stanford.edu

The results show that it took the 6 servers to send the request from the client “the cmd of our computer” to the Stanford server.

e) nslookup www.stanford.edu

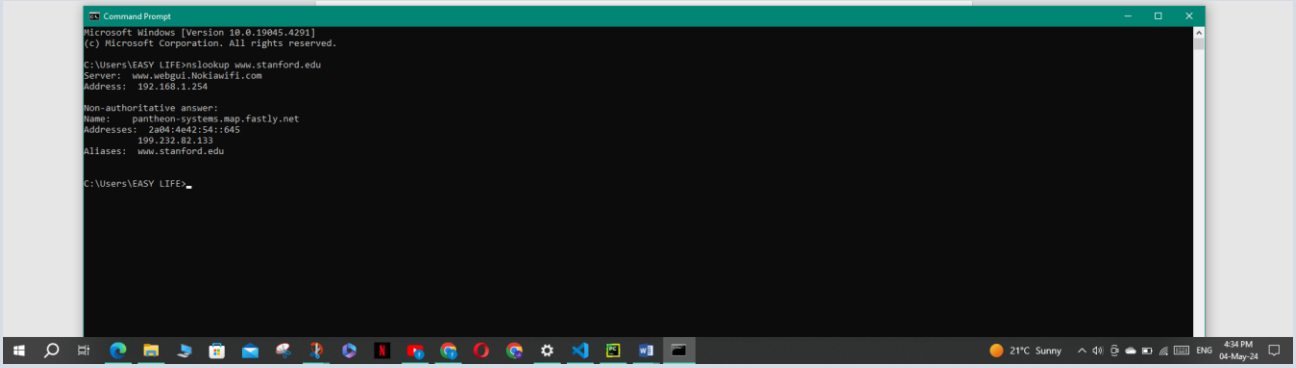


Figure 5- nslookup www.stanford.edu

The results demonstrate the different addresses the Stanford website possesses and the aliases of the server. It is good to note that the IP addresses shown in the nslookup can be used to ping the website if the domain name is not working.

Use Wireshark to capture some DNS messages

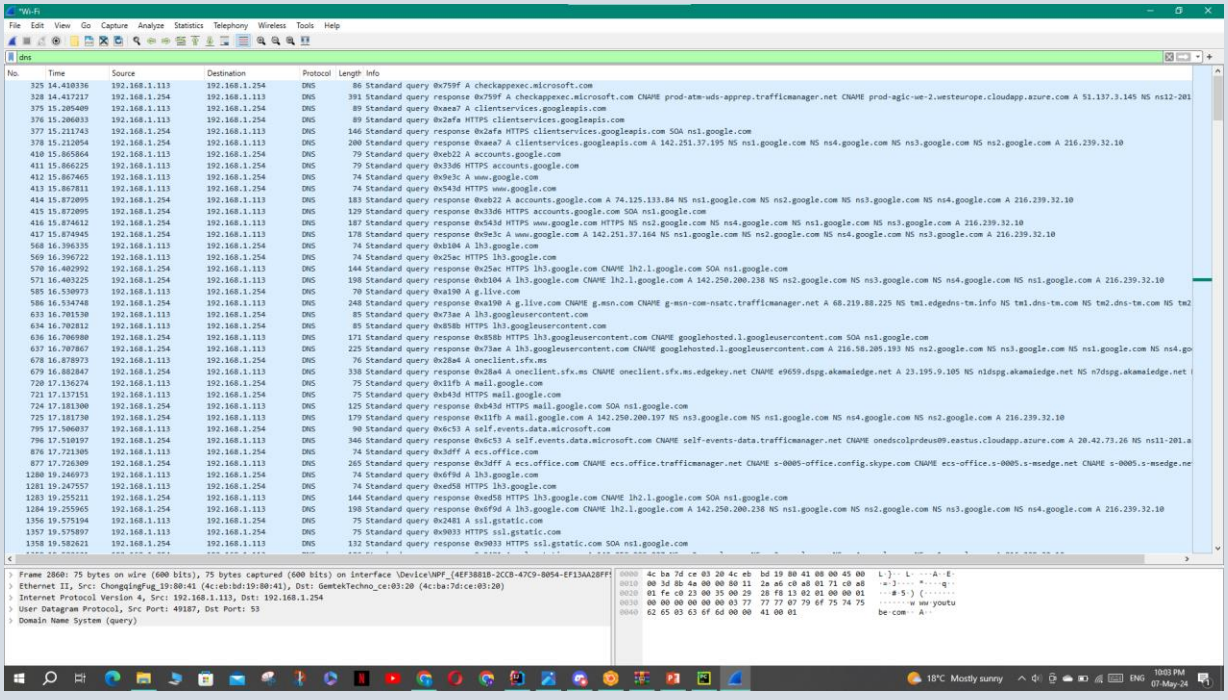


Figure 6- DNS captures by Wireshark

This result shows the DNS messages for some of the activities done on the pc, and it also Shows the response time identified as the DNS time, which is the time it takes the DNS to Receive the request for a domain name's IP address, process it, and return the IP address to the Browser or application requesting it.

Part 2

The code implements a simple UDP (User Datagram Protocol) client-server application that facilitates message exchange between multiple clients. The server listens for messages on a specified port and can broadcast messages to all clients on the local network. Clients can send messages and request to view messages sent by others. Here's a general overview of how the code functions:

Code function explanation

1. **Server Setup**:

- The server is configured to listen on port 5051.
- It uses a broadcast IP (192.168.1.255) to send messages to all clients in the network.
- The server socket is set up to handle UDP packets, which are connectionless and suitable for broadcasting messages.

2. **Client Initialization**:

- Each client is prompted to input their first and last name, which is then combined to form the client_name.
- This name is used to identify messages sent by the client and to filter out the client's own messages from the displayed list of received messages.

3. **Message Handling**:

- **Receiving Messages**:
- The server continuously listens for incoming messages using a separate thread.
- Upon receiving a message, it decodes and logs the message along with the sender's information and timestamp.
- The server also prints a notification of received messages from other clients.

- **Sending Messages**:

- Clients can enter messages which are then sent to the server.
- Each message is formatted with the sender's first name, last name, and the message content, separated by semicolons (;).

4. **Displaying Messages**:

- Clients have the option to view all received messages.
- Messages are listed with a unique index, and clients can select a specific message to view its details.

5. **Concurrency**:

The server uses threading to handle message reception in the background while allowing the main program to continue running, enabling simultaneous sending and receiving of messages.

Key Features

- **Broadcast Communication**: Uses UDP broadcasting to send messages to all clients in the local network.
- **Concurrency**: Utilizes threading to handle incoming messages concurrently with other operations.
- **Message Filtering**: Ensures clients do not see their own messages in the received list.
- **User Interaction**: Provides a simple interface for clients to send messages and view received messages.

Run sample

The first user Husen send his messages to all peers

```
Enter your first name: Husen
Enter your last name: Abugosh
UDP Server listening on port 5051

Options:
1. Send another message
2. Show all messages
Select an option (1/2): 1
Enter your message: Hello Francis

Options:
1. Send another message
2. Show all messages
Select an option (1/2): 1
Enter your message: How are u ?

Options:
1. Send another message
2. Show all messages
Select an option (1/2): 1
Enter your message: Are u good ?
```

Figure 7- the first user message to all peers

The second user Francis received the messages from Husen and he can see the details of the message by chose the line with D char

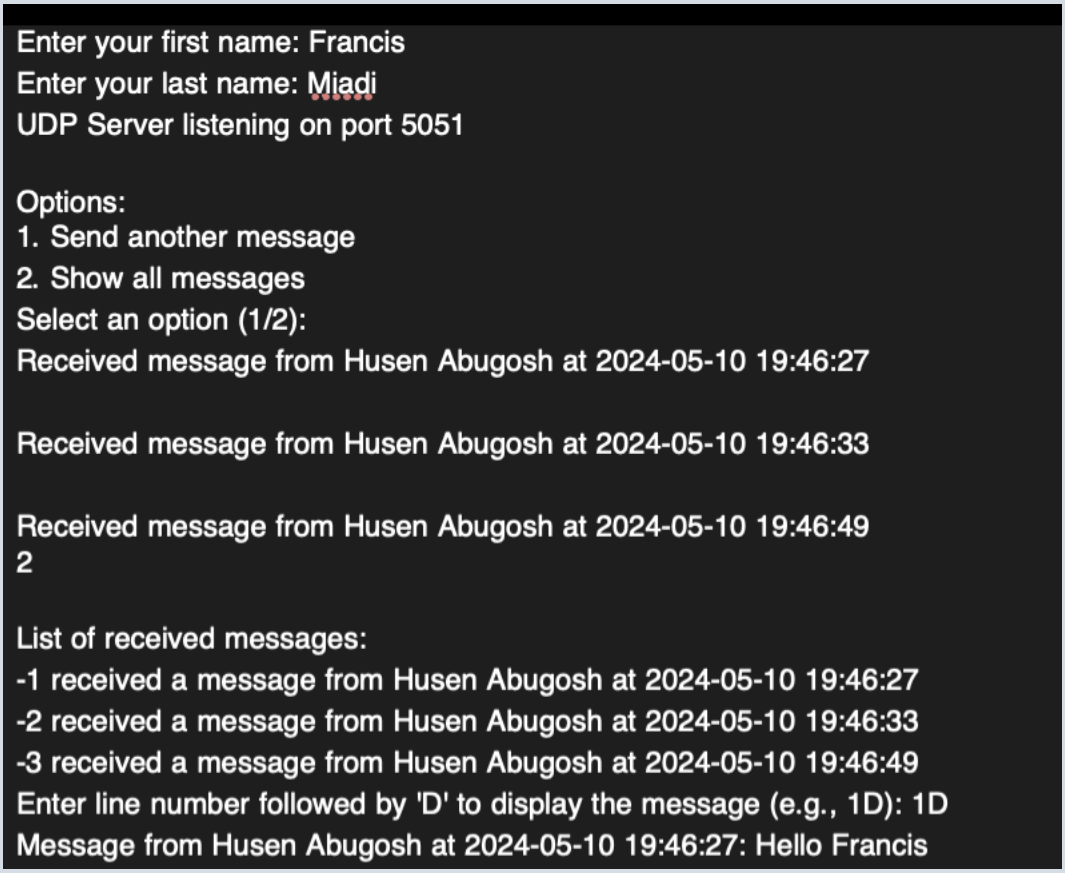


Figure 8- the received message from the first user

The second user Francis chose to send message to the all peers

Figure 9- the second user sends a message to all peers

The first user Husen received Francis message and chose to showed it up

```
Options:
1. Send another message
2. Show all messages
Select an option (1/2):
Received message from Francis Miadi at 2024-05-10 19:51:38
2

List of received messages:
-1 received a message from Francis Miadi at 2024-05-10 19:51:38
Enter line number followed by 'D' to display the message (e.g., 1D): -1
Enter line number followed by 'D' to display the message (e.g., 1D):

Options:
1. Send another message
2. Show all messages
Select an option (1/2): 2

List of received messages:
-1 received a message from Francis Miadi at 2024-05-10 19:51:38
Enter line number followed by 'D' to display the message (e.g., 1D): 1D
Message from Francis Miadi at 2024-05-10 19:51:38: hello husen
```

Figure 10- the received message from the second user

The screenshots demonstrate the following interactions:

1. **Client Initialization**:

- Users input their first and last names.
- The server acknowledges its readiness to listen on the designated port.

2. **Sending Messages**:

- Clients enter messages which are then broadcasted to the network.
- Messages like "Hello Francis," "How are u?", and "Are u good?" are shown being sent.

3. **Receiving Messages**:
- The server receives messages from clients and displays them, excluding the sender's own messages.

4. **Viewing Messages**:
- Clients can list all received messages and view details of specific messages by entering the corresponding index followed by 'D'.

Part 3

Have a look also on rfc2616 (https://datatracker.ietf.org/doc/html/rfc2616). From rfce2616, what is Entity Tag Cache Validators in the HTTP protocol and why do we need it?

The ETag response-header field value, an entity tag, provides for an "opaque" cache validator. This might allow more reliable validation in situations where it is inconvenient to store modification dates, where the one-second resolution of HTTP date values is not sufficient, or where the origin server wishes to avoid certain paradoxes that might arise from the use of modification dates.

This means that In HTTP, resource versions are uniquely identified by Entity Tags (ETags), which are used to manage conditional requests and verify cached content. By enabling clients to determine whether a resource has changed before reloading it, they contribute to bandwidth reduction and increased efficiency. ETags also assist in preventing disputes when editing shared resources.

In this part we will implement a web server, and a website that will act as a client. The web server is built upon a foundation of socket programming, utilizing the Python programming language to create a responsive and efficient server capable of handling multiple simultaneous client requests. Key features include request parsing, content-type determination, file serving, redirections, and error handling, it also can serve a page that works as an image finder, for images that are pre stored on our devices.

The server will receive either GET or POST request from our website through a persistent TCP connection, which will stay open while the website keeps sending request messages through it as long as there is objects to be requested , the server role here is to determine the method that the client used to ask for the object or the file, to determine if the request was valid, and if the file or object name exists in the project, if yes, it will return the desired file, if not, an error web page will pop up to tell the website that the requested file doesn't found. It is also designed using multi-threading so it can handle more than one client at the same time.

The main English web page

In accordance with the specifications outlined in **Part A**, the web server demonstrates precise handling of specific requests: **/**, **/index.html**, **/main_en.html**, and **/en**. For each of these requests, the server consistently serves the main_en.html file. The Content-Type header is appropriately set to **text/html**, aligning with the specified requirements. The accompanying screenshots visually capture the server's responses to these distinct requests, providing a concise and comprehensive overview of the web server's behavior in meeting the outlined criteria.

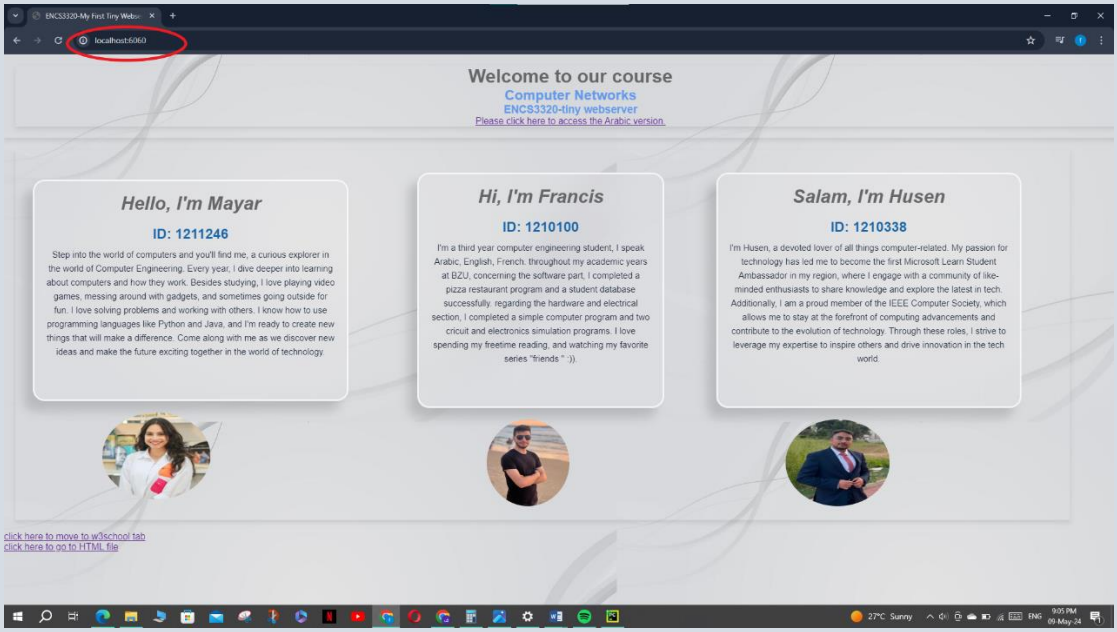


Figure 11- main web page result

When the browser initiates a request for the root path **(/)**, the server seamlessly redirects to the main page without the trailing slash. The automatic removal of the slash by the browser is observed, and the server still responds by serving the main_en.html file with the appropriate Content-Type set to **text/html**.

This following requests were successfully handled and the webpages opened as expected.

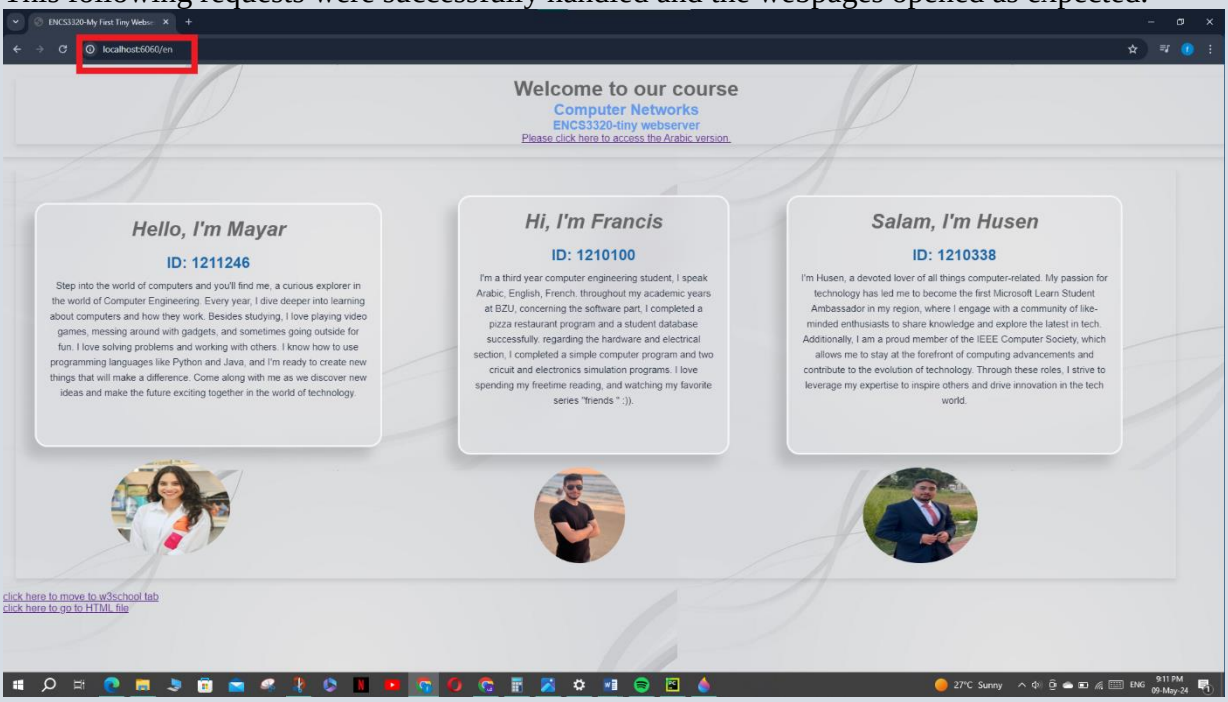


Figure 12- requesting /en

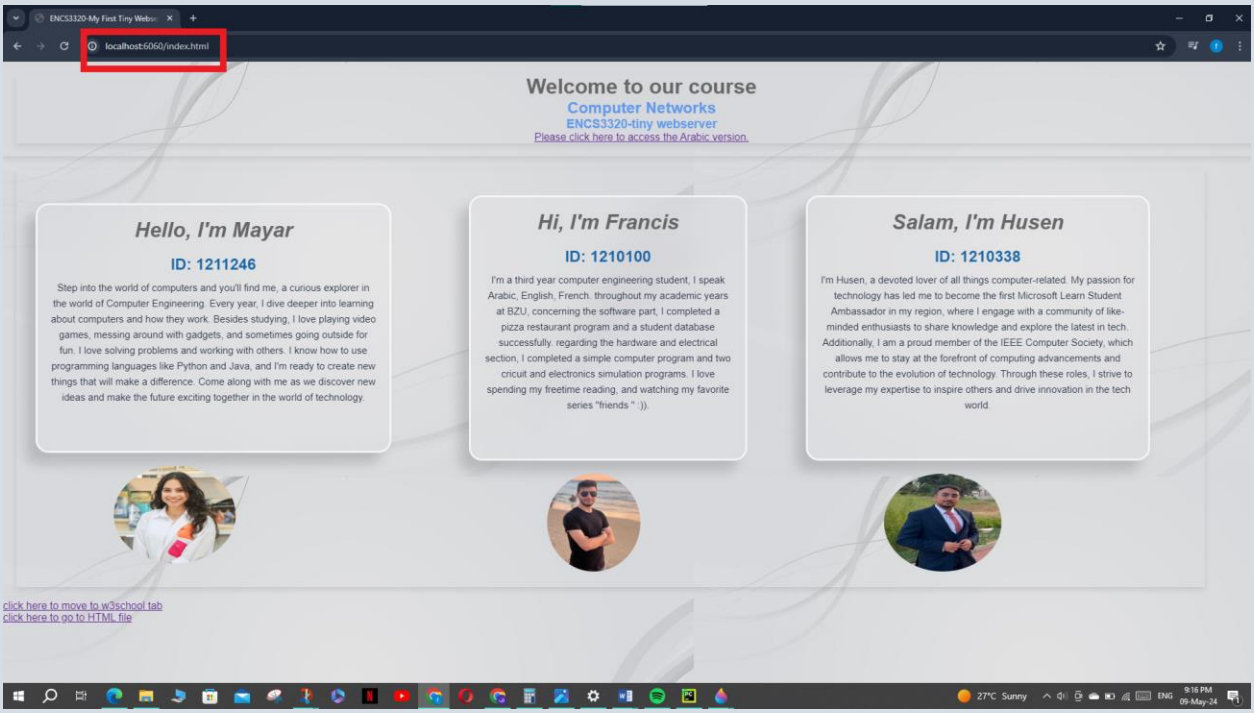


Figure 13- requesting /index.html

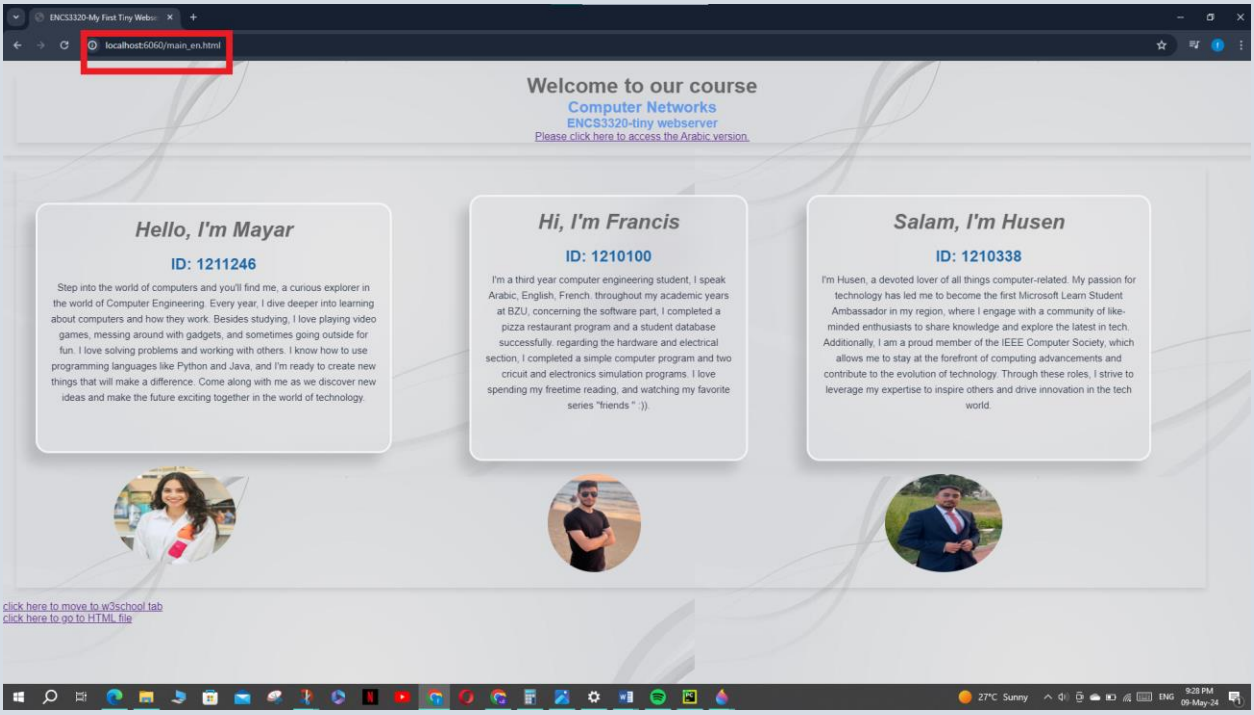


Figure 14- requesting /main_en.html

In the main web page, there is two hyperlinks and here is the results when the user clicks on each one:

When the user clicked on ("click here to move to W3school tab"), this was the result:

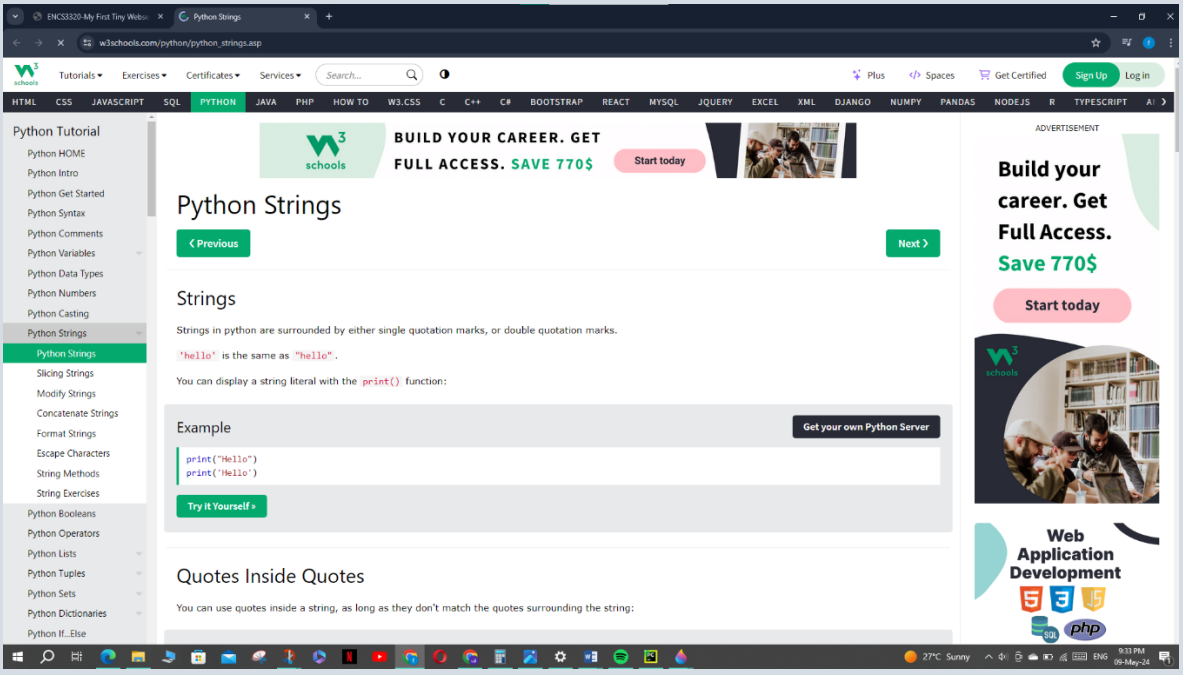


Figure 15-clicking on W3school link result

And when the user clicked on (“click here to go to HTML file”), this was the result:

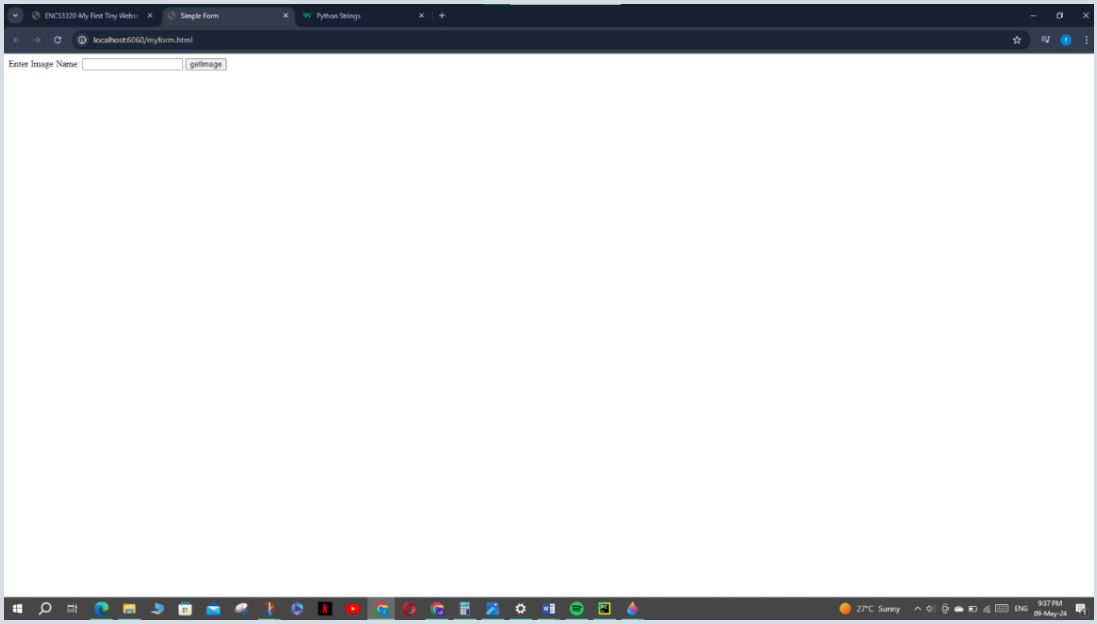


Figure 16- clicking on go to HTML file result

Myform.html page

This web page called “myform.html” it works as an image finder, in which you enter a name of an image that was already saved in a folder called images, and the client (my.form.html) will send a POST request to the server to get the specified image, and if there is no such image, the Error page will pop up.

Here is the pre saved images in “images”:

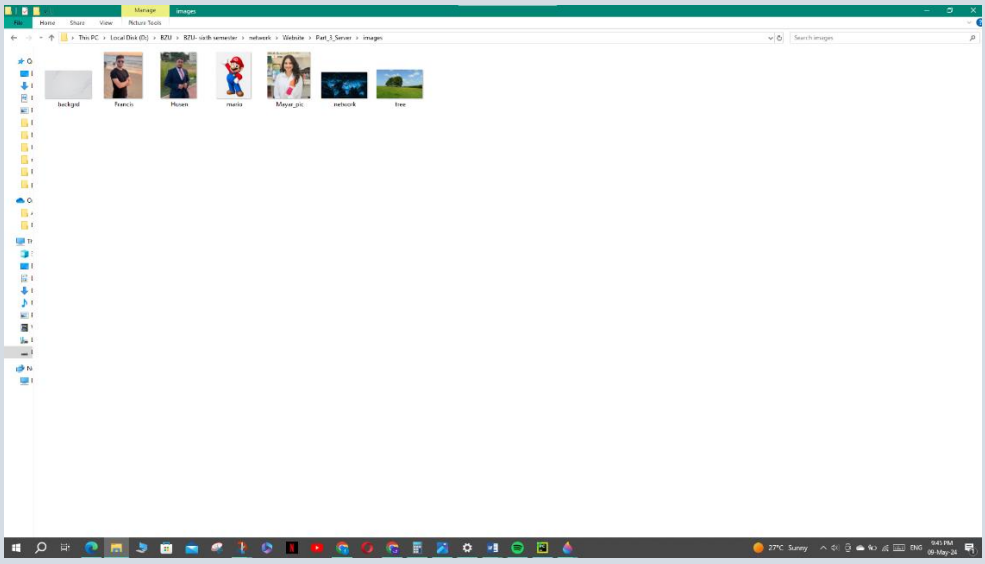


Figure 17- the images folder

And here is the results for searching for a .png and .jpg images and an image that doesn't exist:

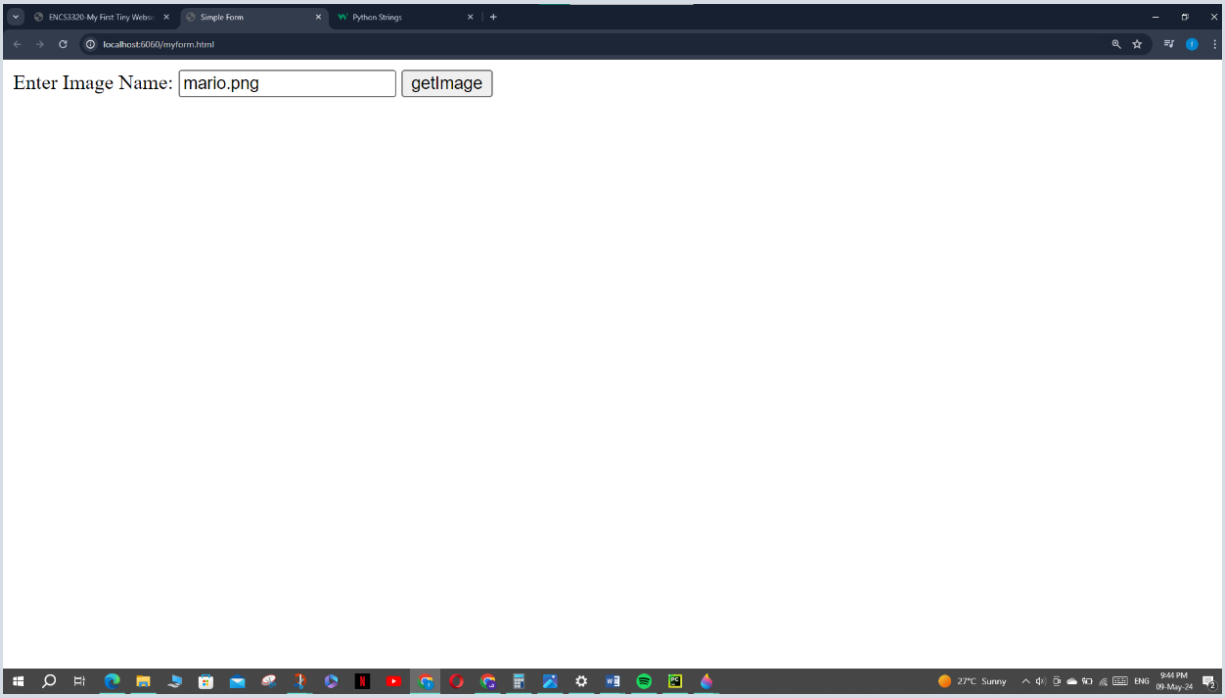


Figure 18- searching for a .png image in myform.html

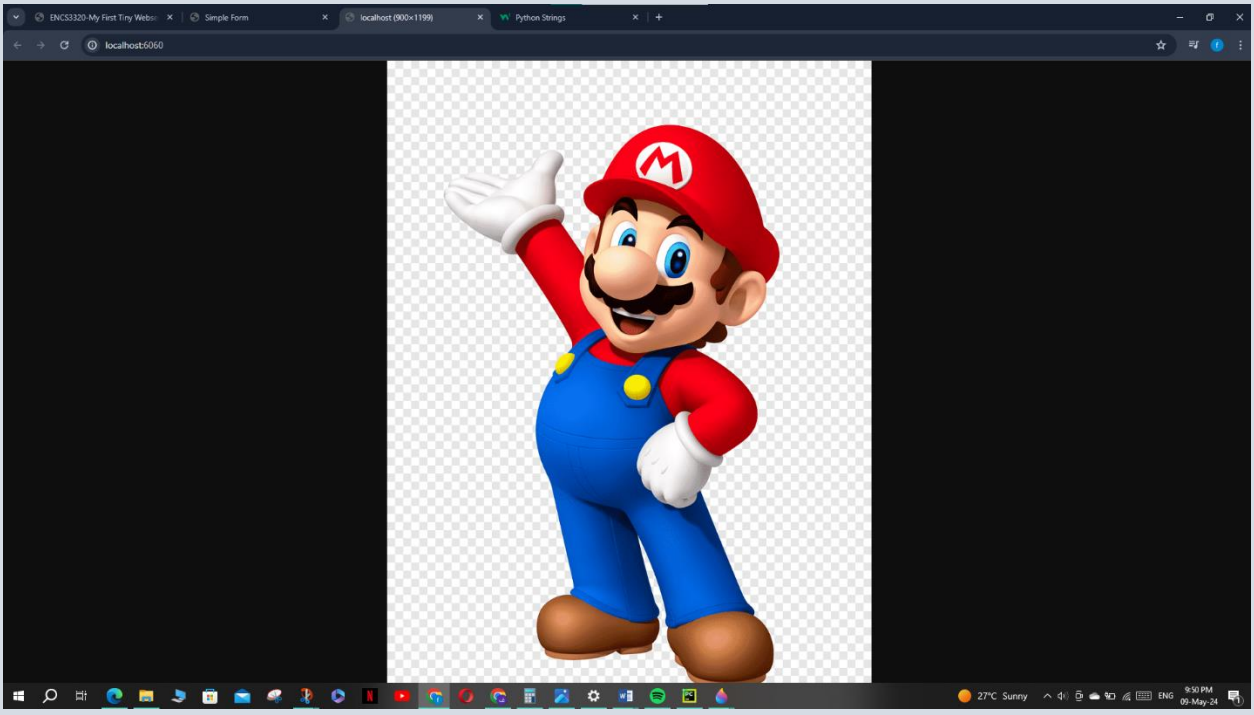


Figure 19- the result of searching for a .png image using myform.html

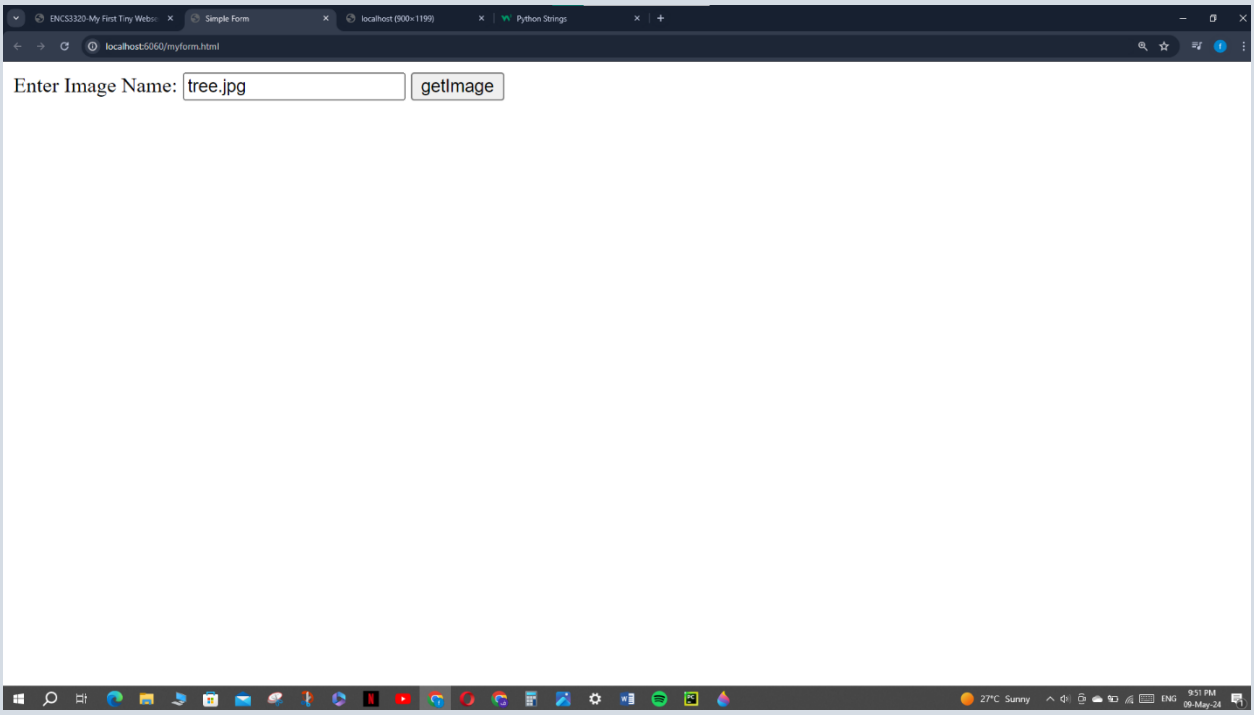


Figure 20- searching for a .jpg image using myform.html

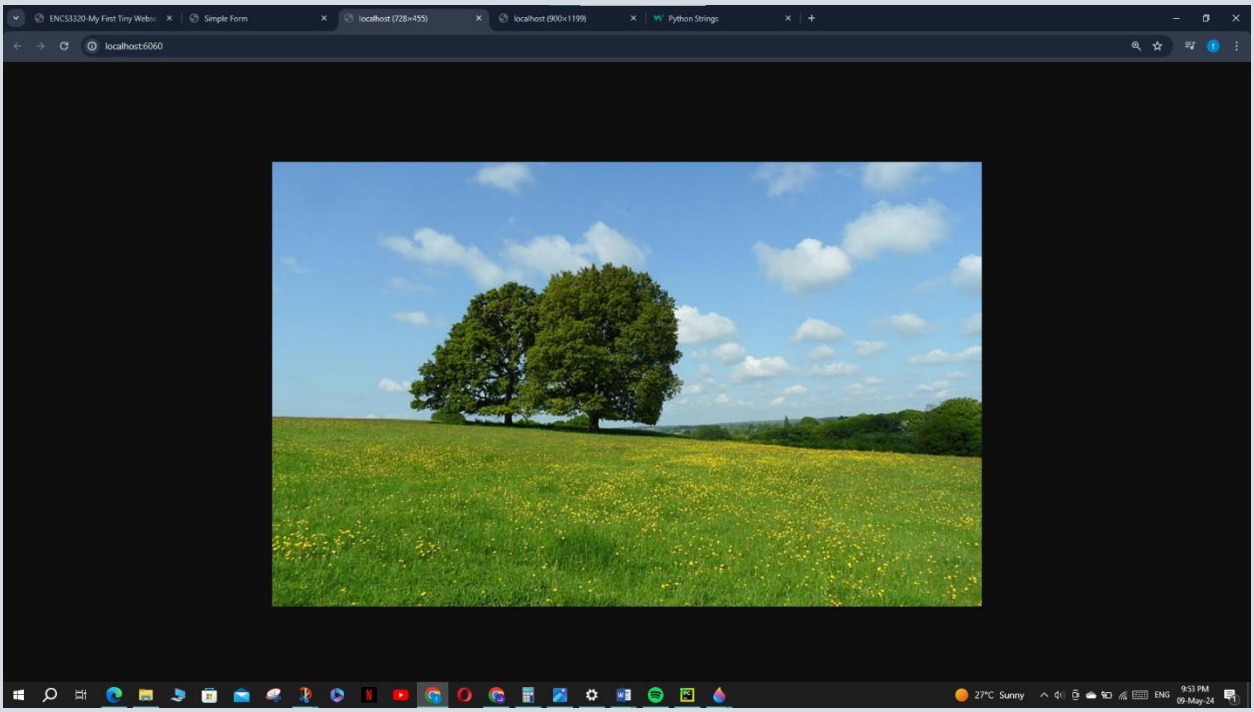


Figure 21- the result of searching for a .jpg image using myform.html

Here is a request for a name of an image that doesn’t exist,

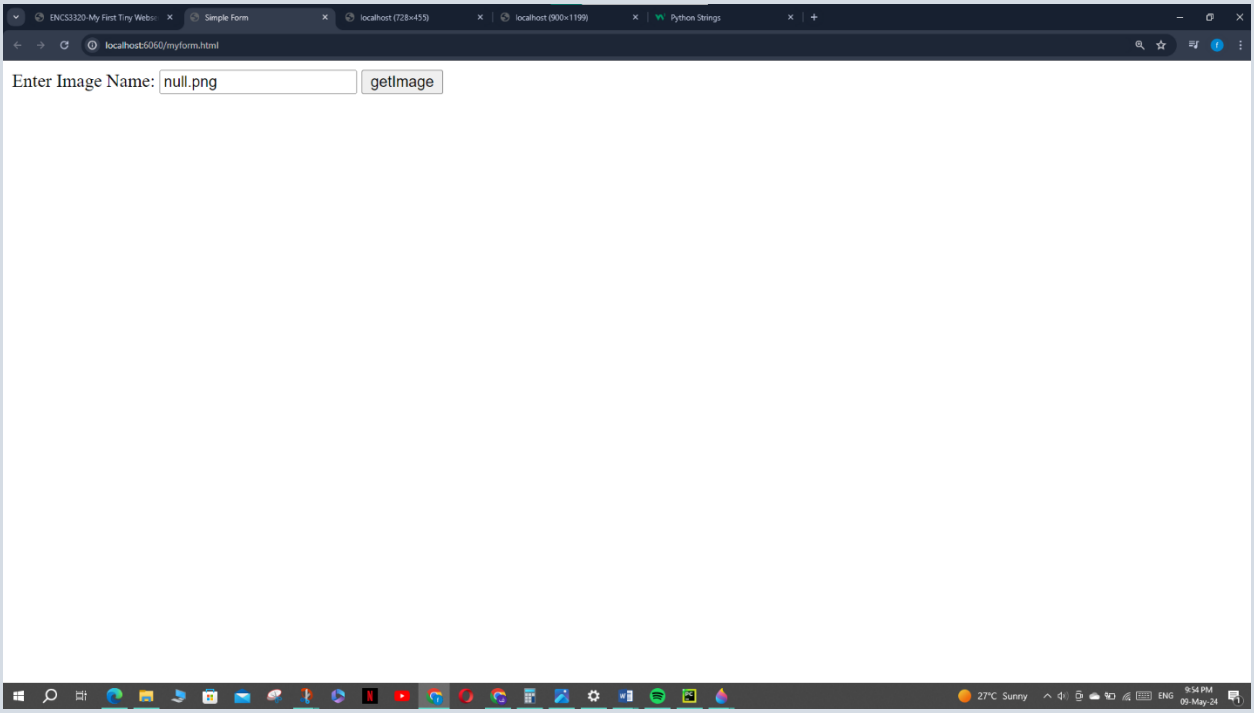


Figure 22- searching for an image that doesn't exist using myform.html

And since it doesn’t exist, then the error (not found) web page will be opened,

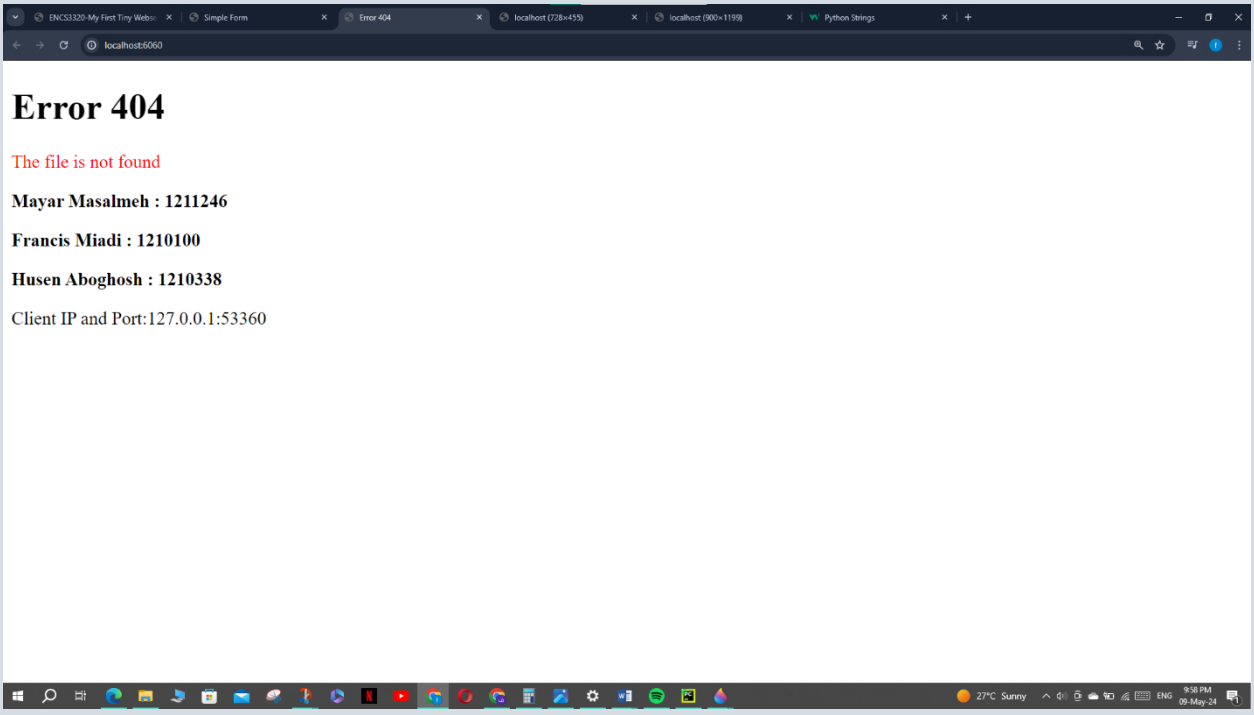


Figure 23- result of searching for an image that doesn't exist using myform.html

The Arabic version of the main page

A shown below ,You can reach this web page by two ways, by clicking on the hyperlink (“Please click here to reach the Arabic version”) in the English version, or by manually type http://localhost:6060/ar in the search bar in the browser.

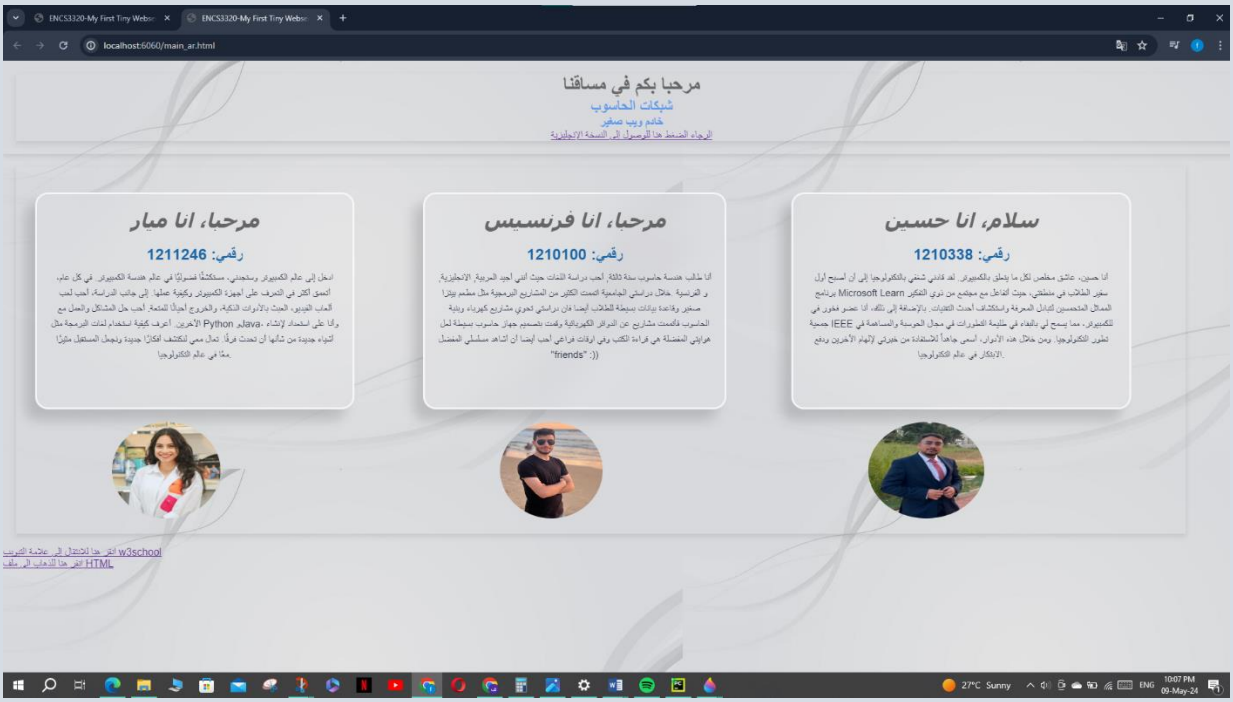


Figure 24- the Arabic version by clicking on the hyperlink

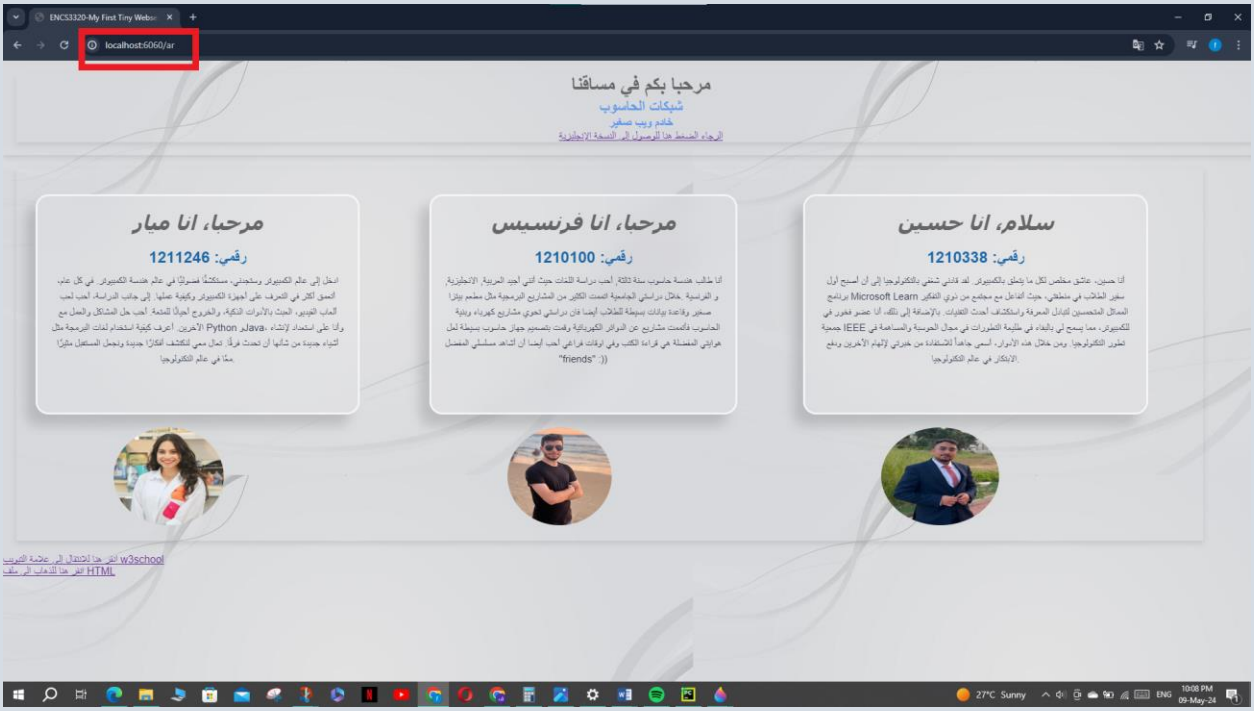


Figure 25- requesting /ar

And when clicking on the hyperlinks in the Arabic version, the same results that we saw in the English version requests will be shown again, as shown below

When the user click on (“انقر هنا للانتقال الى علامة التبويب w3school”),

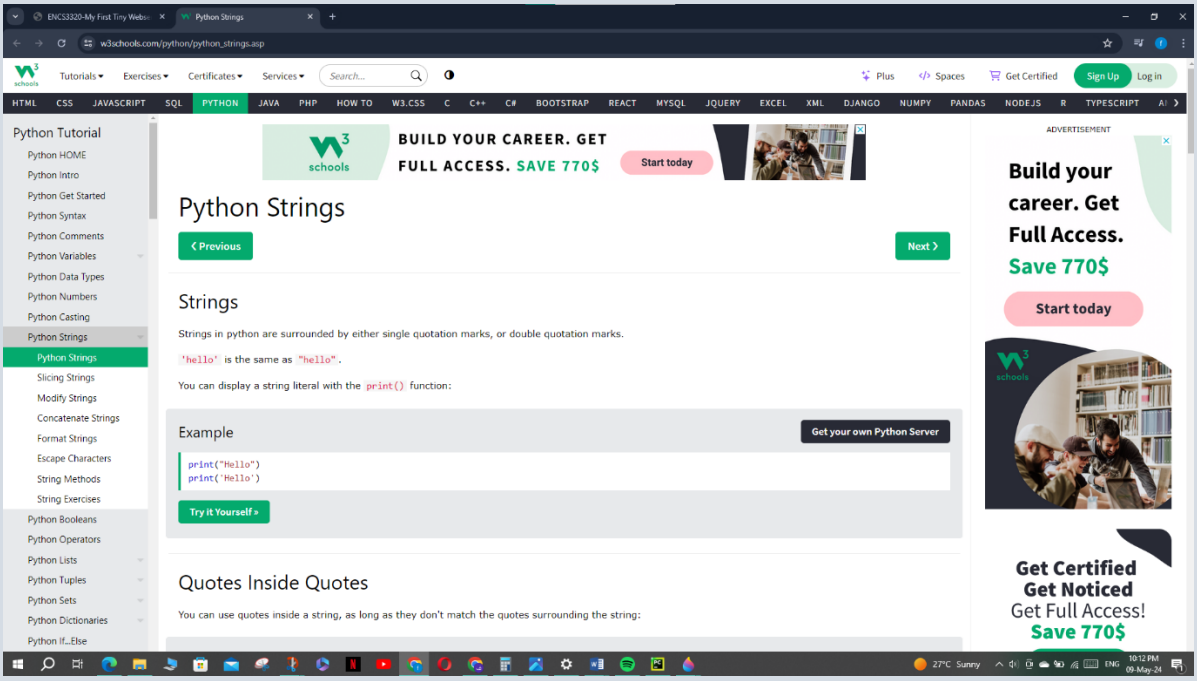


Figure 26- result of clicking on (“انقر هنا للانتقال الى علامة التبويب w3school”)

When the user click on (“انقر هنا للوصول الى ملف HTML”),

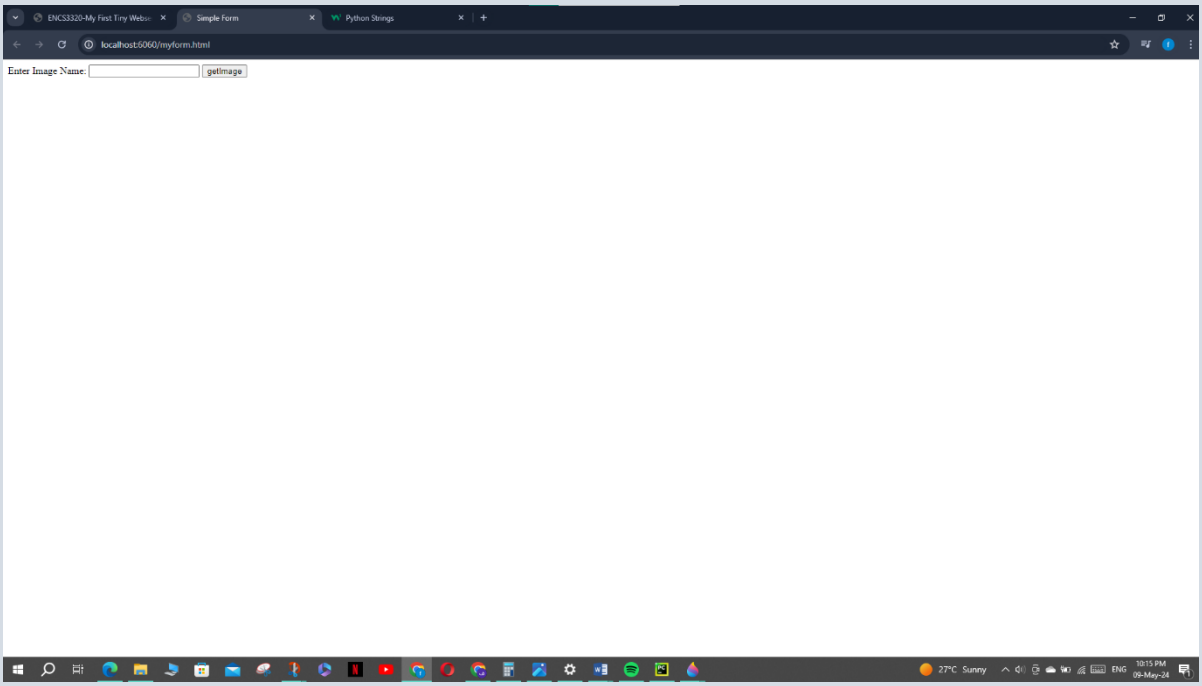


Figure 27-result of clicking on (“انقر هنا للوصول الى ملف HTML”)

And when the user clicks on (“الرجاء الضغط هنا للوصول للنسخة الانجليزية”) , it will go the main_en.html web page,

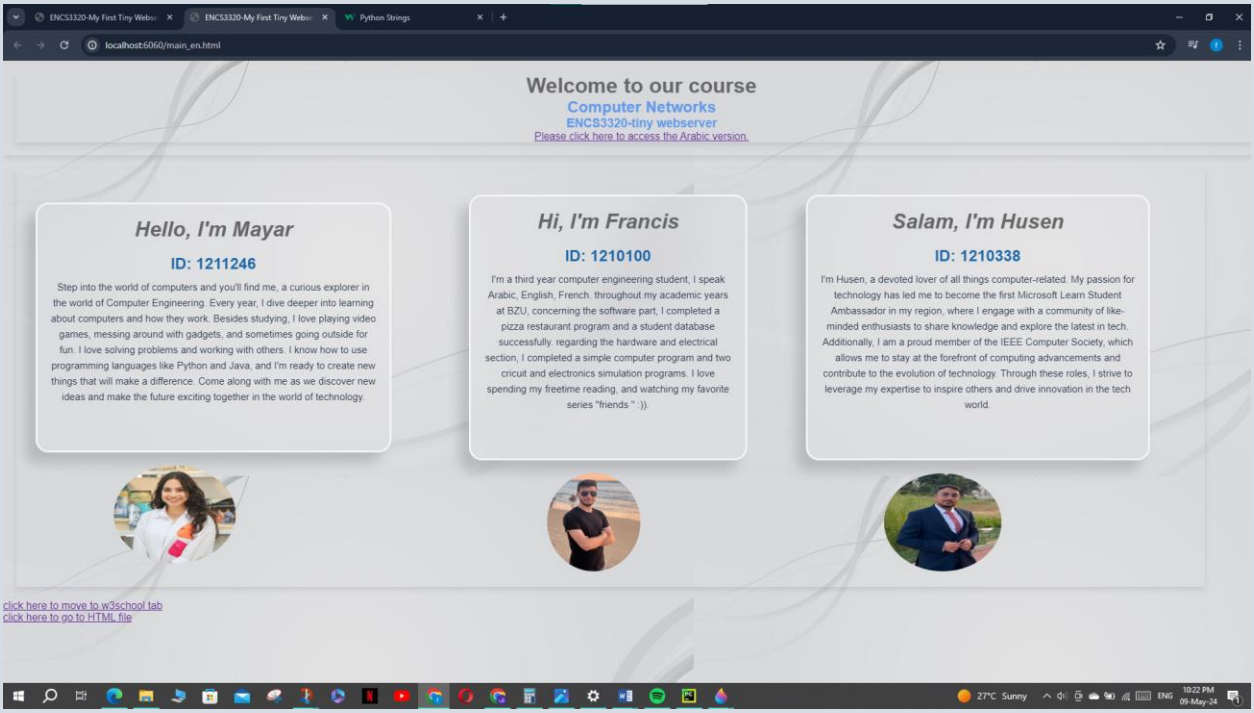


Figure 28- result of clicking on الرجاء الضغط هنا للوصول للنسخة الانجليزية

Request an HTML file

When the browser requests /"anyHTMLfile".html, the server accurately responds by serving the requested HTML file and The Content-Type is appropriately set to text/html. In this screenshot the user requested /myform.html and the webserver responded correctly.

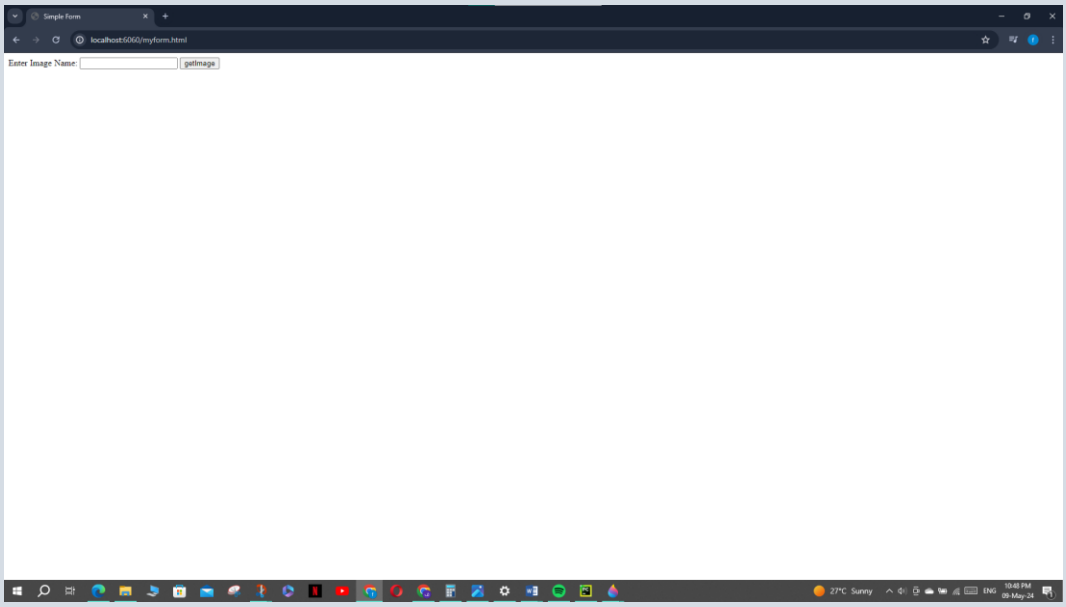


Figure 29- result of requesting an HTML file

Request a CSS file

In the depicted figure, the server responds appropriately to the request for /styles.css by serving the requested CSS file with the Content-Type correctly set to text/css.

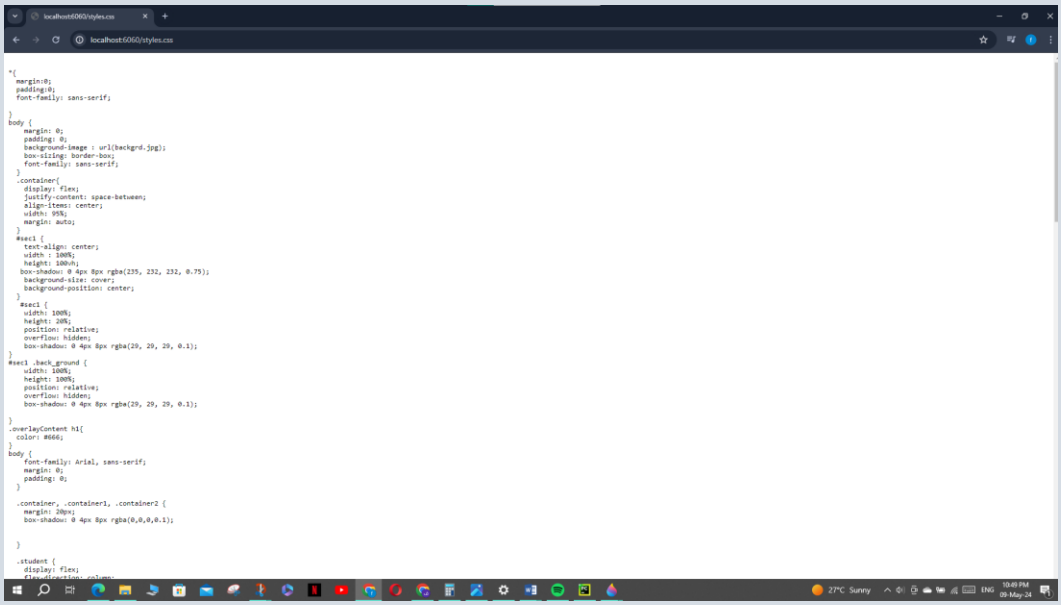


Figure 30- result of requesting a css file

Request a PNG Image

The server appropriately responds to the request for /mario.png by serving the specified PNG image. The Content-Type is correctly set to image /png .

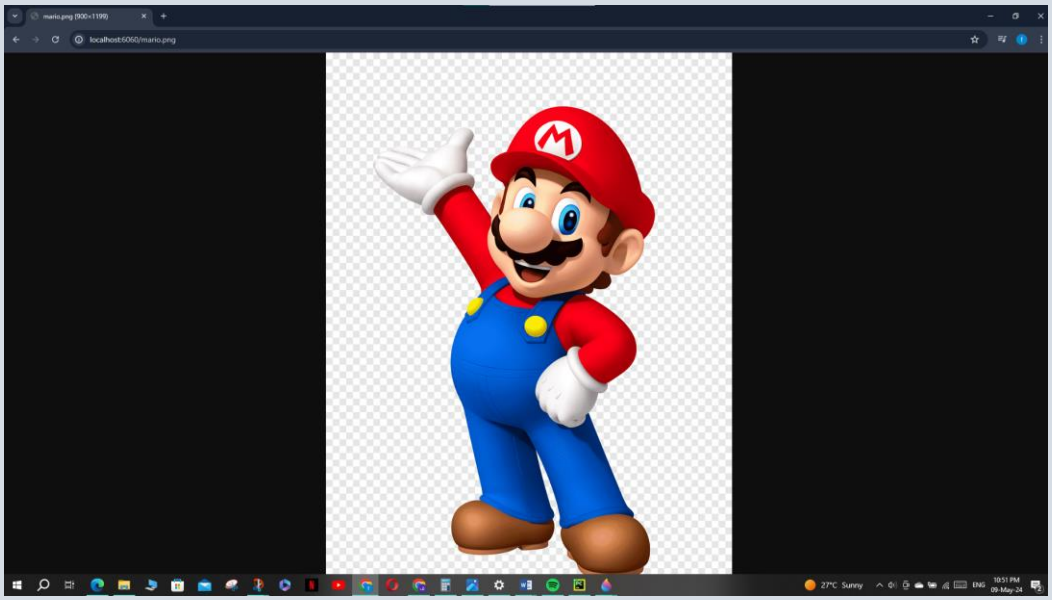


Figure 31- result of requesting a PNG image

Request a JPG image

The server appropriately responds to the request for /tree.jpg by serving the specified JPEG image. The Content-Type is correctly set to image/jpeg.

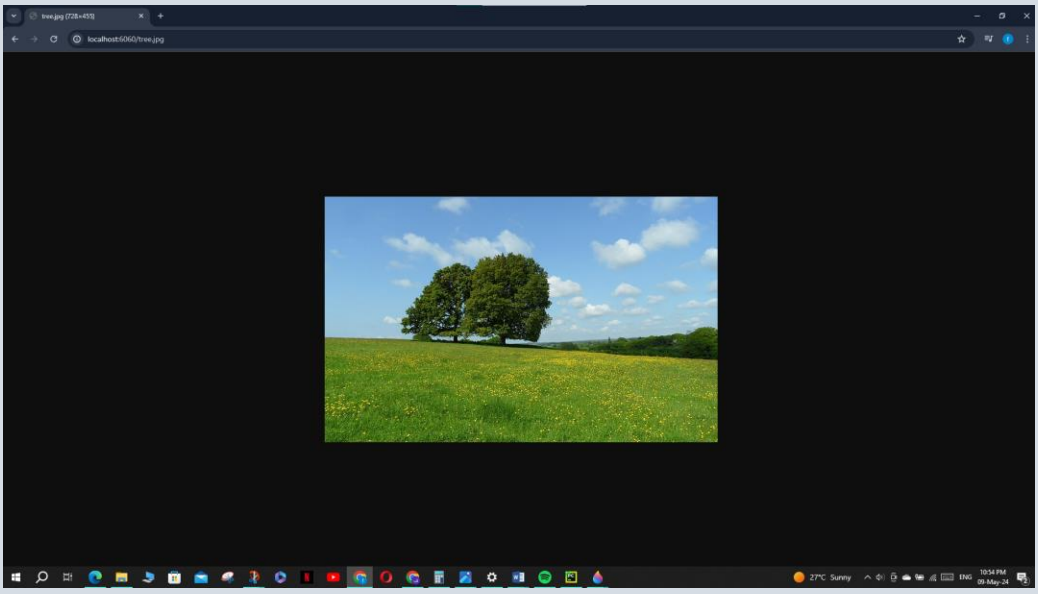


Figure 32- result of requesting a JPG image

Redirects

In response to a request for /itc, the server successfully executes a temporary redirect (status code 307), directing the user to the specified website (itc).

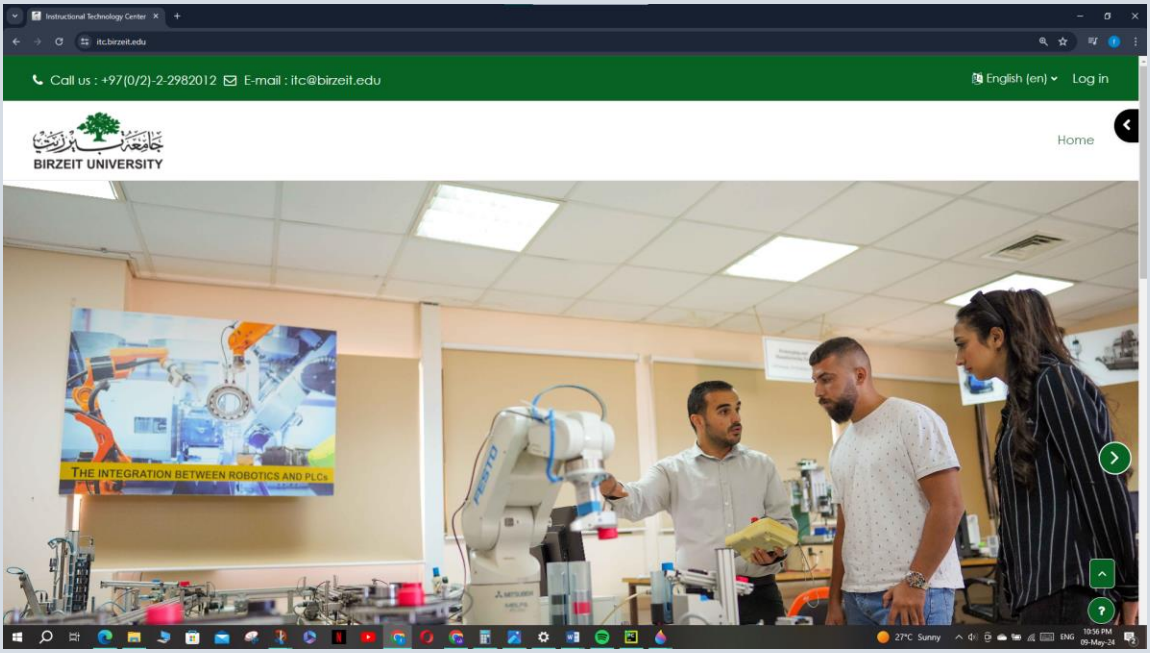


Figure 33- itc request result

When the browser requests /so, the server correctly implements a temporary redirect with a status code of 307, redirecting the user to the specified website (stackoverflow.com)

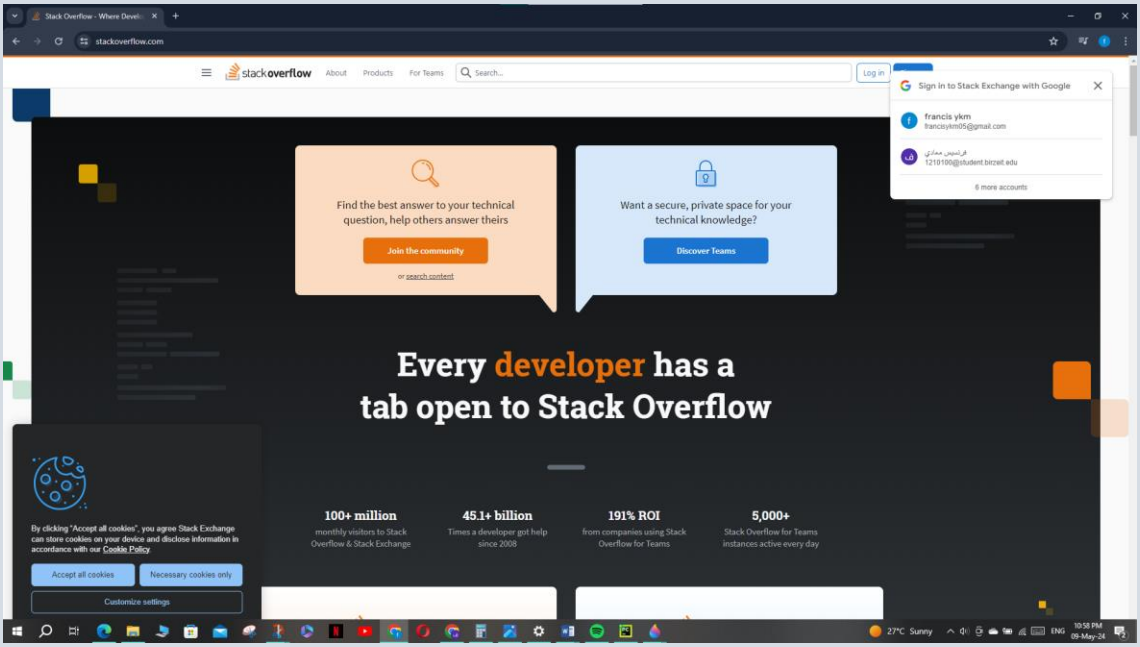
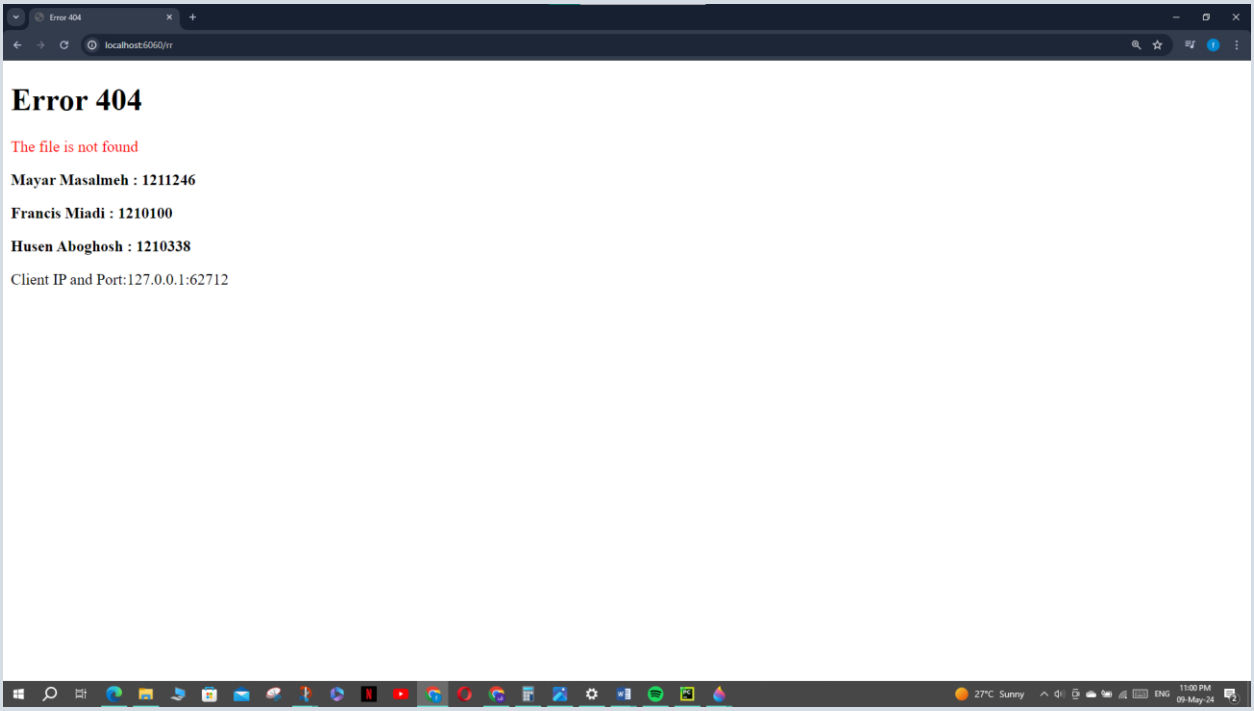


Figure 34- stackoverflow request result

The error page

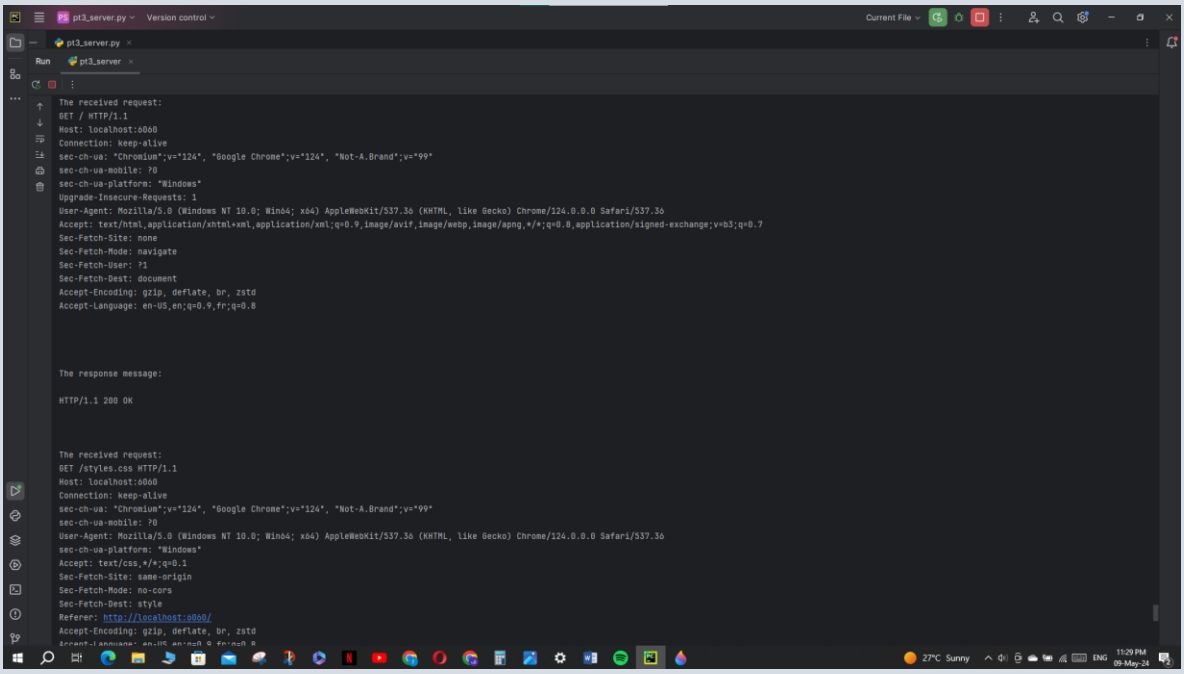
When the browser requests an incorrect path (/rr), the server correctly responds with a simple HTML webpage conveying a 404 Not Found status. The HTML content adheres to the specified structure, featuring the title "Error 404," body text indicating the file is not found, and bolded names and IDs. Additionally, the IP and port information of the client is included.

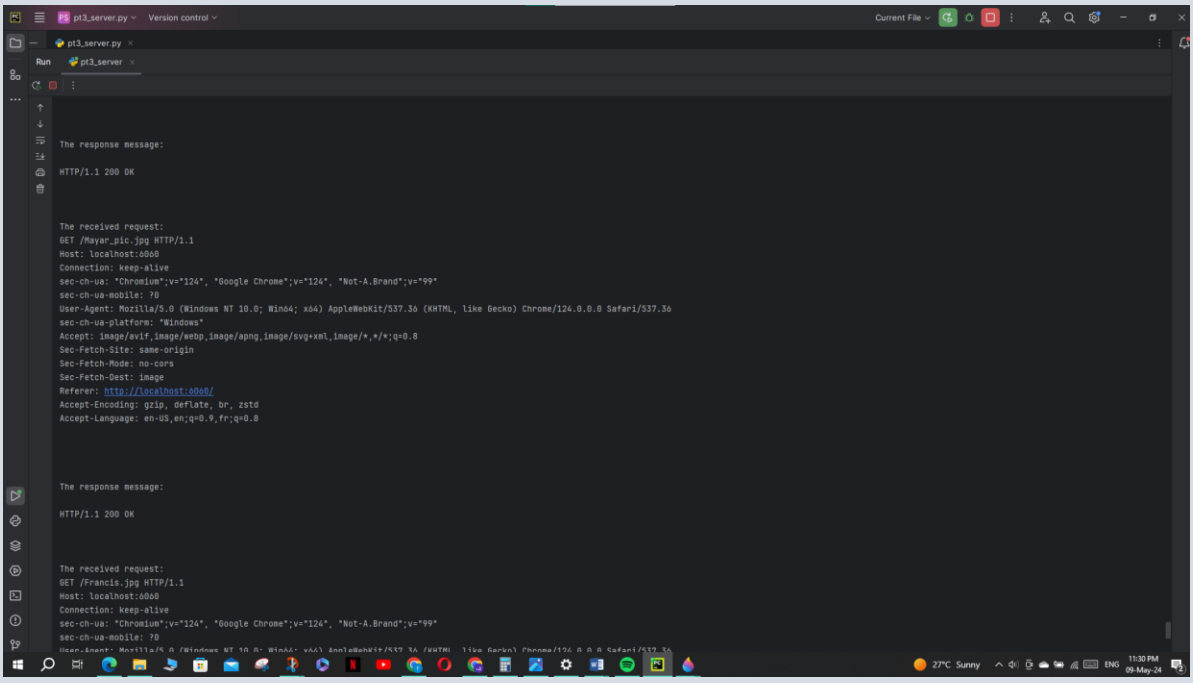


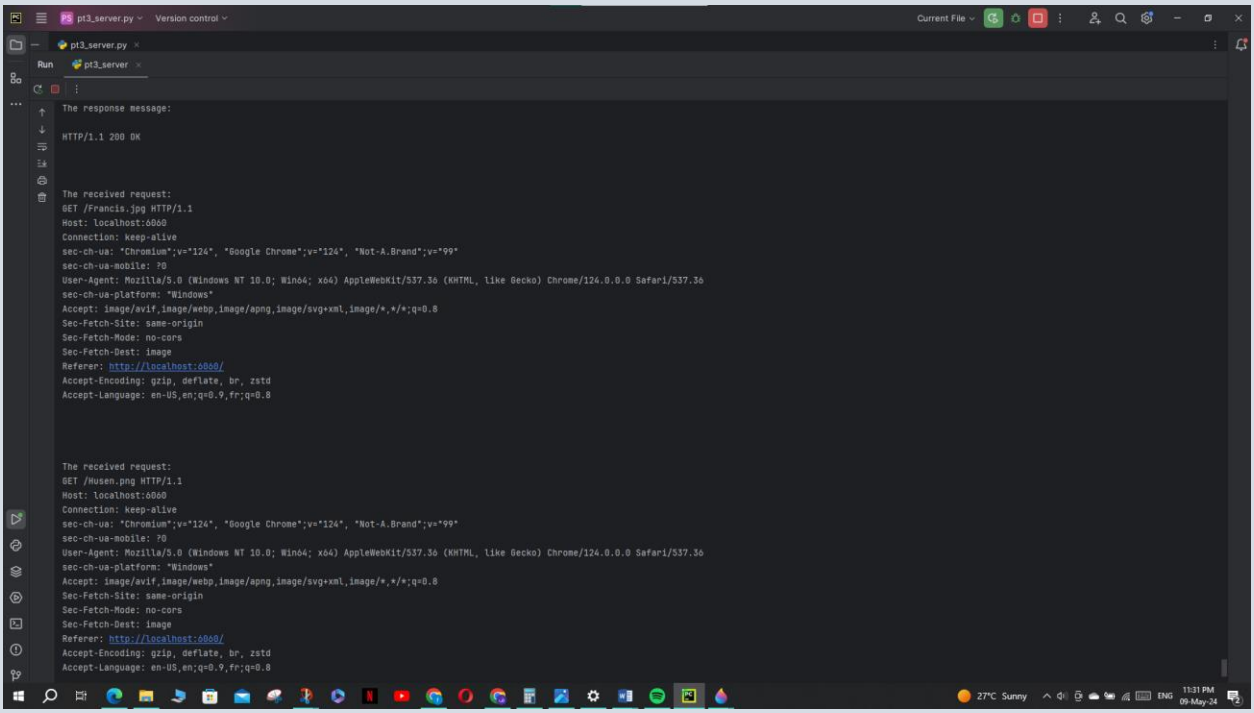
HTTP requests and the response messages

This HTTP GET request is seeking the root resource (/) from the server at localhost: 6060, and the client is capable of understanding HTML, among other formats. The headers provide additional context about the client environment and preferences.

We can also see the response messages







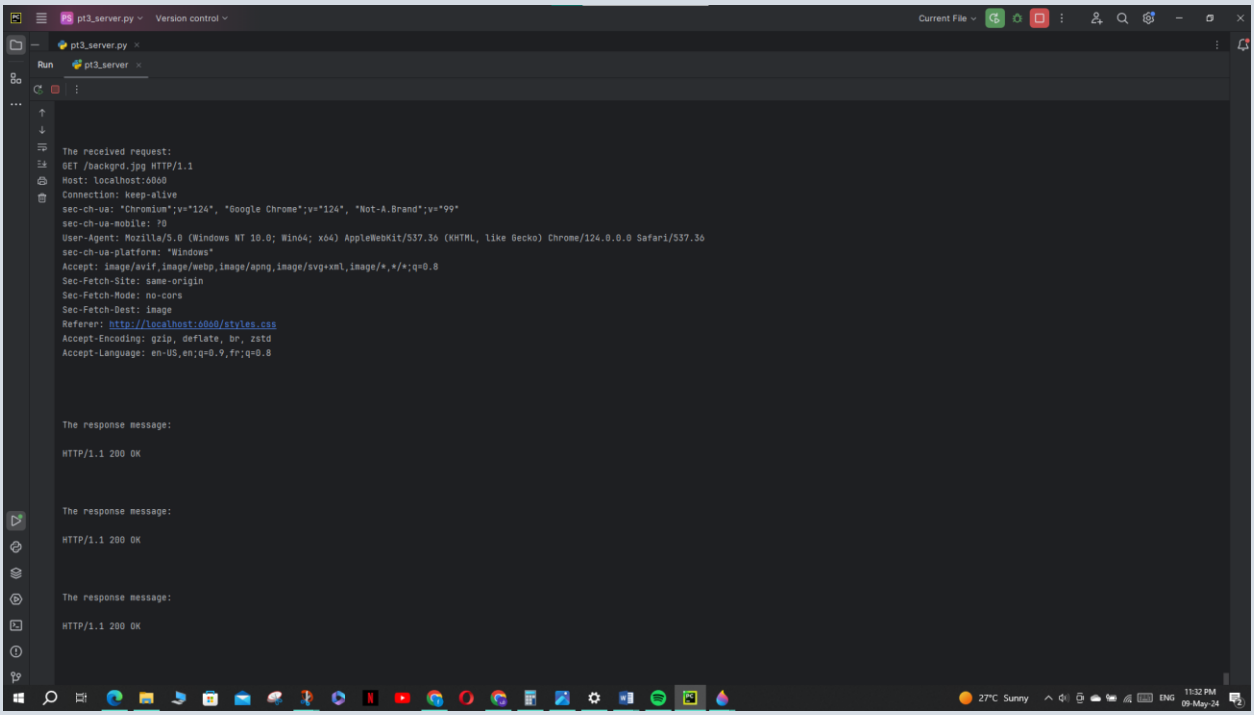


Figure 35- HTTP requests and response messages

Requesting our website from another device

Two devices are connected to the same network, the web server is running on the first device, and the website was requested from the other device, and this was the result

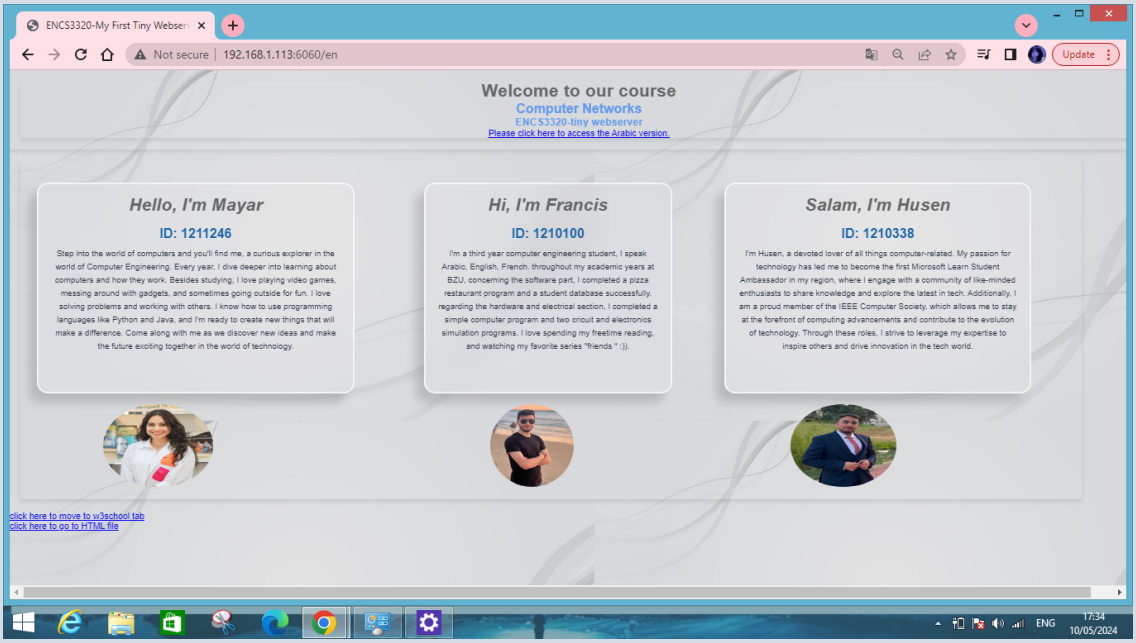
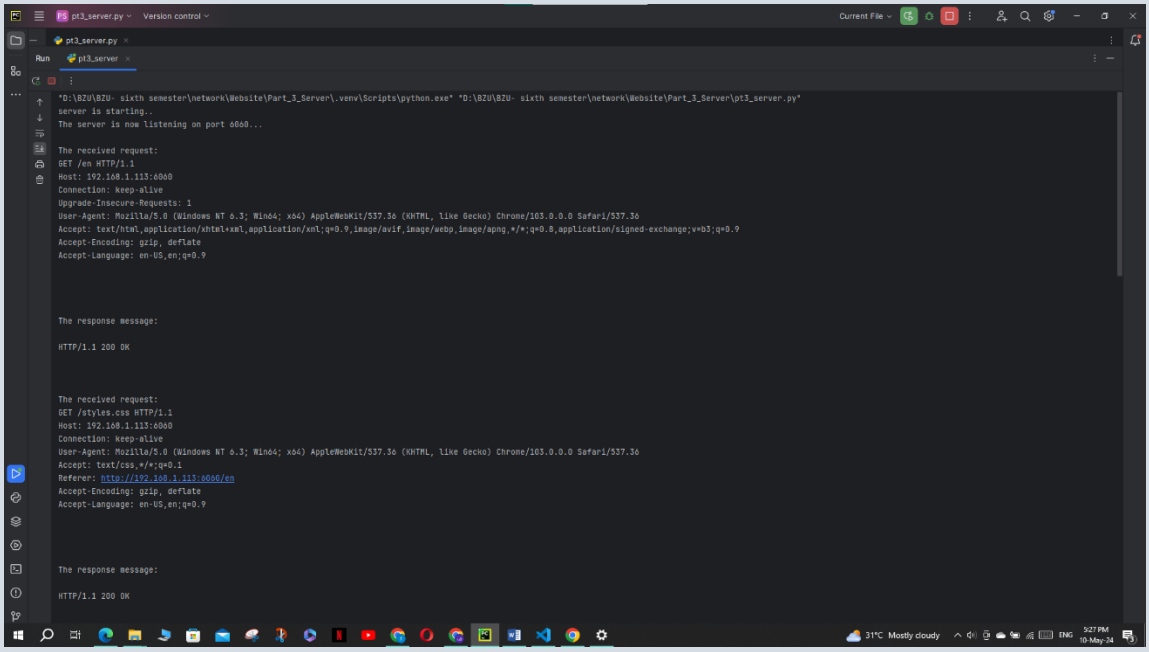
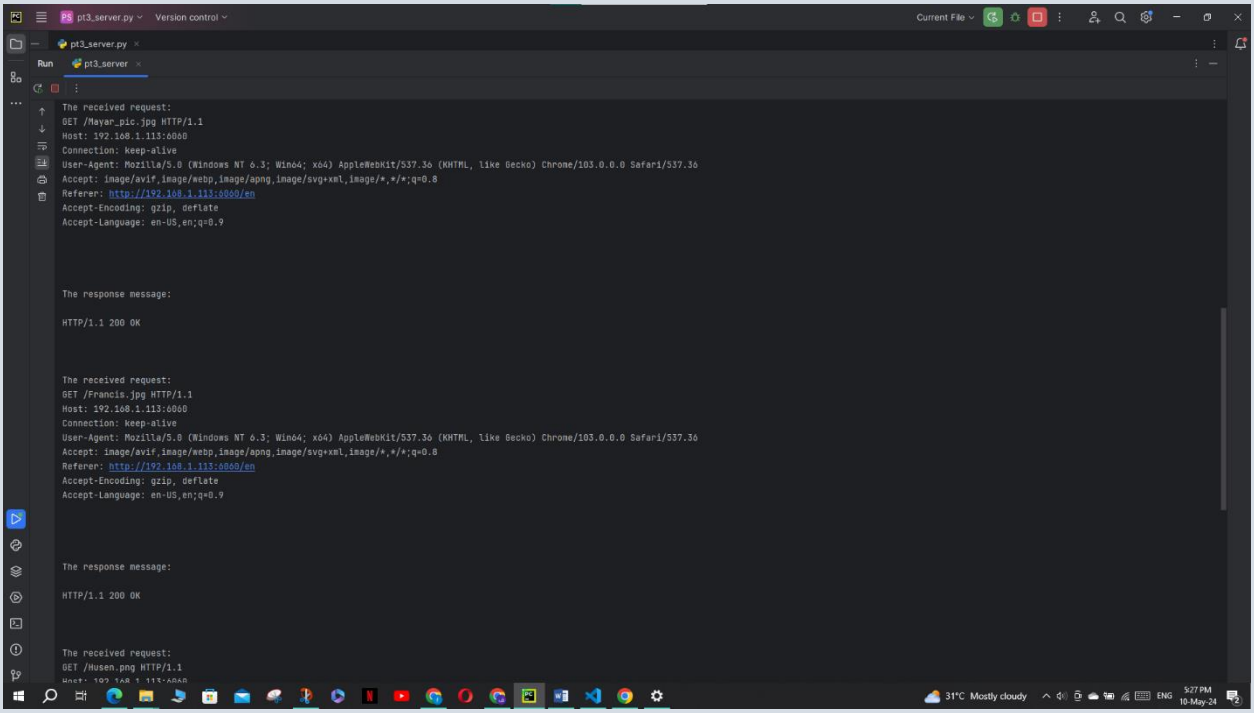


Figure 36- request from another device

The HTTP requests and response messages from the original device that the server is running on





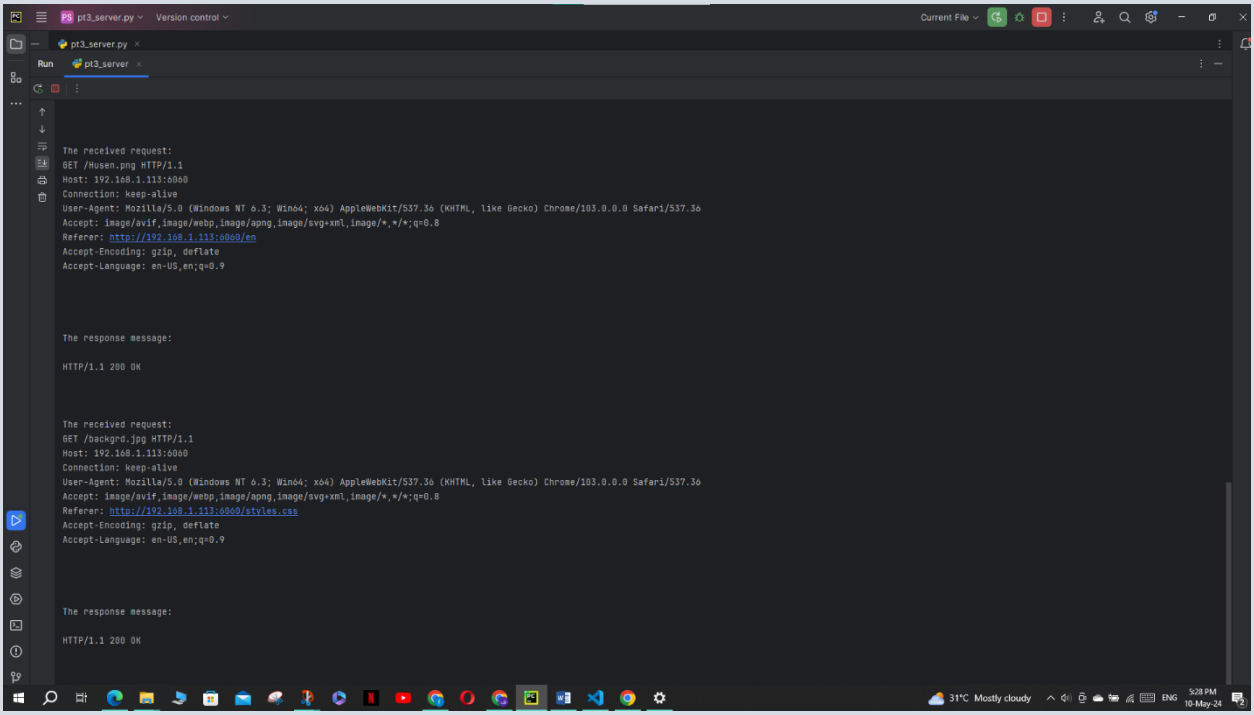


Figure 37- request and response messages of the other device

Appendix

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" />
    <link rel="stylesheet" href="styles.css" />
    <title>ENCS3320-My First Tiny Webserver</title>
  </head>
  <body>
    <div class="student" id="sec1">
        <div class="back_ground">
          <div class="overlayContent">
            <h1>Welcome to our course</h1>
            <h2 style="color: #639aec">Computer Networks</h2>
            <h3 style="color: #639aec">ENCS3320-tiny webserver</h3>
            <p>
              <a href = "main_ar.html" target = "_blank" > Please click here to access the Arabic
version. </a>


            </p>

          </div>
        </div>
      </div>
```

```
      <div class="container">
        <div class="student" id="sec2">
          <div class="content-container">
            <div class="box">

              <div class="content">
                <div class="sec3">
                  <h1>Hello, I'm Mayar</h1>
                  <h3>ID: 1211246</h3>
                  <p>
                    Step into the world of computers and you'll find me, a curious explorer in the world
of Computer Engineering.
                    Every year, I dive deeper into learning about computers and how they work. Besides
studying, I love playing video games,
                    messing around with gadgets, and sometimes going outside for fun. I love solving
problems and working with others.
                    I know how to use programming languages like Python and Java, and I'm ready to create
new things that will make a difference.
                    Come along with me as we discover new ideas and make the future exciting together in
the world of technology.
                  </p>
```

```
            </div>
            <div class="content">
              <div class="sec4">
                <img src="Mayar_pic.jpg" alt="Mayar Masalmeh" />
              </div>
            </div>
          </div>
        </div>
      </div>


      <div class="student" id="sec2">
        <div class="content-container">
          <div class="box1">

            <div class="content">
              <div class="sec3">
                <h1>Hi, I'm Francis</h1>
                <h3>ID: 1210100</h3>
                <p>
                  I'm a third year computer engineering student, I speak Arabic,
                  English, French. throughout my academic years at
                  BZU, concerning the software part, I completed a pizza
                  restaurant program and a student database successfully.
                  regarding the hardware and electrical section, I completed a
                  simple computer program and two cricuit and electronics
                  simulation programs. I love spending my freetime reading, and
                  watching my favorite series "friends " :)).
                </p>
              </div>
            </div>
            <div class="content">
              <div class="sec4">
                <img src="Francis.jpg" alt="Francis Miadi" />
              </div>
            </div>
          </div>
        </div>
      </div>
```

```
      <div class="student" id="sec3">
        <div class="content-container">
          <div class="box2">

            <div class="content">
              <div class="sec3">
                <h1>Salam, I'm Husen</h1>
                <h3>ID: 1210338</h3>
                <p>
                  I'm Husen, a devoted lover of all things computer-related.
                   My passion for technology has led me to become the first  Microsoft Learn
                    Student Ambassador in my region, where I engage with a community of like-minded
                    enthusiasts to share knowledge and explore the latest in tech. Additionally,
                    I am a proud member of the IEEE Computer Society, which allows me to stay
                    at the forefront of computing advancements and contribute to the evolution of
technology.
                     Through these roles, I strive to leverage my expertise to inspire others and drive
innovation in the tech world.
                </p>
```

```
                </div>
                <div class="content">
                  <div class="sec4">
                    <img src="Husen.png" alt="Husen Aboghosh" />
                  </div>
                </div>
              </div>
            </div>
          </div>
        </div>

    </div>
      <p>
        <a href = "https://www.w3schools.com/python/python_strings.asp" target = "_blank" >click here to
move to w3school tab  </a>
        <br>
        <a href = "myform.html" target = _"blank" > click here to go to HTML file </a>
        </p>

  </body>
</head>

  return go(f, seed, [])
}
```

Figure 38-English version HTML code

Main_ar.html

```html
<!DOCTYPE html>

<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <link rel="stylesheet" href="styles.css" />
    <title>ENCS3320-My First Tiny Webserver</title>
  </head>
  <body>
    <div class="student" id="sec1">
        <div class="back_ground">
          <div class="overlayContent">
            <h1>مرحبا بكم في مساقنا</h1>
            <h2 style="color: #639aec">شبكات الحاسوب</h2>
            <h3 style="color: #639aec">خادم ويب صغير</h3>
            <p>
              <a href = "main_en.html" target = "_blank" > الرجاء الضغط هنا للوصول إلى النسخة الإنجليزية </a>


            </p>

          </div>
        </div>
      </div>
```

```html
      <div class="container">
        <div class="student" id="sec2">
          <div class="content-container">
            <div class="box">

              <div class="content">
                <div class="sec3">
                  <h1>مرحبا، انا ميار</h1>
                  <h3>1211246 :رقمي</h3>
                  <p>
                    ادخل إلى عالم الكمبيوتر وستجدني، مستكشفًا فضوليًا في عالم هندسة الكمبيوتر.
                    في كل عام، أتعمق أكثر في التعرف على أجهزة الكمبيوتر وكيفية عملها. إلى جانب الدراسة، أحب لعب ألعاب الفيديو
                    العبث بالأدوات الذكية، والخروج أحيانًا للمتعة. أحب حل المشاكل والعمل مع الآخرين.
                    وأنا على استعداد لإنشاء أشياء جديدة من شأنها أن تحدث فرقًا. أعرف كيفية استخدام لغات البرمجة مثل Python وJava
                    تعال معي لنكتشف أفكارًا جديدة ونجعل المستقبل مثيرًا معًا في عالم التكنولوجيا.
                  </p>
                </div>
                <div class="content">
                  <div class="sec4">
                    <img src="Mayar_pic.jpg" alt="Mayar Masalmeh" />
                  </div>
                </div>
              </div>
            </div>
          </div>
        </div>
```

```
        <div class="student" id="sec2">
          <div class="content-container">
            <div class="box1">

              <div class="content">
                <div class="sec3">
                  <h1>مرحبا، انا فرنسيس</h1>
                  <h3>1210100 :رقمي</h3>
                  <p>
                    أنا طالب هندسة حاسوب سنة ثالثة, أحب دراسة اللغات
                حيث أنني أجيد العربية, الانجليزية, و الفرنسية
                خلال دراستي الجامعية اتممت الكثير من المشاريع البرمجية مثل.
                مطعم بيتزا صغير وقاعدة بيانات بسيطة للطلاب
                أيضا فان دراستي تحوي مشاريع كهرباء وبنية الحاسوب
                فأتممت مشاريع عن الدوائر الكهربائية وقمت بتصميم جهاز حاسوب بسيطة
                لعل هوايتي المفضلة هي قراءة الكتب وفي اوقات فراغي أحب أيضا
                أن أشاهد مسلسلي المفضل
                "friends" :))
                    </p>
                  </div>
                  <div class="content">
                    <div class="sec4">
                      <img src="Francis.jpg" alt="Francis Miadi" />
                    </div>
                  </div>
                </div>
              </div>
            </div>
          </div>
```

```
        <div class="student" id="sec3">
          <div class="content-container">
            <div class="box2">

              <div class="content">
                <div class="sec3">
                  <h1>سلام، انا حسين</h1>
                  <h3>1210338 :رقمي</h3>
                  <p>أنا حسين، عاشق مخلص لكل ما يتعلق بالكمبيوتر.
                    لقد قادني شغفي بالتكنولوجيا إلى أن أصبح أول برنامج Microsoft Learn
                    سفير الطلاب في منطقتي، حيث أتفاعل مع مجتمع من ذوي التفكير المماثل
                    المتحمسين لتبادل المعرفة واستكشاف أحدث التقنيات. بالإضافة إلى ذلك،
                    للكمبيوتر، مما يسمح لي بالبقاء IEEE أنا عضو فخور في جمعية
                    في طليعة التطورات في مجال الحوسبة والمساهمة في تطور التكنولوجيا.
                    ومن خلال هذه الأدوار، أسعى جاهداً للاستفادة من خبرتي لإلهام الآخرين ودفع الابتكار في عالم التكنولوجيا.
                  </p>
                </div>
                <div class="content">
                  <div class="sec4">
                    <img src="Husen.png" alt="Husen Aboghosh" />
                  </div>
                </div>
              </div>
            </div>
          </div>
        </div>

    </div>
      <p>
        <a href = "https://www.w3schools.com/python/python_strings.asp" target = "_blank" >انقر هنا للانتقال إلى
علامة التبويب w3school </a>
        <br>
        <a href = "myform.html" target = _"blank" > انقر هنا للذهاب الى ملف HTML</a>
        </p>

  </body>
</head>
```

Figure 39- Arabic version HTML code

Error.html

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Error 404</title>

</head>
<body>

    <h1>Error 404</h1>
    <p style=" color: rgb(255,0,0)">The file is not found </p>
    <p><strong>Mayar Masalmeh : 1211246</strong> </p>
    <p><strong>Francis Miadi : 1210100</strong></p>
    <p><strong>Husen Aboghosh : 1210338</strong></p>
    <p>Client IP and Port:[Client IP and Port]</p>
</body>
</html>
```

Figure 40-Error page HTML code

Myform.html

```html
<!DOCTYPE html>
<html lang="en">
    <head>
<meta charset="UTF-8">
<meta http-equiv="X-UA-compatible" content="IE=edge">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Simple Form</title>
</head>
<body>
    <form action="http://localhost:6060" method="POST"  target="_blank" >

        <label for="imageName">Enter Image Name:</label>
        <input type="text" id="imageName" name="imageName" required>
        <button type="submit">getImage</button>


    </form>
</body>
</html>
```

Figure 41-Myform page HTML code

CSS

```css
*{
  margin:0;
  padding:0;
  font-family: sans-serif;

}
body {
    margin: 0;
    padding: 0;
    background-image : url(backgrd.jpg);
    box-sizing: border-box;
    font-family: sans-serif;
  }
  .container{
    display: flex;
    justify-content: space-between;
    align-items: center;
    width: 95%;
    margin: auto;
  }
  #sec1 {
    text-align: center;
    width : 100%;
    height: 100vh;
   box-shadow: 0 4px 8px rgba(235, 232, 232, 0.75);
    background-size: cover;
    background-position: center;
  }
   #sec1 {
    width: 100%;
    height: 20%;
    position: relative;
    overflow: hidden;
    box-shadow: 0 4px 8px rgba(29, 29, 29, 0.1);
}
#sec1 .back_ground {
    width: 100%;
    height: 100%;
    position: relative;
    overflow: hidden;
    box-shadow: 0 4px 8px rgba(29, 29, 29, 0.1);

}
```

```css
.overlayContent h1{
  color: #666;
}
body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
  }

  .container, .container1, .container2 {
    margin: 20px;
    box-shadow: 0 4px 8px rgba(0,0,0,0.1);


  }

  .student {
    display: flex;
    flex-direction: column;
    align-items: center;
    padding: 20px;
  }



  .overlayContent h1, .overlayContent h2, .overlayContent h3 {
    margin: 0;

  }


  .content .sec3 {

    flex: 1;  /* Ensures that all boxes take equal width */
    border: 3px solid rgba(255,255,255,0.7);
    border-radius: 20px;
    padding: 20px;
    margin: 10px;
    backdrop-filter: blur(25px);
    box-shadow: -15px 17px 17px rgba(10,10,10,0.15);
    background-color: rgba(255,255,255,0.1);
    display: flex;
    flex-direction: column;
    align-items: center;
    justify-content: center;
  }
```

```css
  .content {
    margin-top: 20px;
    box-sizing: border-box;


  }

  .sec3 h1, .sec3 h3 {
    color: #333;
  }

  .sec3 p {
    color: #666;
    line-height: 1.6;
  }




.blue-text {
    color: rgb(87, 87, 240);
    font-family: 'Brush Script MT', cursive;
}

 .container1 .box1,
.container2 .box2
{
margin:auto;
height : 100%;
width : 100%;
padding: 0 20px;
background-color: rgba(255,255,255,0.1);
z-index: 10;
border: 2px solid rgba(255,255,255,0.7);
border-radius: 20px;
display: flex;
backdrop-filter: blur(25px);
box-shadow: -15px 17px 17px rgba(10,10,10,0.15);

}

.box img, .box1 img, .box2 img {
  width: 100px;
  height: 150px;
  border-radius: 100%;
  margin-left: 130px;

}
```

```css
.box h1, .box1 h1, .box2 h1,
.box h3, .box1 h3, .box2 h3,
.box p, .box1 p, .box2 p {
    text-align: center;
    color:black;
}
.content .sec4 img, .content .sec4 img{
  width: 30%;
}

.student .Mayar_pic,
.student .Francis_pic,
.student .husen_pic{
font-size: 35px;
font-weight: 700;
text-decoration: none;
color:aliceblue;
margin-left: 50px;
}

.section .Mayar_pic::first-letter,
.section .Francis_pic::first-letter,
.section .husen_pic::first-letter{
color:rgb(84, 135, 139);
font-size: 35px;
}
.content .sec3 h1,
.content1 .sec3 h1,
.content2 .sec3 h1{
margin-bottom: 20px;
font-size: 2em;
font-weight: 700;
color:#666;
font-style: italic;
}

.content-container,
.content1-container1,
.content2-container2 {
  display:flex ;
  width: 100%;
}
.content .sec3,
.content1 .sec3,
.content2 .sec3{
  display:flex ;
position: relative;
width: 80%;
}
```

```css
.content .sec3 h3,
.content1 .sec3 h3{
margin-bottom: 10px;
font-size: 24px;
color:rgb(26, 101, 167);
}

.content .sec3 p,
.content1 .sec3 p,
.content2 .sec3 p{
margin-bottom: 50px;
font-size: 15px;
color:rgb(35, 45, 64);
}

.content .sec4,
.content1 .sec4{
width : 100%;
position: relative;

}




*{
  margin:0;
  padding:0;
  font-family: sans-serif;

}
body {
    margin: 0;
    padding: 0;
    background-image : url(backgrd.jpg);
    box-sizing: border-box;
    font-family: sans-serif;
  }
  .container{
    display: flex;
    justify-content: space-between;
    align-items: center;
    width: 95%;
    margin: auto;
  }
```

```
  #sec1 {
    text-align: center;
    width : 100%;
    height: 100vh;
   box-shadow: 0 4px 8px rgba(235, 232, 232, 0.75);
    background-size: cover;
    background-position: center;
  }
   #sec1 {
    width: 100%;
    height: 20%;
    position: relative;
    overflow: hidden;
    box-shadow: 0 4px 8px rgba(29, 29, 29, 0.1);
}
#sec1 .back_ground {
    width: 100%;
    height: 100%;
    position: relative;
    overflow: hidden;
    box-shadow: 0 4px 8px rgba(29, 29, 29, 0.1);

}
.overlayContent h1{
  color: #666;
}
body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
  }

  .container, .container1, .container2 {
    margin: 20px;
    box-shadow: 0 4px 8px rgba(0,0,0,0.1);


  }
```

```css
.student {
  display: flex;
  flex-direction: column;
  align-items: center;
  padding: 20px;
}



.overlayContent h1, .overlayContent h2, .overlayContent h3 {
  margin: 0;

}


.content .sec3 {

  flex: 1;  /* Ensures that all boxes take equal width */
  border: 3px solid rgba(255,255,255,0.7);
  border-radius: 20px;
  padding: 20px;
  margin: 10px;
  backdrop-filter: blur(25px);
  box-shadow: -15px 17px 17px rgba(10,10,10,0.15);
  background-color: rgba(255,255,255,0.1);
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
}

.content {
  margin-top: 20px;
  box-sizing: border-box;


}

.sec3 h1, .sec3 h3 {
  color: #333;
}
```

```css
  .sec3 p {
    color: #666;
    line-height: 1.6;
  }




.blue-text {
    color: rgb(87, 87, 240);
    font-family: 'Brush Script MT', cursive;
}

 .container1 .box1,
.container2 .box2
{
margin:auto;
height : 100%;
width : 100%;
padding: 0 20px;
background-color: rgba(255,255,255,0.1);
z-index: 10;
border: 2px solid rgba(255,255,255,0.7);
border-radius: 20px;
display: flex;
backdrop-filter: blur(25px);
box-shadow: -15px 17px 17px rgba(10,10,10,0.15);

}

.box img, .box1 img, .box2 img {
  width: 100px;
  height: 150px;
  border-radius: 100%;
  margin-left: 130px;

}
```

```css
.box h1, .box1 h1, .box2 h1,
.box h3, .box1 h3, .box2 h3,
.box p, .box1 p, .box2 p {
    text-align: center;
    color:black;
}
.content .sec4 img, .content .sec4 img{
  width: 30%;
}

.student .Mayar_pic,
.student .Francis_pic,
.student .husen_pic{
font-size: 35px;
font-weight: 700;
text-decoration: none;
color:aliceblue;
margin-left: 50px;
}

.section .Mayar_pic::first-letter,
.section .Francis_pic::first-letter,
.section .husen_pic::first-letter{
color:rgb(84, 135, 139);
font-size: 35px;
}
.content .sec3 h1,
.content1 .sec3 h1,
.content2 .sec3 h1{
margin-bottom: 20px;
font-size: 2em;
font-weight: 700;
color:#666;
font-style: italic;
}

.content-container,
.content1-container1,
.content2-container2 {
  display:flex ;
  width: 100%;
}
.content .sec3,
.content1 .sec3,
.content2 .sec3{
  display:flex ;
position: relative;
width: 80%;
}
```

```css
.content .sec3 h3,
.content1 .sec3 h3{
margin-bottom: 10px;
font-size: 24px;
color:rgb(26, 101, 167);
}

.content .sec3 p,
.content1 .sec3 p,
.content2 .sec3 p{
margin-bottom: 50px;
font-size: 15px;
color:rgb(35, 45, 64);
}

.content .sec4,
.content1 .sec4{
width : 100%;
position: relative;

}
```

Figure 42-CSS code

Server-client – part2

```python
import socket
import datetime
import threading

# Server settings:
server_port = 5051
buffer_size = 1024
broadcast_ip = '192.168.1.255'

# to get and save the user names :
client_name = None
if client_name is None:
    first_name = input("Enter your first name: ")
    last_name = input("Enter your last name: ")
    client_name = f"{first_name} {last_name}"

clients = {}  # <--this a  Dictionary to keep track of client addresses and last messages
received_messages = []  # <--  a list to store the messages

# Create a UDP socket
server_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)  # <-- to  creates a UDP socket.
server_socket.setsockopt(socket.SOL_SOCKET, socket.SO_BROADCAST, 1)  # <-- to allow broadcasting.
server_socket.bind(('', server_port))  # <-- to binds the socket to the specified port.

print("UDP Server listening on port", server_port)


#Prints a list of received messages, excluding messages from the client itself.
def display_received_messages():
    if received_messages:
        print("\nList of received messages:")
        for idx, (sender, time, msg) in enumerate(received_messages, start=1):
            if sender != client_name:
                print(f"-{idx} received a message from {sender} at {time}")
    else:
        print("\nNo messages to display.")


def rec():
    # Receive message from any client
    message, client_address = server_socket.recvfrom(buffer_size)
    decoded_message = message.decode()
    current_time = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")

    # Update client's last message and timestamp
    sender_info = decoded_message.split(';')
    if len(sender_info) == 3:
        sender = f"{sender_info[0]} {sender_info[1]}"
        if client_name != sender:
            print(f"\nReceived message from {sender} at {current_time}")
        received_messages.append((sender, current_time, sender_info[2]))
        rec()

# creating a new thread to run the rec function, allowing the server to receive messages concurrently
with other tasks:
t1 = threading.Thread(target=rec)
t1.start()


#to show the options menu:
while True:

    print("\nOptions:\n1. Send another message\n2. Show all messages")
    option = input("Select an option (1/2): ")

    if option == '1':

        message = input("Enter your message: ")
        send_message = f"{first_name};{last_name};{message}"
        server_socket.sendto(send_message.encode(), (broadcast_ip, server_port)) #<-- to send it via
UDP to the broadcast address.

    elif option == '2':
        display_received_messages()
        choice = input("Enter line number followed by 'D' to display the message (e.g., 1D): ")
        if choice.endswith('D') and choice[:-1].isdigit():
            index = int(choice[:-1]) - 1
            if 0 <= index < len(received_messages):
                print(
                    f"Message from {received_messages[index][0]} at {received_messages[index][1]}:
{received_messages[index][2]}")
            else:
                prin"Invalid line number.")
    else:
        print("Invalid option. Please select 1 or 2.")
```

Figure 43- server-client code -part 2

Server – part 3

```python
import socket
import threading
import os


PORTNUM = 6060

SERVER_IP = '127.0.0.1'
#SERVER_IP = '192.168.1.113'
ADDRESS = (SERVER_IP, PORTNUM)
server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)  # OUR SOCKET
server.bind(ADDRESS)  # we connected our socket to ADDRESS , so anything that hit this address will hit
our socket



def handle_requests(request, client_addr):


    content_types = {

        ".html": "text/html",
        ".css": "text/css",
        ".png": "image/png",
        ".jpg": "image/jpeg"

    }


    redirects = {

        "/so": "https://stackoverflow.com/",
        "/itc": "https://itc.birzeit.edu/"

    }

    path_mapping = {

        "/": "main_en.html",
        "/index.html": "main_en.html",
        "/en": "main_en.html",
        "/main_en.html": "main_en.html",
        "/ar": "main_ar.html"

    }
```

```
    if request in redirects:
        redirect_to = redirects[request]
        response = (f"HTTP/1.1 307 Temporary Redirect\r\n"
                    f"Location: {redirect_to}\r\n\r\n")

        return response.encode()


    if request in path_mapping:
        file_path = path_mapping[request]
    else:
        file_path = request[1:]

    if os.path.isfile(file_path):
        _, fextension = os.path.splitext(file_path)  # to store the file extension
        ctype = content_types.get(fextension, "application//octet-stream")  # to store the content type
        with open(file_path, "rb") as file:
            file_content = file.read()
        return f"HTTP/1.1 200 OK\r\nContent-Type: {ctype}\r\n\r\n".encode() + file_content

    return f"HTTP/1.1 404 Not Found\r\nContent-Type: text/html\r\n\r\n".encode() +
error_not_found(client_addr)


def get_image_name(request):
    headers, _, body = request.partition("\r\n\r\n")
    image_name = {kv.split('=')[0]: kv.split('=')[1] for kv in body.split('&')}
    image_name = str(image_name)
    image_name = image_name.split(':')[1]
    image_name= image_name.replace("'","")
    image_name = image_name.replace("{", "")
    image_name = image_name.replace("}", "")
    image_name = image_name.replace(" ", "")
    return image_name
```

```
def get_image(name,client_addr):



     types = {


        ".png": "image/png",
        ".jpg": "image/jpeg"

     }


     path = os.path.join("images",name)



     if os.path.exists(path):
        _, extension = os.path.splitext(path)  # to store the file extension
        ctype = types.get(extension, "application//octet-stream")  # to store the content type
        with open(path,'rb') as image_location:
            image = image_location.read()

        return f"HTTP/1.1 200 OK\r\nContent-Type: {ctype}\r\n\r\n".encode() + image


     return f"HTTP/1.1 404 Not Found\r\nContent-Type: text/html\r\n\r\n".encode() +
error_not_found(client_addr)
```

```python
def handle_client_request(client_socket):

    request = client_socket.recv(2048).decode('utf-8')
    print(f"The received request:\n{request}\n\n")
    # print('\n\n')
    line = request.split('\r\n')[0]
    # print(line)  # the first line
    line1 = line.split(" ")[0]
    if line1 == "GET":
        line = request.split('\r\n')[0]
        # print(line)  # the first line
        line1 = line.split(" ")[0]
        # print(line1)  # the method
        _, path, _ = line.split(' ')
        # print(path)  # object name
        response = handle_requests(path, client_socket.getpeername())
        client_socket.sendall(response)
        print("The response message: \n")
        #  print()
        line2 = response.split(b'\r\n')[0].decode('utf-8')
        print(line2)
        print('\n\n')
        client_socket.close()

    if line1 == "POST":
        # path = path.replace("/","")
        path = get_image_name(request)

        response = get_image(path,client_socket.getpeername())

        client_socket.sendall(response)
        print("The response message: \n")
        #  print()
        line2 = response.split(b'\r\n')[0].decode('utf-8')
        print(line2)
        print('\n\n')
        client_socket.close()
```

```python
def error_not_found(client_addr):

    with open("Error.html", "r") as file:
        error_file_content = file.read()
    error_file_content = error_file_content.replace("[Client IP and Port]", f"{client_addr[0]}:
{client_addr[1]}")

    return error_file_content.encode()


def start():
    server.listen()
    print(f'The server is now listening on port {PORTNUM}...')
    print()
    while True:
        client_socket, client_addr = server.accept()
        thread = threading.Thread(target=handle_client_request, args=(client_socket, ))
        thread.start()

if __name__== "__main__":
    print("server is starting..")
start()
```

Figure 44-Web server python code