Faculty of Engineering & Technology

Electrical & Computer Engineering Department

ENCS3320-Computer Networks

Project#1

Prepared by:

Dana Ismail **1200006**

Jana Herzallah **1201139**

Lana Badwan **1200071**

Instructor: Dr. Abdalkarim Awad

Section: 1

Date: 31Aug .2023

Contents

Table of figure:

Part 1:

General definition:

- **Ping :** check server accessibility and measure packet transmission time.
- **Tracert:** Trace packet routes and determine network paths and hops**.**
- **Nslookup:** Query the DNS server for domain and IP information.
- **Telnet**: Enables text-based remote access to network devices.

Commands & Discussion:

ping :

- command → www.amazon.de :

  The "ping" command is used to check the accessibility and round-trip travel time to the "www.amazon.de" hostname. The command sent four packets to the IP address **162.219.224.163**, which corresponds to the server "www-amazon-de.customer.fastly.net". The round-trip packet transmission time ranged from 75 ms to 123 ms, with an average of 87 ms. All packages have been received successfully without loss.

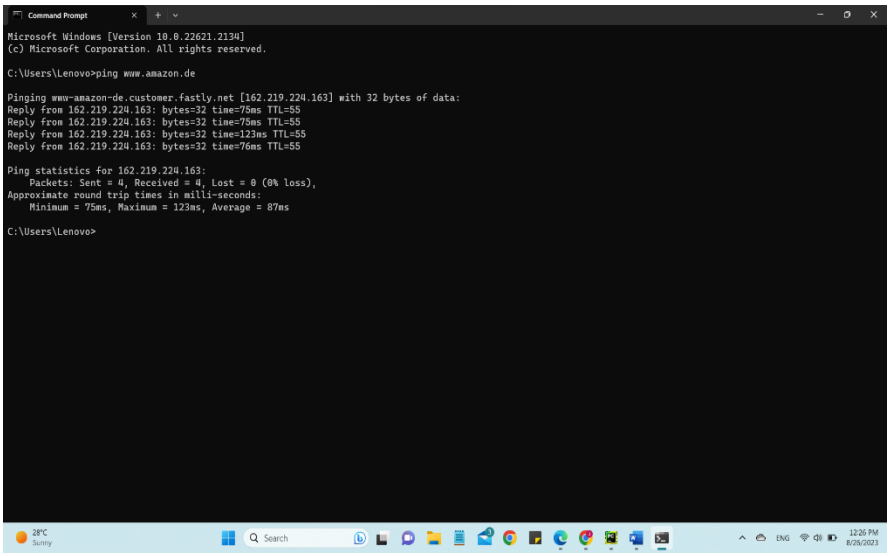


Figure 1 ping amazon.de

- **Discussion:**

In the command prompt output provided, the user 'pings' the hostname 'www.amazon.de'. The system successfully resolved the hostname to the IP address **162.219.224.163**, which is associated with the server "**www-amazon-de.customer.fastly.net**". The following lines show the response received from the server for packets sent. Each response includes data bytes, round trip time in milliseconds (ms) and a **TTL** (Time to Live) value, which indicates how many hops the packet can make before being dropped. The final statistics show that all 4 packets are sent and received without loss, resulting in a loss rate of 0%. The "Approximate round-trip time" section provides the minimum, maximum and average round-trip times in milliseconds for received packets. These metrics give an indication of network latency to the target host. Overall, successful responses and relatively consistent round-trip times indicate a stable and responsive connection to the specified Amazon server.

Tracert :

- command → www.amazon.de :

The "tracert" command, which stands for "trace Route", is used to reveal the route taken by data packets on an IP network from the source system to the specified destination. Specifically, when applied to "tracert www.amazon.de", the command initiates an activity that tracks the "**www.amazon.de**" hostname, revealing the sequence of network devices in between, known as "hops", which packets encounter during their journey to the target server. These hops represent routers or network elements that facilitate data transmission. Command output shows the path, round-trip time for each hop, and potential network anomalies such as high latency or packet loss. This data allows network administrators to assess route efficiency, troubleshoot connectivity issues, and identify areas where delays may occur, thereby optimizing overall network performance.

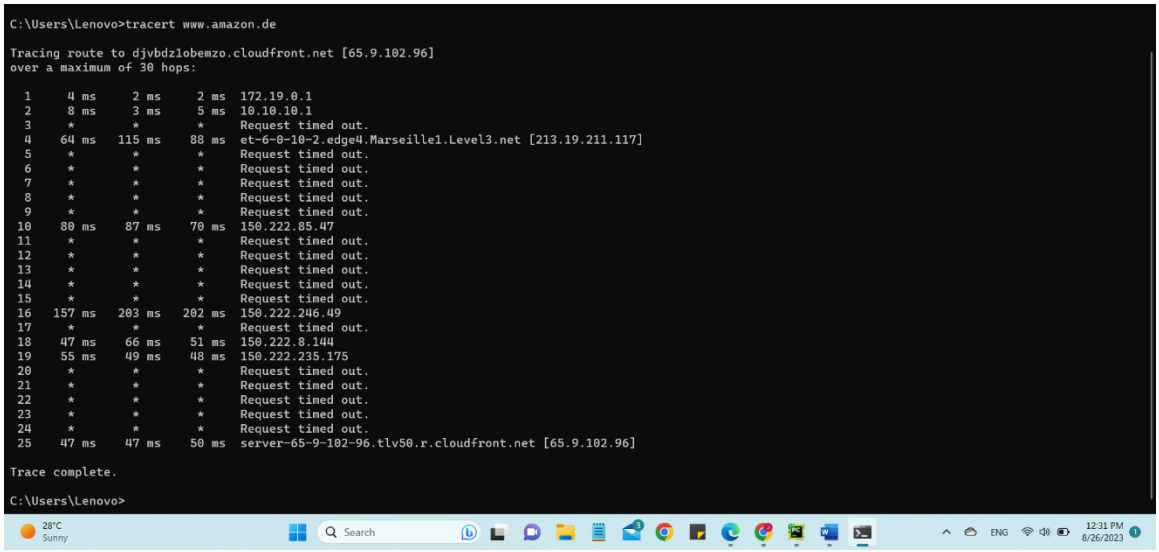


Figure 2 tracert amazon.de

Discussion :

The provided output shows route tracking to the domain "djvbdzlobenzo.cloudfront.net" with IP address 65.9.182.96. This process involves mapping the network path from the source to the destination host, allowing up to 30 hops. The results show the round trip time (in milliseconds) for each hop along with the intermediate IP address or router name. However, some requests experience a timeout, indicating no response within the specified time. Additionally, some hops are marked as "Request timed out", which indicates a lack of response. Despite these interruptions, the tracking journey continues until it reaches its final destination, revealing the path to the cloudfront.net server. In general, the operation represents the network path taken and the responsiveness of the intermediate nodes between the source and destination servers.

Nslookup :

- command → www.amazon.de :

The "nslookup" command was used to query DNS information for the hostname "www.amazon.de". It provides details about the associated IP address and other DNS records. This command is valuable for diagnosing DNS-related issues and verifying domain configurations.

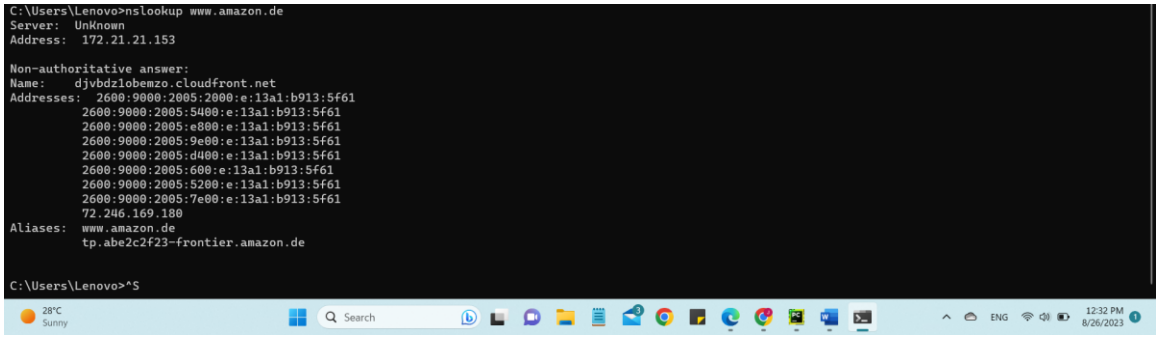


Figure 3 nslookup amazon.de

Discussion :

The "nslookup" command is run to query the DNS information of "www.amazon.de". The results show that the answers are not authoritative, including the IPv6 and IPv4 addresses associated with the domain name. The domain "djvbdzlobenzo.cloudfront.net" is also displayed along with its IPv6 address. Additionally, the IPv4 address "72.246.169.180" is listed with the aliases "www.amazon.de" and "tp.abe2c2f23-frontier.amazon.de". This information represents the various addresses and aliases connected to the queried domain and its association with the cloudfront.net server.

Part 2 :

A brief explanation for the code:

The server listens on a specified port and processes incoming requests from the client. When it receives the request, it parses the request stream, extracts the requested path, and then processes the request accordingly. The server supports multiple response types, including serving HTML files, plain text files, images (PNG and JPEG), and CSS files. In

addition, the server has an error handling mechanism to generate an appropriate error response when the requested resource is not found.

In the create_response function, various conditions are checked against the requested path to determine the appropriate response to generate. For example, if the requested path matches known file paths, such as "/index.html" or "/laptops.txt", the server will read the contents of those files and generate only response with "200 OK" status appropriate content. type. If the file is not found, an error response is generated using the generate_error_response function, which reads the contents of the "error.html" template and replaces the placeholder with the client port.

The main function is the entry point of the server. It configures the server socket, binds it to the specified host and port, and starts listening for incoming connections. When the client connects, the server receives the request, processes it with the create_response function, and returns a response. If the requested path is invalid or the request is malformed, an error response will be generated. The server also includes redirects for specific paths with the status code "Temporary Redirect 307".

Using socket programming, implement a simple but a complete web server that is listening on port **12345.**

main_en.html :

Certainly! The server is designed to respond to specific paths with the "main_en.html" file, accompanied by the appropriate content type. When the program is run using "localhost:12345", "localhost:12345/en", or any variation of these paths, the server will automatically serve the "main_en.html" content with the content type set to "text/html; charset=UTF-8". This functionality ensures that the designated HTML file is displayed correctly for these specific cases.

Case 1 → "/"

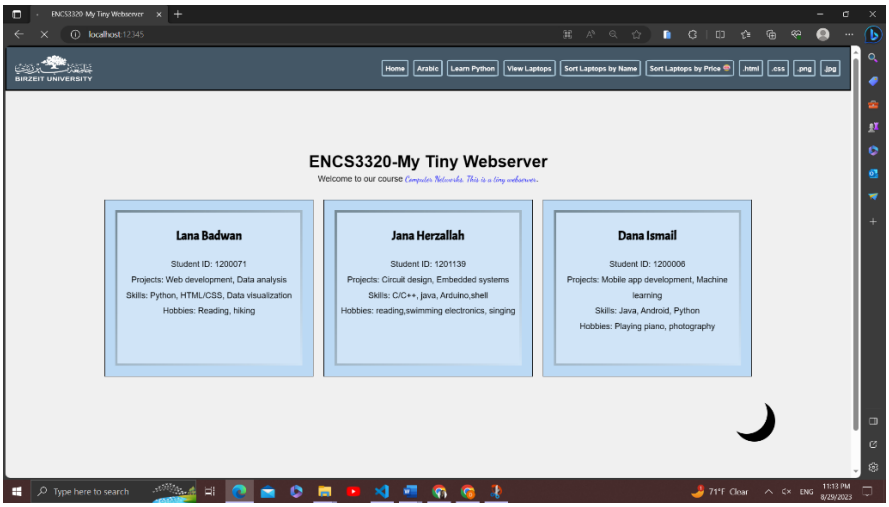


Figure 4 localhost:12345/

Case 2 → "/en"

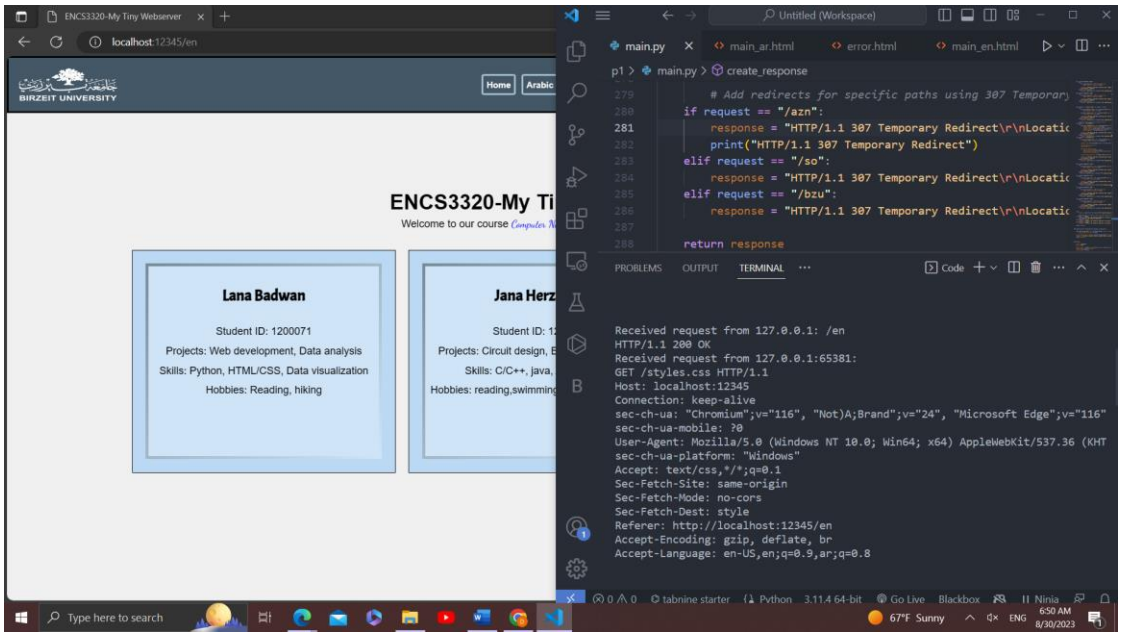


Figure 5 HTTP Request (/en)

The log shows two HTTP requests from the client to the IP address 127.0.0.1. The first request was "/en", received a successful response "200 OK" with "main_en.html". The second request is for "styles.css", with headers showing customer details, user-agent, and accepting compressed responses. The "Referer" header suggests a source of navigation from the "/en" page.

Python code :

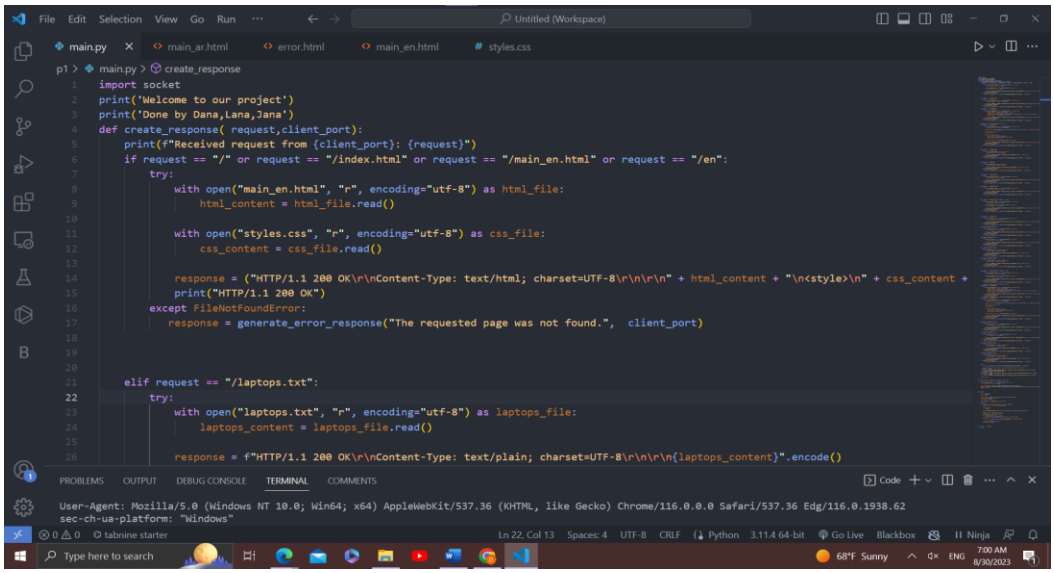


Figure 6 python Request part : (main_en.html)

The code block provided checks if the incoming request is for specific paths ("/", "index.html", "main_en.html", or "/en"). It is intended to respond with content from "main_en.html" along with associated CSS styles. If the requested files are found, it will generate an HTTP response with a status of "200 OK", set the appropriate content type and encoding. In case a file is missing, it will trigger a “404 Not Found” error response using the generate_error_response function. The response status is printed as "HTTP/1.1 200 OK" to indicate successful request processing.

The "main_en.html" file serves as the webpage's core content, featuring a title "ENCS3320-My Tiny Webserver," a welcoming message with a blue-styled phrase, group members' names and IDs, information about their projects, skills, and hobbies, all presented within visually appealing CSS-styled boxes. The page contains images in both ".jpg" and ".png" formats, links to local HTML files, and an external link to a Python-related webpage. The layout is organized and enhanced using CSS, ensuring a visually pleasing presentation.

W3school:

```
learnPythonButton.addEventListener("click", () => {
    window.location.href = "https://www.w3schools.com/python/python_intro.asp";
});
```

Figure 8 W3school button click

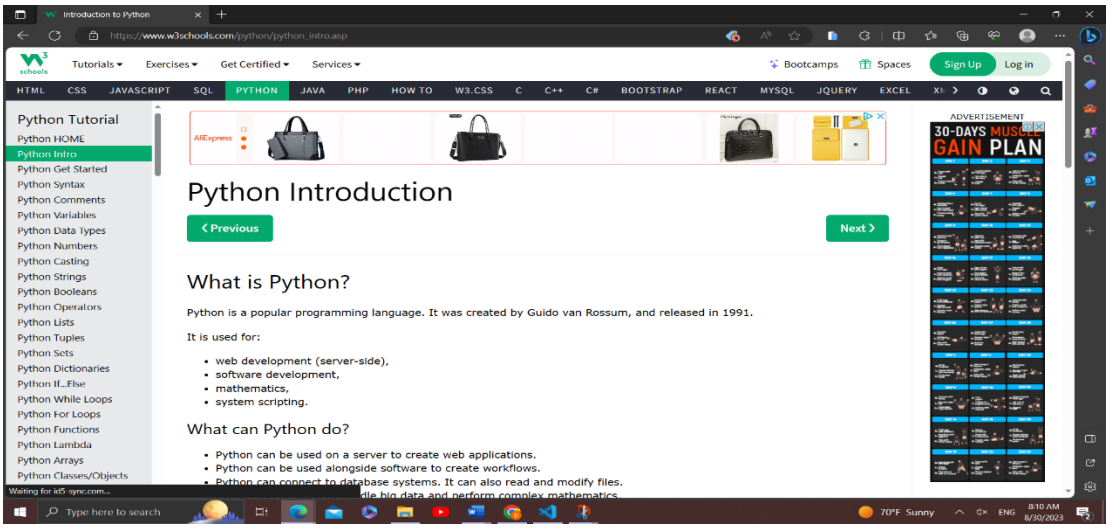


Figure 7 python website

CSS File :

This code block handles the request for the "styles.css" file. It reads the CSS content, constructs an HTTP response with status 200 OK and content type "text/css," and sends the CSS content back to the client. If the CSS file is not found, it generates an error response

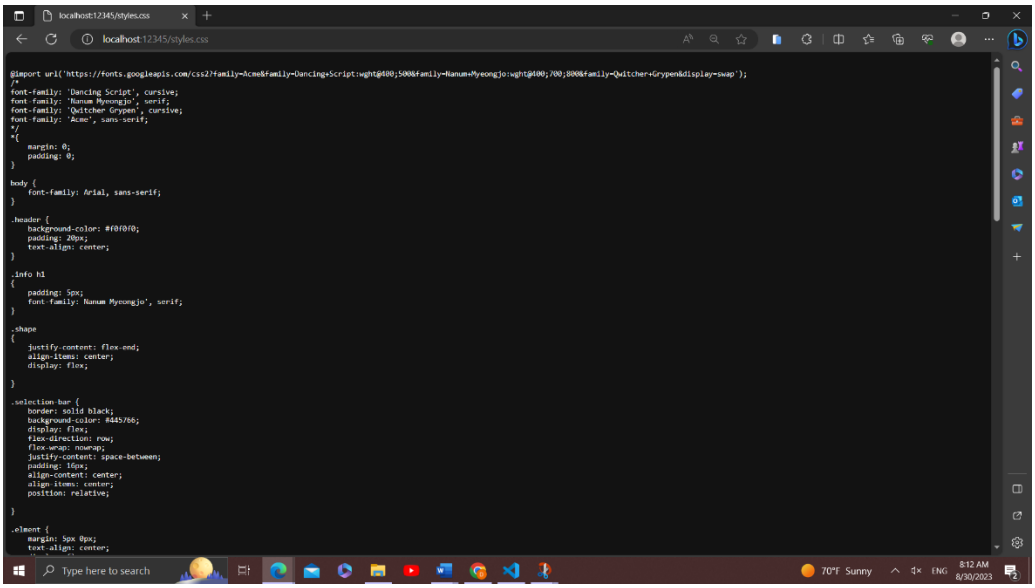
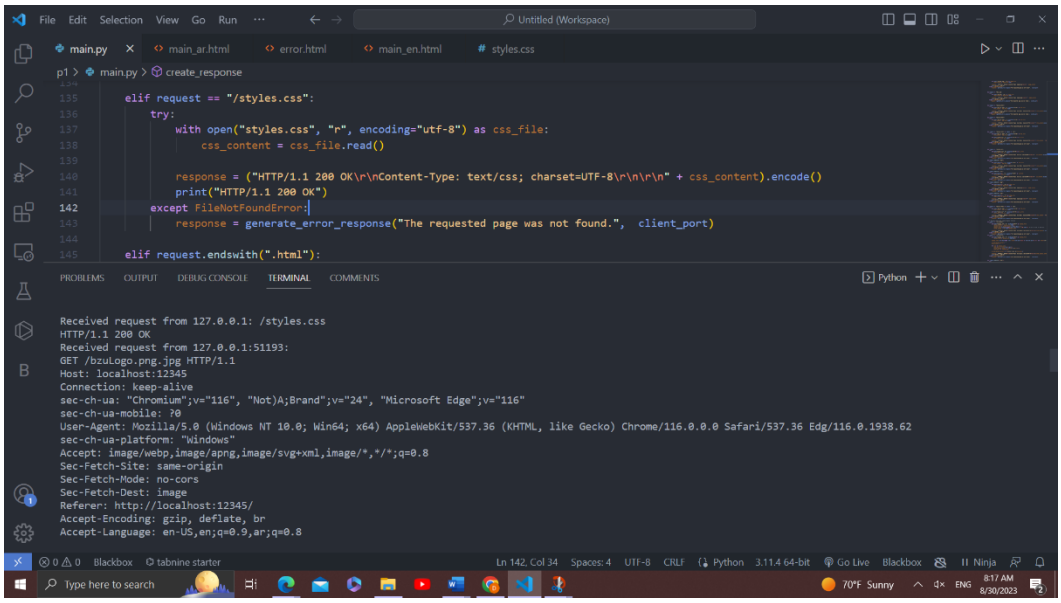
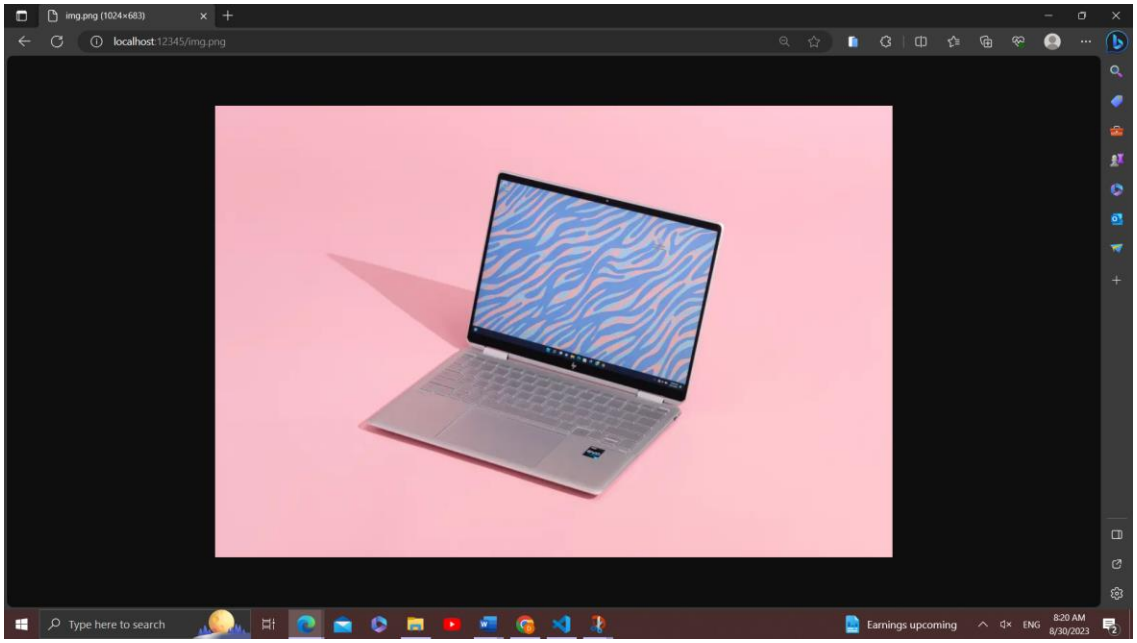
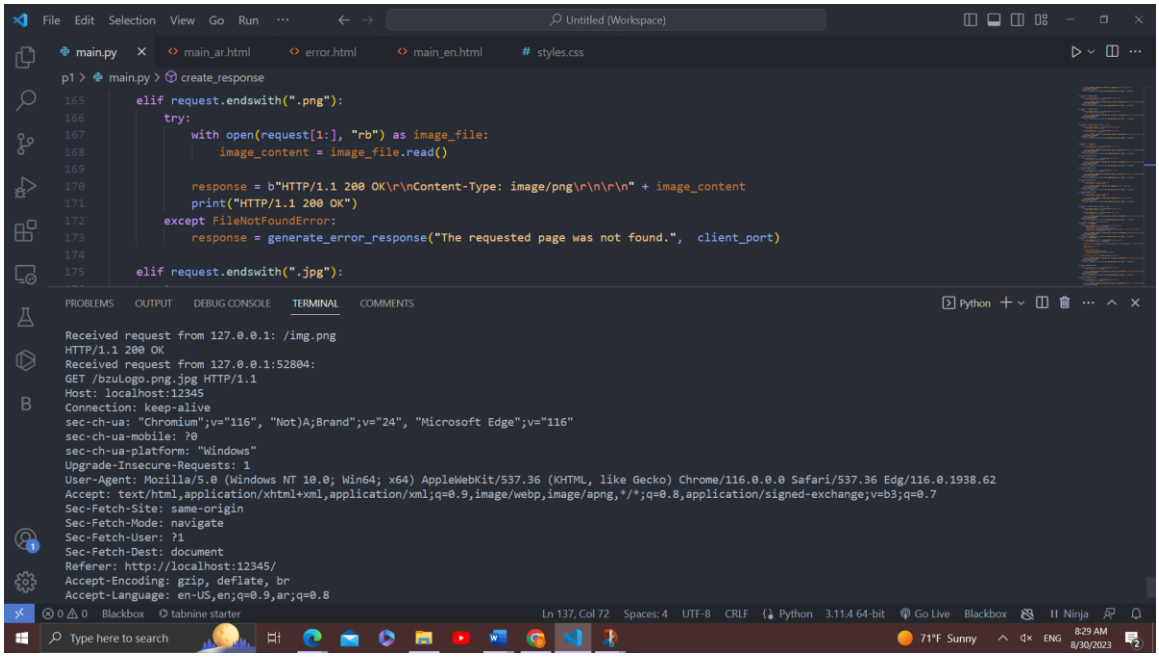


Figure 9 Css file

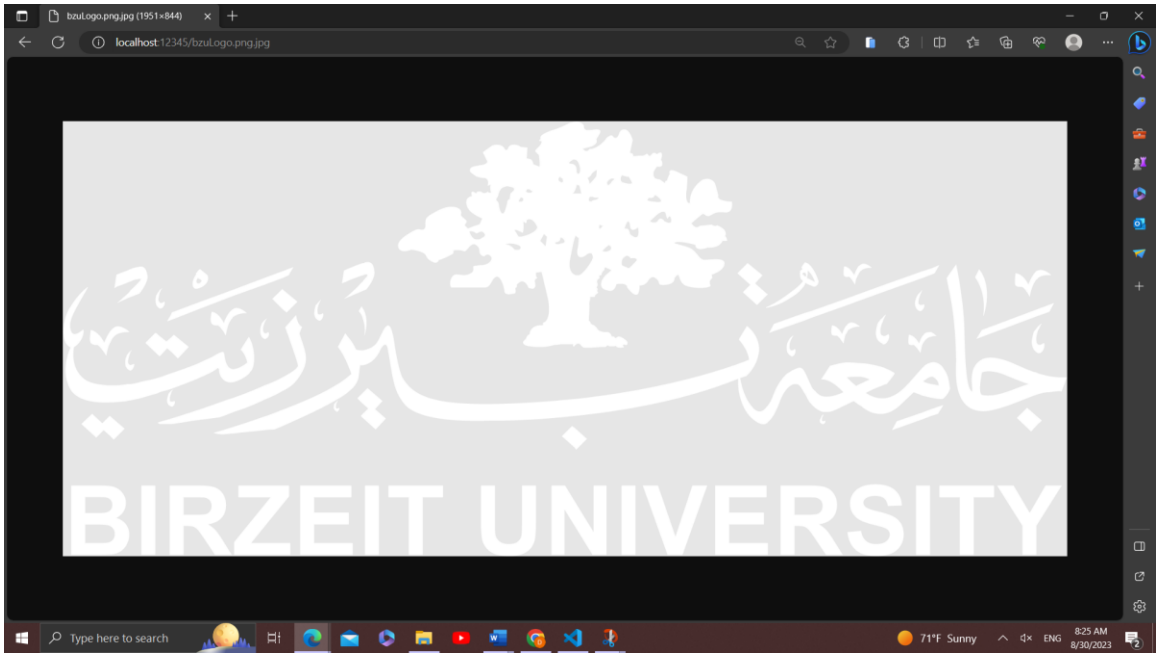


Png image :


img.png (1024×683)
localhost:12345/img.png
Type here to search
Earnings upcoming
ENG
8:20 AM
8/30/2023


File Edit Selection View Go Run
Untitled (Workspace)
main.py
main_ar.html
error.html
main_en.html
styles.css
p1 > main.py > create_response
elif request.endswith(".png"):
try:
with open(request[1:], "rb") as image_file:
image_content = image_file.read()
response = b"HTTP/1.1 200 OK\r\nContent-Type: image/png\r\n\r\n" + image_content
print("HTTP/1.1 200 OK")
except FileNotFoundError:
response = generate_error_response("The requested page was not found.", client_port)
elif request.endswith(".jpg"):
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL COMMENTS
Python
Received request from 127.0.0.1: /img.png
HTTP/1.1 200 OK
Received request from 127.0.0.1:52804:
GET /bzuLogo.png.jpg HTTP/1.1
Host: localhost:12345
Connection: keep-alive
sec-ch-ua: "Chromium";v="116", "Not)A;Brand";v="24", "Microsoft Edge";v="116"
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/116.0.0.0 Safari/537.36 Edg/116.0.1938.62
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
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: http://localhost:12345/
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9,ar;q=0.8
Blackbox
tabnine starter
Ln 137, Col 72
Spaces: 4
UTF-8
CRLF
Python
3.11.4 64-bit
Go Live
Blackbox
Ninja
Type here to search
71°F Sunny
ENG
8:29 AM
8/30/2023

Jpj image



Html request

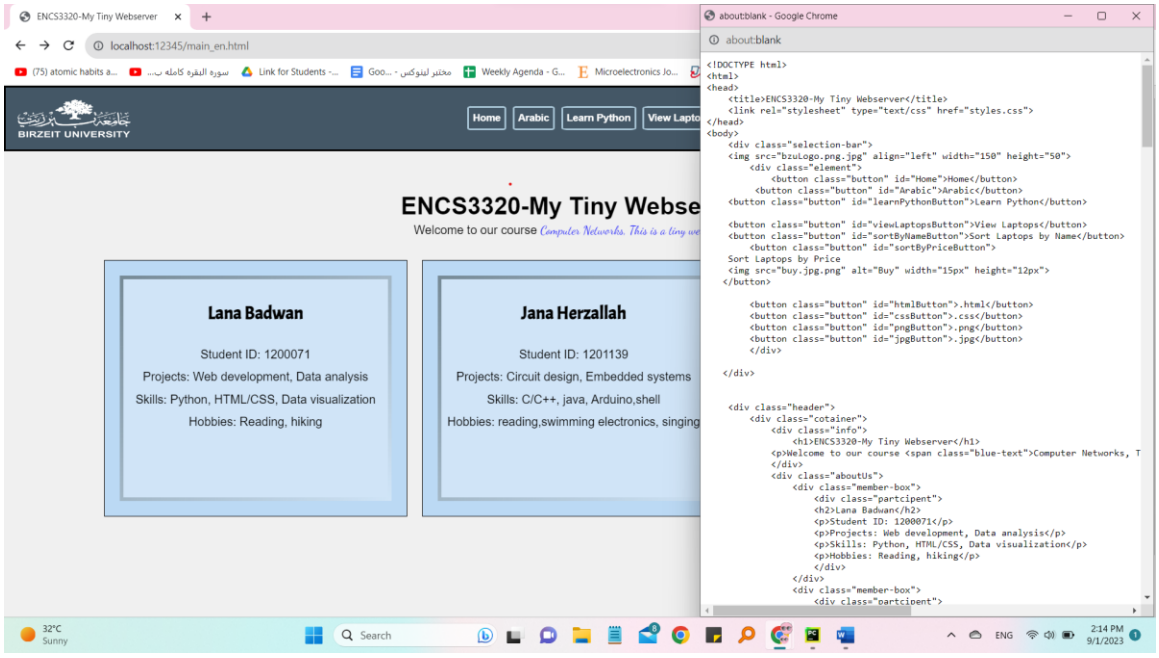


Figure 10: html request

main_ar.html:

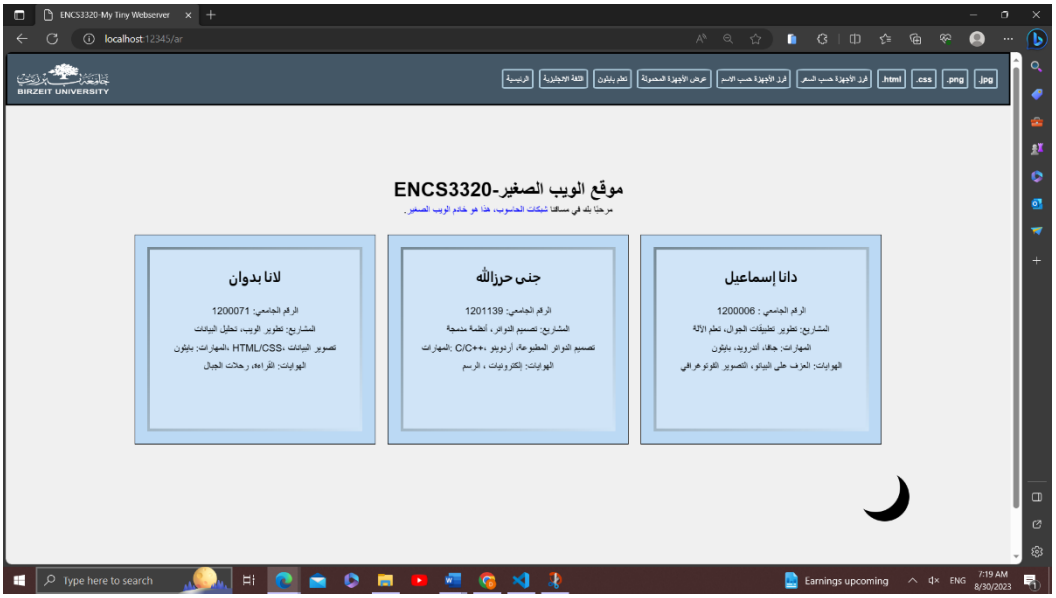


Figure 11 main_ar.html page

When the server receives a request for "/ar", it responds by providing the file "main_ar.html". This file is the Arabic version of "main_en.html" and contains similar content but adapted for the Arabic audience. It includes the title, welcome message, names and IDs of team members, information about their projects, skills and interests, all presented in visually distinct boxes using CSS style. This page also contains images, links to local HTML files, and external links to Python related web pages, all presented in Arabic. The layout and presentation are designed to provide an engaging experience for Arabic-speaking users.

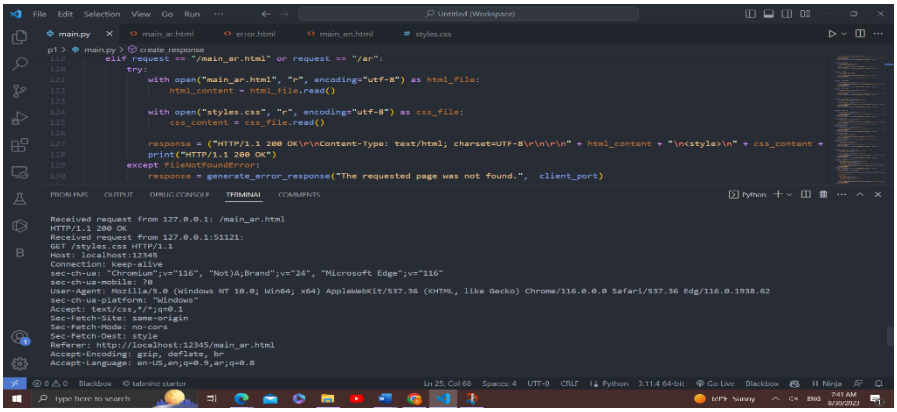


Figure 12 HTTP Request + python code

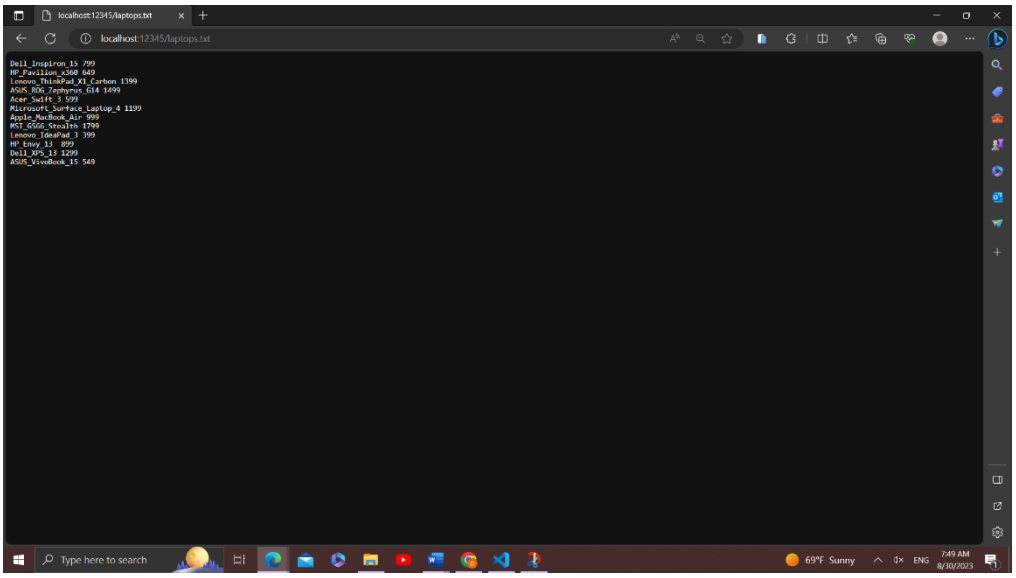


Figure 13 laptop.txt

Laptops Text File:

The script handles the situation when the client requests the resource "/laptops.txt". If the file "laptops.txt" exists, it is read and its contents sent in response with the status line "HTTP/1.1 200 OK". The content type is set to "text/plain" with UTF-8 encoding. The contents of the file are split into lines, injected into the response body, and returned to the client. If the file is not found, an error response will be generated using the 'generate_error_response' function. This mechanism allows the server to deliver the content of "laptops.txt" to the client in plain text format when a specific resource is requested.

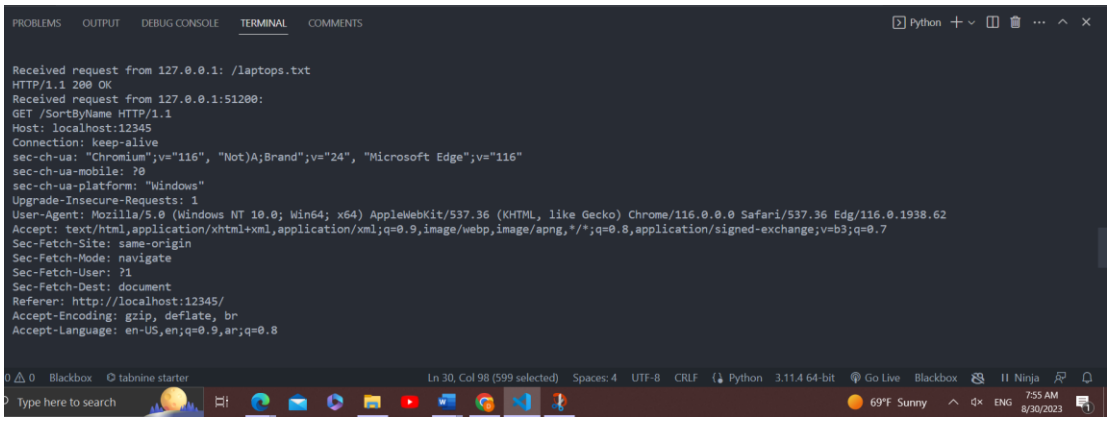


Figure 14 HTTP request for text file

Sort By Name :

This code section handles a request to sort a list of laptops by their names in alphabetical order. It reads the content of the "laptops.txt" file, sorts the lines based on laptop names (ignoring case), and generates a response with a status of 200 OK. The response contains the sorted list of laptops in plain text format. If the file is not found, an error response is generated. The printed message confirms the successful handling of the request.

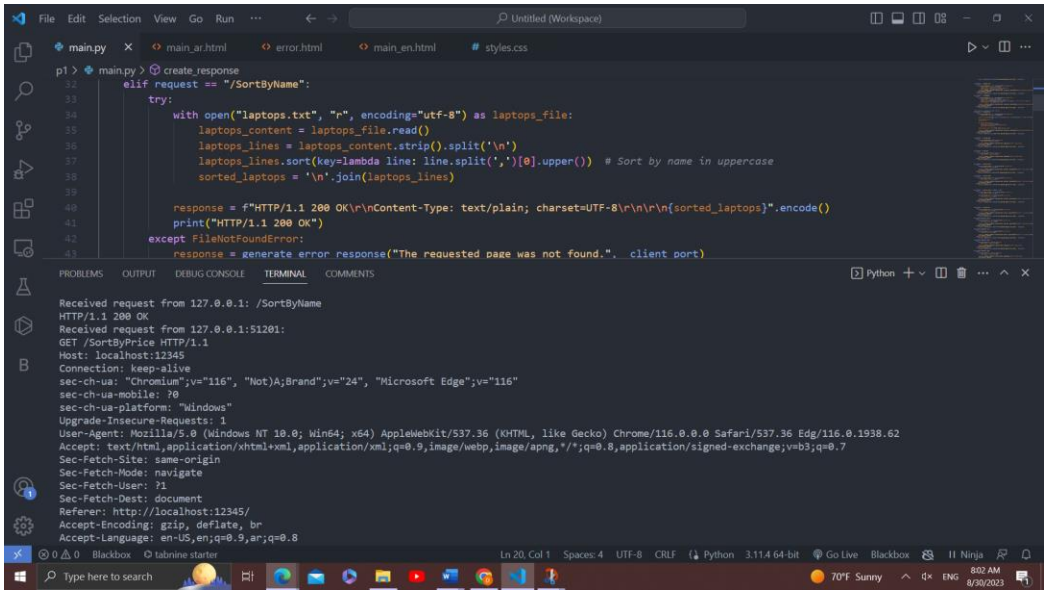


Figure 15 HTTP Request + python code (SortByName)

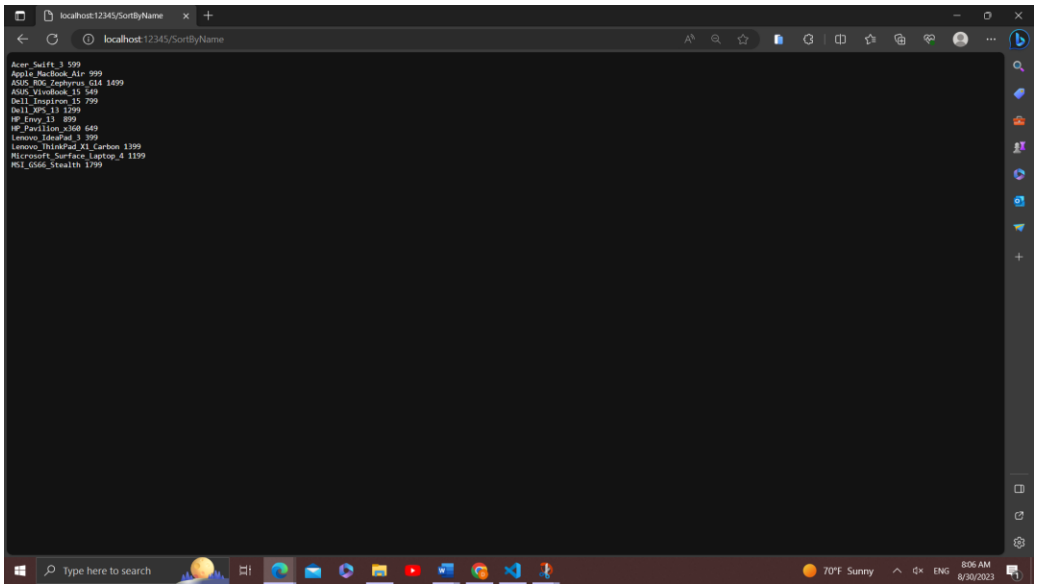


Figure 16 Sort By Name result

Sort By Price ;

This snippet handles the request to sort the list of laptops based on their price. It reads the contents of the "laptops.txt" file, sorts the lines by ascending price, and generates a response with a status of 200 OK. The response contains a sorted list of laptops with their prices and a total price calculation, all in plain text format. If the file is not found, an error response will be generated. The message is printed indicating that the request has been processed successfully.

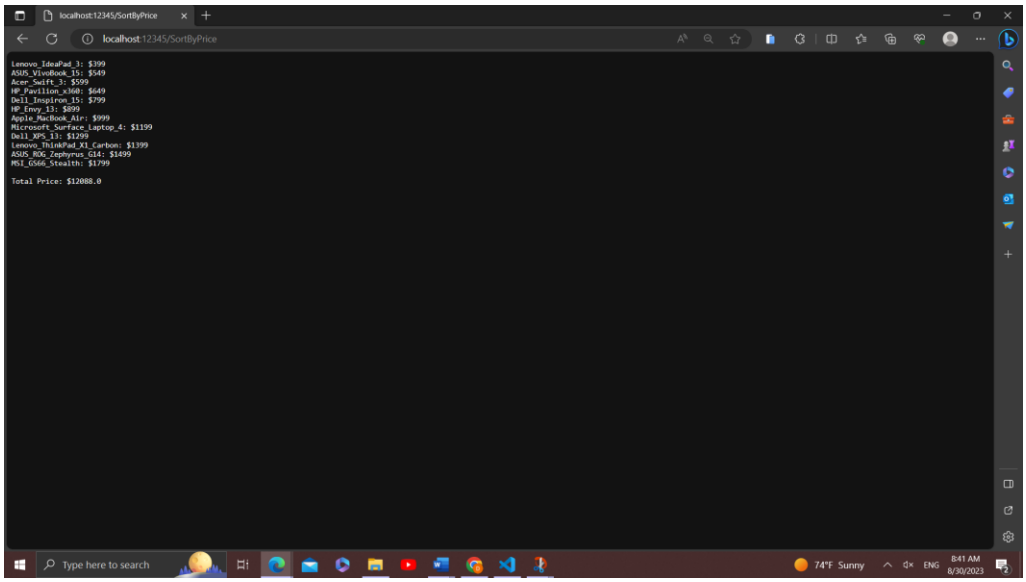


Figure 17 Sort by price res

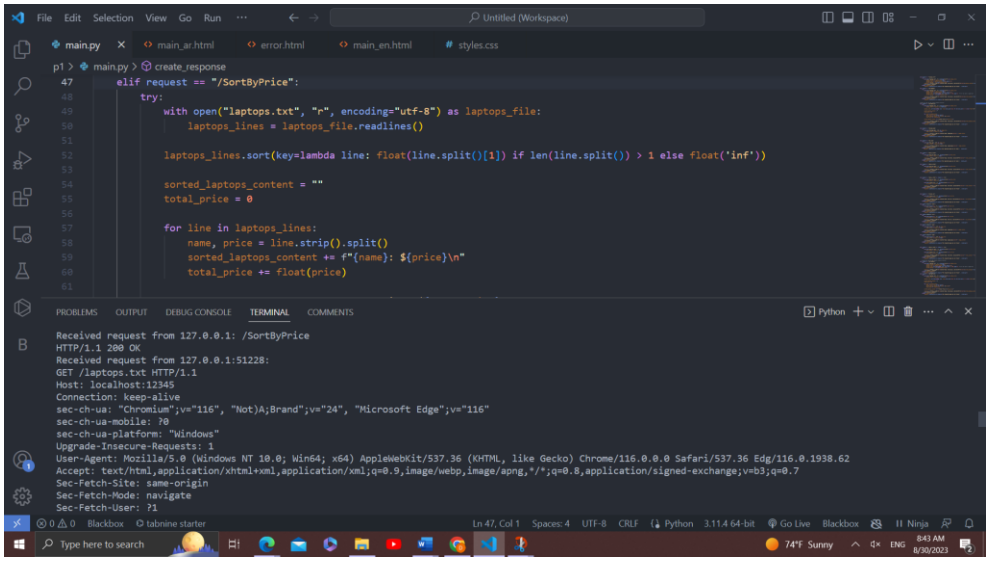


Figure 18 Http Request + python (Sort by name)

Using the status code 307 Temporary Redirect to redirect the following:

If the request is /azn then redirect to amazon website

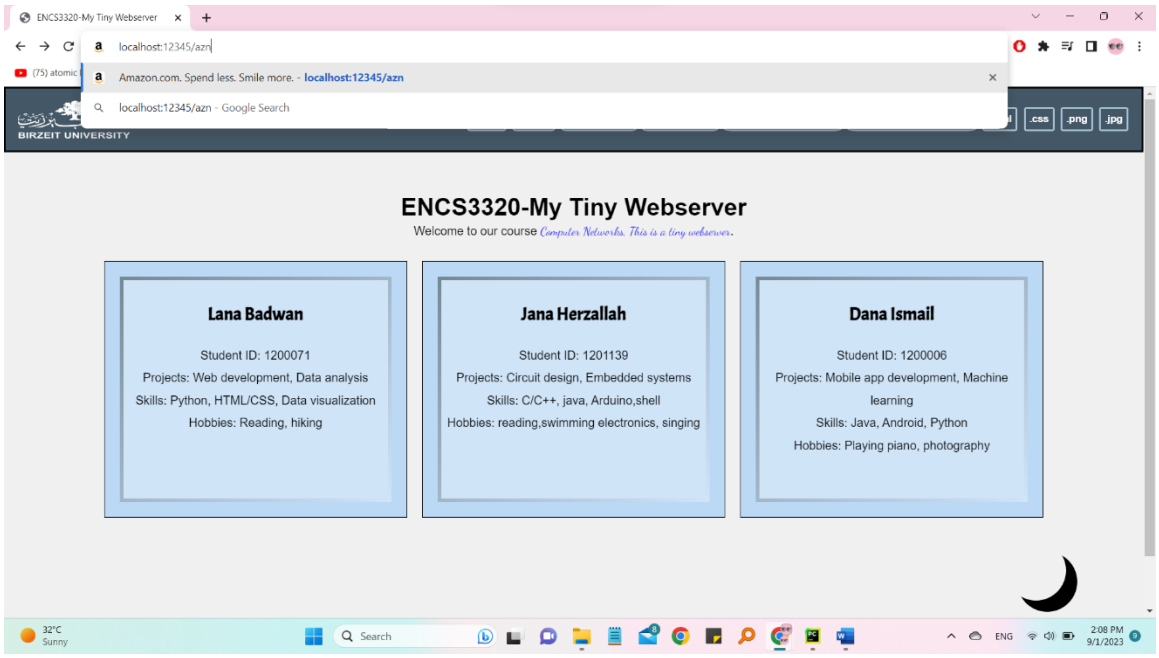
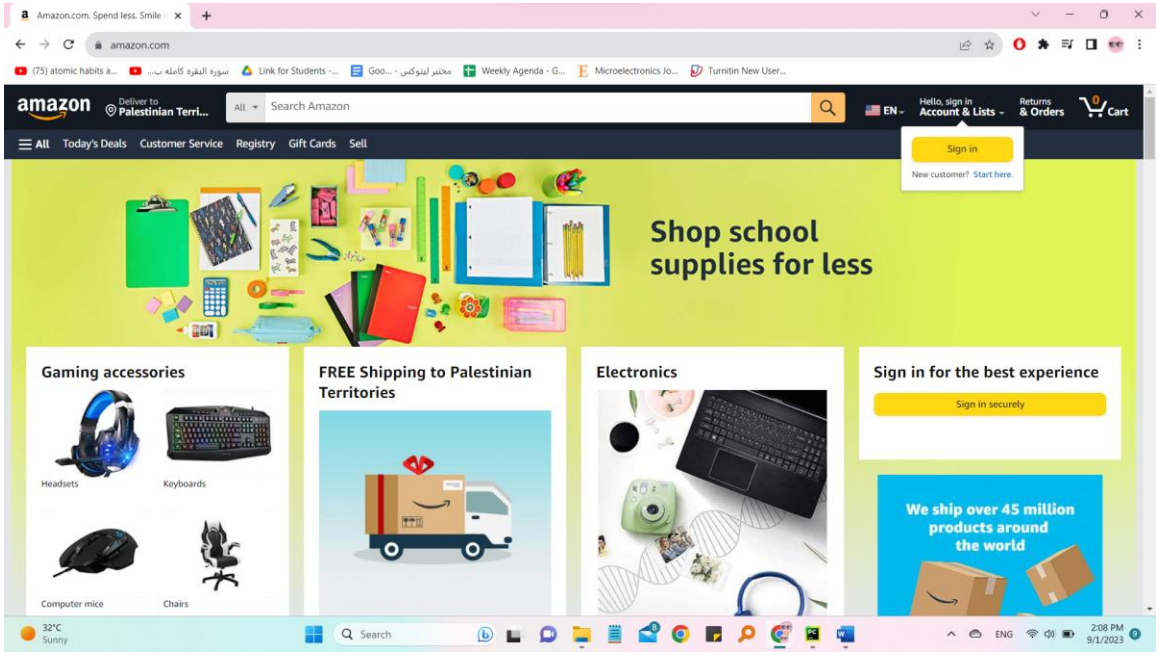


Figure 19: /azn redirection



If the request is /so then redirect to stackoverflow.com website

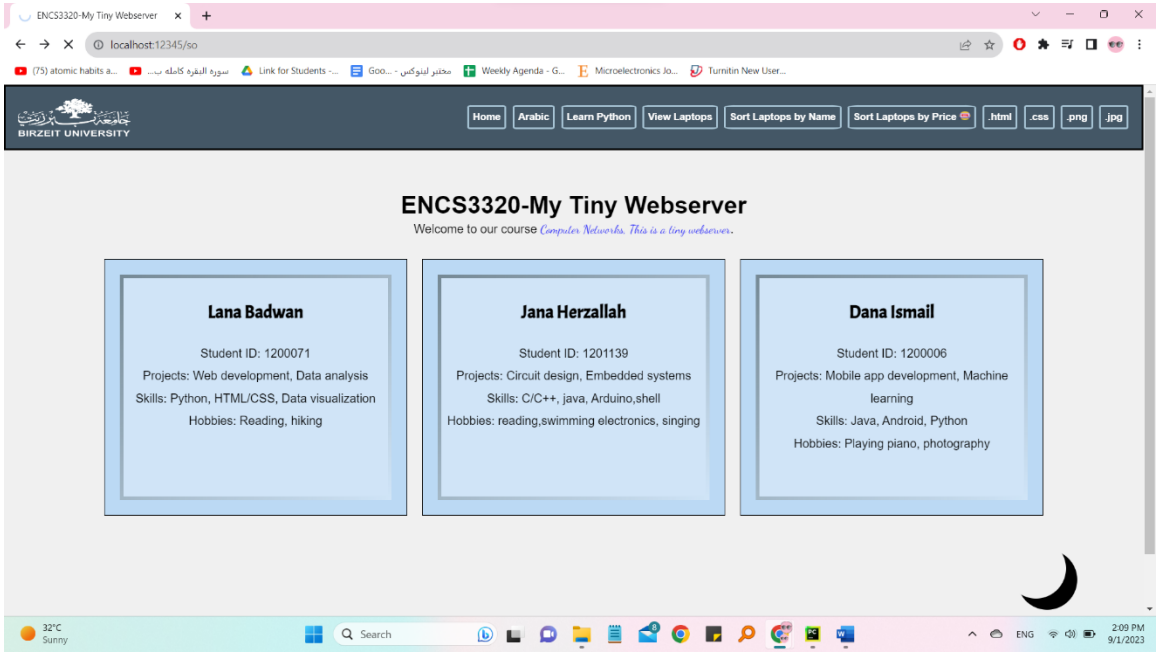
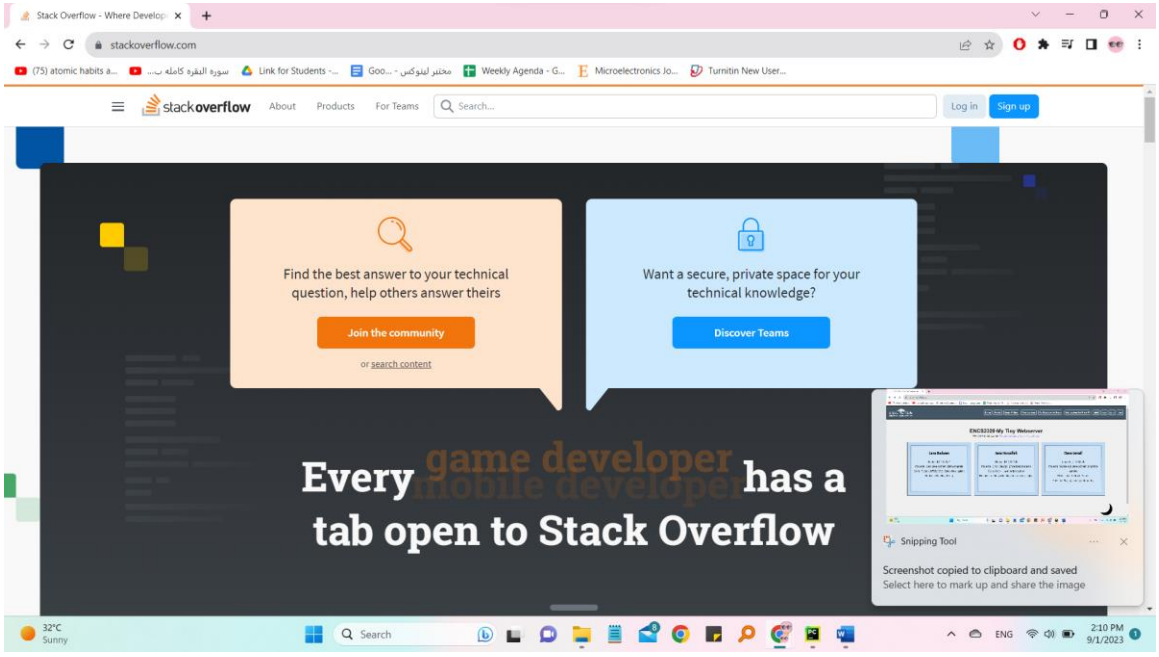


Figure 20 : /so rdirection



If the request is /bzu then redirect to birzeit website

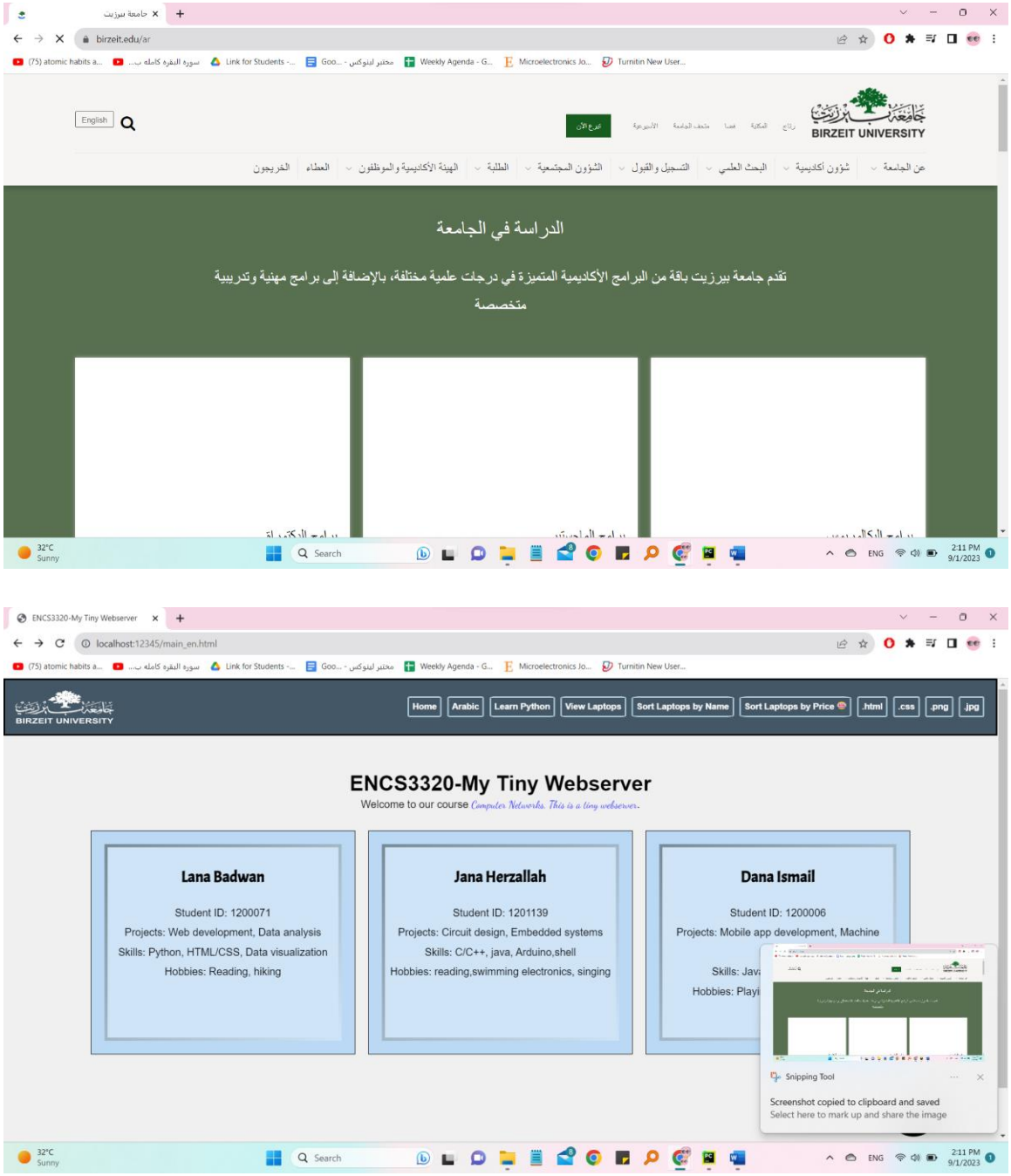


Figure 21: redirection to birzeit.edu

Error message

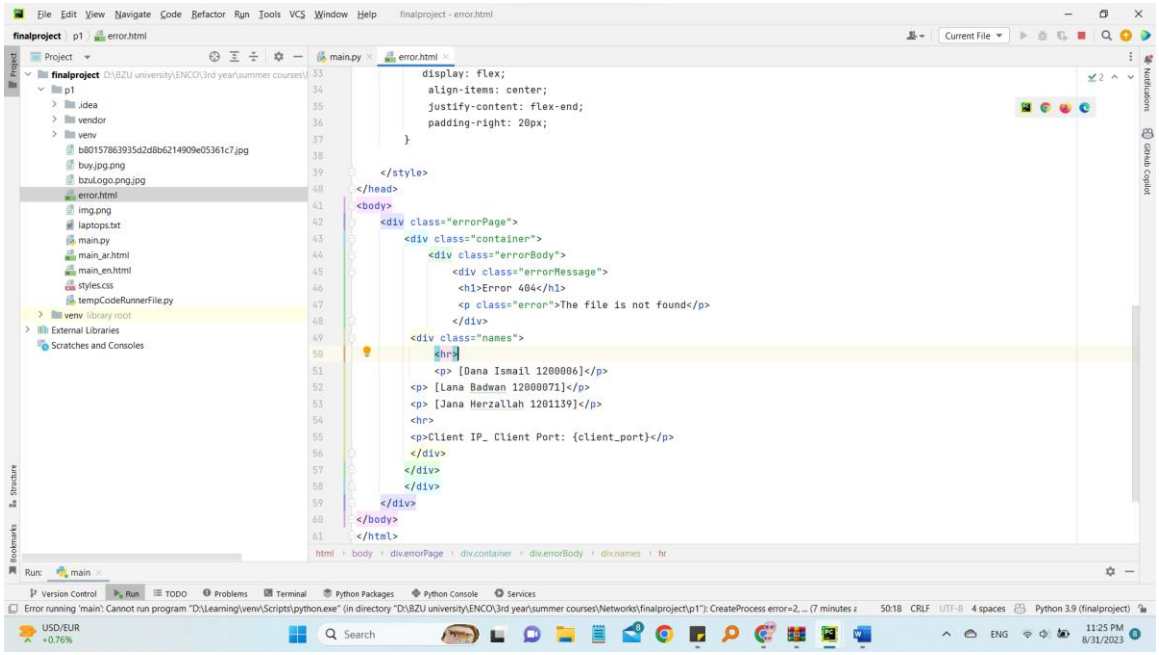


Figure 22 : error html code

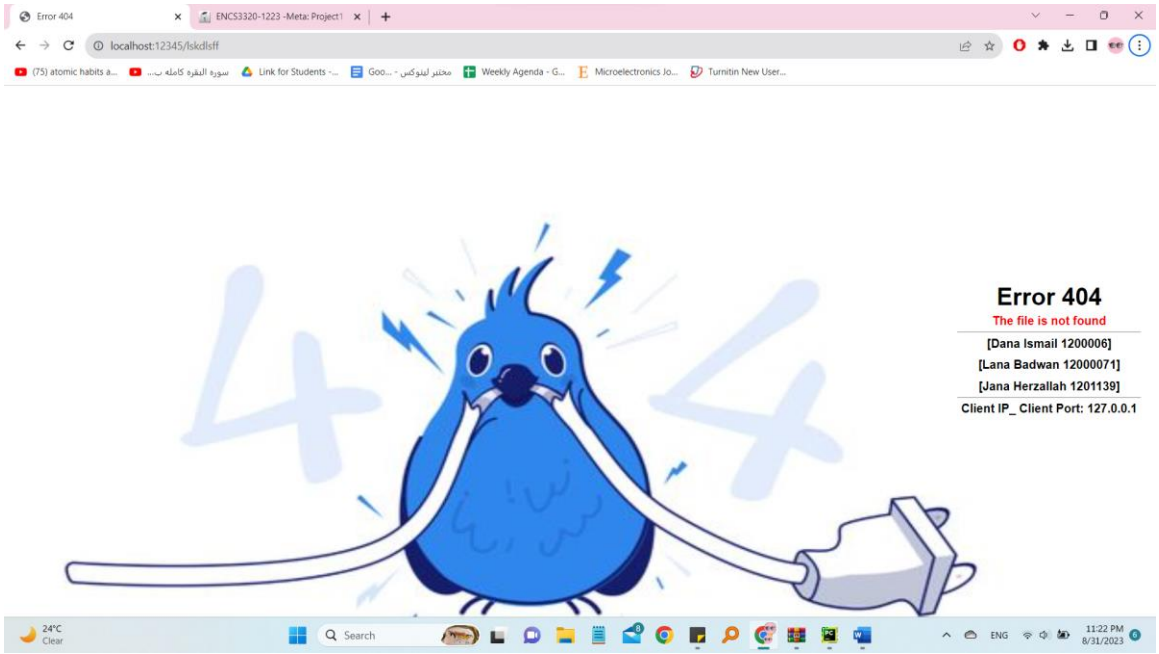


Figure 23: display error screen

Error code in python:

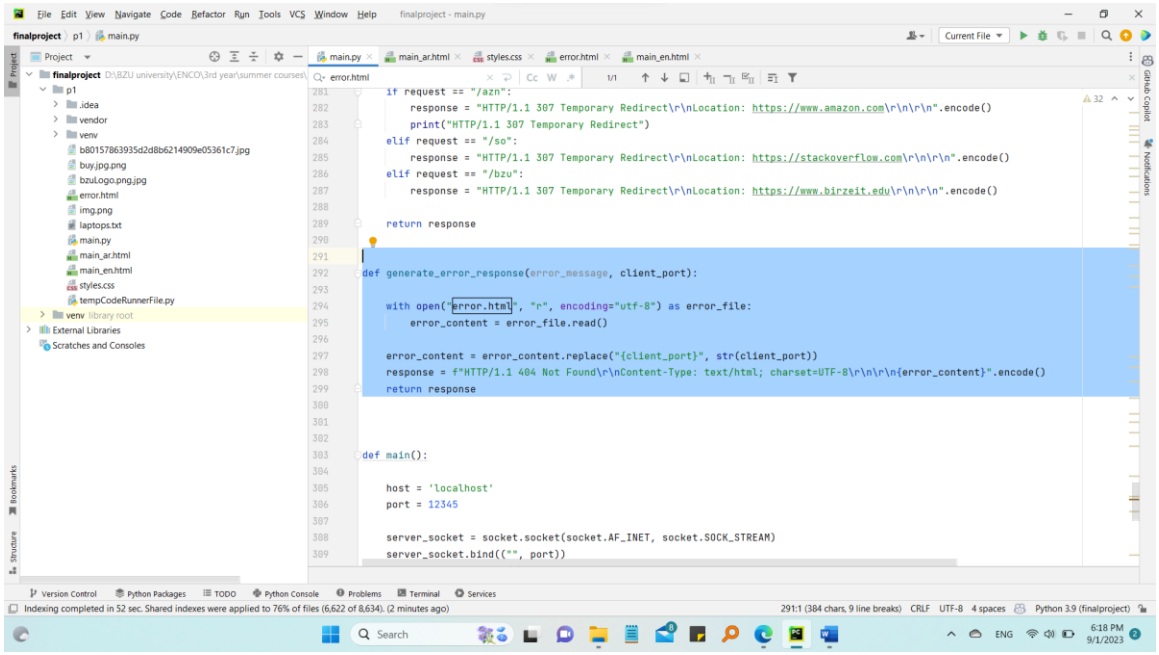


Figure 24 : error code python

This function is designed to generate customized HTTP error responses. It begins by opening a file named "error.html" with UTF-8 encoding, assuming that this file contains an HTML template for error messages. The HTML content of the file is then read and stored in the **error_content** variable. Next, any occurrences of the placeholder "{client_port}" within the **error_content** are replaced with the actual **client_port** value passed as an argument. The function then constructs an HTTP response with a status code of "404 Not Found," specifies the content type as "text/html; charset=UTF-8," and includes the modified HTML content. Finally, the response is encoded into bytes and returned.

Appendix:

Main.py

```
import socket
print('Welcome to our project')
print('Done by Dana,Lana,Jana')
def create_response( request,client_port):
    print(f"Received request from {client_port}: {request}")
```

```
    if request == "/" or request == "/index.html" or request ==
"/main_en.html" or request == "/en":
        try:
            with open("main_en.html", "r", encoding="utf-8") as html_file:
                html_content = html_file.read()

            with open("styles.css", "r", encoding="utf-8") as css_file:
                css_content = css_file.read()

            response = ("HTTP/1.1 200 OK\r\nContent-Type: text/html;
charset=UTF-8\r\n\r\n" + html_content + "\n<style>\n" + css_content +
"\n</style>").encode()
            print("HTTP/1.1 200 OK")
        except FileNotFoundError:
           response = generate_error_response("The requested page was not
found.",  client_port)



    elif request == "/laptops.txt":
        try:
            with open("laptops.txt", "r", encoding="utf-8") as
laptops_file:
                laptops_content = laptops_file.read()
                laptops_lines = laptops_content.strip().split('\n')
                sorted_laptops = '\n'.join(laptops_lines)
            response = f"HTTP/1.1 200 OK\r\nContent-Type: text/plain;
charset=UTF-8\r\n\r\n{laptops_content}".encode()
            print("HTTP/1.1 200 OK")
        except FileNotFoundError:
            response = generate_error_response("The requested page was not
found.",  client_port)

    elif request == "/SortByName":
        try:
            with open("laptops.txt", "r", encoding="utf-8") as
laptops_file:
                laptops_content = laptops_file.read()
                laptops_lines = laptops_content.strip().split('\n')
                laptops_lines.sort(key=lambda line:
line.split(',')[0].upper())  # Sort by name in uppercase
                sorted_laptops = '\n'.join(laptops_lines)

            response = f"HTTP/1.1 200 OK\r\nContent-Type: text/plain;
charset=UTF-8\r\n\r\n{sorted_laptops}".encode()
```

```
            print("HTTP/1.1 200 OK")
        except FileNotFoundError:
            response = generate_error_response("The requested page was not
found.",  client_port)


    # Add this case to the create_response function
    elif request == "/SortByPrice":
        try:
            with open("laptops.txt", "r", encoding="utf-8") as
laptops_file:
                laptops_lines = laptops_file.readlines()

            laptops_lines.sort(key=lambda line: float(line.split()[1]) if
len(line.split()) > 1 else float('inf'))

            sorted_laptops_content = ""
            total_price = 0

            for line in laptops_lines:
                name, price = line.strip().split()
                sorted_laptops_content += f"{name}: ${price}\n"
                total_price += float(price)

            sorted_laptops_content += f"\nTotal Price: ${total_price}"

            response = f"HTTP/1.1 200 OK\r\nContent-Type: text/plain;
charset=UTF-8\r\n\r\n{sorted_laptops_content}".encode()
            print("HTTP/1.1 200 OK")
        except FileNotFoundError:
            response = generate_error_response("The requested page was not
found.",  client_port)



    elif request == "/bzuLogo.png":
        try:
            with open("bzuLogo.png", "rb") as image_file:
                image_content = image_file.read()

            response = b"HTTP/1.1 200 OK\r\nContent-Type:
image/png\r\n\r\n" + image_content
            print("HTTP/1.1 200 OK")
        except FileNotFoundError:
```

```
            response = generate_error_response("The requested page was not
found.",  client_port)



    elif request == "/buy.jpg":
        try:
            with open("buy.jpg", "rb") as image_file:
                image_content = image_file.read()

            response = b"HTTP/1.1 200 OK\r\nContent-Type:
image/jpeg\r\n\r\n" + image_content
            print("HTTP/1.1 200 OK")
        except FileNotFoundError:
           response = generate_error_response("The requested page was not
found.",  client_port)



    elif request == "/main_en.html":
        try:
            with open("main_en.html", "r", encoding="utf-8") as html_file:
                html_content = html_file.read()

            response = ("HTTP/1.1 200 OK\r\nContent-Type: text/html;
charset=UTF-8\r\n\r\n" + html_content).encode()
            print("HTTP/1.1 200 OK")
        except FileNotFoundError:
           response = generate_error_response("The requested page was not
found.",  client_port)


    elif request == "/main_en.html":
        try:
            with open("main_en.html", "r", encoding="utf-8") as html_file:
                html_content = html_file.read()

            response = ("HTTP/1.1 200 OK\r\nContent-Type: text/html;
charset=UTF-8\r\n\r\n" + html_content).encode()
            print("HTTP/1.1 200 OK")
        except FileNotFoundError:
            response = generate_error_response("The requested page was not
found.",  client_port)
```

```
    elif request == "/main_ar.html" or request == "/ar":
        try:
            with open("main_ar.html", "r", encoding="utf-8") as html_file:
                html_content = html_file.read()

            with open("styles.css", "r", encoding="utf-8") as css_file:
                css_content = css_file.read()

            response = ("HTTP/1.1 200 OK\r\nContent-Type: text/html;
charset=UTF-8\r\n\r\n" + html_content + "\n<style>\n" + css_content +
"\n</style>").encode()
            print("HTTP/1.1 200 OK")
        except FileNotFoundError:
            response = generate_error_response("The requested page was not
found.",  client_port)



    elif request == "/styles.css":
        try:
            with open("styles.css", "r", encoding="utf-8") as css_file:
                css_content = css_file.read()

            response = ("HTTP/1.1 200 OK\r\nContent-Type: text/css;
charset=UTF-8\r\n\r\n" + css_content).encode()
            print("HTTP/1.1 200 OK")
        except FileNotFoundError:
            response = generate_error_response("The requested page was not
found.",  client_port)

    elif request.endswith(".html"):
        try:
            with open(request[1:], "r", encoding="utf-8") as html_file:
                html_content = html_file.read()

            response = ("HTTP/1.1 200 OK\r\nContent-Type: text/html;
charset=UTF-8\r\n\r\n" + html_content).encode()
            print("HTTP/1.1 200 OK")
        except FileNotFoundError:
```

```
            response = generate_error_response("The requested page was not
found.",  client_port)

    elif request.endswith(".css"):
        try:
            with open(request[1:], "r", encoding="utf-8") as css_file:
                css_content = css_file.read()

            response = ("HTTP/1.1 200 OK\r\nContent-Type: text/css;
charset=UTF-8\r\n\r\n" + css_content).encode()
            print("HTTP/1.1 200 OK")
        except FileNotFoundError:
            response = generate_error_response("The requested page was not
found.",  client_port)

    elif request.endswith(".png"):
        try:
            with open(request[1:], "rb") as image_file:
                image_content = image_file.read()

            response = b"HTTP/1.1 200 OK\r\nContent-Type:
image/png\r\n\r\n" + image_content
            print("HTTP/1.1 200 OK")
        except FileNotFoundError:
            response = generate_error_response("The requested page was not
found.",  client_port)

    elif request.endswith(".jpg"):
        try:
            with open(request[1:], "rb") as image_file:
                image_content = image_file.read()

            response = b"HTTP/1.1 200 OK\r\nContent-Type:
image/jpeg\r\n\r\n" + image_content
            print("HTTP/1.1 200 OK")
        except FileNotFoundError:
            response = generate_error_response("The requested page was not
found.",  client_port)



    elif request == "/main_ar.html" or request == "/ar":
        try:
            with open("main_ar.html", "r", encoding="utf-8") as html_file:
                html_content = html_file.read()
```

```
            with open("styles.css", "r", encoding="utf-8") as css_file:
                css_content = css_file.read()

            response = ("HTTP/1.1 200 OK\r\nContent-Type: text/html;
charset=UTF-8\r\n\r\n" + html_content + "\n<style>\n" + css_content +
"\n</style>").encode()
            print("HTTP/1.1 200 OK")
        except FileNotFoundError:
            response = generate_error_response("The requested page was not
found.",  client_port)

        # Add this case to the create_response function
    elif request == "/SortByName":
        try:
            with open("laptops.txt", "r", encoding="utf-8") as
laptops_file:
                laptops_content = laptops_file.read()
                laptops_lines = laptops_content.strip().split('\n')
                laptops_lines.sort(key=lambda line:
line.split(',')[0].upper())  # Sort by name in uppercase
                sorted_laptops = '\n'.join(laptops_lines)

            response = f"HTTP/1.1 200 OK\r\nContent-Type: text/plain;
charset=UTF-8\r\n\r\n{sorted_laptops}".encode()
            print("HTTP/1.1 200 OK")
        except FileNotFoundError:
            response = generate_error_response("The requested page was not
found.",  client_port)
    elif request == "/SortByPrice":
        try:
            with open("laptops.txt", "r", encoding="utf-8") as
laptops_file:
                laptops_lines = laptops_file.readlines()

            laptops_lines.sort(key=lambda line: float(line.split()[1]) if
len(line.split()) > 1 else float('inf'))

            sorted_laptops_content = ""
            total_price = 0

            for line in laptops_lines:
                name, price = line.strip().split()
                sorted_laptops_content += f"{name}: ${price}\n"
                total_price += float(price)
```

```
            sorted_laptops_content += f"\nTotal Price: ${total_price}"

            response = f"HTTP/1.1 200 OK\r\nContent-Type: text/plain;
charset=UTF-8\r\n\r\n{sorted_laptops_content}".encode()
            print("HTTP/1.1 200 OK")
        except FileNotFoundError:
            response = generate_error_response("The requested page was not
found.",  client_port)


    elif request.endswith(".html"):
        try:
            with open(request[1:], "r", encoding="utf-8") as html_file:
                html_content = html_file.read()

            response = ("HTTP/1.1 200 OK\r\nContent-Type: text/html;
charset=UTF-8\r\n\r\n" + html_content).encode()
            print("HTTP/1.1 200 OK")
        except FileNotFoundError:
           response = generate_error_response("The requested page was not
found.",  client_port)

    elif request.endswith(".css"):
        try:
            with open(request[1:], "r", encoding="utf-8") as css_file:
                css_content = css_file.read()

            response = ("HTTP/1.1 200 OK\r\nContent-Type: text/css;
charset=UTF-8\r\n\r\n" + css_content).encode()
            print("HTTP/1.1 200 OK")
        except FileNotFoundError:
           response = generate_error_response("The requested page was not
found.",  client_port)

    elif request.endswith(".png"):
        try:
            with open(request[1:], "rb") as image_file:
                image_content = image_file.read()

            response = b"HTTP/1.1 200 OK\r\nContent-Type:
image/png\r\n\r\n" + image_content
            print("HTTP/1.1 200 OK")
        except FileNotFoundError:
```

```
            response = generate_error_response("The requested page was not
found.",  client_port)

    elif request.endswith(".jpg"):
        try:
            with open(request[1:], "rb") as image_file:
                image_content = image_file.read()

            response = b"HTTP/1.1 200 OK\r\nContent-Type:
image/jpeg\r\n\r\n" + image_content
            print("HTTP/1.1 200 OK")
        except FileNotFoundError:
            response = generate_error_response("The requested page was not
found.",  client_port)


    else:
        response = generate_error_response("The requested page was not
found.",  client_port)

        # Add redirects for specific paths using 307 Temporary Redirect
status code
    if request == "/azn":
        response = "HTTP/1.1 307 Temporary Redirect\r\nLocation:
https://www.amazon.com\r\n\r\n".encode()
        print("HTTP/1.1 307 Temporary Redirect")
    elif request == "/so":
        response = "HTTP/1.1 307 Temporary Redirect\r\nLocation:
https://stackoverflow.com\r\n\r\n".encode()
    elif request == "/bzu":
        response = "HTTP/1.1 307 Temporary Redirect\r\nLocation:
https://www.birzeit.edu\r\n\r\n".encode()

    return response


def generate_error_response(error_message, client_port):

    with open("error.html", "r", encoding="utf-8") as error_file:
        error_content = error_file.read()

    error_content = error_content.replace("{client_port}",
str(client_port))
    response = f"HTTP/1.1 404 Not Found\r\nContent-Type: text/html;
charset=UTF-8\r\n\r\n{error_content}".encode()
```

```
    return response



def main():

    host = 'localhost'
    port = 12345

    server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server_socket.bind(("", port))
    server_socket.listen(1)

    print(f"Server is listening on {host}:{port}")

    while True:
        client_socket, client_address = server_socket.accept()
        request = client_socket.recv(1024).decode()

        if not request:
            continue

        print(f"Received request from
{client_address[0]}:{client_address[1]}:\n{request}")

        request_line = request.split('\n')[0]
        request_parts = request_line.split()
        if len(request_parts) > 1:
            requested_path = request_parts[1]
            response = create_response(requested_path, client_address[0])
        else:
            response = generate_error_response("Bad Request",
client_address[0])

        client_socket.sendall(response)
        client_socket.close()



if __name__ == "__main__":
    main()
```

main_en.html

```
<!DOCTYPE html>
<html>
<head>
    <title>ENCS3320-My Tiny Webserver</title>
    <link rel="stylesheet" type="text/css" href="styles.css">
</head>
<body>
    <div class="selection-bar">
    <img src="bzuLogo.png.jpg" align="left" width="150" height="50">
        <div class="element">
            <button class="button" id="Home">Home</button>
         <button class="button" id="Arabic">Arabic</button>
    <button class="button" id="learnPythonButton">Learn Python</button>

    <button class="button" id="viewLaptopsButton">View Laptops</button>
    <button class="button" id="sortByNameButton">Sort Laptops by
Name</button>
        <button class="button" id="sortByPriceButton">
    Sort Laptops by Price
    <img src="buy.jpg.png" alt="Buy" width="15px" height="12px">
   </button>

        <button class="button" id="htmlButton">.html</button>
        <button class="button" id="cssButton">.css</button>
        <button class="button" id="pngButton">.png</button>
        <button class="button" id="jpgButton">.jpg</button>
        </div>

   </div>


    <div class="header">
        <div class="cotainer">
            <div class="info">
                <h1>ENCS3320-My Tiny Webserver</h1>
            <p>Welcome to our course <span class="blue-text">Computer
Networks, This is a tiny webserver</span>.</p>
            </div>
            <div class="aboutUs">
                <div class="member-box">
                    <div class="partcipent">
                    <h2>Lana Badwan</h2>
                    <p>Student ID: 1200071</p>
                    <p>Projects: Web development, Data analysis</p>
                    <p>Skills: Python, HTML/CSS, Data visualization</p>
                    <p>Hobbies: Reading, hiking</p>
                    </div>
                </div>
                <div class="member-box">
                    <div class="partcipent">
                        <h2>Jana Herzallah</h2>
                    <p>Student ID: 1201139</p>
                    <p>Projects: Circuit design, Embedded systems</p>
                    <p>Skills: C/C++, java, Arduino,shell</p>
                    <p>Hobbies: reading,swimming electronics,
```

```
singing</p>
                    </div>
                </div>
                <div class="member-box">
                    <div class="partcipent">
                        <h2>Dana Ismail</h2>
                    <p>Student ID: 1200006</p>
                    <p>Projects: Mobile app development, Machine
learning</p>
                    <p>Skills: Java, Android, Python</p>
                    <p>Hobbies: Playing piano, photography</p>
                    </div>
                </div>
            </div>
            <div class="shape">
                <div class="moon">

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



    <script>
    const learnPythonButton =
document.getElementById("learnPythonButton");
    const viewLaptopsButton =
document.getElementById("viewLaptopsButton");
    const sortByNameButton =
document.getElementById("sortByNameButton");
    const sortByPriceButton =
document.getElementById("sortByPriceButton");

        const htmlButton = document.getElementById("htmlButton");
        const cssButton = document.getElementById("cssButton");
        const pngButton = document.getElementById("pngButton");
        const jpgButton = document.getElementById("jpgButton");

    learnPythonButton.addEventListener("click", () => {
        window.location.href =
"https://www.w3schools.com/python/python_intro.asp";
    });

    viewLaptopsButton.addEventListener("click", async () => {

            window.location.href = "/laptops.txt"
    });

    sortByNameButton.addEventListener("click", () => {
        window.location.href = "/SortByName"; // Redirect to
/SortByName
    });

    sortByPriceButton.addEventListener("click", () => {
        window.location.href = "/SortByPrice"; // Redirect to
/SortByPrice
```

```
    });

    const arabicButton = document.getElementById("Arabic");

    arabicButton.addEventListener("click", () => {
        window.location.href = "main_ar.html"; // Redirect to Arabic
version
    });

    htmlButton.addEventListener("click", async () => {
       // Fetch the content of main_en.html
       const response = await fetch("/main_en.html");
       const htmlCode = await response.text();

       // Create a new window or dialog and display the HTML code
       const codeWindow = window.open("", "_blank",
"width=600,height=400");
        codeWindow.document.write("<pre>" + escapeHtml(htmlCode) +
"</pre>");
    });

        // Function to escape HTML entities
     function escapeHtml(unsafe) {
       return unsafe.replace(/</g, "&lt;").replace(/>/g, "&gt;");
     }

    cssButton.addEventListener("click", () => {
        window.location.href = "/styles.css"; // Redirect to styles.css
    });

    pngButton.addEventListener("click", () => {
        window.location.href = "img.png"; // Redirect to img.png
    });

    jpgButton.addEventListener("click", () => {
        window.location.href = "bzuLogo.png.jpg"; // Redirect to
buy.jpg
    });
    </script>
</body>
</html>
```

main_ar.html

```
<!DOCTYPE html>
<html lang="ar">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>ENCS3320-My Tiny Webserver</title>
    <link rel="stylesheet" type="text/css" href="styles.css">
</head>
<body>
```

```
    <div class="selection-bar">
        <img src="bzuLogo.png.jpg" align="left" width="150"
height="50">
        <div class="element">
            <button class="button" id="homeButton">الرئيسية</button>
        <button class="button" id="arabicButton">اللغة
الانجليزية</button>
        <button class="button" id="learnPythonButton">تعلم
بايثون</button>
        <button class="button" id="viewLaptopsButton">عرض الأجهزة
المحمولة</button>
        <button class="button" id="sortByNameButton">حسب الأجهزة فرز
الاسم</button>
        <button class="button" id="sortByPriceButton">حسب الأجهزة فرز
السعر</button>
        <button class="button" id="htmlButton">.html</button>
        <button class="button" id="cssButton">.css</button>
        <button class="button" id="pngButton">.png</button>
        <button class="button" id="jpgButton">.jpg</button>
        </div>

    </div>

    <div class="header">
        <div class="container">
            <div class="info">
                <h1>ENCS3320-موقع الويب الصغير</h1>
    <p>.مساقنا في بك مرحبًا <span class="blue-text">شبكات، الحاسوب هذا
الصغير الويب خادم هو</span></p>
            </div>

        <div class="aboutUs">
            <div class="member-box">
                <div class="partcipent">
                    <h2>بدوان لانا</h2>
                    <p>الجامعي الرقم: 1200071</p>
                    <p>البيانات تحليل الويب، تطوير :المشاريع</p>
                    <p>البيانات تصوير ،HTML/CSS بايثون، :المهارات</p>
                    <p>الجبال رحلات القراءة، :الهوايات</p>
                </div>
            </div>
            <div class="member-box">
                <div class="partcipent">
                    <h2>حرزالله جنى</h2>
                <p>الجامعي الرقم: 1201139</p>
                <p>مدمجة أنظمة الدوائر، تصميم :المشاريع</p>
                <p>أردوينو المطبوعة، الدوائر تصميم ،C/C++ :المهارات</p>
                <p>الرسم ، إلكترونيات :الهوايات</p>
                </div>
            </div>
            <div class="member-box">
                <div class="partcipent">
                    <h2>إسماعيل دانا</h2>
                    <p>الجامعي الرقم : 1200006</p>
                <p>الآلة تعلم الجوال، تطبيقات تطوير :المشاريع</p>
                <p>بايثون أندرويد، جافا، :المهارات</p>
                <p>الفوتوغرافي التصوير البيانو، على العزف :الهوايات</p>
```

```
                </div>
            </div>

        </div>
        <div class="shape">
            <div class="moon">

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

    <script>
        const learnPythonButton =
document.getElementById("learnPythonButton");
        const viewLaptopsButton =
document.getElementById("viewLaptopsButton");
        const sortByNameButton =
document.getElementById("sortByNameButton");
        const sortByPriceButton =
document.getElementById("sortByPriceButton");
        const homeButton = document.getElementById("homeButton");

        learnPythonButton.textContent = "بايثون تعلم";
        viewLaptopsButton.textContent = "المحمولة الأجهزة عرض";
        sortByNameButton.textContent = "الاسم حسب الأجهزة فرز";
        sortByPriceButton.textContent = "السعر حسب الأجهزة فرز";
        homeButton.textContent = "الرئيسية";

        learnPythonButton.addEventListener("click", () => {
            window.location.href =
"https://www.w3schools.com/python/python_intro.asp";
        });

        viewLaptopsButton.addEventListener("click", async () => {

            window.location.href = "/laptops.txt"
        });

        sortByNameButton.addEventListener("click", () => {
            window.location.href = "/SortByName";
        });

        sortByPriceButton.addEventListener("click", () => {
            window.location.href = "/SortByPrice";
        });

        homeButton.addEventListener("click", () => {
            window.location.href = "main_en.html"; // Replace with the
actual file path
        });

        htmlButton.addEventListener("click", async () => {
       // Fetch the content of main_en.html
       const response = await fetch("/main_en.html");
       const htmlCode = await response.text();
```

```
        // Create a new window or dialog and display the HTML code
        const codeWindow = window.open("", "_blank",
"width=600,height=400");
         codeWindow.document.write("<pre>" + escapeHtml(htmlCode) +
"</pre>");
    });

         // Function to escape HTML entities
     function escapeHtml(unsafe) {
       return unsafe.replace(/</g, "&lt;").replace(/>/g, "&gt;");
     }

         cssButton.addEventListener("click", () => {
             window.location.href = "/styles.css"; // Redirect to
styles.css
         });

         pngButton.addEventListener("click", () => {
             window.location.href = "/bzuLogo.png.jpg"; // Redirect to
bzuLogo.png
         });

         jpgButton.addEventListener("click", () => {
             window.location.href = "/buy.jpg.png"; // Redirect to
buy.jpg
         });

         arabicButton.addEventListener("click", () => {
             window.location.href = "main_en.html"; // Replace with the
actual file path
         });


    </script>
</body>
</html>
```

Cssfile

```
@import
url('https://fonts.googleapis.com/css2?family=Acme&family=Dancing+Scrip
t:wght@400;500&family=Nanum+Myeongjo:wght@400;700;800&family=Qwitcher+G
rypen&display=swap');
/*
font-family: 'Dancing Script', cursive;
font-family: 'Nanum Myeongjo', serif;
font-family: 'Qwitcher Grypen', cursive;
font-family: 'Acme', sans-serif;
*/
*{
```

```
    margin: 0;
    padding: 0;
}

body {
    font-family: Arial, sans-serif;
}

.header {
    background-color: #f0f0f0;
    padding: 20px;
    text-align: center;
}

.info h1
{
    padding: 5px;
    font-family: Nanum Myeongjo', serif;
}

.shape
{
    justify-content: flex-end;
    align-items: center;
    display: flex;

}

.selection-bar {
    border: solid black;
    background-color: #445766;
    display: flex;
    flex-direction: row;
    flex-wrap: nowrap;
    justify-content: space-between;
    padding: 16px;
    align-content: center;
    align-items: center;
    position: relative;

}

.elment {
    margin: 5px 0px;
    text-align: center;
    display: flex;
    position: relative;
}

.element button {
    margin: 2px;
    padding: 5px;
    border: solid #AAC7D8;
    background-color: transparent;
    font-weight: bold !important;
    color: white;
    border-radius: 9%;
```

```
    transition: 3ms;
    height: 32px;
}

.element button:hover {
    background-color: #AAC7D8;
    border-color: black;
    color: black;
}




.aboutUs{
    display: flex;
    justify-content: space-between;
    margin: 20px;
}

.member-box {
    flex: 1;
    border: 1px solid;
    padding: 20px;
    margin: 10px;
   /* background-image:  repeating-radial-gradient(circle at 0 0,
transparent 0, #bbd9f4 14px), repeating-linear-gradient(#768a96,
#1b3854);
    */background-color: #bbd9f4;
    height: 300px;



    }




    .partcipent {
        background: rgba(255, 255, 255, 0.3);
        -webkit-backdrop-filter: blur(8px);
        backdrop-filter: blur(8px);
        height: 280px;
        padding: 5px;
        line-height: 30px;
        justify-content: center;
        position: relative; /* Added for positioning the pseudo-element
*/

        /* Initial gradient border */
        border-width: 5px;
        border-style: solid;
        border-image: linear-gradient(to bottom right, #768A96,
transparent) 1;
        transition: border-image 0.3s; /* Added transition for the
border effect */
    }
```

```
    .partcipent  h2{
        font-family: 'Acme', sans-serif;
    }

    /* Hover effect */
    .partcipent:hover {
        border-image: linear-gradient(to bottom right, transparent,
#768A96) 1;
        background: transparent;
    }




.partcipent h2
{
    padding: 25px 0px;

}



.blue-text {
    color: blue;
    font-family: 'Dancing Script', cursive;
}

.container
{
    width: 80%;
}

.header
{
    height: 100vh;
    display: flex;
    justify-content: center;
    align-items: center;
    margin-top: -50px;
}

.moon
{
    width: 80px;
      height: 80px;
      border-radius: 50%;
      box-shadow: 15px 15px 0 0 black;
      justify-content: right;

}
```

Error.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>
    <style>
        body {
            font-family: Arial, sans-serif;
            text-align: center;

        }
        *
        {
            margin: 0px;
            padding: 0px;
        }
        .error {
            color: red;
            font-weight: bold;
        }

        .errorMessage p , .names p{
            font-weight: bold;
            padding: 5px;
        }
        .errorPage
        {
            background-image: url(b80157863935d2d8b6214909e05361c7.jpg)
;
            background-repeat: no-repeat;
            height: 100vh;
           background-size: 1380px 900px;
           display: flex;
            align-items: center;
            justify-content: flex-end;
            padding-right: 20px;
        }

    </style>
</head>
<body>
    <div class="errorPage">
        <div class="container">
            <div class="errorBody">
                <div class="errorMessage">
                 <h1>Error 404</h1>
                 <p class="error">The file is not found</p>
                </div>
         <div class="names">
             <hr>
```

```
            <p> [Dana Ismail 1200006]</p>
          <p> [Lana Badwan 12000071]</p>
          <p> [Jana Herzallah 1201139]</p>
          <hr>
          <p>Client IP_ Client Port: {client_port}</p>
          </div>
        </div>
        </div>
    </div>
</body>
</html>
```