<u>TCP Socket Programming</u>

Group Members: Akshat Gian(2022051), Abhinav Kumar Saxena(2022018)

Course Name: Computer Networks(CN CSE232)

In this assignment, we are implementing ***Transmission Control Protocol (TCP)*** sockets to send and receive Stream packets in python by Establishing a TCP connection between the Server and the Client by sending and processing **HTTP** requests.

We will begin by implementing a simple server that handles **HTTP** requests and establishes a **TCP** connection when a client initiates a request. The client sends **HTTP** requests to the server to initiate the connection. This process mirrors the way modern operating systems manage network communications.

Transmission Control Protocol :

The TCP protocol provides a shared Network between two endpoints for reliable packet transmission between the server and the client. It ensures data integrity by establishing a connection-oriented communication, where both parties must agree on the connection before data is sent. TCP uses mechanisms like sequence numbers and acknowledgments to guarantee that packets arrive in order and without errors. If any packets are lost or corrupted during transmission, TCP will automatically retransmit them. This ensures that communication between the server and the client is both reliable and error-free, which is essential for applications like web browsing, email, and file transfers. Additionally, TCP uses flow control and congestion control to manage data flow and avoid overwhelming the network.

Assumptions(in part 1) (in reference to the code as well)

- **Message** received by the server is in the following format (if connection Established):

  GET /HelloWorld.html HTTP/1.1

  Host: 192.168.29.97:12000

  Connection: keep-alive

  Upgrade-Insecure-Requests: 1

  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/129.0.0.0 Safari/537.36

  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7

  Accept-Encoding: gzip, deflate

  Accept-Language: en-US,en;q=0.9

- **Server port = 12000**
- queueing up to 1 pending packet meaning which is getting processed.
- Web server handling only 1 HTTP request at a time.
- Printing "**404 Not Found"** in case of IOError.
- To better demonstrate the server handling only 1 TCP request at a time, we close the socket after handling 1 HTTP request.

Assumptions(in part 2) (in reference to the code as well)

- We will create a multithreaded **`client.py`** script to simulate multiple **HTTP** requests being sent to the server. This will allow us to test the server's ability to handle and process multiple requests concurrently, demonstrating its multithreading capabilities.

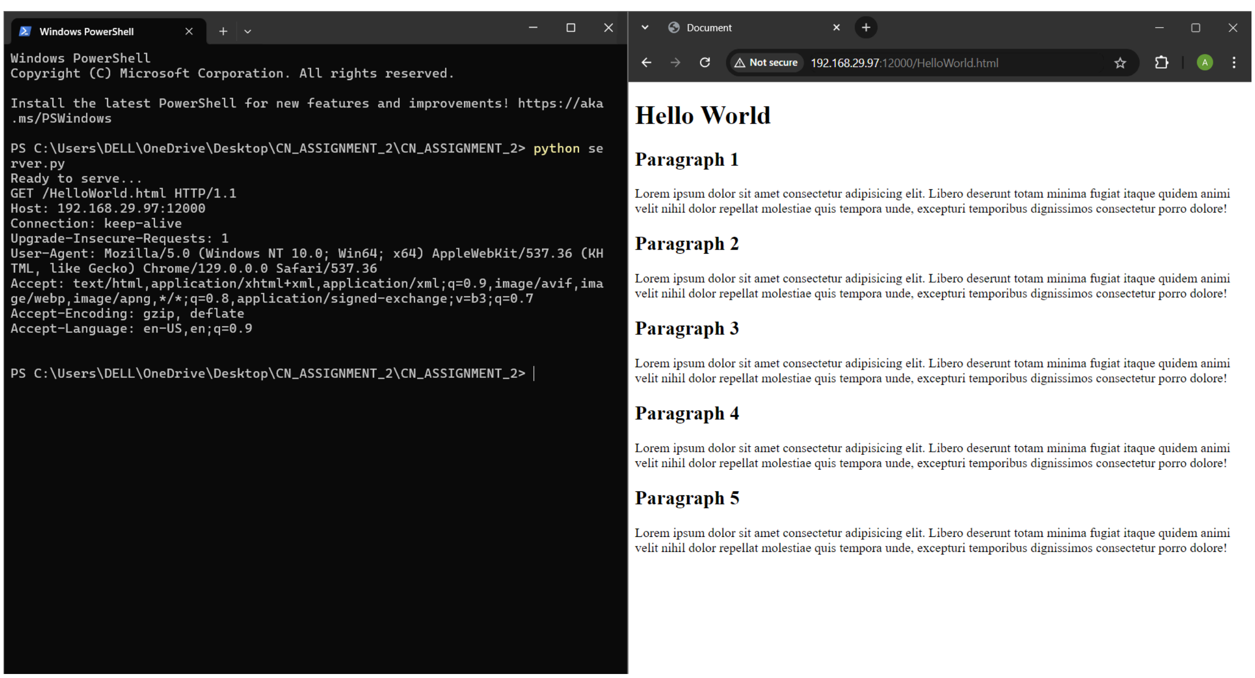
Assumptions(in part 3) (in reference to the code as well)

- Using multithreading in **`client.py`**, for sending multiple requests simultaneously, allowing us to test the **server's** ability to handle concurrent requests through its own multithreading.

Note : In all the images, the **server** is on the **left** and the **client/web browser** is on the right.

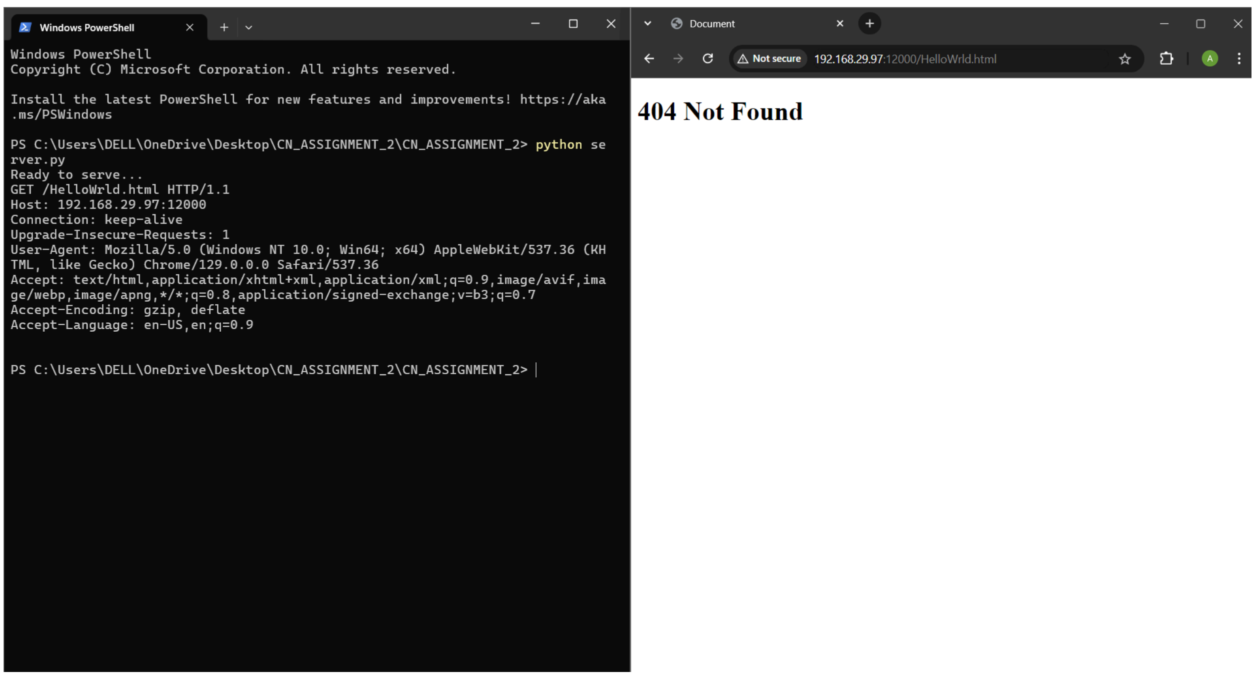
Part A

1. Requested File is present in the server (in our case, in the same folder as the server.py).



We have printed out the message received by the server from the web browser after establishing the TCP connection.

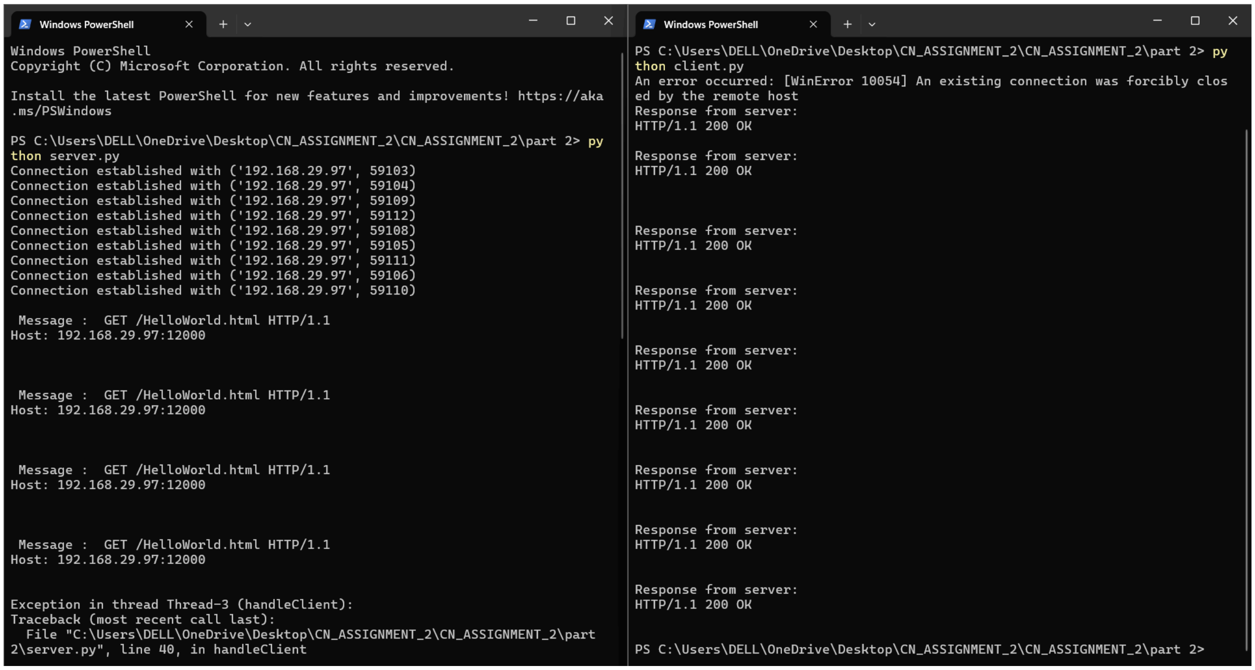
2. The Requested File is not present in the server.



If the requested file is not available on the server, it should return an HTTP status code '**404 Not Found**' in response to the client.

Part B

Note : We developed a custom multithreaded **`client.py`** to simulate multiple simultaneous requests sent to the server. The server, utilizing multithreading, efficiently handles and parses these concurrent requests.



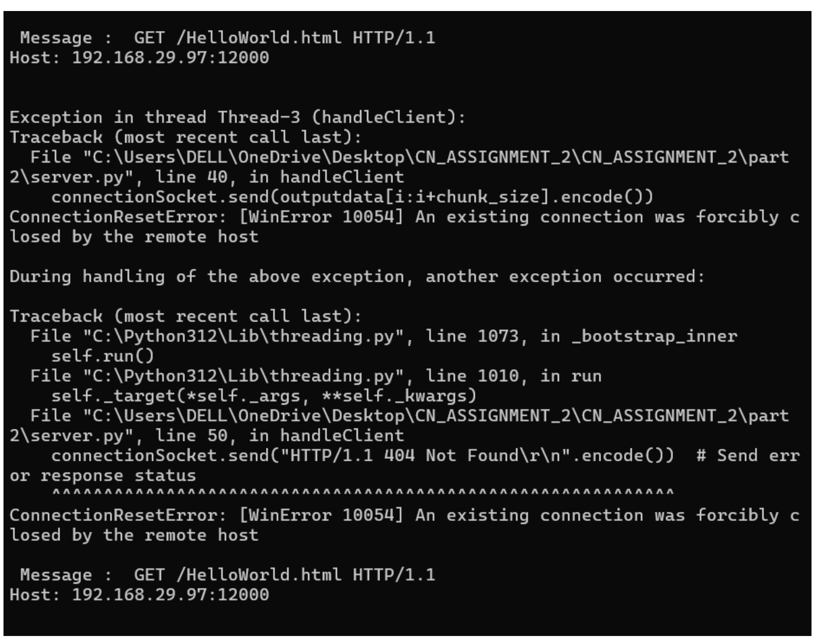
```
Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka
.ms/PSWindows

PS C:\Users\DELL\OneDrive\Desktop\CN_ASSIGNMENT_2\CN_ASSIGNMENT_2\part 2> py
thon server.py
Connection established with ('192.168.29.97', 59103)
Connection established with ('192.168.29.97', 59104)
Connection established with ('192.168.29.97', 59109)
Connection established with ('192.168.29.97', 59112)
Connection established with ('192.168.29.97', 59108)
Connection established with ('192.168.29.97', 59105)
Connection established with ('192.168.29.97', 59111)
Connection established with ('192.168.29.97', 59106)
Connection established with ('192.168.29.97', 59110)

 Message :  GET /HelloWorld.html HTTP/1.1
Host: 192.168.29.97:12000


 Message :  GET /HelloWorld.html HTTP/1.1
Host: 192.168.29.97:12000


 Message :  GET /HelloWorld.html HTTP/1.1
Host: 192.168.29.97:12000


 Message :  GET /HelloWorld.html HTTP/1.1
Host: 192.168.29.97:12000

Exception in thread Thread-3 (handleClient):
Traceback (most recent call last):
  File "C:\Users\DELL\OneDrive\Desktop\CN_ASSIGNMENT_2\CN_ASSIGNMENT_2\part
2\server.py", line 40, in handleClient
```

```
Windows PowerShell
PS C:\Users\DELL\OneDrive\Desktop\CN_ASSIGNMENT_2\CN_ASSIGNMENT_2\part 2> py
thon client.py
An error occurred: [WinError 10054] An existing connection was forcibly clos
ed by the remote host
Response from server:
HTTP/1.1 200 OK

Response from server:
HTTP/1.1 200 OK


Response from server:
HTTP/1.1 200 OK


Response from server:
HTTP/1.1 200 OK


Response from server:
HTTP/1.1 200 OK


Response from server:
HTTP/1.1 200 OK


Response from server:
HTTP/1.1 200 OK


Response from server:
HTTP/1.1 200 OK


Response from server:
HTTP/1.1 200 OK


PS C:\Users\DELL\OneDrive\Desktop\CN_ASSIGNMENT_2\CN_ASSIGNMENT_2\part 2>
```

We are sending **10** simultaneous requests to the server on the **left side**.

The **server** efficiently handles all requests concurrently, thanks to the implementation of multithreading as specified in the task.

Exception :

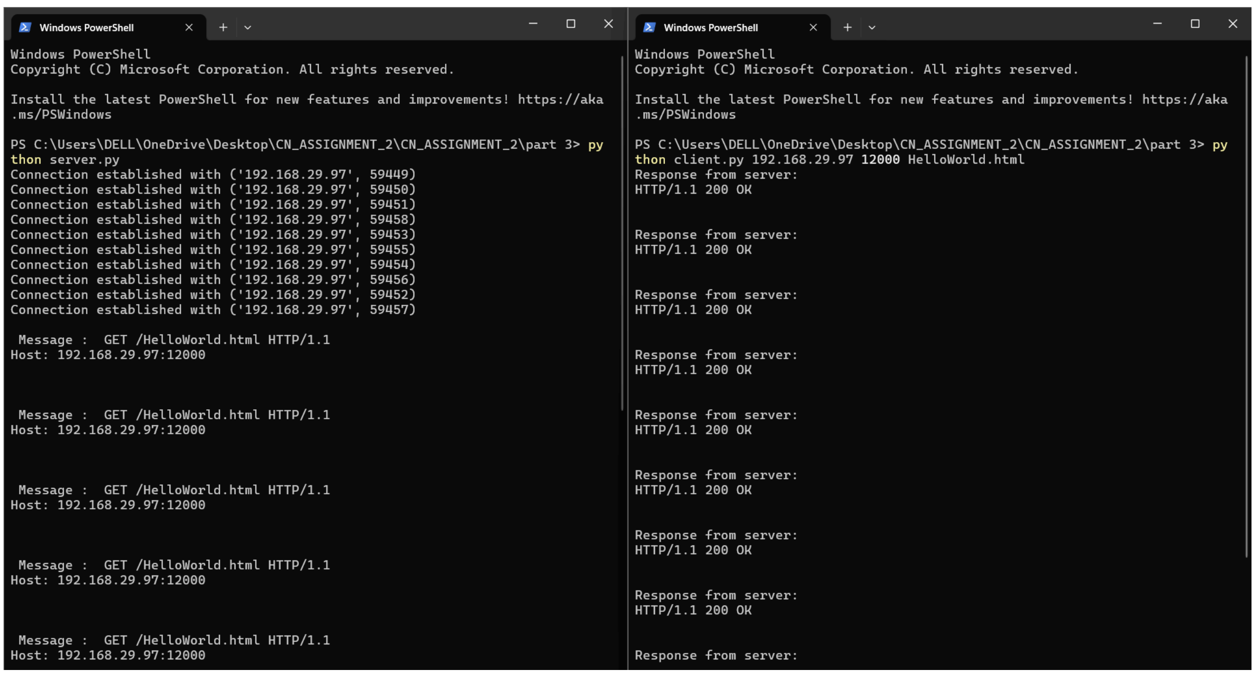


```
 Message :  GET /HelloWorld.html HTTP/1.1
Host: 192.168.29.97:12000


Exception in thread Thread-3 (handleClient):
Traceback (most recent call last):
  File "C:\Users\DELL\OneDrive\Desktop\CN_ASSIGNMENT_2\CN_ASSIGNMENT_2\part
2\server.py", line 40, in handleClient
    connectionSocket.send(outputdata[i:i+chunk_size].encode())
ConnectionResetError: [WinError 10054] An existing connection was forcibly c
losed by the remote host

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "C:\Python312\Lib\threading.py", line 1073, in _bootstrap_inner
    self.run()
  File "C:\Python312\Lib\threading.py", line 1010, in run
    self._target(*self._args, **self._kwargs)
  File "C:\Users\DELL\OneDrive\Desktop\CN_ASSIGNMENT_2\CN_ASSIGNMENT_2\part
2\server.py", line 50, in handleClient
    connectionSocket.send("HTTP/1.1 404 Not Found\r\n".encode())  # Send err
or response status
    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
ConnectionResetError: [WinError 10054] An existing connection was forcibly c
losed by the remote host

 Message :  GET /HelloWorld.html HTTP/1.1
Host: 192.168.29.97:12000
```

It can be seen from the **Server** output that if there are too many requests then there is a possibility of an existing connection being forcefully closed by the remote host. (as is visible in the image, an **exception** occurred while processing the request for **Thread-3**).

Part C

We send 10 HTTP requests to the server using our custom **HTTP client**. This demonstrates the capabilities of both the **client** and **server**, with the client utilizing multithreading to send multiple requests and the server simultaneously processing them through its own multithreaded implementation.



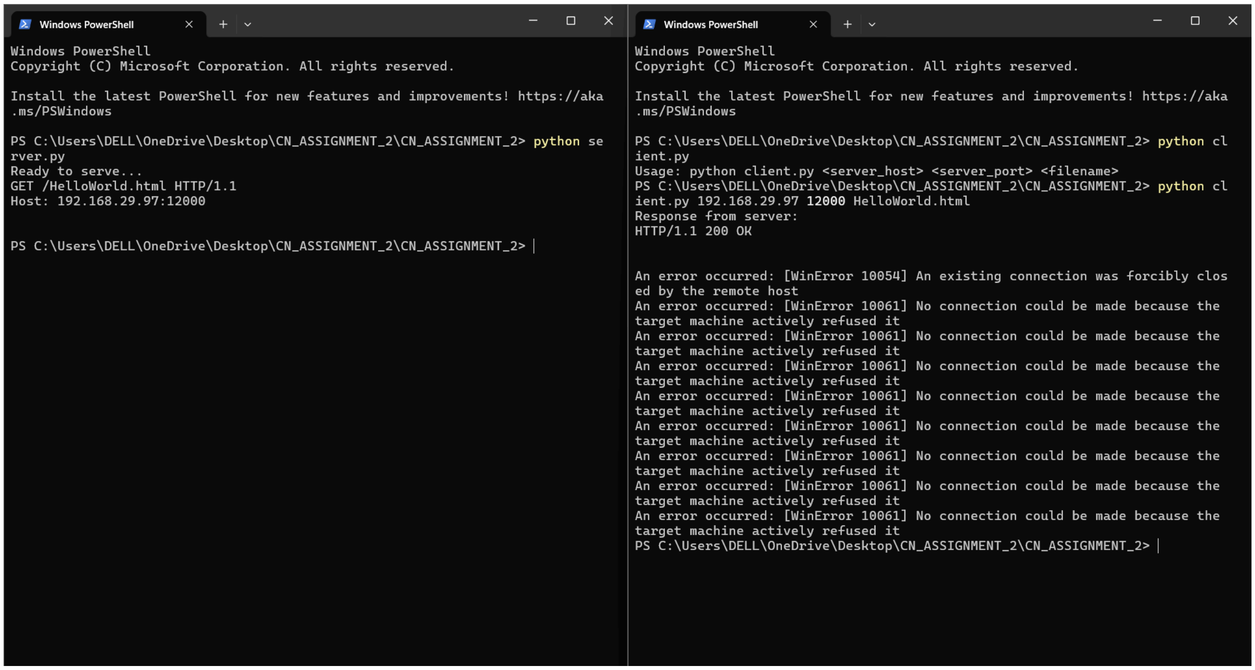
Input command for the client is of the following format :

client.py server_host server_port filename

Does this client work with the server implemented in Part 1 or Part 2 or both?

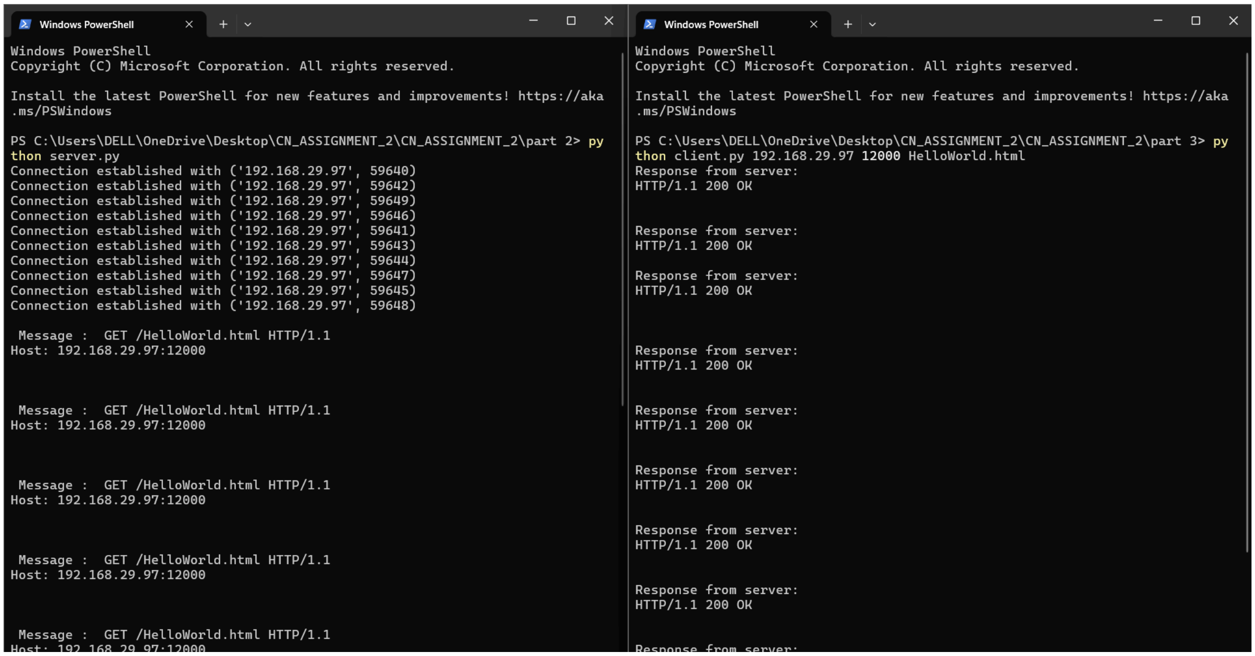
- **Part 1 - Yes**, this **HTTP Client** will work with the server implemented in **Part 1,** but we will have to modify it to send only **1 HTTP request.**

  If we try to run this HTTP client with the Server code written in Part 1 then we get the following output :



  The Server in Part 1 closes the socket after processing 1 HTTP request. As you can see the rest of the HTTP requests send the message **“No connection could be made because the target machine actively refused it.”**

- **Part 2 - Yes**, the HTTP client works completely fine with the Server implemented in Part-2 as it has been implemented to handle multiple requests simultaneously using multithreading.



As we can see from the Image, the server code from Part-2 runs completely fine with our custom **HTTP client** that was implemented in **Part -3**.

Conclusion :

This assignment enhanced our understanding of how **HTTP requests** are sent and processed, as well as how dedicated **TCP** connections are established for transmitting packets in real-world scenarios.

In **Part A**, we applied our knowledge of TCP connections in the context of web browsers, learning how the client (browser) sends HTTP requests and how the server processes and responds to those requests.

Part B expanded our comprehension by having us implement a multithreaded server capable of handling multiple HTTP requests simultaneously, deepening our grasp of how HTTP and TCP connections work in parallel.

In **Part C**, we developed our own HTTP client to send requests to the server. This experience provided valuable insight into how real-world HTTP clients interact with servers, establishing TCP connections to facilitate the sending and receiving of packets.

Overall, this assignment significantly broadened our understanding of the **Transmission Control Protocol (TCP)** and its real-world applications.