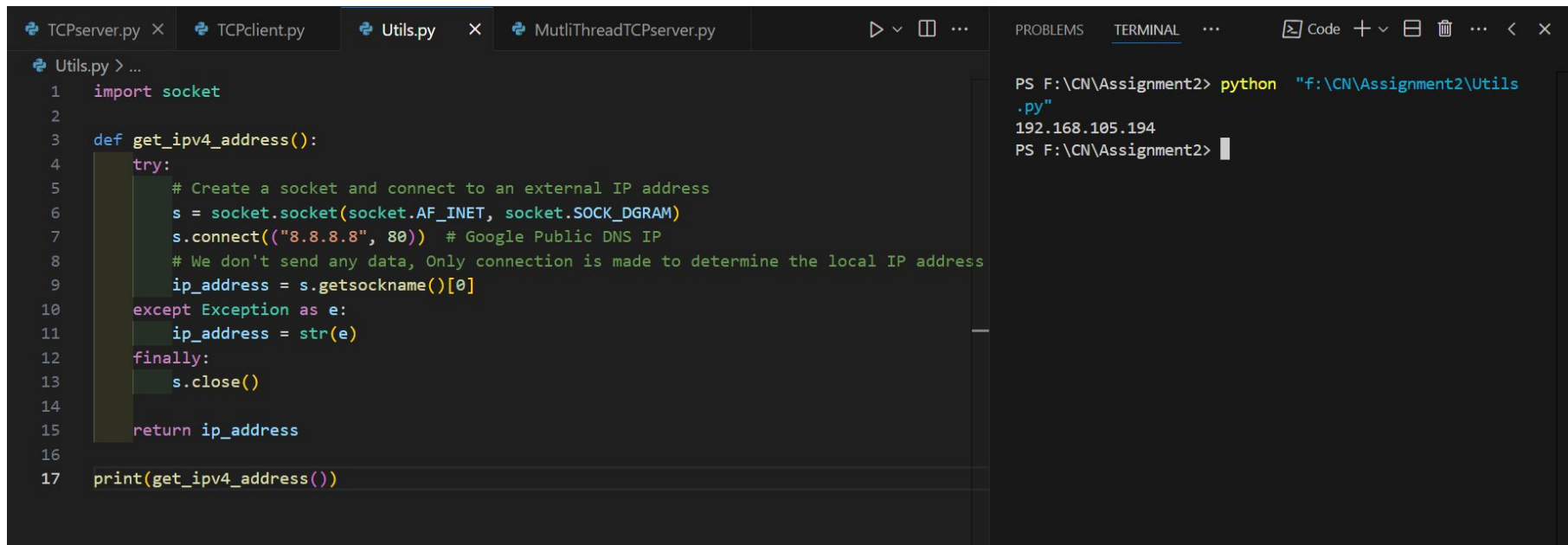


CSE 232 Section B,
Computer Networks; Programming Assignment 2:
TCP based Web application Lab

Harsh Rajput (2022201)
Aryan Singla (2022112)

Utils.get_ipv4_address

This function determines and returns the local machine's IPv4 address by creating a UDP socket connection to an external IP address (Google DNS) without sending data.

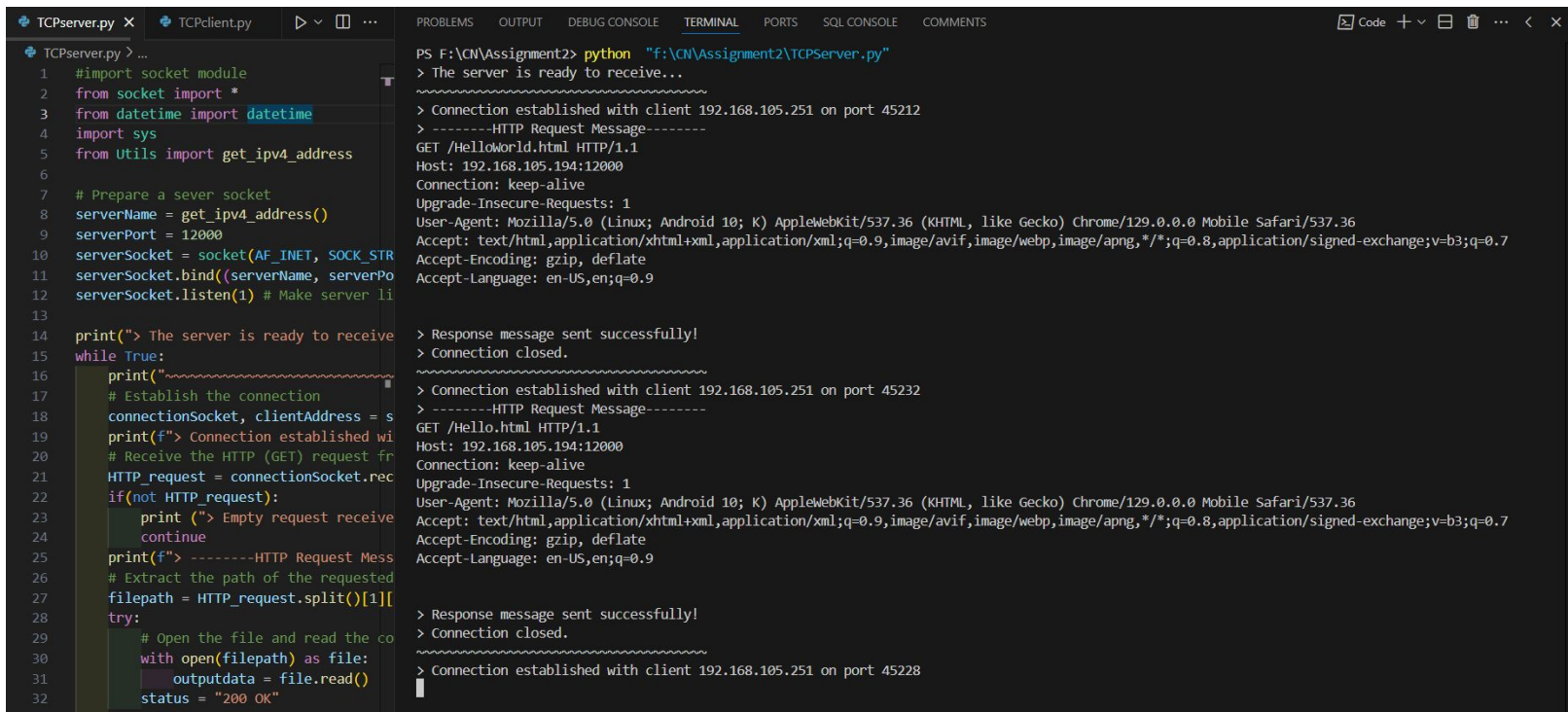


```
Utils.py > ...
1  import socket
2
3  def get_ipv4_address():
4      try:
5          # Create a socket and connect to an external IP address
6          s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
7          s.connect(("8.8.8.8", 80)) # Google Public DNS IP
8          # We don't send any data, Only connection is made to determine the local IP address
9          ip_address = s.getsockname()[0]
10     except Exception as e:
11         ip_address = str(e)
12     finally:
13         s.close()
14
15     return ip_address
16
17 print(get_ipv4_address())
```

```
PS F:\CN\Assignment2> python "f:\CN\Assignment2\Utils.py"
192.168.105.194
PS F:\CN\Assignment2>
```

TCPserver.py

The `TCPserver.py` script sets up a TCP server that listens for client connections, processes a single HTTP GET request at a time, and serves the requested file or returns a 404 error if the file is not found.



The screenshot displays a code editor with two tabs: `TCPserver.py` and `TCPclient.py`. The `TCPserver.py` tab is active, showing a Python script that sets up a TCP server. The script imports the `socket`, `datetime`, and `sys` modules, and uses the `Utils` module for `get_ipv4_address`. It prepares a server socket, binds it to an address and port (12000), and starts listening. A `while` loop handles incoming connections, printing status messages and processing HTTP GET requests. For each request, it extracts the file path and attempts to open and read the file, returning a "200 OK" status. The terminal output shows the server's execution, including connection establishment, request processing, and response sending.

```
TCPserver.py > ...
1  #import socket module
2  from socket import *
3  from datetime import datetime
4  import sys
5  from Utils import get_ipv4_address
6
7  # Prepare a sever socket
8  serverName = get_ipv4_address()
9  serverPort = 12000
10 serverSocket = socket(AF_INET, SOCK_STR
11 serverSocket.bind((serverName, serverPo
12 serverSocket.listen(1) # Make server li
13
14 print("> The server is ready to receive
15 while True:
16     print("~~~~~")
17     # Establish the connection
18     connectionSocket, clientAddress = s
19     print(f"> Connection established wi
20     # Receive the HTTP (GET) request fr
21     HTTP_request = connectionSocket.rec
22     if(not HTTP_request):
23         print ("> Empty request receive
24         continue
25     print(f"> -----HTTP Request Mess
26     # Extract the path of the requested
27     filepath = HTTP_request.split()[1][
28     try:
29         # Open the file and read the co
30         with open(filepath) as file:
31             outputdata = file.read()
32         status = "200 OK"
```

```
PS F:\CN\Assignment2> python "f:\CN\Assignment2\TCPServer.py"
> The server is ready to receive...
~~~~~
> Connection established with client 192.168.105.251 on port 45212
> ----- HTTP Request Message-----
GET /HelloWorld.html HTTP/1.1
Host: 192.168.105.194:12000
Connection: keep-alive
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Linux; Android 10; K) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/129.0.0.0 Mobile 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

> Response message sent successfully!
> Connection closed.
~~~~~
> Connection established with client 192.168.105.251 on port 45232
> ----- HTTP Request Message-----
GET /Hello.html HTTP/1.1
Host: 192.168.105.194:12000
Connection: keep-alive
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Linux; Android 10; K) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/129.0.0.0 Mobile 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

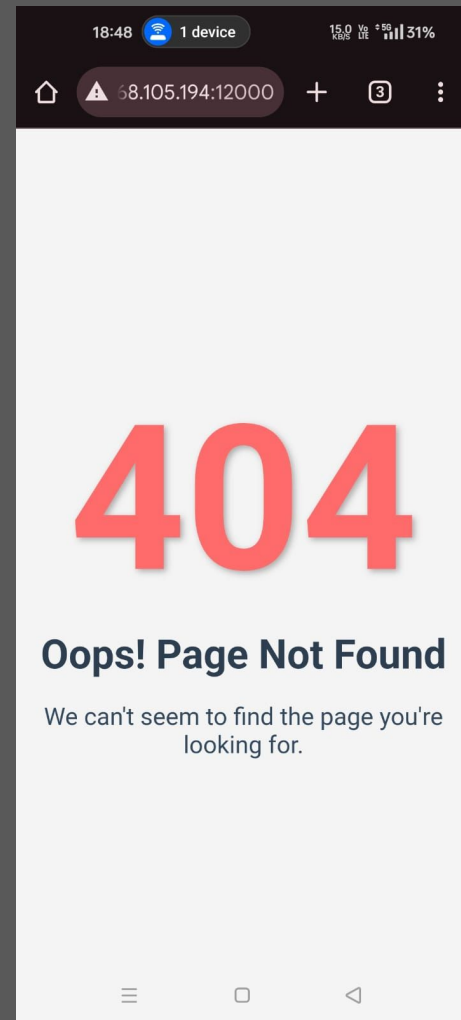
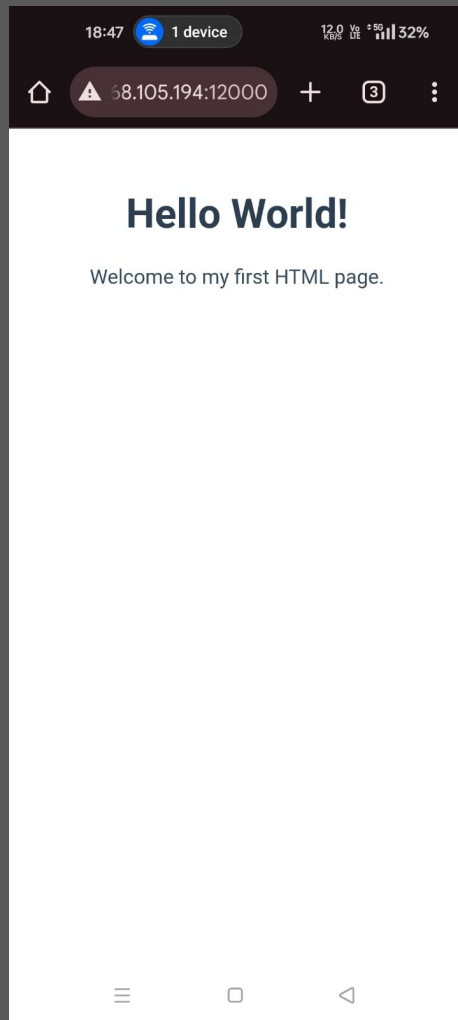
> Response message sent successfully!
> Connection closed.
~~~~~
> Connection established with client 192.168.105.251 on port 45228
```

Request 1

`http://192.168.105.194:12000/HelloWorld.html`

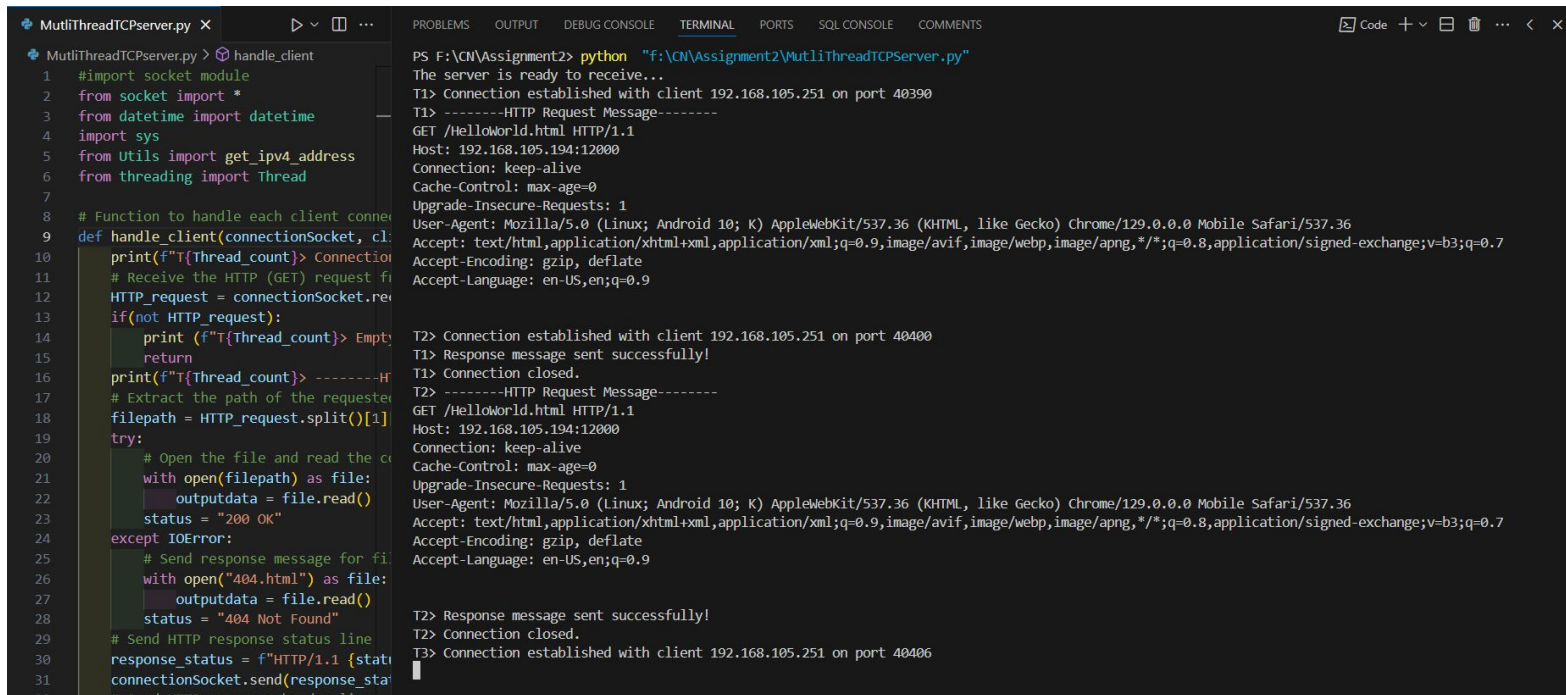
Request 2

`http://192.168.105.194:12000/Hello.html`



MutliThreadTCPserver.py

The `multithreadedTCPserver.py` script sets up a TCP server that uses multithreading to handle multiple client connections concurrently, processing HTTP GET requests and serving requested files or returning a 404 error if the file is not found.

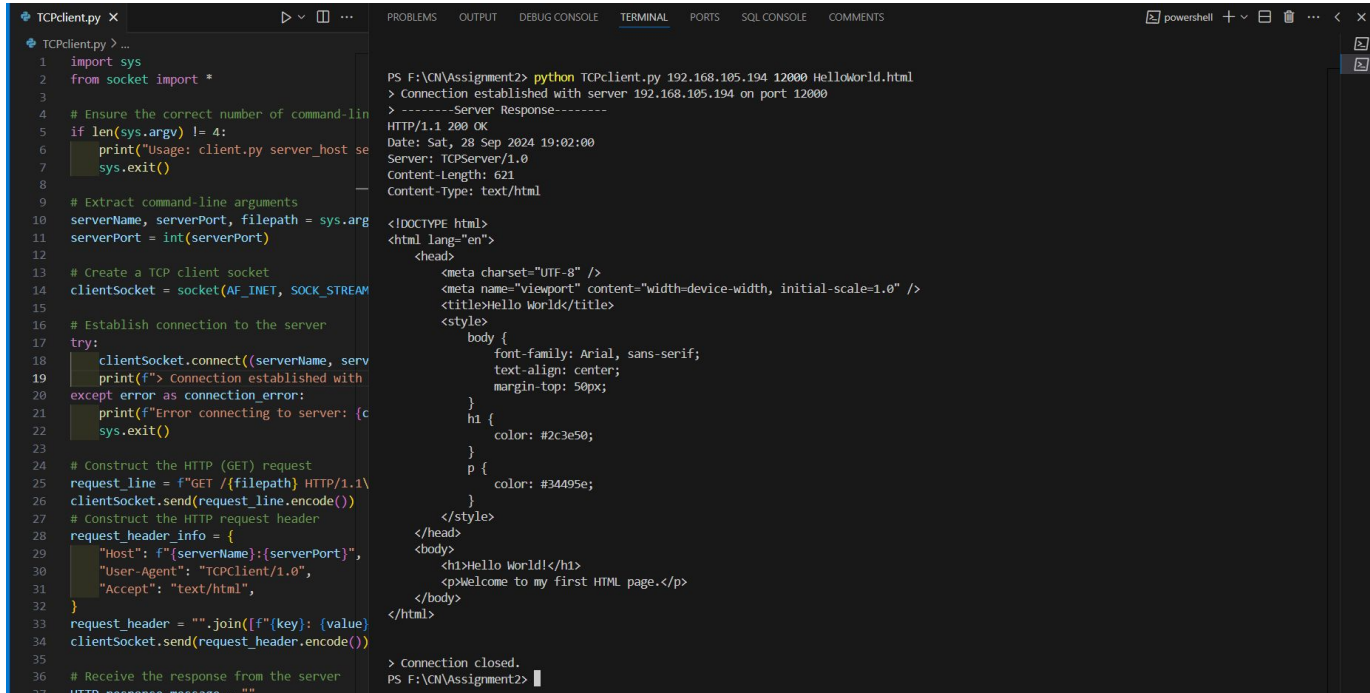


```
MutliThreadTCPserver.py x  ▾ ▢ ...
MutliThreadTCPserver.py > handle_client
1  #import socket module
2  from socket import *
3  from datetime import datetime
4  import sys
5  from Utils import get_ipv4_address
6  from threading import Thread
7
8  # Function to handle each client connection
9  def handle_client(connectionSocket, clientAddress):
10     print(f"T{Thread_count}> Connection established with client {clientAddress} on port {connectionSocket.getsockname()[1]}")
11     # Receive the HTTP (GET) request from the client
12     HTTP_request = connectionSocket.recv(4096)
13     if(not HTTP_request):
14         print(f"T{Thread_count}> Empty request received")
15         return
16     print(f"T{Thread_count}> -----HTTP Request Message-----")
17     # Extract the path of the requested file from the HTTP request line
18     filepath = HTTP_request.split()[1]
19     try:
20         # Open the file and read the data
21         with open(filepath) as file:
22             outputdata = file.read()
23             status = "200 OK"
24     except IOError:
25         # Send response message for file not found
26         with open("404.html") as file:
27             outputdata = file.read()
28             status = "404 Not Found"
29     # Send HTTP response status line
30     response_status = f"HTTP/1.1 {status}"
31     connectionSocket.send(response_status.encode())
32     # Send HTTP response header line
33     connectionSocket.send(f"Content-Type: text/html\r\n\r\n".encode())
34     # Send the data to the client
35     connectionSocket.send(outputdata.encode())
36     # Close the connection
37     connectionSocket.close()
38     print(f"T{Thread_count}> Connection closed.")

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  SQL CONSOLE  COMMENTS
PS F:\CN\Assignment2> python "f:\CN\Assignment2\MutliThreadTCPServer.py"
The server is ready to receive...
T1> Connection established with client 192.168.105.251 on port 40390
T1> -----HTTP Request Message-----
GET /HelloWorld.html HTTP/1.1
Host: 192.168.105.194:12000
Connection: keep-alive
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Linux; Android 10; K) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/129.0.0.0 Mobile 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
T2> Connection established with client 192.168.105.251 on port 40400
T1> Response message sent successfully!
T1> Connection closed.
T2> -----HTTP Request Message-----
GET /HelloWorld.html HTTP/1.1
Host: 192.168.105.194:12000
Connection: keep-alive
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Linux; Android 10; K) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/129.0.0.0 Mobile 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
T2> Response message sent successfully!
T2> Connection closed.
T3> Connection established with client 192.168.105.251 on port 40406
```

TCPclient.py

The `TCPclient.py` script connects to a TCP server, sends an HTTP GET request for a specified file, receives the server's response, and displays it.



```

TCPclient.py X
TCPclient.py > ...
1 import sys
2 from socket import *
3
4 # Ensure the correct number of command-line arguments
5 if len(sys.argv) != 4:
6     print("Usage: client.py server_host server_port filepath")
7     sys.exit()
8
9 # Extract command-line arguments
10 serverName, serverPort, filepath = sys.argv[1:]
11 serverPort = int(serverPort)
12
13 # Create a TCP client socket
14 clientSocket = socket(AF_INET, SOCK_STREAM)
15
16 # Establish connection to the server
17 try:
18     clientSocket.connect((serverName, serverPort))
19     print(f"> Connection established with server {serverName} on port {serverPort}")
20 except error as connection_error:
21     print(f"> Error connecting to server: {connection_error}")
22     sys.exit()
23
24 # Construct the HTTP (GET) request
25 request_line = f"GET /{filepath} HTTP/1.1\r\n"
26 clientSocket.send(request_line.encode())
27
28 # Construct the HTTP request header
29 request_header_info = {
30     "Host": f"{serverName}:{serverPort}",
31     "User-Agent": "TCPclient/1.0",
32     "Accept": "text/html",
33 }
34 request_header = "".join([f"{key}: {value}\r\n" for key, value in request_header_info.items()])
35 clientSocket.send(request_header.encode())
36
37 # Receive the response from the server
38 HTTP_response_message = ""

```

```

PS F:\CN\Assignment2> python TCPclient.py 192.168.105.194 12000 HelloWorld.html
> Connection established with server 192.168.105.194 on port 12000
> -----Server Response-----
HTTP/1.1 200 OK
Date: Sat, 28 Sep 2024 19:02:00
Server: TCPServer/1.0
Content-Length: 621
Content-Type: text/html

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>Hello World</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      text-align: center;
      margin-top: 50px;
    }
    h1 {
      color: #2c3e50;
    }
    p {
      color: #34495e;
    }
  </style>
</head>
<body>
  <h1>Hello World!</h1>
  <p>Welcome to my first HTML page.</p>
</body>
</html>

> Connection closed.
PS F:\CN\Assignment2>

```