Quinn Roemer

Professor Xuyu Wang

CSC 138

November 4, 2020

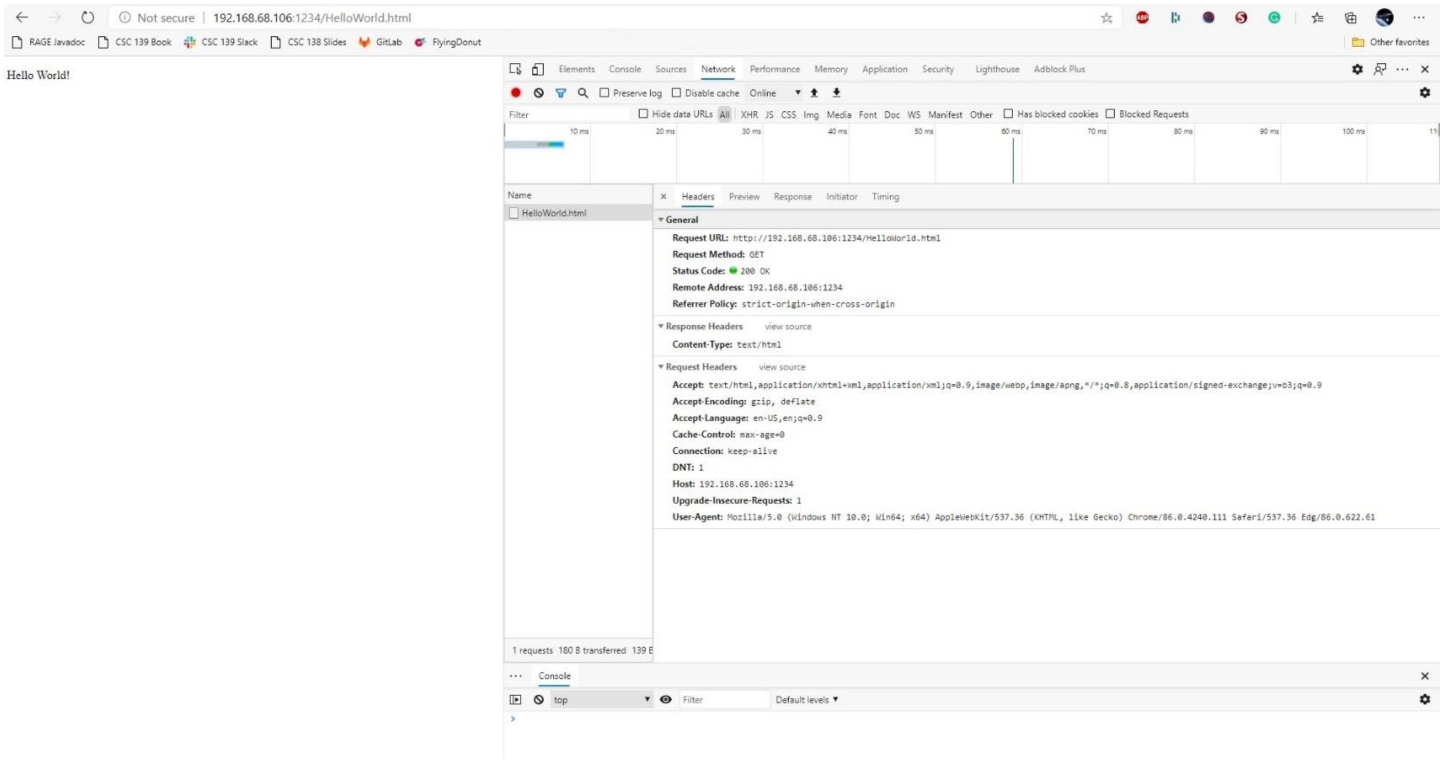
Description:

In this programming assignment I was tasked with building an HTML web server using Python. This web server sends an HTML file through a socket which is then displayed in a web browser. Below you can find my code and screenshots of my browser to prove successful completion of this assignment.

Web Server Code:

```python
#Import socket
from socket import *

#Prepare a server socket at this address
serverSocket = socket(AF_INET, SOCK_STREAM)
serverPort = 1234
serverSocket.bind(('',serverPort))

#Queue up to a maximum of 1 request
serverSocket.listen(1)

while True:
    try:
        #Print that the server is ready to service requests
        print("Ready to serve...")

        #Accept any requests and decode them
        connectionSocket, addr = serverSocket.accept()
        msg = connectionSocket.recv(1024).decode()
        fileName = msg.split()[1]

        #Slice '/' from filename and attempt to open
        f  = open(fileName[1:])
        outputData = f.readlines()

        #File was found... send OK response with content type header
        connectionSocket.send('HTTP/1.1 200 OK\n'.encode())
        connectionSocket.send('Content-Type: text/html\n\n'.encode())

        #Feed file into socket
        for line in range (0, len(outputData)):
            connectionSocket.send(outputData[line].encode())
        #Close the file
        f.close()

    #The server couldn't find the file, send a 404
    except IOError:
        print(fileName + " not found")
        connectionSocket.send('HTTP/1.1 404 NOT FOUND\n'.encode())

    #If server failed to parse the message
    except IndexError:
        print("Failed to index msg: " + msg)

    #Close the connection
    connectionSocket.close()
```

Screenshots:



192.168.68.106:1234/HelloWorld.html

Hello World!

General

Request URL: http://192.168.68.106:1234/HelloWorld.html
Request Method: GET
Status Code: 200 OK
Remote Address: 192.168.68.106:1234
Referrer Policy: strict-origin-when-cross-origin

Response Headers

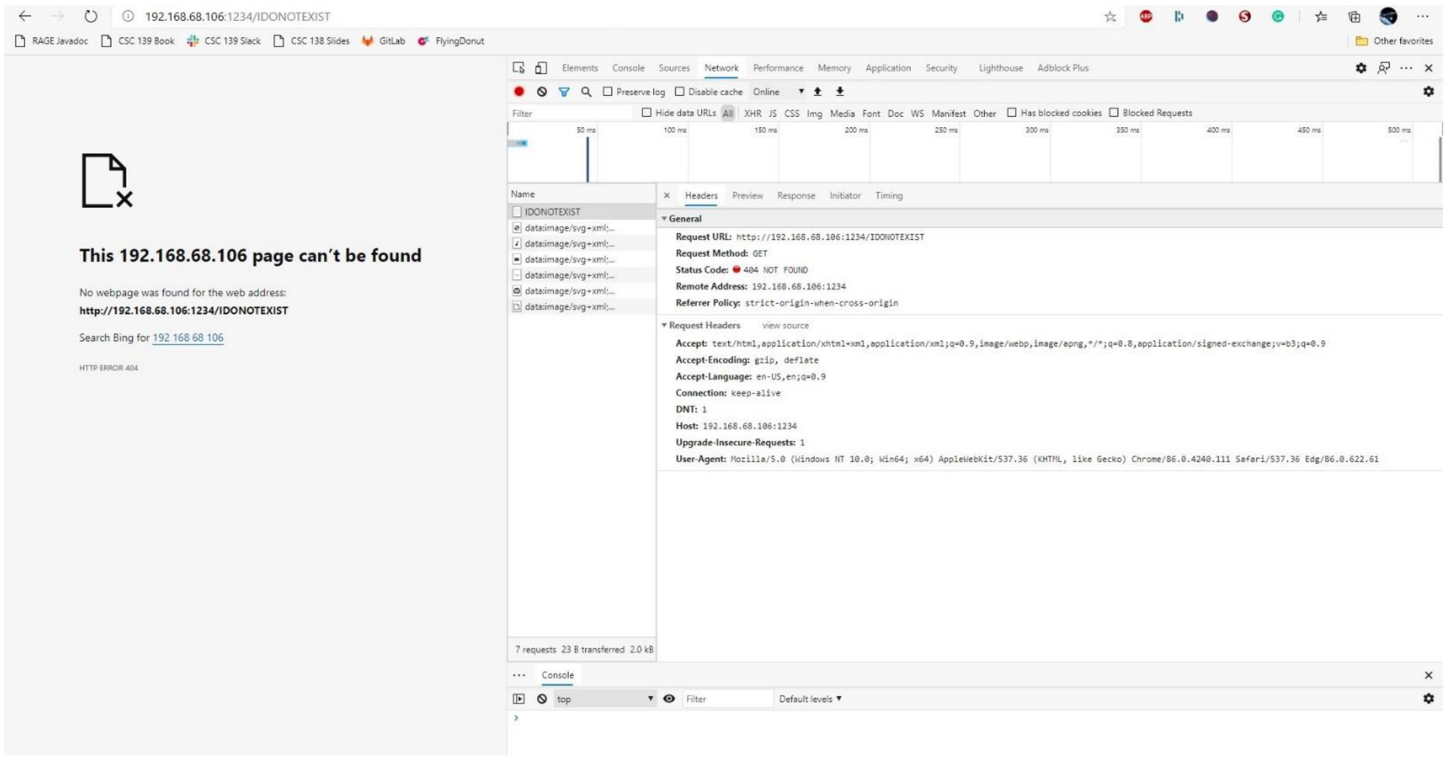
Content-Type: text/html

Request Headers

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cache-Control: max-age=0
Connection: keep-alive
DNT: 1
Host: 192.168.68.106:1234
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/86.0.4240.111 Safari/537.36 Edg/86.0.622.61

1 requests 180 B transferred 139 B

(HelloWorld.html being displayed in the browser after receiving the file)



This 192.168.68.106 page can’t be found

No webpage was found for the web address: **http://192.168.68.106:1234/IDONOTEXIST**

Search Bing for 192 168 68 106

HTTP ERROR 404

```
General
Request URL: http://192.168.68.106:1234/IDONOTEXIST
Request Method: GET
Status Code: 404 NOT FOUND
Remote Address: 192.168.68.106:1234
Referrer Policy: strict-origin-when-cross-origin

Request Headers
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: keep-alive
DNT: 1
Host: 192.168.68.106:1234
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/86.0.4240.111 Safari/537.36 Edg/86.0.622.61
```

(404 response being displayed after requesting a nonexistent file)

OPTIONAL:

In addition to the required deliverables, two optional exercises were outlined. The first was to create a version of the server where the client request was handled in another thread. The second, was to create a client that could communicate with the server and print out the received message. This client takes command line arguments. The code can be seen below respectively, along with an example of the client displaying the response message received from the threaded server.

Threaded Server Code:

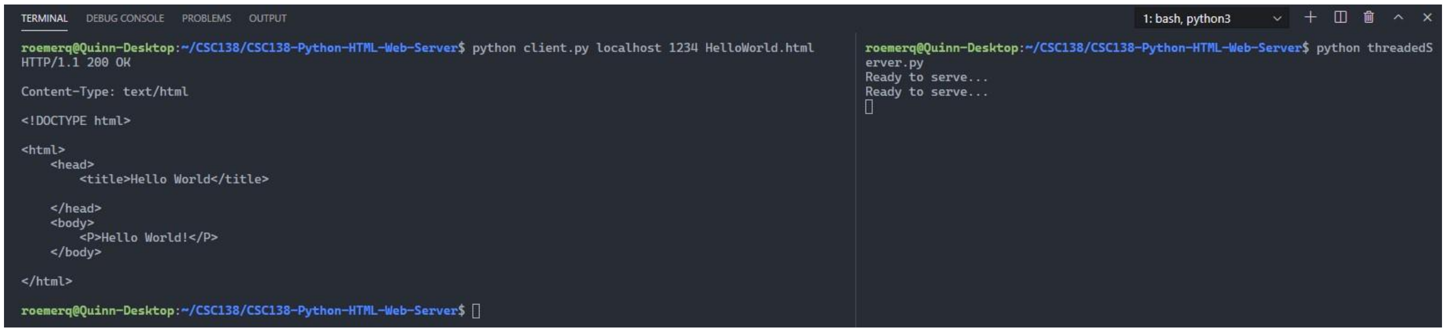
```
#Import socket
from socket import *
import _thread

#Function ran by a thread
def serviceClient(clientSocket):
    try:
        #Receive data from passed socket
        msg = connectionSocket.recv(1024).decode()
        fileName = msg.split()[1]

        #Slice '/' from filename and attempt to open
        f  = open(fileName[1:])
        outputData = f.readlines()

        #File was found... send OK response with content type header
        clientSocket.send('HTTP/1.1 200 OK\n'.encode())
        clientSocket.send('Content-Type: text/html\n\n'.encode())

        #Feed file into socket
        for line in range (0, len(outputData)):
            clientSocket.send(outputData[line].encode())

        #Close the file
        f.close()

    #The server couldn't find the file, send a 404
    except IOError:
        print(fileName + " not found")
        clientSocket.send('HTTP/1.1 404 NOT FOUND\n'.encode())

    #If server failed to parse the message
    except IndexError:
        print("Failed to index msg: " + msg)

    #Close the connection
    clientSocket.close()

#Prepare a server socket at this address
serverSocket = socket(AF_INET, SOCK_STREAM)
serverPort = 1234
serverSocket.bind(('',serverPort))

#Queue up to a maximum of 5 requests
serverSocket.listen(5)

while True:
    #Print that the server is ready to service requests
    print("Ready to serve...")

    #Accept any requests and decode them
    connectionSocket, addr = serverSocket.accept()

    #Spin up a new thread to service the received request
    _thread.start_new_thread(serviceClient, (connectionSocket,))
```
(Threaded web server code. Requires Python3+)

Client Code:

```
from socket import *
import sys

#Read command line arguments, make sure 4 were sent
if (len(sys.argv) != 4):
    print("Usage: client.py server_host server_port filename")
    exit()

try:
    #Create socket
    clientSocket = socket(AF_INET, SOCK_STREAM)
    clientSocket.connect((sys.argv[1], int(sys.argv[2])))

    #Send an HTTP GET request
    msg = 'Get /' + sys.argv[3] + ' HTTP/1.1'
    clientSocket.send(msg.encode())

    #Keep receiving until an empty string
    while True:
        modifiedSentence = clientSocket.recv(1024)
        print(modifiedSentence.decode())
        if not modifiedSentence:
            break;

    #Close the connection
    clientSocket.close()

except Exception:
    print("An error occured... Does the server exist?")
```

(Client code. Prints out the received message)

Output:



```
roemerq@Quinn-Desktop:~/CSC138/CSC138-Python-HTML-Web-Server$ python client.py localhost 1234 HelloWorld.html
HTTP/1.1 200 OK

Content-Type: text/html

<!DOCTYPE html>

<html>
    <head>
        <title>Hello World</title>

    </head>
    <body>
        <P>Hello World!</P>
    </body>

</html>

roemerq@Quinn-Desktop:~/CSC138/CSC138-Python-HTML-Web-Server$ 
```

```
roemerq@Quinn-Desktop:~/CSC138/CSC138-Python-HTML-Web-Server$ python threadedS
erver.py
Ready to serve...
Ready to serve...
```

(Threaded Server responding to a message from the client code which is then printed out)