ENSF 462

Laboratory 2

Title: Web Server and UDP Pinger

Name: Hongwoo Yoon

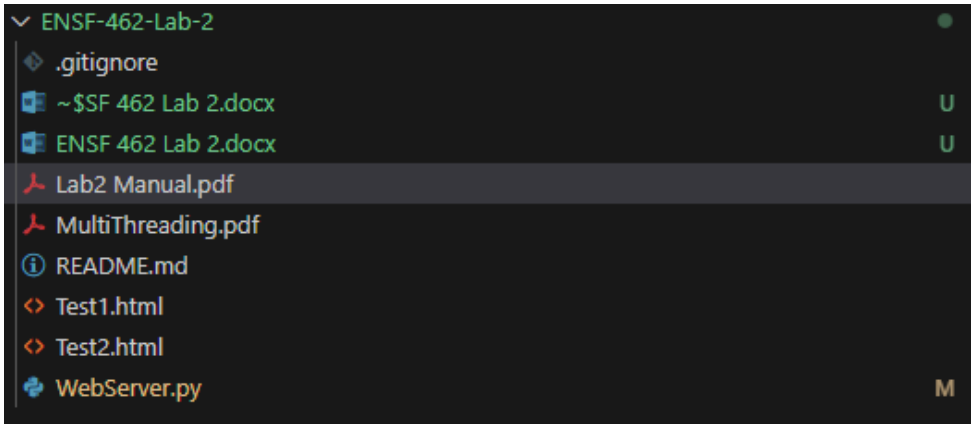
UCID: 30113779

Section: B02

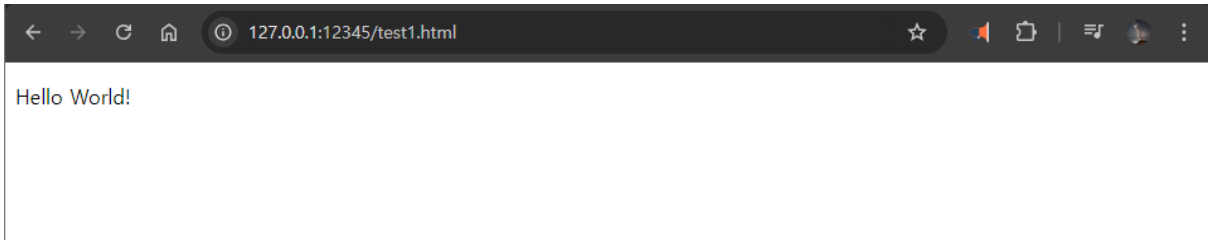
Date: Oct 9, 2024

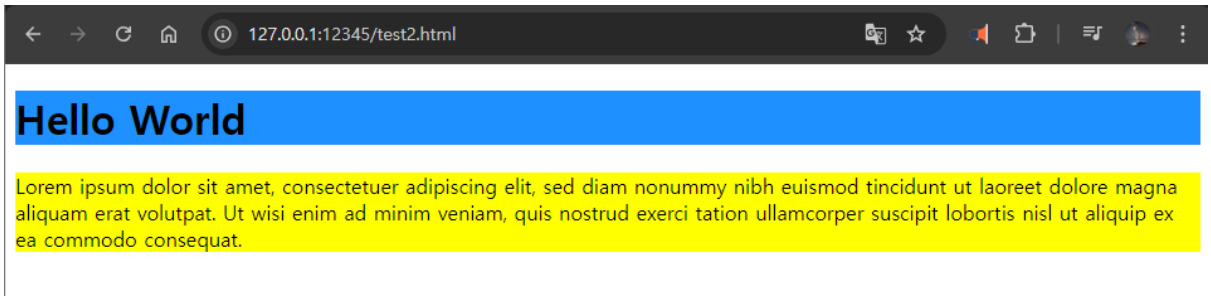
Part 1 Web Server

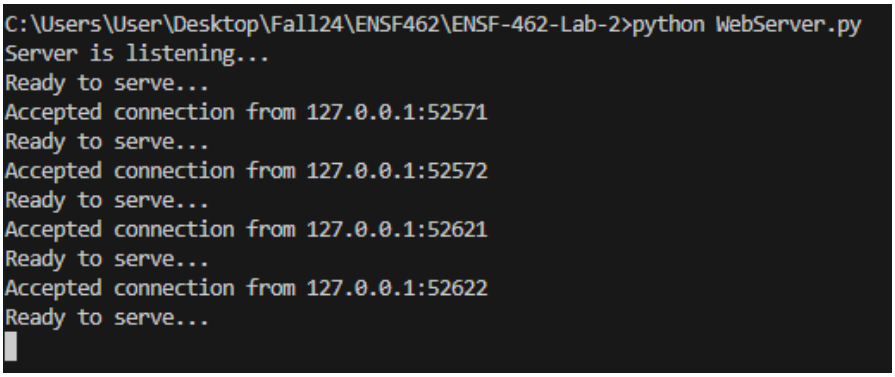
Server Directory



Output







Code

```
#import socket module
from socket import *
import threading

def handle_client(connectionSocket, addr):
    try:
        print(f"Accepted connection from {addr[0]}:{addr[1]}")
        message =  connectionSocket.recv(1024)
        filename = message.split()[1]
        f = open(filename[1:])

        outputdata = f.read()

        #Send one HTTP header line into socket
        connectionSocket.send('HTTP/1.x 200 OK\r\n'.encode())
        connectionSocket.send('Content-Type: text/html\r\n\r\n'.encode())
        #Send the content of the requested file to the client
        for i in range(0, len(outputdata)):
            connectionSocket.send(outputdata[i].encode())
        connectionSocket.send("\r\n".encode())
        connectionSocket.close()
    except IOError:
        #Send response message for file not found
        connectionSocket.send('404 Page Not Found\r\n'.encode())
        connectionSocket.send('Content-Type: text/html\r\n\r\n'.encode())
        connectionSocket.send('<html><body><h1>404 Page Not
Found</h1></body></html>'.encode())

        #Close client socket
        connectionSocket.close()

serverSocket = socket(AF_INET, SOCK_STREAM)
#Prepare a sever socket
serverSocket.bind(('127.0.0.1', 12345))
serverSocket.listen(1)
print("Server is listening...")
while True:
    #Establish the connection
    print('Ready to serve...')
    connectionSocket, addr =  serverSocket.accept()

    thread = threading.Thread(target=handle_client, args=(connectionSocket,
addr))
    thread.start()

serverSocket.close()
```

Part 2

```
C:\Users\User\Desktop\Fall24\ENSF462\ENSF-462-Lab-2>python client.py
Sequence Number: 1
Response: PING 1 1728580538.7470329
Round-Trip Time: 0.0004923343658447266
Sequence Number: 2
Request time out
Sequence Number: 3
Response: PING 3 1728580539.7582645
Round-Trip Time: 0.00040268898010253906
Sequence Number: 4
Request time out
Sequence Number: 5
Response: PING 5 1728580540.7672334
Round-Trip Time: 0.0003063678741455078
Sequence Number: 6
Request time out
Sequence Number: 7
Response: PING 7 1728580541.7704954
Round-Trip Time: 0.00019073486328125
Sequence Number: 8
Response: PING 8 1728580541.7710226
Round-Trip Time: 0.000102996826171875
Sequence Number: 9
Response: PING 9 1728580541.7713974
Round-Trip Time: 9.369850158691406e-05
Sequence Number: 10
Response: PING 10 1728580541.7717643
Round-Trip Time: 8.559226989746094e-05
Minimum RTT: 8.559226989746094e-05
Maximum RTT: 0.0004923343658447266
Average RTT: 0.00023920195443289622
Packet loss rate: 30.0%
```

```
from socket import *
import time
import numpy as np

UDP_IP = "127.0.0.1"
UDP_PORT = 12000
times = []
success = 0
fail = 0
#message = Ping sequence_number time
#sequence num starts from 1 and ends at 10.
#time = when packet is sent to the server
socket = socket(AF_INET, SOCK_DGRAM)
socket.settimeout(1)
for sequence_number in range(1, 11):
    start = time.time()
    socket.sendto((f'Ping {sequence_number} {time.time()}').encode(), (UDP_IP,
UDP_PORT))
    try:
        response, address = socket.recvfrom(1024)
        time_taken = time.time() - start
        times.append(time_taken)
        print(f'Sequence Number: {sequence_number}')
        print(f'Response: {response.decode()}')
        print(f'Round-Trip Time: {time_taken}')
    except timeout:
        print(f'Sequence Number: {sequence_number}')
        print("\033[91mRequest time out\033[0m")
        fail += 1
loss_rate = ((fail / 10) * 100)
print(f'Minimum RTT: {np.min(times)}')
print(f'Maximum RTT: {np.max(times)}')
print(f'Average RTT: {np.mean(times)}')
print(f'Packet loss rate: {loss_rate}%')
socket.close()
```