COMPUTER NETWORKS LAB –WEEK 5

Name: Tushar Y S

SRN: PES1UG19CS545

**Socket Programming**

**1.1 TCP Connection**

**1.1.1 TCP Server**

from socket import \*

serverPort = 12000

serverSocket = socket(AF\_INET,SOCK\_STREAM) serverSocket.bind(('',serverPort)) serverSocket.listen(1)

print("The server is ready to receive")

while 1:

connectionSocket, addr = serverSocket.accept() sentence = connectionSocket.recv(1024) capitalizedSentence = sentence.upper() connectionSocket.send(capitalizedSentence) connectionSocket.close()

**1.1.2 TCP Client**

from socket import \* serverName = '' serverPort = 12000

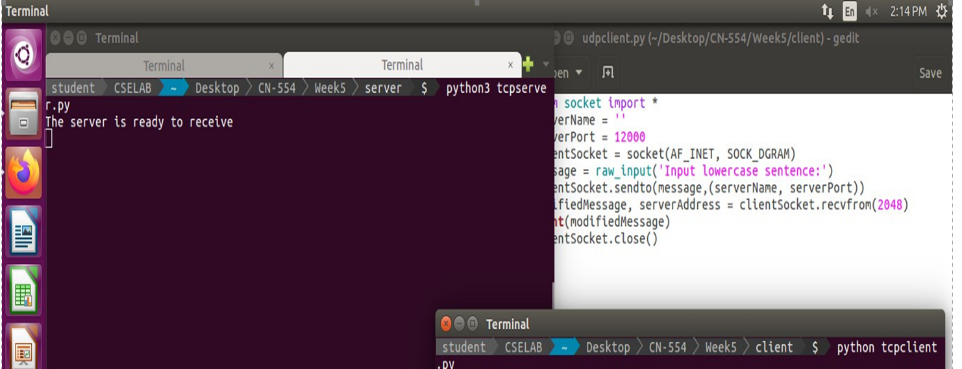
clientSocket = socket(AF\_INET, SOCK\_STREAM) clientSocket.connect((serverName,serverPort)) sentence = raw\_input('Input lowercase sentence:') clientSocket.send(sentence)

modifiedSentence = clientSocket.recv(1024)

print('From Server:', modifiedSentence)

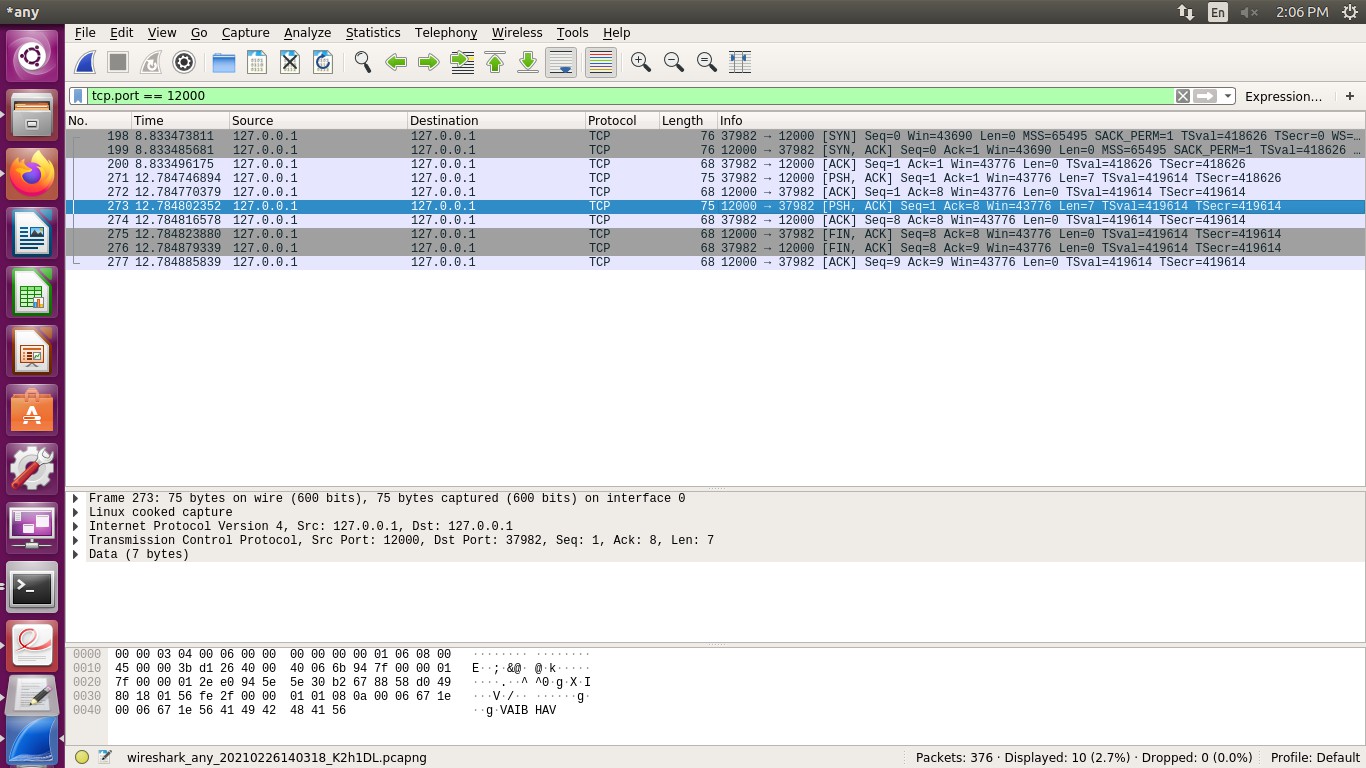
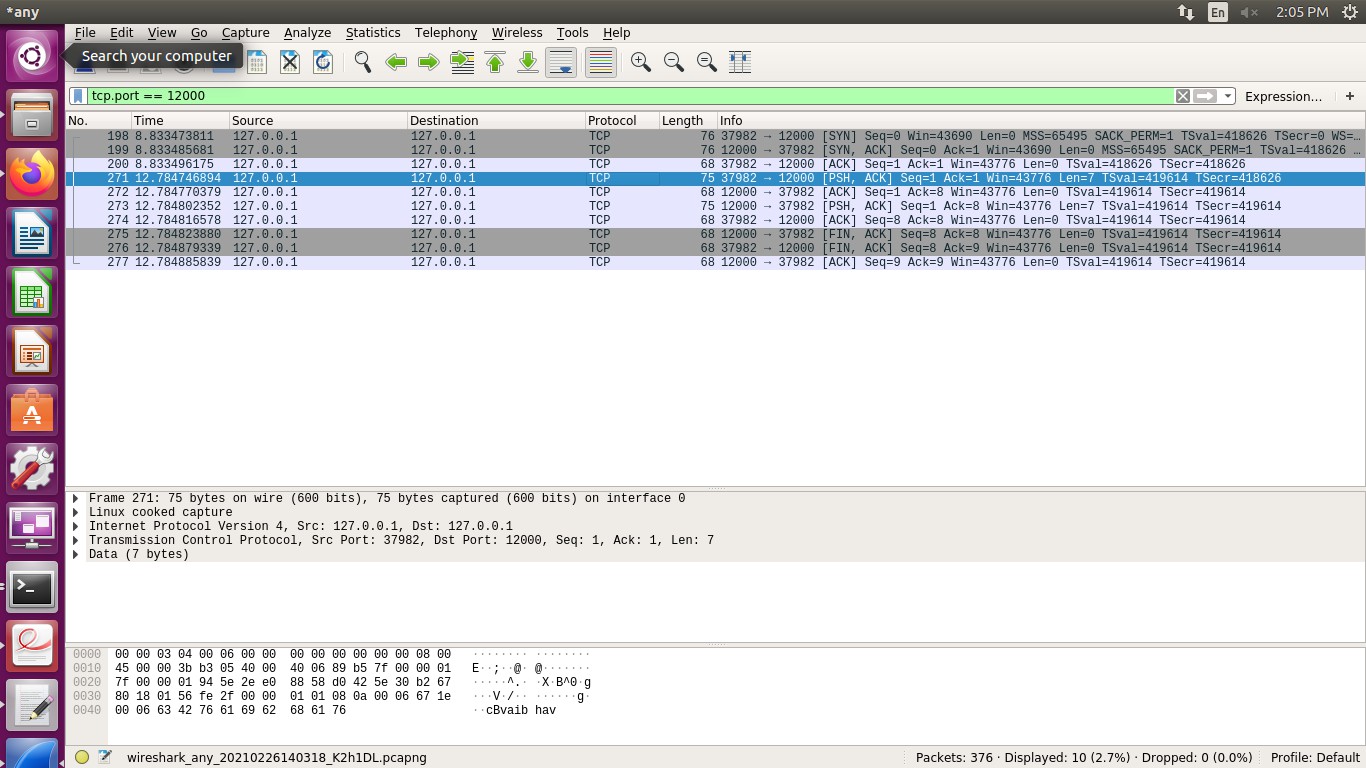
clientSocket.close()

**1.1.3 TCP Connection between Server and Client**



TCP Server and Client

**1.1.4 Wireshark Capture for TCP Connection**



**1.2 UDP Connection**

**1.2.1 UDP Server**

from socket import \*

serverPort = 12000

serverSocket = socket(AF\_INET, SOCK\_DGRAM) serverSocket.bind(('', serverPort)) print("The server is ready to receive") while 1:

message, clientAddress = serverSocket.recvfrom(2048) modifiedMessage = message.upper() serverSocket.sendto(modifiedMessage, clientAddress)

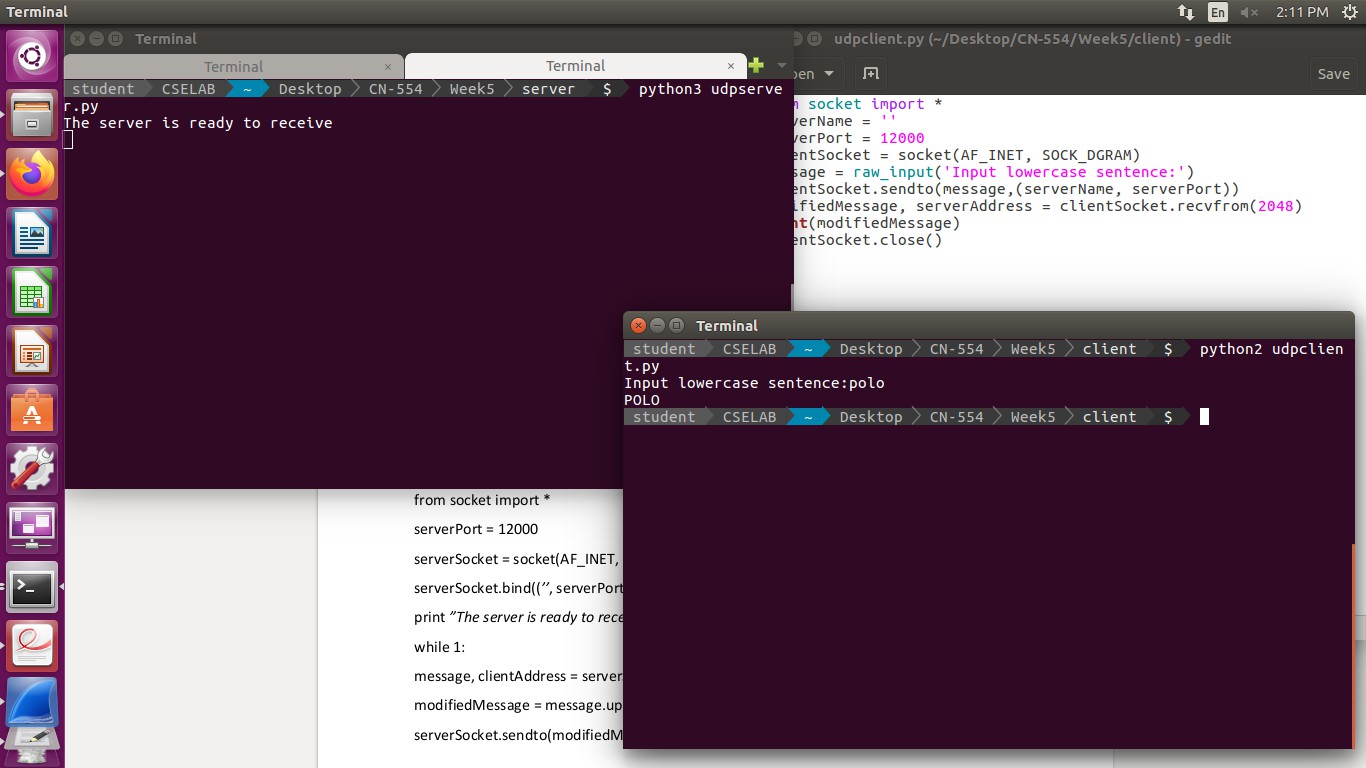
**1.2.2 UDP Client**

from socket import \* serverName = '' serverPort = 12000

clientSocket = socket(AF\_INET, SOCK\_DGRAM) message = raw\_input('Input lowercase sentence:') clientSocket.sendto(message,(serverName, serverPort)) modifiedMessage, serverAddress = clientSocket.recvfrom(2048) print(modifiedMessage)

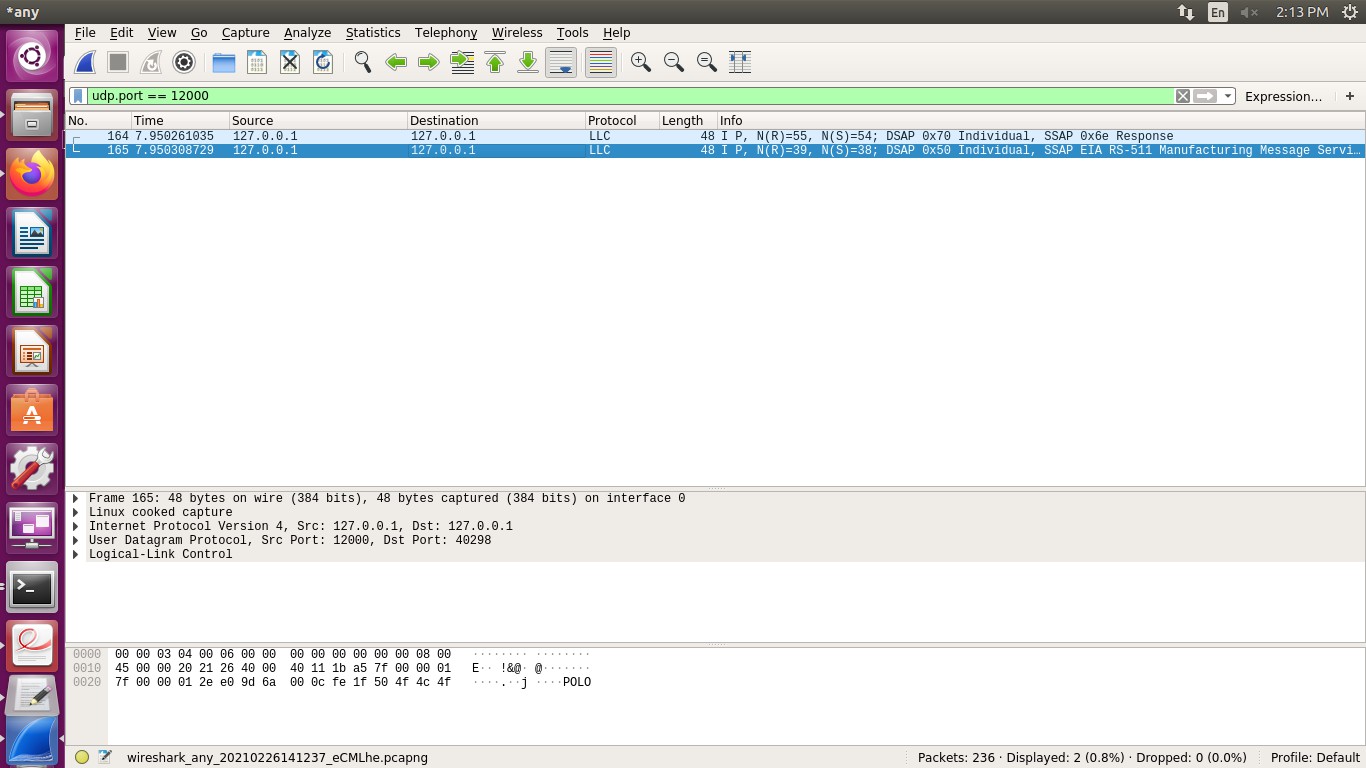
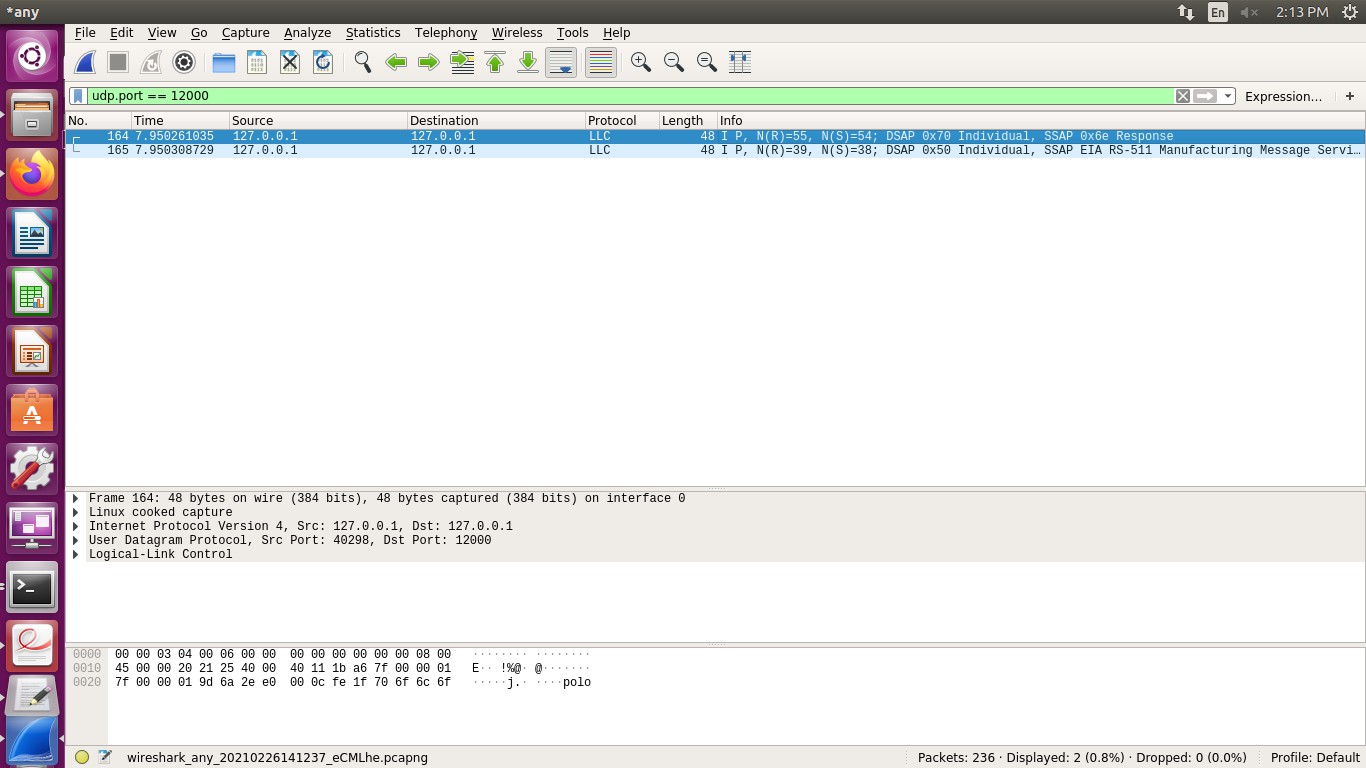
clientSocket.close()

**1.2.3 UDP Connection between Server and Client**



UDP Server and Client

**1.2.4 Wireshark Capture for UDP Connection**



**1. Task 2 – Web Server**

**2.1 Setting up the Web Server**

• After making relevant changes in the skeleton code (adding IPv4 address of the machine)

# Import socket module from socket import \*

# Create a TCP server socket

#(AF\_INET is used for IPv4 protocols)

#(SOCK\_STREAM is used for TCP)

serverSocket = socket(AF\_INET, SOCK\_STREAM)

# Assign a port number

serverPort = 6789

# Bind the socket to server address and server port

serverSocket.bind(("10.2.20.198", serverPort))

# Listen to at most 1 connection at a time

serverSocket.listen(1)

# Server should be up and running and listening to the incoming connections while True:

print 'Ready to serve...'

# Set up a new connection from the client

connectionSocket, addr = serverSocket.accept()

fer

# If an exception occurs during the execution of try clause

# the rest of the clause is skipped

# If the exception type matches the word after except

# the except clause is executed try:

# Receives the request message from the client message = connectionSocket.recv(1024)

# Extract the path of the requested object from the message

# The path is the second part of HTTP header, identified by [1]

filename = message.split()[1]

# Because the extracted path of the HTTP request includes

# a character '\', we read the path from the second character f = open(filename[1:])

# Store the entire contenet of the requested file in a temporary buf

outputdata = f.read()

# Send the HTTP response header line to the connection socket connectionSocket.send("HTTP/1.1 200 OK\r\n\r\n")

# Send the content of the requested file to the connection socket for i in range(0, len(outputdata)):

connectionSocket.send(outputdata[i])

connectionSocket.send("\r\n")

# Close the client connection socket connectionSocket.close()

except IOError:

# Send HTTP response message for file not found connectionSocket.send("HTTP/1.1 404 Not Found\r\n\r\n") connectionSocket.send("<html><head></head><body><h1>404 Not Found</h

1></body></html>\r\n")

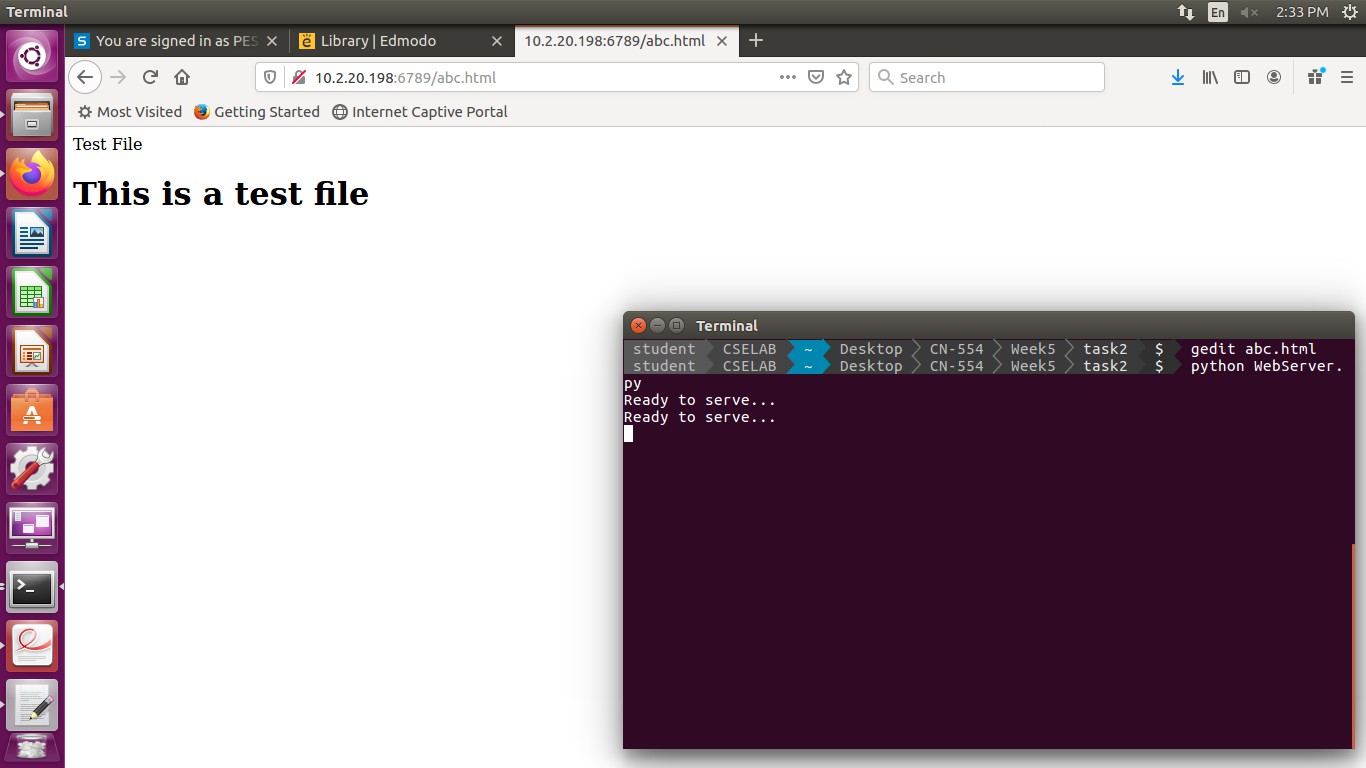
# Close the client connection socket connectionSocket.close()

serverSocket.close()

**2.2 File Present on the Server**

• The IP address along with path of the test file is entered in the browser after running the web server.

• Since the file actually exists on the server, we are able to get the requested file.

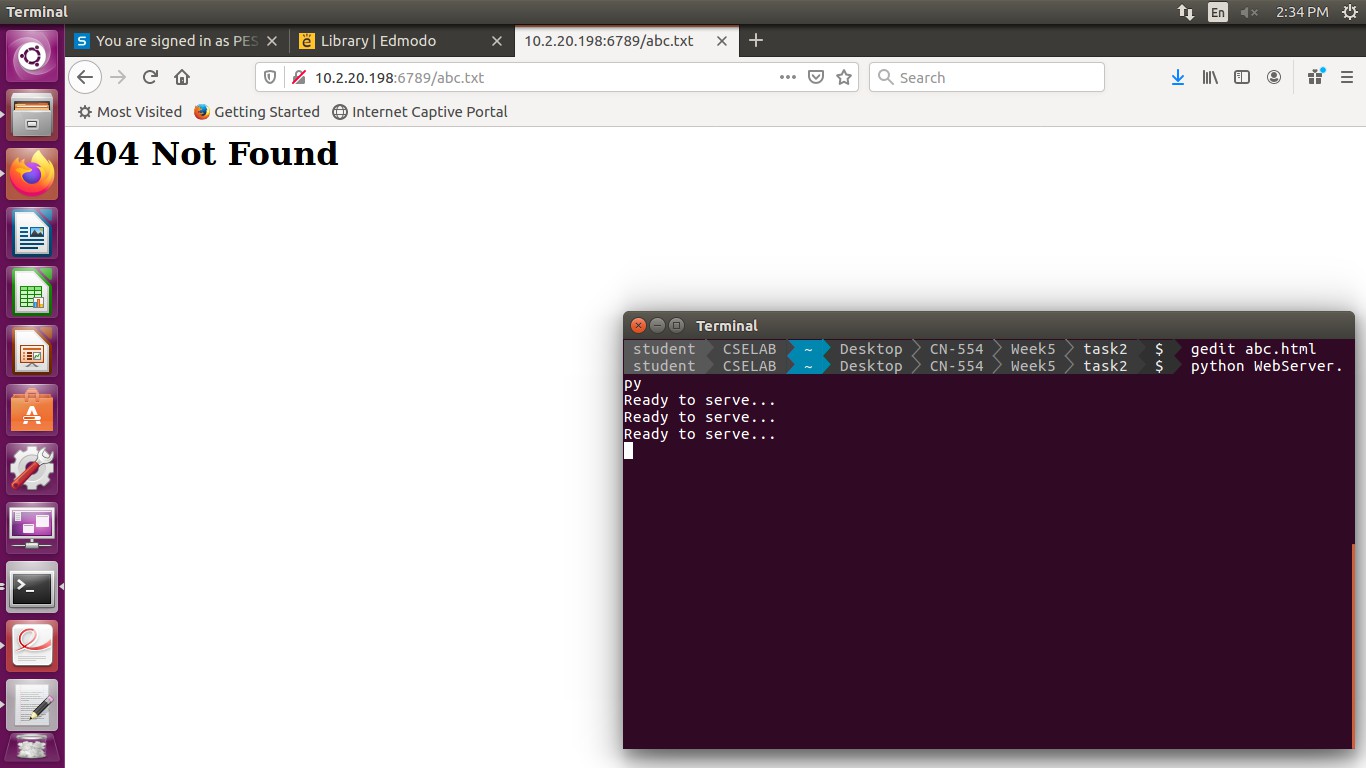


Response of Web Server when we access a file present on the server.

**2.3 File not present on the Server**

• The IP address along with the path of the wrong file is entered in the browser.

• Since the file not exists on the server, we get a 404 Not Found response by the Web Server



Response of Web Server when we access a file not on the server

**2.4 Wireshark Capture for Web Server**

