CSE 476
MOBILE COMMUNICATION NETWORKS

MIDTERM PROJECT

FINAL REPORT

ÖMER ÇEVİK
161044004

1. Assignment 1: Web Server

In this assignment, main idea is to create a web server which serves an HTML file. I used the skeleton of web server which is shared as a lab of Web Server for this assignment and filled the empty places.

First of all, creating a socket in Python programming language is in **socket[11]** library and using **socket(…)[13]** returns a socket. To use TCP socket we give parameter **SOCK_STREAM[13]** to socket function. We use Web Server so it needs internet connection we use **AF_INET[13]** for socket parameter as address from the internet.

After server socket is created then we create the **HOST** and **PORT** which are represented ***"0.0.0.0"*****[18]** as host address and ***"6789"*****[19]** as port number. Then socket is **bind**ing**[20]** to host with its port number and starting to **listen[21]** socket. Server is ready for serve**[28]** right now infinitely**[25]**. Then server **accept[29]** the requests, creates connection socket**[29]** and reads message from clients using **recv[32]** function on that connection socket. Clients requests the file name of example HTML file. In try catch block**[31, 50]** that happens. Because if the file is not exist**[35]** then we **send[54]** client ***"404 Not Found"*** information message on socket and closing connection socket**[61]**. If it is exist then it reads whole HTML file**[37]**, **send**ing **200 HTTP Success OK Code[42]** to client and outputs to client the HTML file using socket's **send** function**[47, 48]**. Then it closes connection socket**[49]** and closes server socket**[64]**.

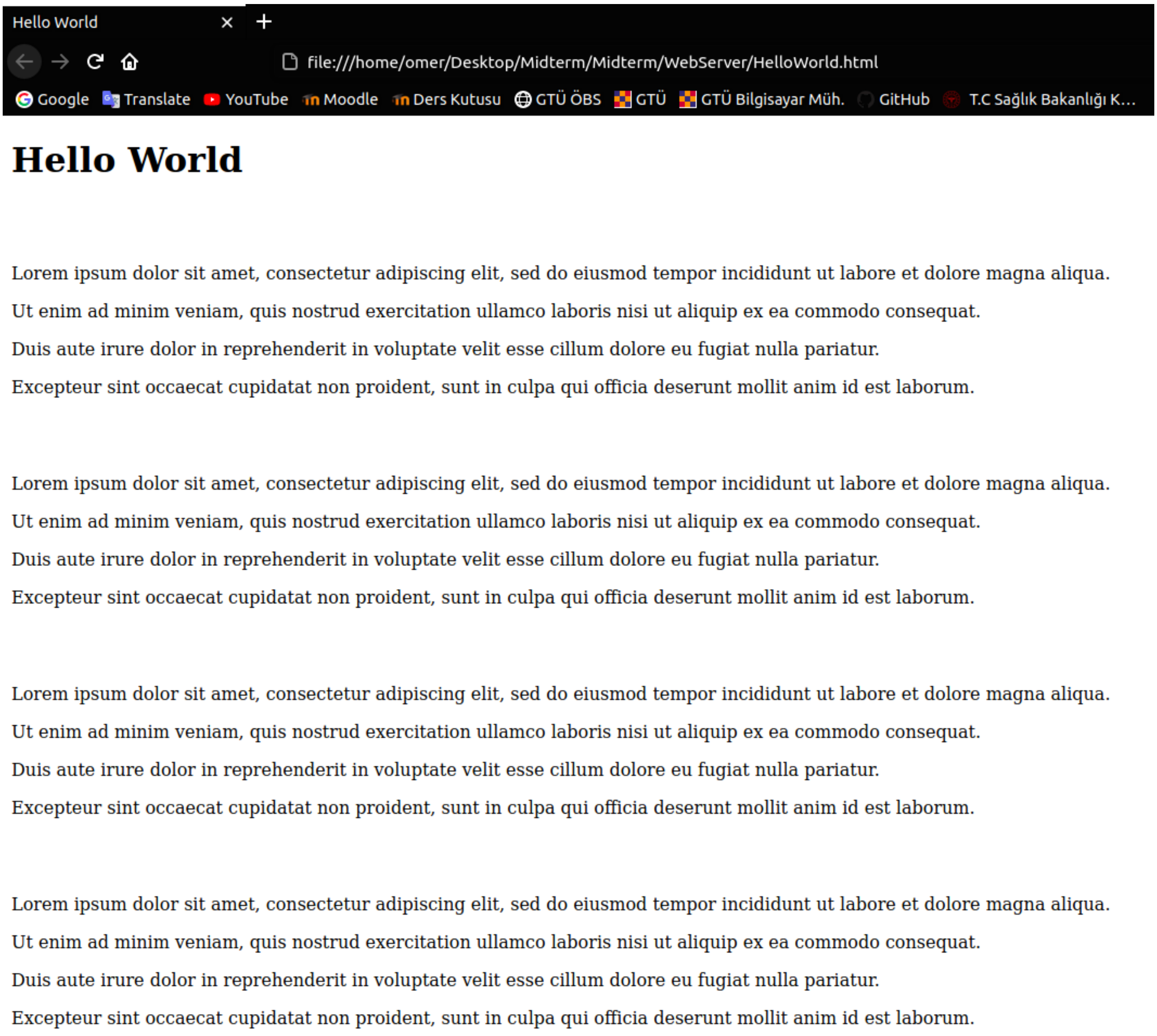
Notes:

- Code lines in WebServer.py are represented inside bold square brackets.
- Python 2.7.18 version is used to run code.
- Ubuntu 20.04.1 LTS operating system and its Terminal is workspace.
- To run code ***python WebServer.py*** command is realized in Terminal.
- No optional exercises are implemented in Web Server part.

WebServer.py :

```
"""
CSE 476 Mobile Communication Networks
Midterm Project
Web Server

WebServer.py
@author Omer CEVIK
161044004
"""
#import socket module
from socket import *

serverSocket = socket(AF_INET, SOCK_STREAM)

#Prepare a server socket
#Fill in start

HOST = '0.0.0.0'
PORT = 6789
serverSocket.bind((HOST, PORT))
serverSocket.listen(1)

#Fill in end

while True:

    #Establish the connection
    print('Ready to serve...')
    connectionSocket, addr = serverSocket.accept()

    try:
        message = connectionSocket.recv(1024)

        filename = message.split()[1]
        f = open(filename[1:])

        outputdata = f.read()

        #Send one HTTP header line into socket
        #Fill in start

        connectionSocket.send('HTTP/1.1 200 OK\r\n\r\n'.encode())

        #Fill in end

        #Send the content of the requested file to the client
        for i in range(0, len(outputdata)):
            connectionSocket.send(outputdata[i])
        connectionSocket.close()
    except IOError:
        #Send response message for file not found
        #Fill in start

        connectionSocket.send('404 Not Found')

        #Fill in end

        #Close client socket
        #Fill in start

        connectionSocket.close()

        #Fill in end
serverSocket.close()
```

Web Server Program Test Results :


Hello World
file:///home/omer/Desktop/Midterm/Midterm/WebServer/HelloWorld.html
Google
Translate
YouTube
Moodle
Ders Kutusu
GTÜ ÖBS
GTÜ
GTÜ Bilgisayar Müh.
GitHub
T.C Sağlık Bakanlığı K...

Hello World

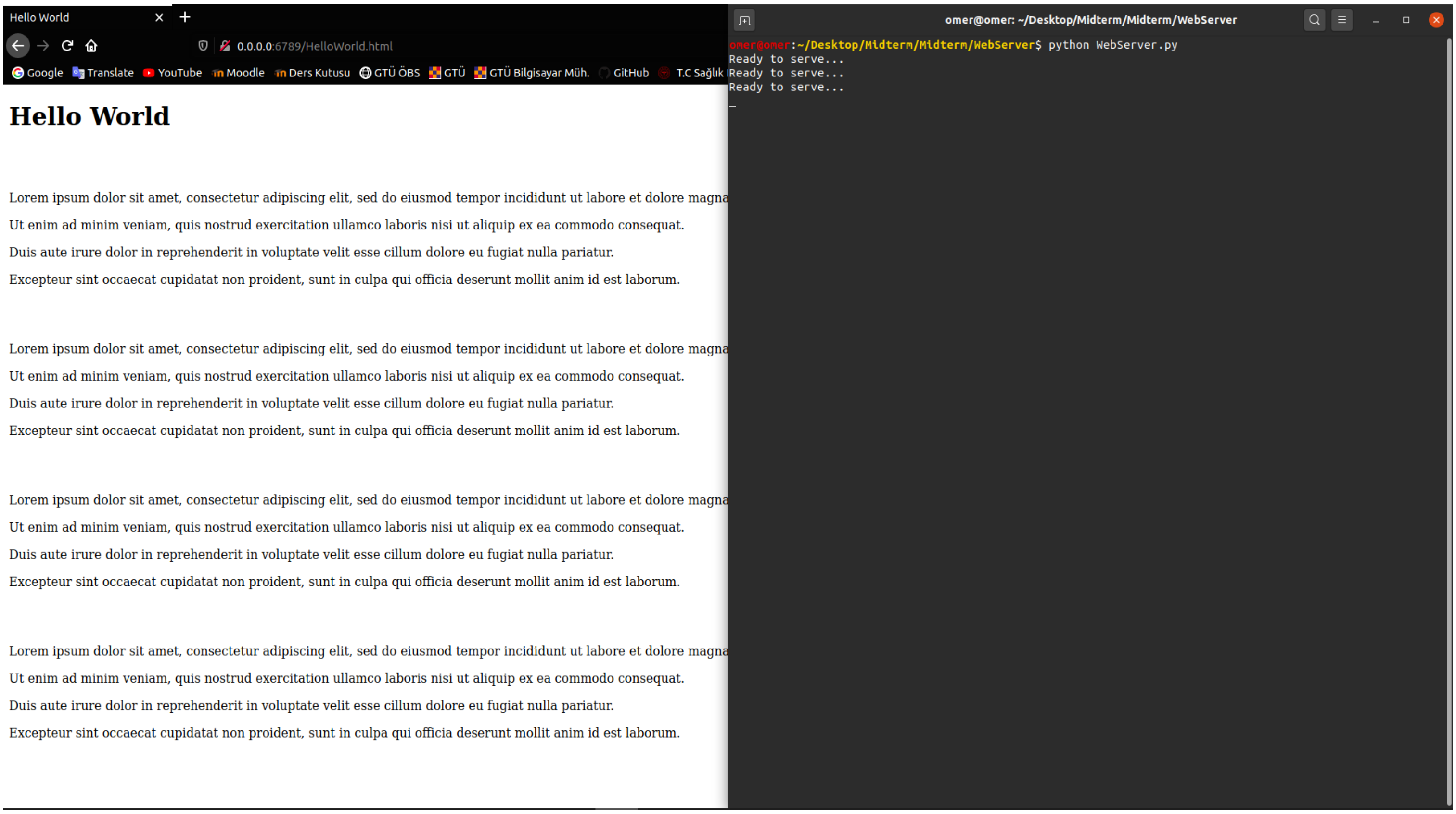
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.
Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.
Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.
Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.
Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.
Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.
Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.
Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.
Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

1. HelloWorld.html File



Hello World

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna
Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.
Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.
Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

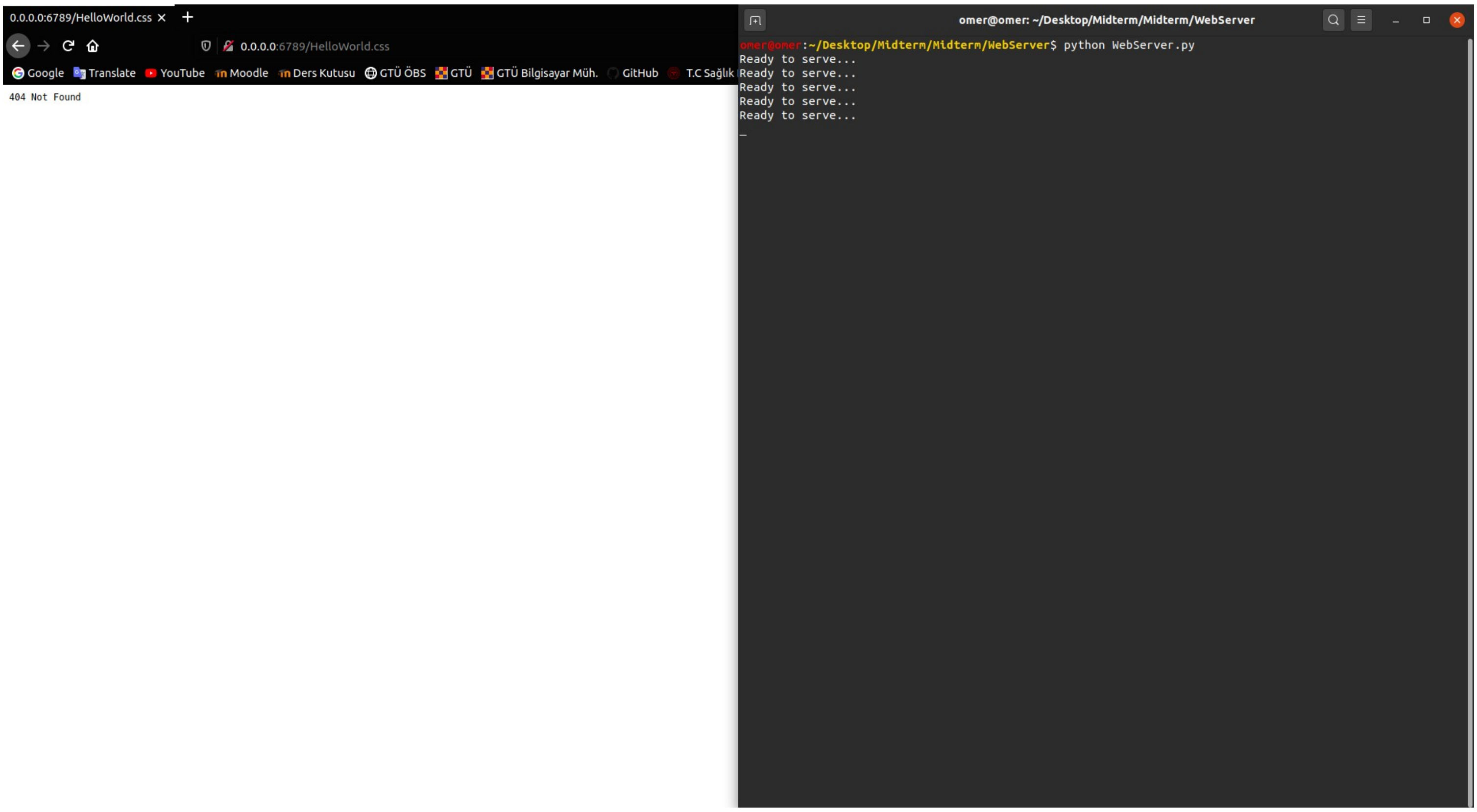
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna
Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.
Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.
Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna
Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.
Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.
Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna
Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.
Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.
Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

```
omer@omer:~/Desktop/Midterm/Midterm/WebServer$ python WebServer.py
Ready to serve...
Ready to serve...
Ready to serve...
_
```

2. Running Web Server and Browser Client Result


0.0.0.0:6789/HelloWorld.css
0.0.0.0:6789/HelloWorld.css
Google
Translate
YouTube
Moodle
Ders Kutusu
GTÜ ÖBS
GTÜ
GTÜ Bilgisayar Müh.
GitHub
T.C Sağlık
omer@omer: ~/Desktop/Midterm/Midterm/WebServer

```
404 Not Found
```

```
omer@omer:~/Desktop/Midterm/Midterm/WebServer$ python WebServer.py
Ready to serve...
Ready to serve...
Ready to serve...
Ready to serve...
Ready to serve...
_
```

3. Running Web Server and Browser Client Requests HelloWorld.css File

2. Assignment 2: UDP Pinger

In this assignment, main idea is to create a server-client relation using UDP between that processes. UDPPingerServer.py is attached to assignment file as lab work which creates random integer values and if the value is less then 4 it keeps creating random values**[31]**. Otherwise, server reads the message**[26]** from socket which is hosted in ***"127.0.0.1"*** in "***12000"*** port number and makes the message upper case each character**[20]**. If random value is **not** less than 4 then it sends the upper cased message to client which is connected by socket**[34]**. To make the protocol as UDP is provided using **SOCK_DGRAM** while creating server's socket in UDPPingerServer.py**[18]**.

In client side which is inside UDPPingerClient.py, host address and port number are set to ***"127.0.0.1"*****[17]**, ***"12000"*****[18]**. To read from socket buffer, initialized the buffer size as 1024**[19]**. The ping message is declared as **"Ping Message "[23]** and it is encoded**[24]**. For request to server 10 times is declared using a message counter**[27]**. To create a socket using UDP is provided using **SOCK_DGRAM** in client side too**[30]**. If server doesn't response in 1 second, the socket is set the time out for 1 second using **settimeout(1)** function of socket**[33]**. Then for each message request to server is applied using **sendto()** function of socket passing parameter as encoded ping message**[42]**. While doing that, using **time** library it starts to count **RTT** time**[39]** before sending ping message then tries to get **recvfrom** server**[46]**. If the time out is realized then it will be catched in **except[55]** and printing the message **"Request timed out"[57]**. If everything is fine, then it gets the end of **RTT** time**[49]** and distinct of end and starting time of RTT represents the message's RTT time**[50]**. After evaluating RTT time, printing the message which is upper cased and received by server, message counter and distinct RTT time**[53, 54]**. Then counting the messages for 10 times**[60]**. Finally, closing UDP socket**[63]**.

Notes:

- Code lines in UDPPingerServer.py and UDPPingerClient.py are represented inside bold square brackets.
- Python 2.7.18 version is used to run code.
- Ubuntu 20.04.1 LTS operating system and its Terminal is workspace.
- To run UDPPingerServer code ***python UDPPingerServer.py*** command is realized in Terminal.
- To run UDPPingerClient code ***python UDPPingerClient.py*** command is realized in another Terminal.
- No optional exercises are implemented in UDP Pinger Server/Client part.

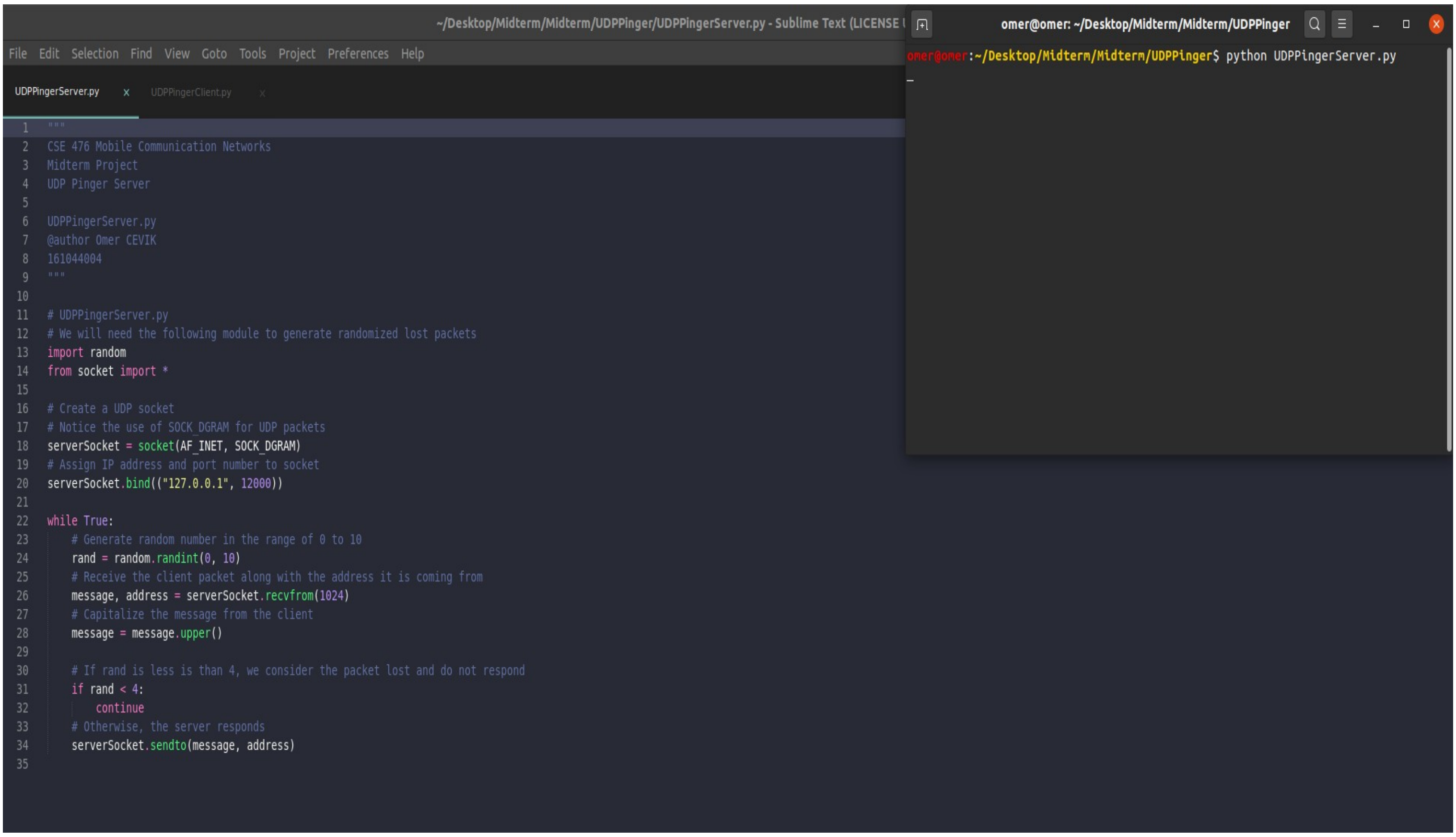
UDPPingerServer.py :

```python
"""
CSE 476 Mobile Communication Networks
Midterm Project
UDP Pinger Server

UDPPingerServer.py
@author Omer CEVIK
161044004
"""

# UDPPingerServer.py
# We will need the following module to generate randomized lost packets
import random
from socket import *

# Create a UDP socket
# Notice the use of SOCK_DGRAM for UDP packets
serverSocket = socket(AF_INET, SOCK_DGRAM)
# Assign IP address and port number to socket
serverSocket.bind(("127.0.0.1", 12000))

while True:
    # Generate random number in the range of 0 to 10
    rand = random.randint(0, 10)
    # Receive the client packet along with the address it is coming from
    message, address = serverSocket.recvfrom(1024)
    # Capitalize the message from the client
    message = message.upper()

   # If rand is less is than 4, we consider the packet lost and do not respond
    if rand < 4:
        continue
    # Otherwise, the server responds
    serverSocket.sendto(message, address)

# Closing socket.
serverSocket.close()
```

UDPPingerClient.py :

```
"""
CSE 476 Mobile Communication Networks
Midterm Project
UDP Pinger Client

UDPPingerClient.py
@author Omer CEVIK
161044004
"""

# Import time, socket, UDP protocol
from socket import AF_INET, SOCK_DGRAM
import time
import socket

# Init host as localhost, port, buffersize as 1024 and server address.
HOST = "127.0.0.1"
PORT = 12000
bufferSize = 1024
serverAddressPort = (HOST, PORT)

# Initialize simple ping message of client and encode it.
pingMsg = "Ping Message "
bytedPingMsg = pingMsg.encode()

# Message counter initialization with 1.
messageCounter = 1

# Creating UDP client socket.
UDPClientSocket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

# Setting the wait time to 1 second.
UDPClientSocket.settimeout(1)

# Creating a loop for 10 messages of client.
while messageCounter <= 10:

    # RTT time begin.
    start_rtt = time.time()

    # Sending client byted message to server.
    UDPClientSocket.sendto(bytedPingMsg, serverAddressPort)

    try:
        # Waiting response of server on UDP socket.
        msgFromServer = UDPClientSocket.recvfrom(bufferSize)

        # RTT time end and distinct of start-end time of message trip.
        end_rtt = time.time()
        distinct_rtt = end_rtt - start_rtt

        # Printing the message which is upper cased by server, message counter and RTT time.
        msg = str(msgFromServer[0]) + str(messageCounter) + " " + str(distinct_rtt)
        print(msg)
    except socket.timeout:
        # If time out is generated about 1 second then printing the information.
        print("Request timed out")

    # Increasing message counter.
    messageCounter += 1

# Closing socket.
UDPClientSocket.close()
```

UDPPingerServer and UDPPingerClient Program Test Results :

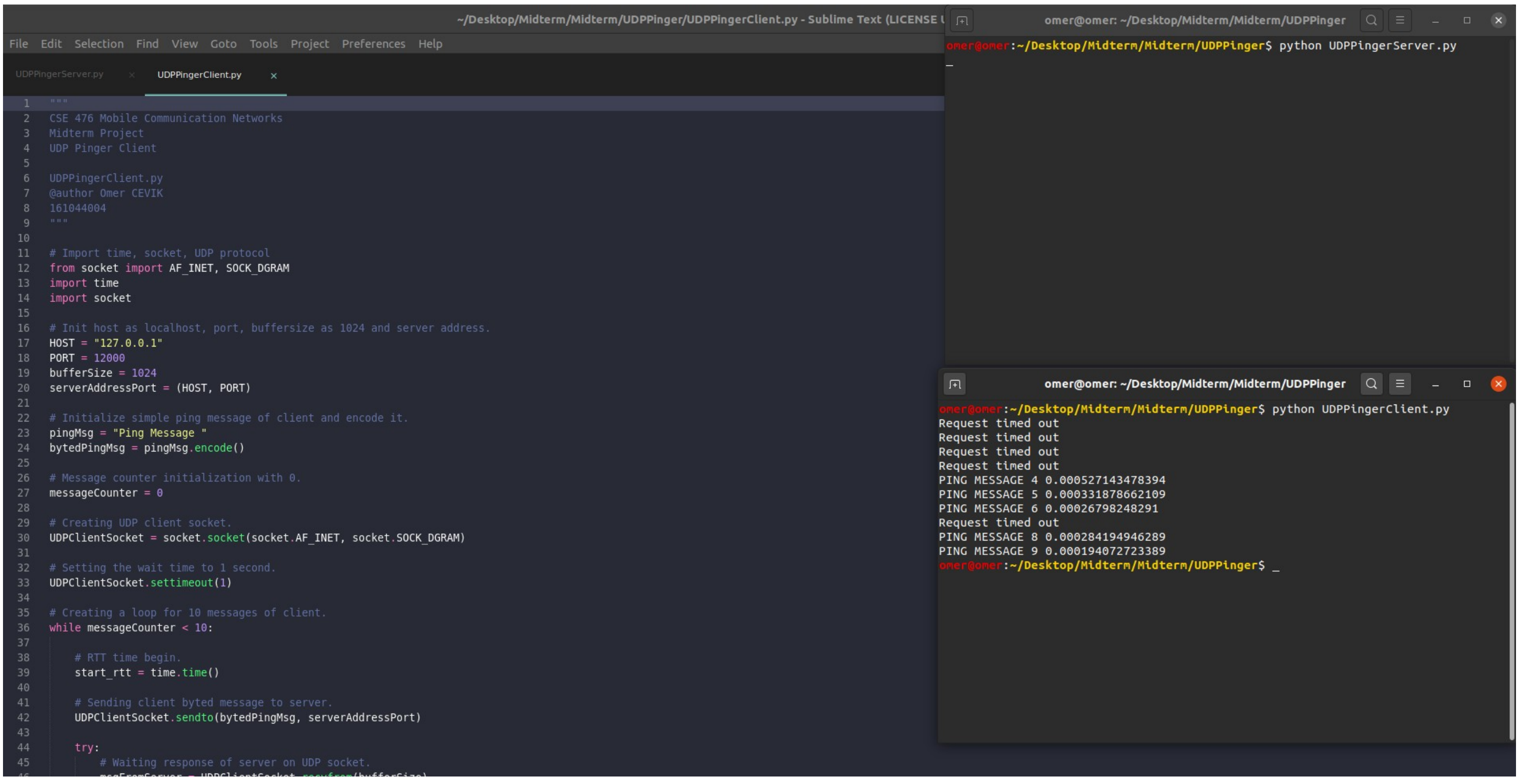


~/Desktop/Midterm/Midterm/UDPPinger/UDPPingerServer.py - Sublime Text (LICENSE U
File Edit Selection Find View Goto Tools Project Preferences Help
UDPPingerServer.py
UDPPingerClient.py
omer@omer: ~/Desktop/Midterm/Midterm/UDPPinger

```python
"""
CSE 476 Mobile Communication Networks
Midterm Project
UDP Pinger Server

UDPPingerServer.py
@author Omer CEVIK
161044004
"""

# UDPPingerServer.py
# We will need the following module to generate randomized lost packets
import random
from socket import *

# Create a UDP socket
# Notice the use of SOCK_DGRAM for UDP packets
serverSocket = socket(AF_INET, SOCK_DGRAM)
# Assign IP address and port number to socket
serverSocket.bind(("127.0.0.1", 12000))

while True:
    # Generate random number in the range of 0 to 10
    rand = random.randint(0, 10)
    # Receive the client packet along with the address it is coming from
    message, address = serverSocket.recvfrom(1024)
    # Capitalize the message from the client
    message = message.upper()

    # If rand is less is than 4, we consider the packet lost and do not respond
    if rand < 4:
        continue
    # Otherwise, the server responds
    serverSocket.sendto(message, address)

```

```
omer@omer:~/Desktop/Midterm/Midterm/UDPPinger$ python UDPPingerServer.py
_
```

1. Running UDPPingerServer.py in Terminal



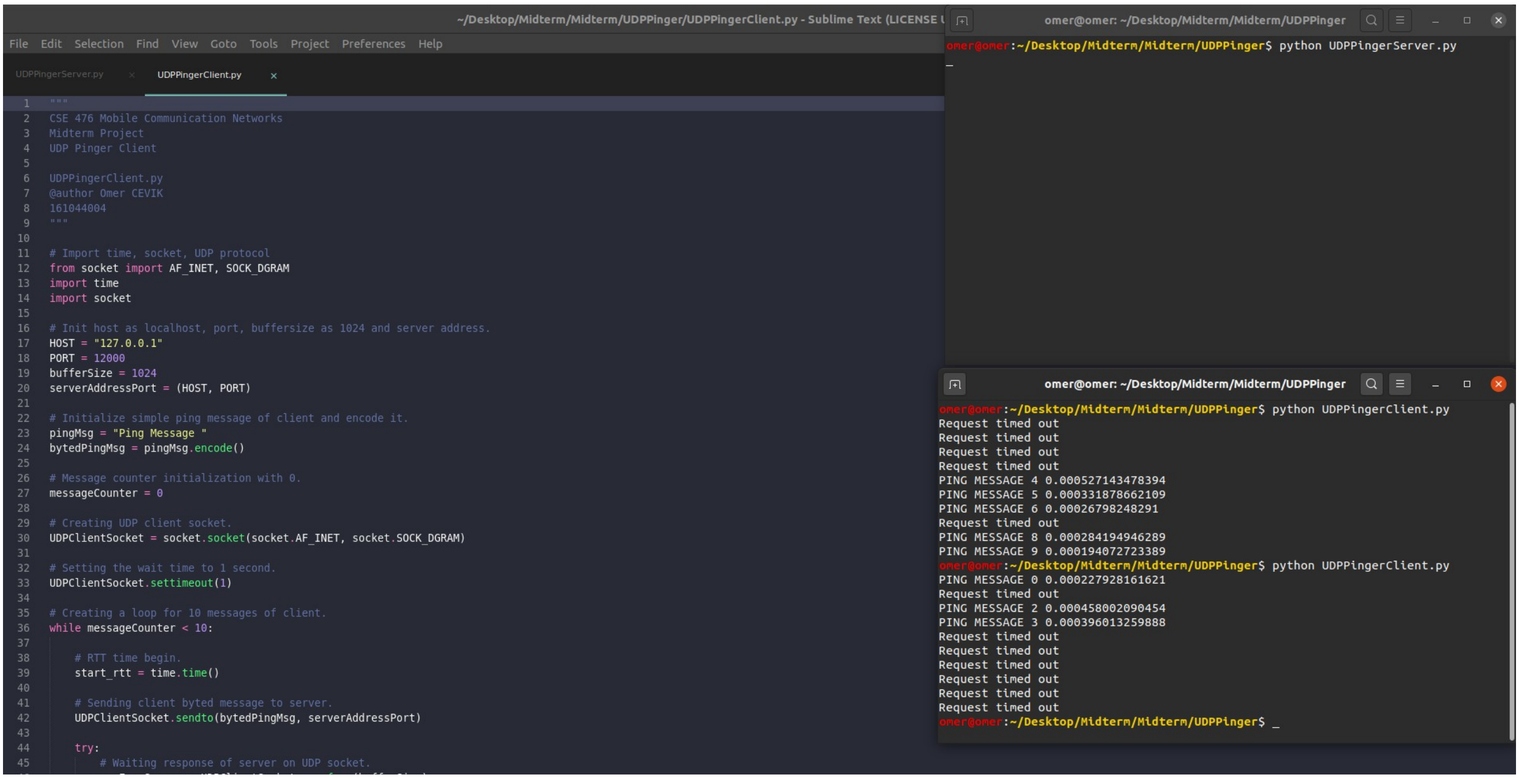
~/Desktop/Midterm/Midterm/UDPPinger/UDPPingerClient.py - Sublime Text (LICENSE U
File Edit Selection Find View Goto Tools Project Preferences Help
UDPPingerServer.py
UDPPingerClient.py
omer@omer: ~/Desktop/Midterm/Midterm/UDPPinger
omer@omer: ~/Desktop/Midterm/Midterm/UDPPinger

```python
"""
CSE 476 Mobile Communication Networks
Midterm Project
UDP Pinger Client

UDPPingerClient.py
@author Omer CEVIK
161044004
"""

# Import time, socket, UDP protocol
from socket import AF_INET, SOCK_DGRAM
import time
import socket

# Init host as localhost, port, buffersize as 1024 and server address.
HOST = "127.0.0.1"
PORT = 12000
bufferSize = 1024
serverAddressPort = (HOST, PORT)

# Initialize simple ping message of client and encode it.
pingMsg = "Ping Message "
bytedPingMsg = pingMsg.encode()

# Message counter initialization with 0.
messageCounter = 0

# Creating UDP client socket.
UDPClientSocket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

# Setting the wait time to 1 second.
UDPClientSocket.settimeout(1)

# Creating a loop for 10 messages of client.
while messageCounter < 10:

    # RTT time begin.
    start_rtt = time.time()

    # Sending client byted message to server.
    UDPClientSocket.sendto(bytedPingMsg, serverAddressPort)

    try:
        # Waiting response of server on UDP socket.
        msgFromServer = UDPClientSocket.recvfrom(bufferSize)
```

```
omer@omer:~/Desktop/Midterm/Midterm/UDPPinger$ python UDPPingerServer.py
_
```

```
omer@omer:~/Desktop/Midterm/Midterm/UDPPinger$ python UDPPingerClient.py
Request timed out
Request timed out
Request timed out
Request timed out
PING MESSAGE 4 0.000527143478394
PING MESSAGE 5 0.000331878662109
PING MESSAGE 6 0.00026798248291
Request timed out
PING MESSAGE 8 0.000284194946289
PING MESSAGE 9 0.000194072723389
omer@omer:~/Desktop/Midterm/Midterm/UDPPinger$ _
```

2. Running UDPPingerClient.py in Another Terminal

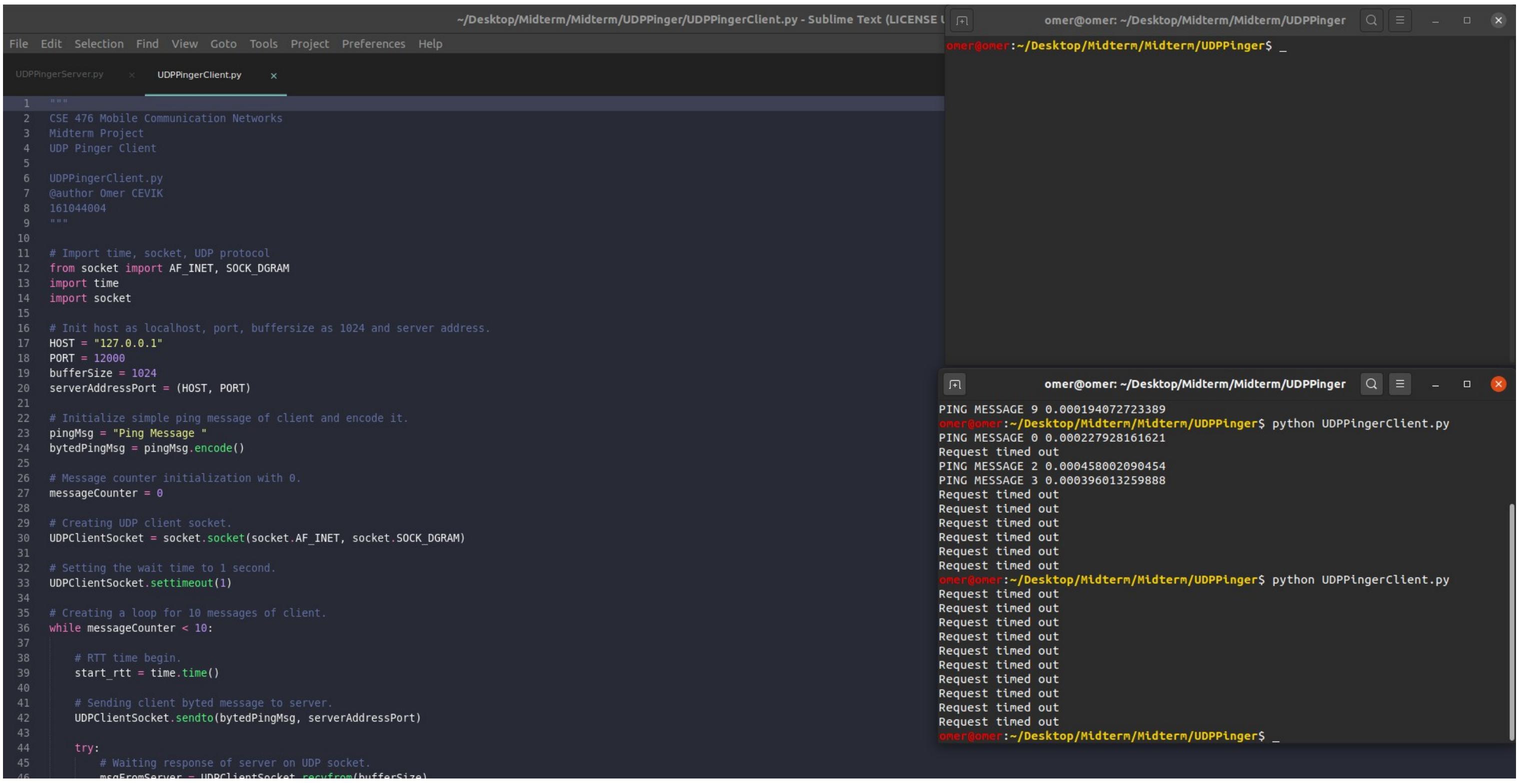

~/Desktop/Midterm/Midterm/UDPPinger/UDPPingerClient.py - Sublime Text (LICENSE U
File Edit Selection Find View Goto Tools Project Preferences Help
UDPPingerServer.py
UDPPingerClient.py
omer@omer: ~/Desktop/Midterm/Midterm/UDPPinger
omer@omer: ~/Desktop/Midterm/Midterm/UDPPinger

```python
"""
CSE 476 Mobile Communication Networks
Midterm Project
UDP Pinger Client

UDPPingerClient.py
@author Omer CEVIK
161044004
"""

# Import time, socket, UDP protocol
from socket import AF_INET, SOCK_DGRAM
import time
import socket

# Init host as localhost, port, buffersize as 1024 and server address.
HOST = "127.0.0.1"
PORT = 12000
bufferSize = 1024
serverAddressPort = (HOST, PORT)

# Initialize simple ping message of client and encode it.
pingMsg = "Ping Message "
bytedPingMsg = pingMsg.encode()

# Message counter initialization with 0.
messageCounter = 0

# Creating UDP client socket.
UDPClientSocket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

# Setting the wait time to 1 second.
UDPClientSocket.settimeout(1)

# Creating a loop for 10 messages of client.
while messageCounter < 10:

    # RTT time begin.
    start_rtt = time.time()

    # Sending client byted message to server.
    UDPClientSocket.sendto(bytedPingMsg, serverAddressPort)

    try:
        # Waiting response of server on UDP socket.
```

```
omer@omer:~/Desktop/Midterm/Midterm/UDPPinger$ python UDPPingerServer.py
_
```

```
omer@omer:~/Desktop/Midterm/Midterm/UDPPinger$ python UDPPingerClient.py
Request timed out
Request timed out
Request timed out
Request timed out
PING MESSAGE 4 0.000527143478394
PING MESSAGE 5 0.000331878662109
PING MESSAGE 6 0.00026798248291
Request timed out
PING MESSAGE 8 0.000284194946289
PING MESSAGE 9 0.000194072723389
omer@omer:~/Desktop/Midterm/Midterm/UDPPinger$ python UDPPingerClient.py
PING MESSAGE 0 0.000227928161621
Request timed out
PING MESSAGE 2 0.000458002090454
PING MESSAGE 3 0.000396013259888
Request timed out
Request timed out
Request timed out
Request timed out
Request timed out
Request timed out
omer@omer:~/Desktop/Midterm/Midterm/UDPPinger$ _
```

3. Running UDPPingerClient.py in Another Terminal Again


~/Desktop/Midterm/Midterm/UDPPinger/UDPPingerClient.py - Sublime Text (LICENSE U
File Edit Selection Find View Goto Tools Project Preferences Help
UDPPingerServer.py
UDPPingerClient.py
omer@omer: ~/Desktop/Midterm/Midterm/UDPPinger
omer@omer: ~/Desktop/Midterm/Midterm/UDPPinger

```python
"""
CSE 476 Mobile Communication Networks
Midterm Project
UDP Pinger Client

UDPPingerClient.py
@author Omer CEVIK
161044004
"""

# Import time, socket, UDP protocol
from socket import AF_INET, SOCK_DGRAM
import time
import socket

# Init host as localhost, port, buffersize as 1024 and server address.
HOST = "127.0.0.1"
PORT = 12000
bufferSize = 1024
serverAddressPort = (HOST, PORT)

# Initialize simple ping message of client and encode it.
pingMsg = "Ping Message "
bytedPingMsg = pingMsg.encode()

# Message counter initialization with 0.
messageCounter = 0

# Creating UDP client socket.
UDPClientSocket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

# Setting the wait time to 1 second.
UDPClientSocket.settimeout(1)

# Creating a loop for 10 messages of client.
while messageCounter < 10:

    # RTT time begin.
    start_rtt = time.time()

    # Sending client byted message to server.
    UDPClientSocket.sendto(bytedPingMsg, serverAddressPort)

    try:
        # Waiting response of server on UDP socket.
        msgFromServer = UDPClientSocket.recvfrom(bufferSize)
```

```
omer@omer:~/Desktop/Midterm/Midterm/UDPPinger$ _
```

```
PING MESSAGE 9 0.000194072723389
omer@omer:~/Desktop/Midterm/Midterm/UDPPinger$ python UDPPingerClient.py
PING MESSAGE 0 0.000227928161621
Request timed out
PING MESSAGE 2 0.000458002090454
PING MESSAGE 3 0.000396013259888
Request timed out
Request timed out
Request timed out
Request timed out
Request timed out
Request timed out
omer@omer:~/Desktop/Midterm/Midterm/UDPPinger$ python UDPPingerClient.py
Request timed out
Request timed out
Request timed out
Request timed out
Request timed out
Request timed out
Request timed out
Request timed out
Request timed out
Request timed out
omer@omer:~/Desktop/Midterm/Midterm/UDPPinger$ _
```

4. Running UDPPingerClient.py While Server is Closed

3. Assignment 3: Mail Client

In this assignment, main idea is to create a client mail program which use **SMTP** to connect a mail server using socket.

Firstly, to send a message content the message is declared**[16]** and to represent the end of message it is declared**[17]**. Then the mail server's address and port number are declared (**gmail** server is used in this example using port number as **587**)**[22, 23]**. With **TCP** client socket is created**[29]** and **connect**ed**[32]** to mail server. Waiting the connection result using **recv[34]** on socket and **decode[35]** it about receive from server is applied. Then printing the receive result from server**[36]**. If result starts with '**220**'**[37]** it means client is not received by server**[38]**.

After connection succeed, **HELO** command is **sent** to server**[44, 45]** and its response is got using **recv[47, 48, 49]** function on client socket. If receive from server starts with '**250**' it means client is not reply received by server**[51, 52]**.

If **HELO** command is succeed then **STARTTLS** command is **sent** to server using **encoding[59]** and getting receive from server using **recv[60]** on socket. Printing the receive from socket is applied**[61]**.

After starting TLS, using **ssl** library socket is wrapped to **SSL** socket using ssl's **SSLv23** protocol version**[64]**. If mail server is live then it has to send to server **HELO** command**[67, 68]**. Otherwise it continues to **AUTH LOGIN** command sending and receiving by server with printing message of received command**[71, 72, 73, 74]**.

After authenticate login command, it needs user name and password (mail address and password) which is declared and encoded using **base64** library**[77, 78, 81, 82]**. Then **send**ing encoded user name and password with printing receive from socket**[85, 86, 87, 90, 91, 92]**.

Then it sends **MAIL FROM** command to socket with user name while **encoding** and printing receive from socket**[95, 96, 97, 98]**.

RCPT TO command is used to **send**ing**[111, 112]** mail to whom using mail address. In that example sending mail to itself**[107]**. You can try other mail address which is commented line**[108]**. RCPT TO command is **decode**d and printed the receive from socket**[113, 114]**.

DATA command is **sent** to socket and printed the receive message from socket**[123, 124, 125, 126]**. Then message and end of message is **sent** to socket using **encode** and printed the receive from socket**[135, 143, 145, 146]**.

After all **QUIT** command is **sent** to socket using **encode** and printing the receive from socket**[154, 156, 157, 158]**.

Finally, closing the wrapped socket and client socket successfully**[161, 162]**.

Notes:

- Code lines in MailClient.py is represented inside bold square brackets.
- Python 2.7.18 version is used to run code.
- Ubuntu 20.04.1 LTS operating system and its Terminal is workspace.
- To run MailClient code ***python MailClient.py*** command is realized in Terminal.

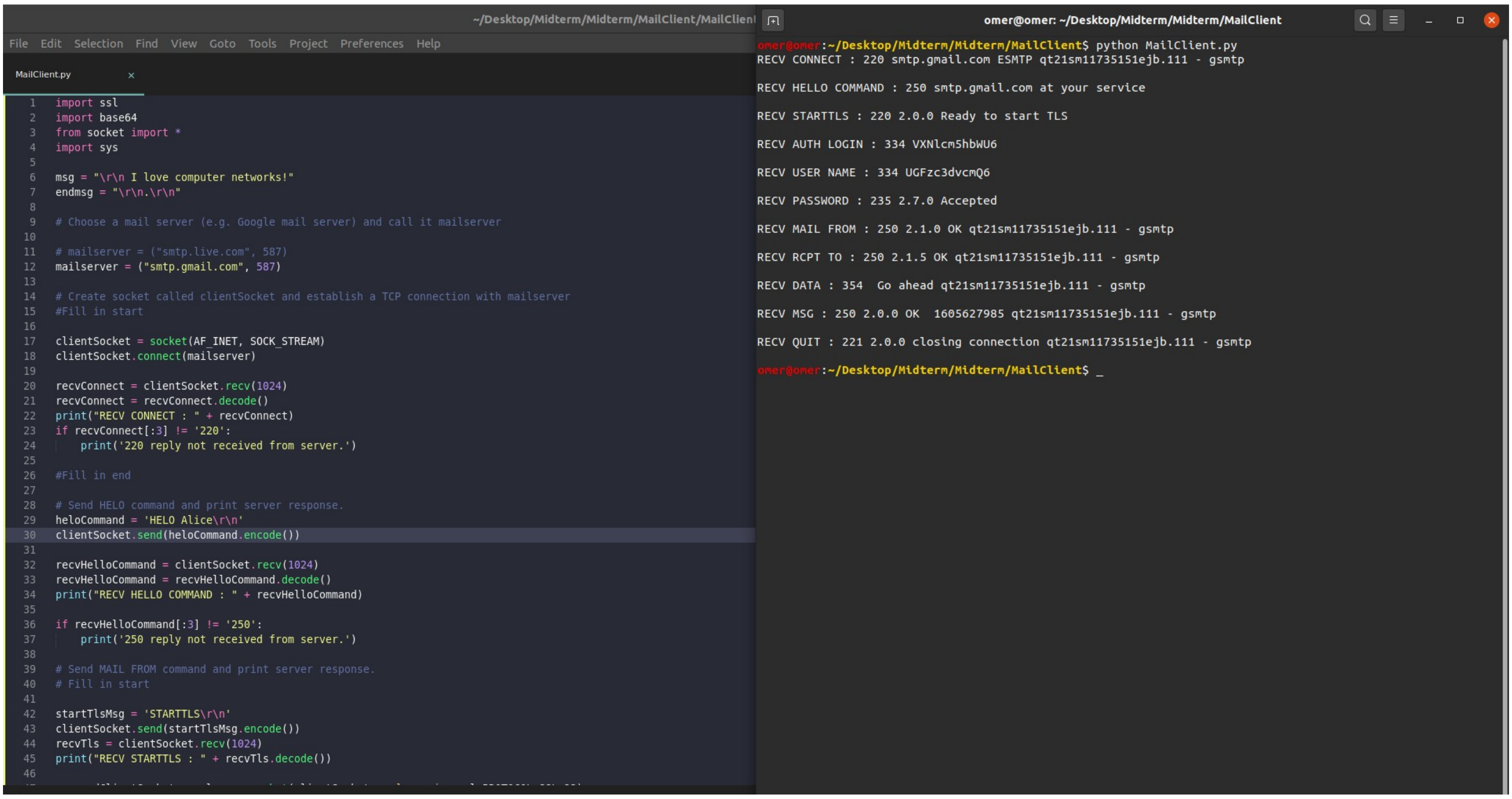
- First optional exercise is implemented in Mail Client part (SSL).

MailClient.py :

```
"""
CSE 476 Mobile Communication Networks
Midterm Project
Mail Client

MailClient.py
@author Omer CEVIK
161044004
"""

from socket import *
import base64
import sys
import ssl

msg = "\r\n I love computer networks!"
endmsg = "\r\n.\r\n"

# Choose a mail server (e.g. Google mail server) and call it mailserver

# Choose mailserver as live or gmail.
# mailserver = ("smtp.live.com", 587)
mailserver = ("smtp.gmail.com", 587)

# Create socket called clientSocket and establish a TCP connection with mailserver
#Fill in start

# Creating socket TCP protocoled.
clientSocket = socket(AF_INET, SOCK_STREAM)

# Connecting to mail server and printing the response.
clientSocket.connect(mailserver)

recvConnect = clientSocket.recv(1024)
recvConnect = recvConnect.decode()
print("RECV CONNECT : " + recvConnect)
if recvConnect[:3] != '220':
    print('220 reply not received from server.')

#Fill in end


# Send HELO command and print server response.
heloCommand = 'HELO Alice\r\n'
clientSocket.send(heloCommand.encode())

recvHelloCommand = clientSocket.recv(1024)
recvHelloCommand = recvHelloCommand.decode()
print("RECV HELLO COMMAND : " + recvHelloCommand)

if recvHelloCommand[:3] != '250':
    print('250 reply not received from server.')

# Send MAIL FROM command and print server response.
# Fill in start

# Start TLS sending STARTTLS command and printing the response.
startTlsMsg = 'STARTTLS\r\n'
clientSocket.send(startTlsMsg.encode())
recvTls = clientSocket.recv(1024)
print("RECV STARTTLS : " + recvTls.decode())

# Wraps the client socket using SSL.
```

```
wrappedClientSocket = ssl.wrap_socket(clientSocket, ssl_version=ssl.PROTOCOL_SSLv23)

# If hotmail/outlook/live server is used then it sends hello command again.
if "live" in mailserver[0]:
    wrappedClientSocket.send(heloCommand.encode())

# Sending wrapped socket AUTH LOGIN command.
authLoginMsg = 'AUTH LOGIN\r\n'
wrappedClientSocket.send(authLoginMsg.encode())
recvAuthLogin = wrappedClientSocket.recv(1024)
print("RECV AUTH LOGIN : " + recvAuthLogin.decode())

# Test mail username and password.
username = "omerceviktest@gmail.com"
password = "KjhdveB7CMqwH7r"

# Encode the username and password.
encodedUserName = (base64.b64encode(username.encode()))+('\r\n').encode()
encodedPassword = (base64.b64encode(password.encode()))+('\r\n').encode()

# Sending encoded username and printing the response.
wrappedClientSocket.send(encodedUserName)
recvUserName = wrappedClientSocket.recv(1024)
print("RECV USER NAME : " + recvUserName.decode())

# Sending encoded password and printing the response.
wrappedClientSocket.send(encodedPassword)
recvPassword = wrappedClientSocket.recv(1024)
print("RECV PASSWORD : " + recvPassword.decode())

# Sending MAIL FROM command with username and printing the response.
mailFromMsg = "MAIL FROM: <"+ username +">\r\n"
wrappedClientSocket.send(mailFromMsg.encode())
recvMailFrom = wrappedClientSocket.recv(1024)
print("RECV MAIL FROM : " + recvMailFrom.decode())

# Fill in end


# Send RCPT TO command and print server response.
# Fill in start

# Using rcpt to mail with itself or other one.
rcptToMail = username
# rcptToMail = "omer.cevik2016@gtu.edu.tr"

# Sending RCPT TO command with rcpt to mail and printing the response.
rcptToMsg = "RCPT TO: <"+ rcptToMail +">\r\n"
wrappedClientSocket.send(rcptToMsg.encode())
recvRcptTo = wrappedClientSocket.recv(1024)
print("RECV RCPT TO : " + recvRcptTo.decode())

# Fill in end


# Send DATA command and print server response.
# Fill in start

# Sending DATA command and printing the response.
dataMsg = "DATA\r\n"
wrappedClientSocket.send(dataMsg.encode())
recvData = wrappedClientSocket.recv(1024)
print("RECV DATA : " + recvData.decode())

# Fill in end


# Send message data.
# Fill in start

# Sending message.
wrappedClientSocket.send(msg.encode())

# Fill in end

# Message ends with a single period.
# Fill in start

# Sending end of message and printing the response.
wrappedClientSocket.send(endmsg.encode())

recvMsg = wrappedClientSocket.recv(1024)
print("RECV MSG : " + recvMsg.decode())

# Fill in end

# Send QUIT command and get server response.
# Fill in start

# Sending QUIT command and printing the response.
quitMsg = "QUIT\r\n"

wrappedClientSocket.send(quitMsg.encode())
recvQuit = wrappedClientSocket.recv(1024)
print("RECV QUIT : " + recvQuit.decode())

# Closing wrapper socket and client socket.
wrappedClientSocket.close()
clientSocket.close()

# Fill in end
```

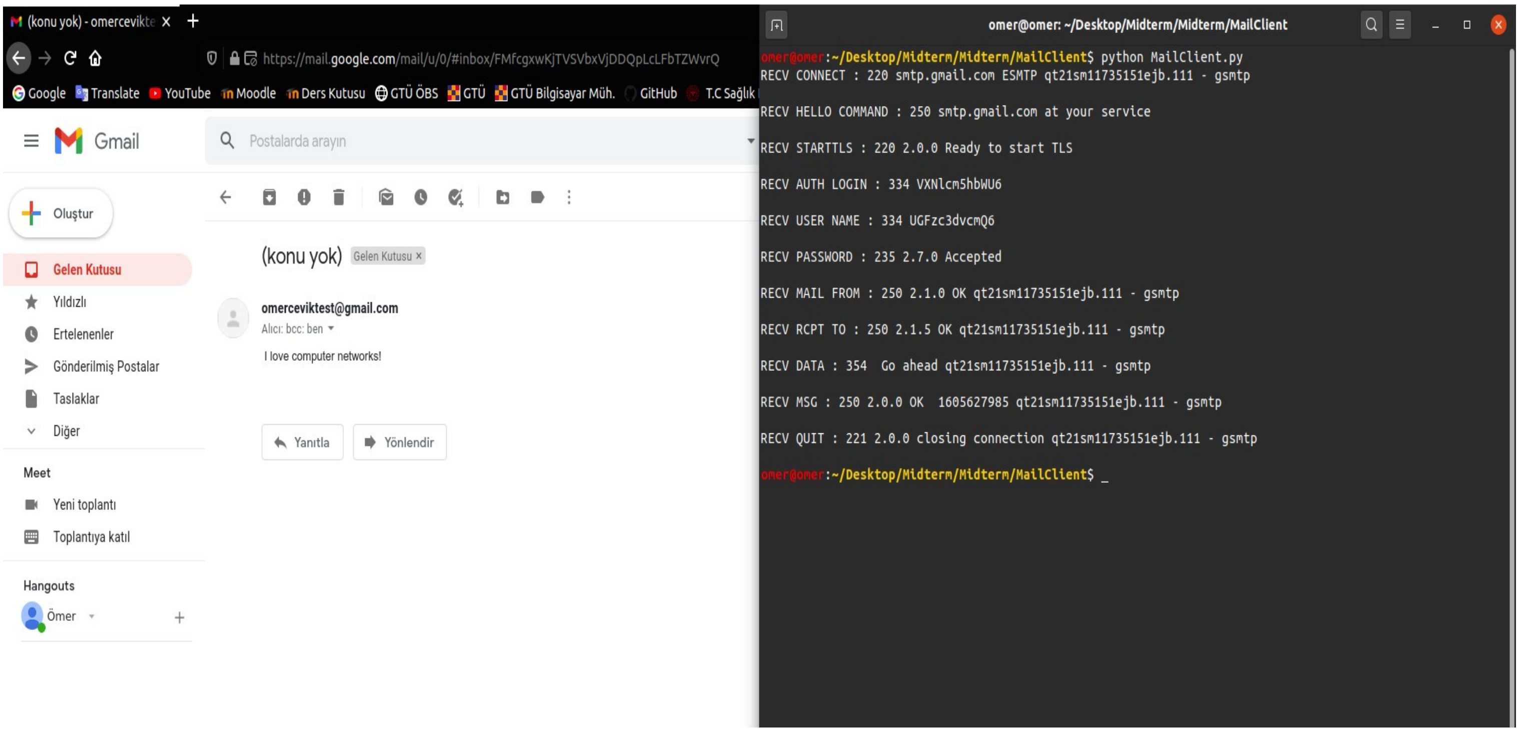
MailClient Program Test Results :


~/Desktop/Midterm/Midterm/MailClient/MailClient
File
Edit
Selection
Find
View
Goto
Tools
Project
Preferences
Help
MailClient.py
omer@omer: ~/Desktop/Midterm/Midterm/MailClient

```
import ssl
import base64
from socket import *
import sys

msg = "\r\n I love computer networks!"
endmsg = "\r\n.\r\n"

# Choose a mail server (e.g. Google mail server) and call it mailserver

# mailserver = ("smtp.live.com", 587)
mailserver = ("smtp.gmail.com", 587)

# Create socket called clientSocket and establish a TCP connection with mailserver
#Fill in start

clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect(mailserver)

recvConnect = clientSocket.recv(1024)
recvConnect = recvConnect.decode()
print("RECV CONNECT : " + recvConnect)
if recvConnect[:3] != '220':
    print('220 reply not received from server.')

#Fill in end

# Send HELO command and print server response.
heloCommand = 'HELO Alice\r\n'
clientSocket.send(heloCommand.encode())

recvHelloCommand = clientSocket.recv(1024)
recvHelloCommand = recvHelloCommand.decode()
print("RECV HELLO COMMAND : " + recvHelloCommand)

if recvHelloCommand[:3] != '250':
    print('250 reply not received from server.')

# Send MAIL FROM command and print server response.
# Fill in start

startTlsMsg = 'STARTTLS\r\n'
clientSocket.send(startTlsMsg.encode())
recvTls = clientSocket.recv(1024)
print("RECV STARTTLS : " + recvTls.decode())

```

```
omer@omer:~/Desktop/Midterm/Midterm/MailClient$ python MailClient.py
RECV CONNECT : 220 smtp.gmail.com ESMTP qt21sm11735151ejb.111 - gsmtp

RECV HELLO COMMAND : 250 smtp.gmail.com at your service

RECV STARTTLS : 220 2.0.0 Ready to start TLS

RECV AUTH LOGIN : 334 VXNlcm5hbWU6

RECV USER NAME : 334 UGFzc3dvcmQ6

RECV PASSWORD : 235 2.7.0 Accepted

RECV MAIL FROM : 250 2.1.0 OK qt21sm11735151ejb.111 - gsmtp

RECV RCPT TO : 250 2.1.5 OK qt21sm11735151ejb.111 - gsmtp

RECV DATA : 354  Go ahead qt21sm11735151ejb.111 - gsmtp

RECV MSG : 250 2.0.0 OK  1605627985 qt21sm11735151ejb.111 - gsmtp

RECV QUIT : 221 2.0.0 closing connection qt21sm11735151ejb.111 - gsmtp

omer@omer:~/Desktop/Midterm/Midterm/MailClient$ _
```

1. Running MailClient.py on Terminal



(konu yok)

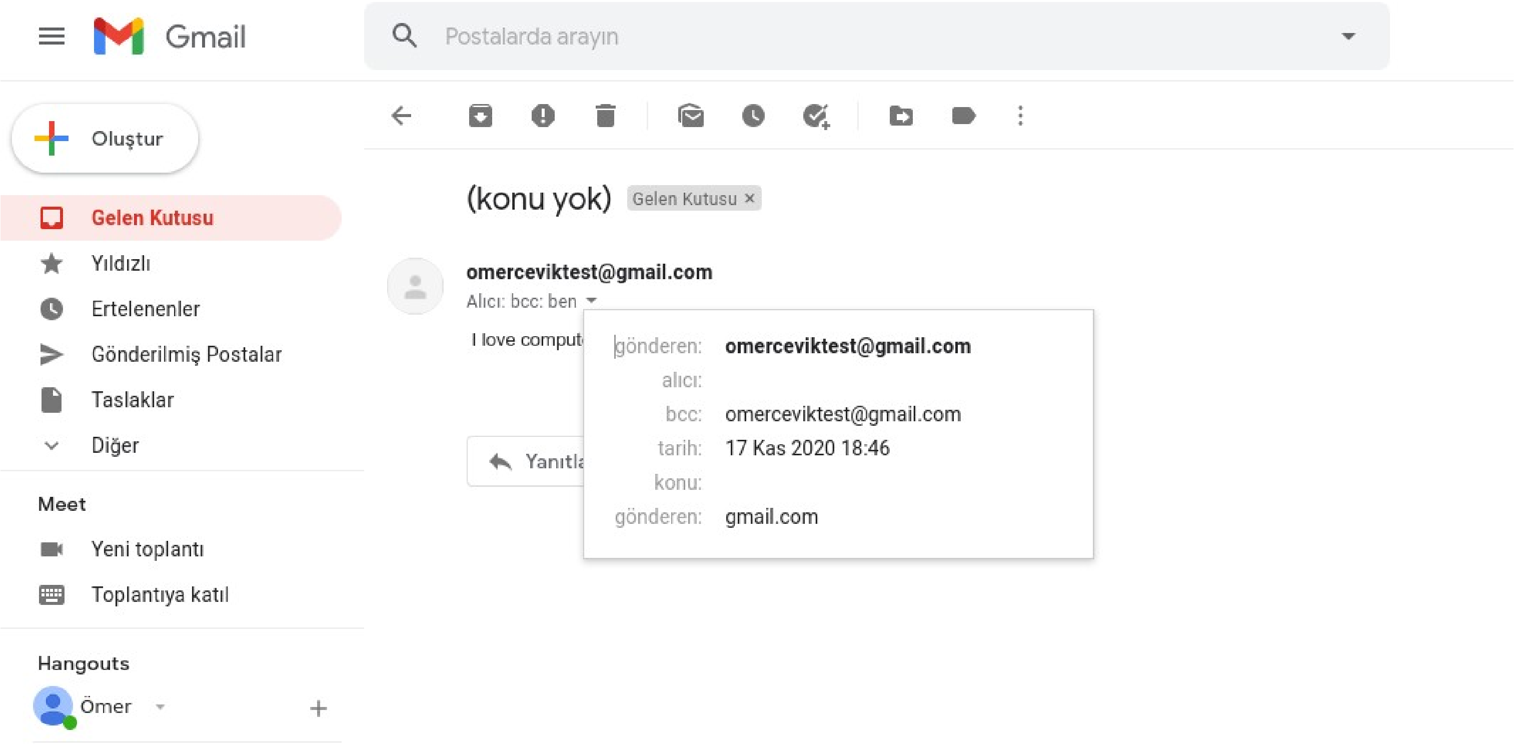
omerceviktest@gmail.com

Alıcı: bcc: ben

I love computer networks!

```
omer@omer:~/Desktop/Midterm/Midterm/MailClient$ python MailClient.py
RECV CONNECT : 220 smtp.gmail.com ESMTP qt21sm11735151ejb.111 - gsmtp

RECV HELLO COMMAND : 250 smtp.gmail.com at your service

RECV STARTTLS : 220 2.0.0 Ready to start TLS

RECV AUTH LOGIN : 334 VXNlcm5hbWU6

RECV USER NAME : 334 UGFzc3dvcmQ6

RECV PASSWORD : 235 2.7.0 Accepted

RECV MAIL FROM : 250 2.1.0 OK qt21sm11735151ejb.111 - gsmtp

RECV RCPT TO : 250 2.1.5 OK qt21sm11735151ejb.111 - gsmtp

RECV DATA : 354  Go ahead qt21sm11735151ejb.111 - gsmtp

RECV MSG : 250 2.0.0 OK  1605627985 qt21sm11735151ejb.111 - gsmtp

RECV QUIT : 221 2.0.0 closing connection qt21sm11735151ejb.111 - gsmtp

omer@omer:~/Desktop/Midterm/Midterm/MailClient$ _
```

2. After Running Program on Gmail



(konu yok)

omerceviktest@gmail.com

Alıcı: bcc: ben

I love comput

gönderen: omerceviktest@gmail.com
alıcı:
bcc: omerceviktest@gmail.com
tarih: 17 Kas 2020 18:46
konu:
gönderen: gmail.com

3. After Running Program Sender and Reciever on Gmail