SMTP Protocol-Building Python Mail Assignment Report

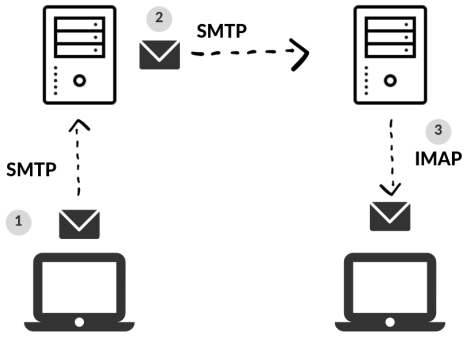
Team Members:
Alize De Matas 500745506
Reiyyan Nizami 500944046

Course Code: CCPS706
Course Name: Computer Networks I
Professor: Dr. Sattar Hussain

Objective:

Successfully created a mail client that sends email to any recipient with a better understanding of SMTP protocol.

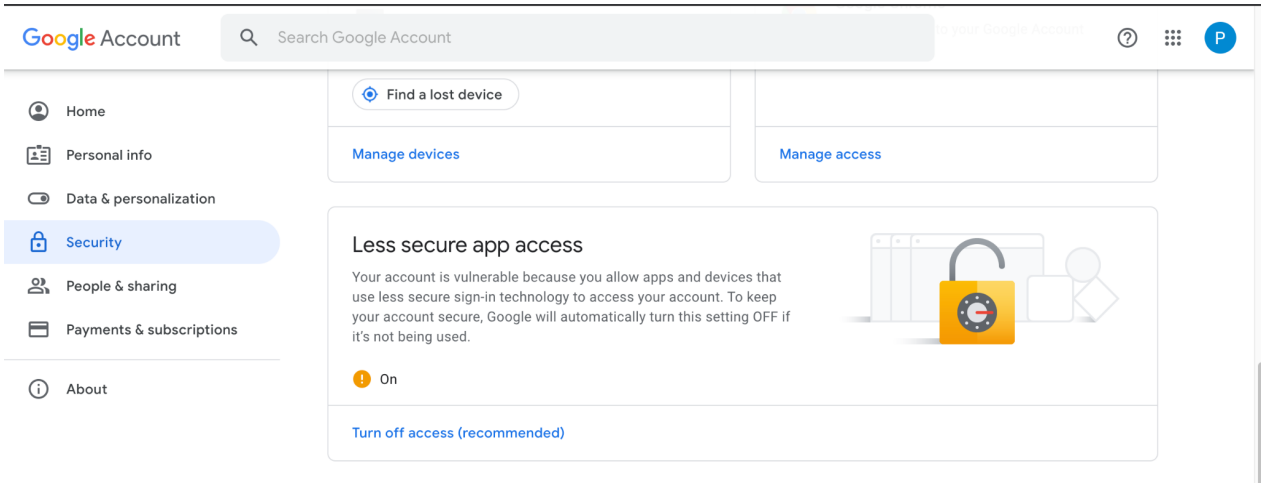
Overview:



Simple Mail Transfer Protocol (SMTP) is an internet standard communication protocol for email transmission. Mail servers use SMTP to send and receive mail messages and it is very reliable and secure. The SMTP server can be broken down into 2 steps. The first step is the set-up and connections (granting permissions). The second step is to send out the message and make sure the delivery of the email is successful. If for some reason, the email fails it will be returned to the sender.

- The SMTP server utilizes simple text commands
- HELO - Introduce yourself
- EHLO - Introduce yourself and request extended mode
- MAIL FROM - Specify the sender
- RCPT TO - Specify the recipient
- DATA - Specify the body of the email

For this project we utilized the python module, called the smtplib, which has built in methods to send mail using SMTP protocol. We connected to the GMAIL server, we changed our settings to allow for Python script authorization.



Contributions:

Alize De Matas (Contributions)	**Reiyyan Nizami (Contributions)**
- Wrote SmtpPython.py	- Wrote SocketMail.py
- Wrote project report	- Made project presentation
- Discussed and assisted in understanding python code and SMTP protocol.	- Discussed and assisted in understanding python code and SMTP protocol.

SmtpPython.py - Alize:

Code:

```python
from socket import *
import smtplib
import ssl
import base64

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.gmail.com'

# Create socket called clientSocket and establish a TCP connection with mailserver
mailport = 587
hostname = 'pythonsmtpserver123@gmail.com'
password = 'PythonProject#1'

clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect(("smtp.gmail.com", 587))
print("successful connection to " + mailserver)

recv = clientSocket.recv(1024).decode()

print(recv)
if recv[:3] != '220':
    print('220 reply not received from server.')

```

```python

user = base64.b64encode(hostname.encode()).decode() + '\r\n'
sslclientSocket.write(user.encode())
user_rev = sslclientSocket.read(1024).decode()
print(user_rev)
if user_rev[:3] != '334':
    print('334 reply not received from server.')

p = base64.b64encode(password.encode()).decode() + '\r\n'
sslclientSocket.write(p.encode())
p_rev = sslclientSocket.read(1024).decode()
print(p_rev)
if p_rev[:3] != '235':
    print('235 reply not received from server.')

# Send MAIL FROM command and print server response
sslclientSocket.send("MAIL FROM:<pythonsmtpserver123@gmail.com>\r\n")
recv1 = sslclientSocket.recv(1024)
print (recv1)
if recv1[:3] != '250':
    print ('250 reply not received from server.')

# Send RCPT TO command and print server response
sslclientSocket.send('RCPT TO:<pythonsmtpserver123@gmail.com>\r\n')
recv1 = sslclientSocket.recv(1024)
print (recv1)
if recv1[:3] != '250':
```

SmtpPython.py smtpImage.py

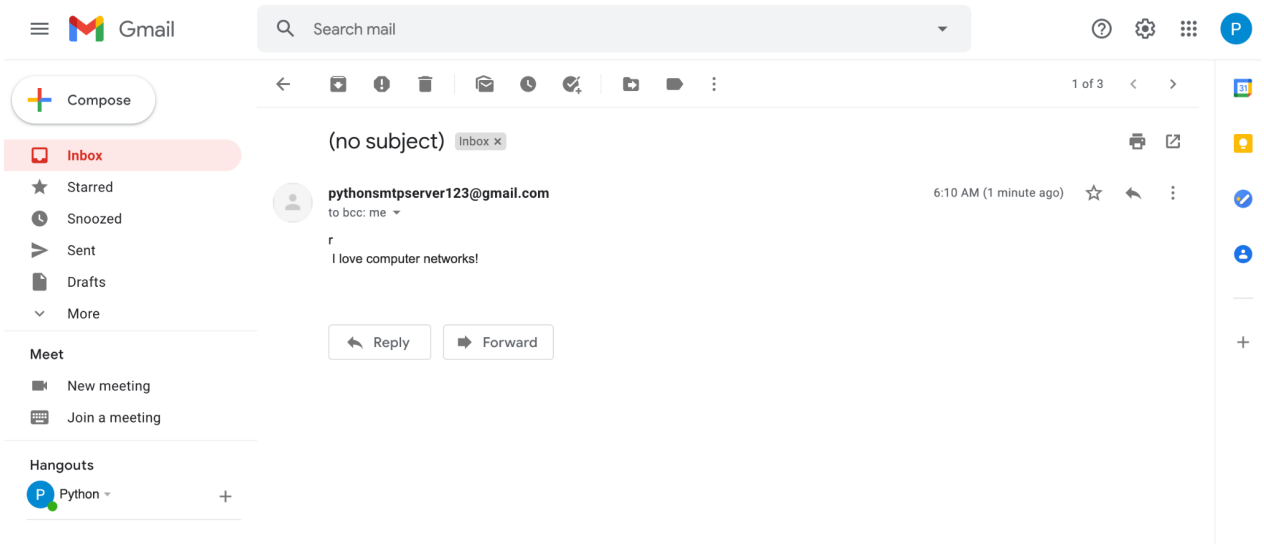
```
76      if recv1[:3] != '250':
77          print('250 reply not received from server.')
78
79      # Send DATA command and print server response.
80      sslclientSocket.send('DATA\r\n')
81      recv1 = sslclientSocket.recv(1024)
82      print recv1
83      if recv1[:3] != '354':
84          print('354 reply not received from server.')
85
86      # Send message data.
87      sslclientSocket.send('\r\n')
88      sslclientSocket.send(msg)
89      sslclientSocket.send(endmsg)
90
91      # Message ends with a single period
92      sslclientSocket.send('.\r\n')
93      recv1 = sslclientSocket.recv(1024)
94      print (recv1)
95      if recv1[:3] != '250':
96          print('250 reply not received from server.')
97
98      # Send QUIT command and get server response
99      sslclientSocket.send('QUIT\r\n')
100     clientSocket.close()
```

```
26
27      # Send HELO command and print server response.
28      heloCommand = 'HELO Alize\r\n'
29      clientSocket.send(heloCommand.encode())
30      recv1 = clientSocket.recv(1024).decode()
31      print(recv1)
32      if recv1[:3] != '250':
33          print('250 reply not received from server.')
34
35      starttls_command = 'STARTTLS\r\n'
36      clientSocket.send(starttls_command.encode())
37      tls_rev = clientSocket.recv(1024).decode()
38      print(tls_rev)
39      if tls_rev[:3] != '220':
40          print('220 reply not received from server.')
41
42
43      sslclientSocket = ssl.wrap_socket(clientSocket)
44      auth = 'AUTH LOGIN\r\n'
45      sslclientSocket.write(auth.encode())
46      auth_rev = sslclientSocket.read(1024).decode()
47      print(auth_rev)
48      if auth_rev[:3] != '334':
49          print('334 reply not received from server.')
50
```

Terminal Output:

```
Terminal:  Local  +

(venv) Alizes-MacBook-Pro:pythonProject6 alizedematas$ python SmtpPython.py
successful connection to smtp.gmail.com
220 smtp.gmail.com ESMTP o12sm1283392qki.44 - gsmtp

250 smtp.gmail.com at your service

220 2.0.0 Ready to start TLS

334 VXNlcm5hbWU6

334 UGFzc3dvcmQ6

235 2.7.0 Accepted

250 2.1.0 OK o12sm1283392qki.44 - gsmtp

250 2.1.5 OK o12sm1283392qki.44 - gsmtp

354  Go ahead o12sm1283392qki.44 - gsmtp

250 2.0.0 OK  1623838214 o12sm1283392qki.44 - gsmtp
```

Email Received:



SocketMail.py - Reiyyan:

Code:

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

# Debug
verbose = False

# Login Info - Must have less secure apps allowed in Gmail.
username = "ccps706test@gmail.com"
password = "ryerson2021"

# Mail Details
if(len(sys.argv) == 2):
    receiver = sys.argv[1]
    msg = "\r\nI love computer networks!"
elif(len(sys.argv) == 3):
    receiver = sys.argv[1]
    msg = f"{sys.argv[2]}\r\n"
else:
    receiver = "reiyyan@gmail.com"
    msg = "\r\nI love computer networks!"

endmsg = "\r\n.\r\n"

# Choose a mail server (e.g. Google mail server) and call it mailserver
#Fill in start
mailserver = ("smtp.gmail.com", 465)
#Fill in end

# Create socket called clientSocket and establish a TCP connection with
mailserver
# Wrap socket with TLS wrapper to get security
#Fill in start
clientSocket = ssl.wrap_socket(socket(AF_INET, SOCK_STREAM),
ssl_version=ssl.PROTOCOL_TLSv1)
clientSocket.connect(mailserver)
```

```
#Fill in end

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

# Send HELO command and print server response.
heloCommand = 'HELO Alice\r\n'
clientSocket.send(heloCommand.encode())
recv1 = clientSocket.recv(1024).decode()
print(recv1)
if recv1[:3] != '250':
    print('250 reply not received from server.')

# Authentication
authMesg = 'AUTH LOGIN\r\n'
crlfMesg = '\r\n'

# Tell server we are trying to authenticate
clientSocket.send(authMesg.encode('utf-8'))
recv2 = clientSocket.recv(2048).decode()
if verbose:
    print(recv2)

# Encode Username and Pw for server must be in base64
user64 = base64.b64encode(username.encode('utf-8'))
pass64 = base64.b64encode(password.encode('utf-8'))

# Tell server our Username
clientSocket.send(user64)
clientSocket.send(crlfMesg.encode('utf-8'))
recv3 = clientSocket.recv(2048).decode()
if verbose:
    print(recv3)

# Tell server our Password
clientSocket.send(pass64)
clientSocket.send(crlfMesg.encode('utf-8'))
recv4 = clientSocket.recv(2048).decode()
```

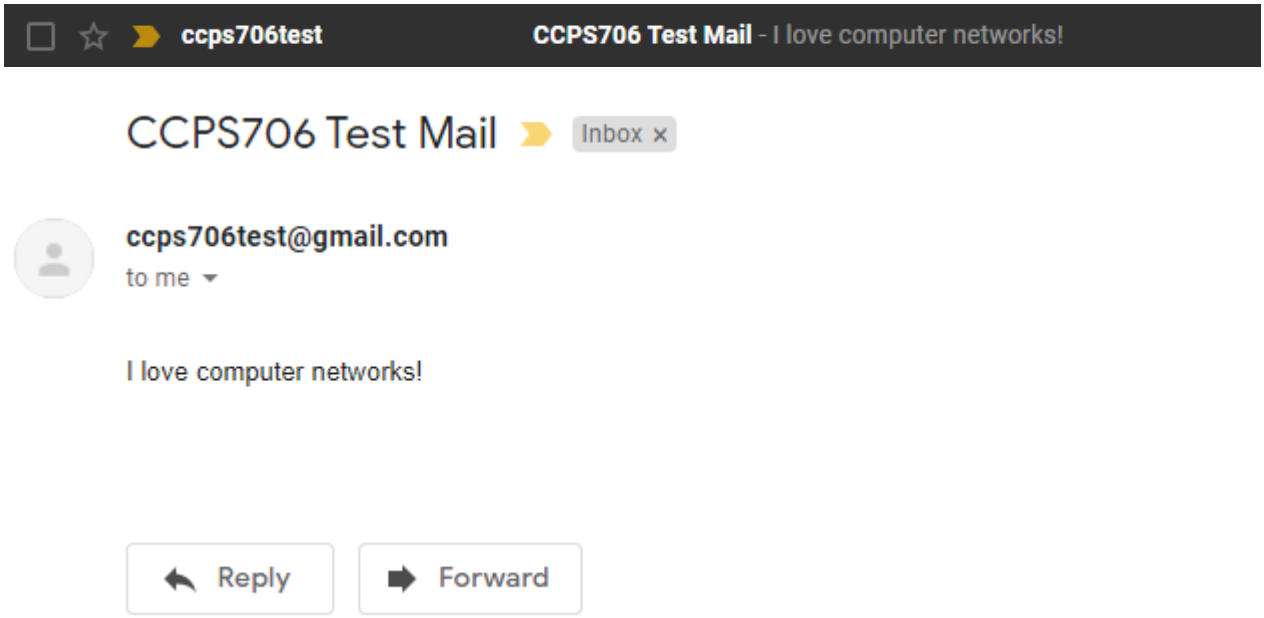
```
if verbose:
    print(recv4)

# Send MAIL FROM command and print server response.
# Fill in start
mailFrom = f"MAIL FROM: <{username}>\r\n"
clientSocket.send(mailFrom.encode())
recv5 = clientSocket.recv(2048).decode()
if verbose:
    print(recv5)
# Fill in end

# Send RCPT TO command and print server response.
# Fill in start
mailTo = f"RCPT TO: <{receiver}>\r\n"
clientSocket.send(mailTo.encode())
recv6 = clientSocket.recv(2048).decode()
if verbose:
    print(recv6)
# Fill in end

# Send DATA command and print server response.
# Fill in start
mailData = "DATA\r\n"
clientSocket.send(mailData.encode())
recv7 = clientSocket.recv(2048).decode()
if verbose:
    print(recv7)
# Fill in end

# Rei: Adding in section for to and subject (Optional)
subject = f"To: {receiver}\r\n"
clientSocket.send(subject.encode())

subject = "Subject: CCPS706 Test Mail\r\n\r\n"
clientSocket.send(subject.encode())

# Send message data.
# Fill in start
body = f"{msg} \r\n"
```

```
clientSocket.send(body.encode())
# Fill in end

# Message ends with a single period.
# Fill in start
clientSocket.send(endmsg.encode())
recv8 = clientSocket.recv(2048).decode()
if verbose:
    print(recv8)
# Fill in end

# Send QUIT command and get server response.
# Fill in start
quitMail = "QUIT\r\n"
clientSocket.send(quitMail.encode())
recv9 = clientSocket.recv(2048).decode()
print(recv9)
# Fill in end

clientSocket.close()
```

Terminal Output:

```
PS F:\RYERSON\Spring 2021\CCPS706\Programming\Project 3> Python SocketMail.py
220 smtp.gmail.com ESMTP g82sm374403qke.119 - gsmtp

250 smtp.gmail.com at your service

221 2.0.0 closing connection g82sm374403qke.119 - gsmtp

PS F:\RYERSON\Spring 2021\CCPS706\Programming\Project 3>
```

Result:



References Used:

- https://www.javatpoint.com/python-sending-email
- https://realpython.com/python-send-email/
- https://www.tutorialspoint.com/python/python_sending_email.htm
- https://docs.python.org/3/library/smtplib.html
- https://www.androidauthority.com/gmail-smtp-settings-801100/
- Lee D., Kang J., Dahouda M.K., Joe I., Lee K. (2020) DTN-SMTP: A Novel Mail Transfer Protocol with Minimized Interactions for Space Internet. In: Gervasi O. et al. (eds) Computational Science and Its Applications – ICCSA 2020. ICCSA 2020. Lecture Notes in Computer Science, vol 12249. Springer, Cham. https://doi.org/10.1007/978-3-030-58799-4_24