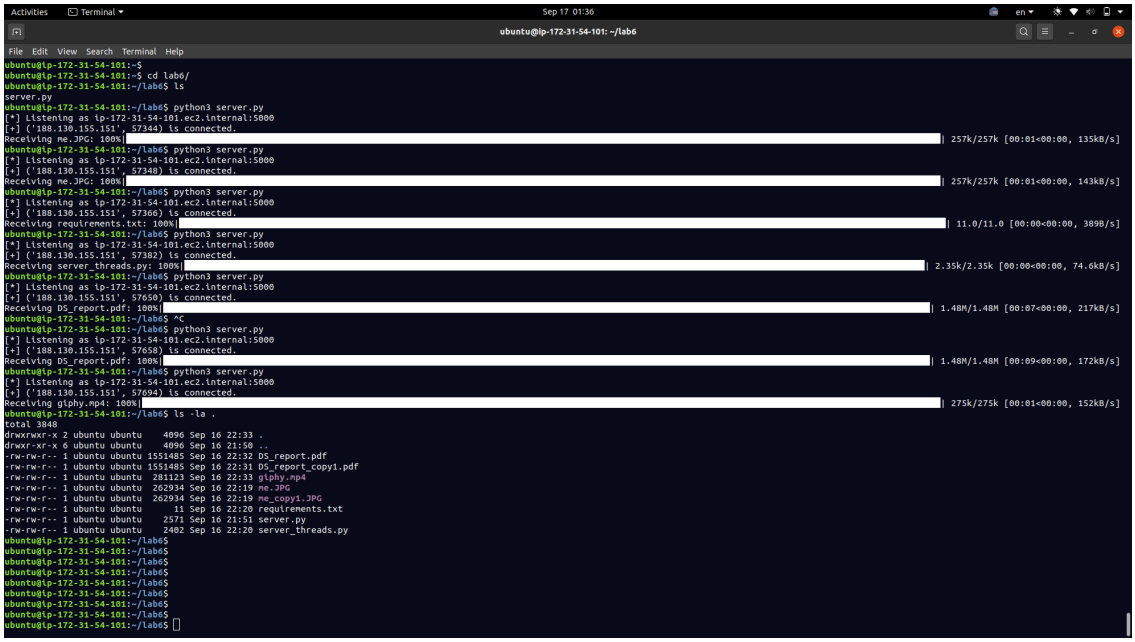
DS Lab 6- Message-oriented Communication

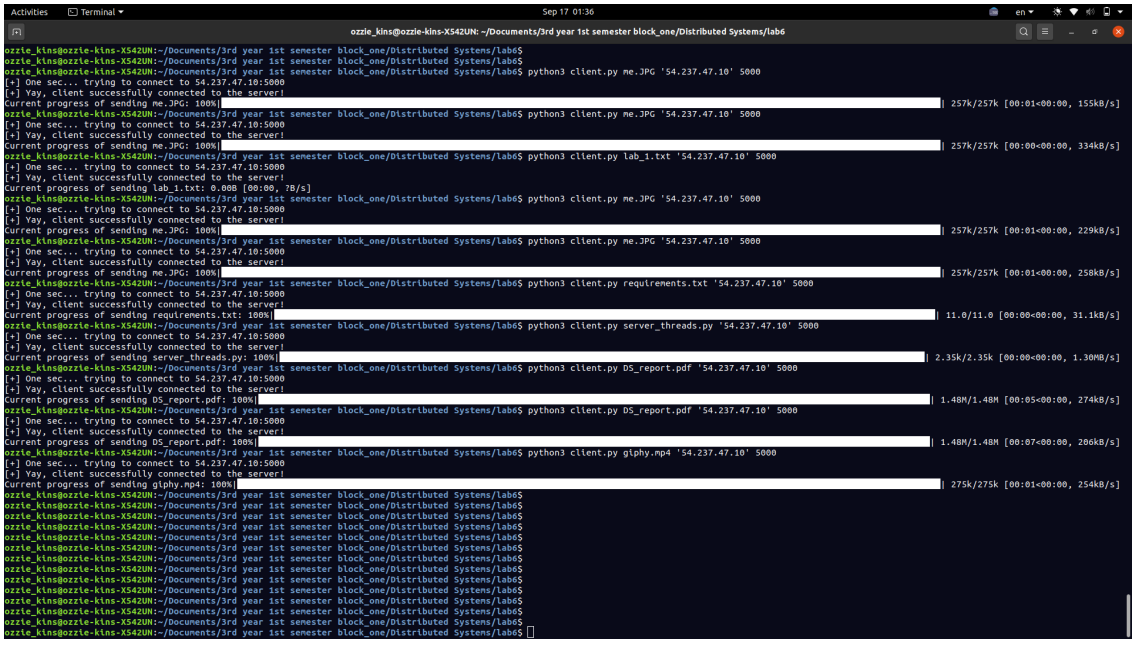
Name: Ozioma Okonicha

Course: SE-01

- Server/Receiver terminal



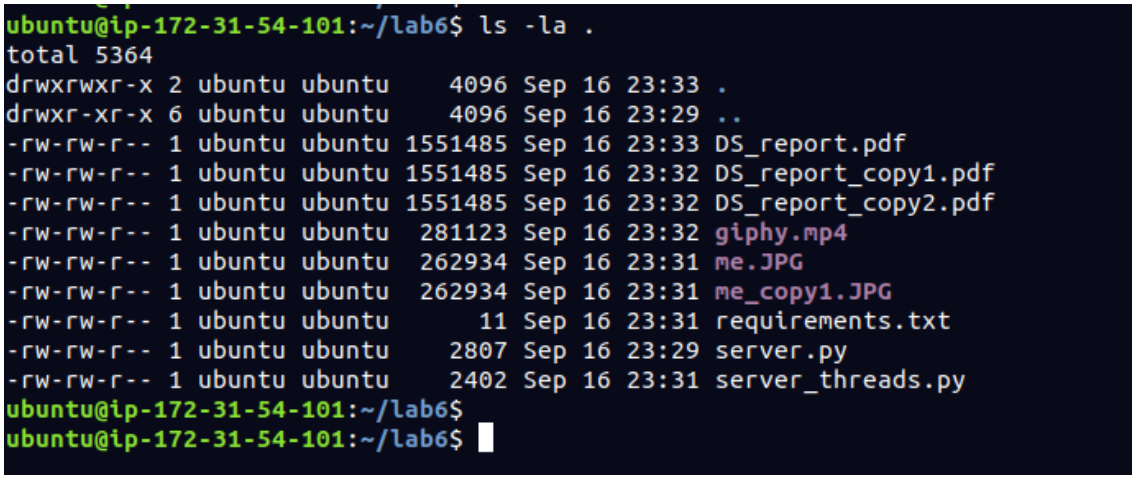
- Client/Sender terminal



- Screenshot of sending progress(receiver terminal)

- Screenshot of sending progress(sender terminal)

- Screenshot of ls -la . output



- Well formatted source code

  client.py

```
# import the necessary librabries
import socket
import tqdm
import os
import argparse

SEPARATOR = "<SEPARATOR>"
BUFFER_SIZE = 4096

def send_file(filename, host, port):
    # total file size read
    total = 0

    # get the file size
    filesize = os.path.getsize(filename)

    # create the client socket
    client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    print(f"[+] One sec... trying to connect to {host}:{port}")
    client_socket.connect((host, int(port)))
    print("[+] Yay, client successfully connected to the server!")

    # send the filename and filesize
    client_socket.send(f"{filename}{SEPARATOR}{filesize}".encode())

    # progress bar to show transfer in percentage
    progress = tqdm.tqdm(range(filesize), f"Current progress of sending
{filename}", unit="B", unit_scale=True, unit_divisor=1024, leave=True)


    # prepare to read the file
    with open(filename, "rb") as f:
        for _ in progress:
            while total != filesize:
                # read the bytes from the file
                bytes_read = f.read(BUFFER_SIZE)

                # to check when file is done transmitting
                if total == filesize:
```

```
                    break

                # send the bytes from the client
                client_socket.sendall(bytes_read)

                # update the progress bar
                progress.update(len(bytes_read))
                total += len(bytes_read)

    # close the file
    f.close()

    # close the socket
    client_socket.close()

if __name__ == "__main__":
    # get the arguments to use in the function call
    parser = argparse.ArgumentParser(description="DS Lab 6")
    parser.add_argument("file")
    parser.add_argument("host")
    parser.add_argument("port")
    args = parser.parse_args()
    filename = args.file
    host = args.host
    port = args.port
    # call the function to send the file
    send_file(filename, host, port)
```

server.py

```
# import necessary librabries
import socket
import tqdm
import os

# total file size written
total = 0

# device's IP address
SERVER_HOST =
socket.gethostbyaddr(socket.gethostbyname(socket.gethostname()))[0]
SERVER_PORT = 5000      # remember to ensure that this port has an inbonud
rule in the ec2 instance

# receive 4096 bytes each time
BUFFER_SIZE = 4096
SEPARATOR = "<SEPARATOR>"

# create the server socket
server_socket = socket.socket()

# bind the socket to our address
server_socket.bind((SERVER_HOST, SERVER_PORT))

# enabling our server to accept connections
server_socket.listen(5)
print(f"[*] Listening as {SERVER_HOST}:{SERVER_PORT}")
```

```
# accept the client connection
client_socket, address = server_socket.accept()

# proof that the client is connected
print(f"[+] {address} is connected.")

# receive the file information
received = client_socket.recv(BUFFER_SIZE).decode()
filename, filesize = received.split(SEPARATOR)

# remove absolute path
filename = os.path.basename(filename)

# get the filesize
filesize = int(filesize)

# RENAMING DUPLICATES
# get the base name
base, ext = os.path.splitext(filename)

# get all contents of the current directory
contents = os.listdir()

copies = base + "_copy"
num = []

# check if any copies already exist
for c in contents:
    # get all numbers of the copies
    if copies in c:
        i = int(''.join(x for x in c if x.isdigit()))
        num.append(i)

# if the file name already exists
if os.path.isfile(filename):
    # and there is no copy, create the first copy
    if len(num) == 0:
        base_copy = base + f'_copy{1}'
        os.rename(filename, base_copy + ext)
    else:
        # if there is a copy get the latest copy and increase the number by
one
        j = max(num) + 1
        base_copy = copies + f'{j}'
        os.rename(filename, base_copy + ext)
else:
    # doesn't exist, so we save it with the same name
    filename = filename


# start receiving the file from the client and writing to the file stream
progress = tqdm.tqdm(range(filesize), f"Receiving {filename}", unit="B",
unit_scale=True, unit_divisor=1024, leave=True)
with open(filename, "wb") as f:
    for _ in progress:
        while total != filesize:
                    # read bytes from the socket
```

```
                    bytes_read = client_socket.recv(BUFFER_SIZE)

                    if total == filesize:
                        # finish transmission
                        break
                    # write to the file the received bytes
                    f.write(bytes_read)

                    # update the progress bar
                    progress.update(len(bytes_read))
                    total += len(bytes_read)

# close the file
f.close()

# close the client socket
client_socket.close()

# close the server socket
server_socket.close()
```

- Link to GitHub repo or gist with source code.

  lab6