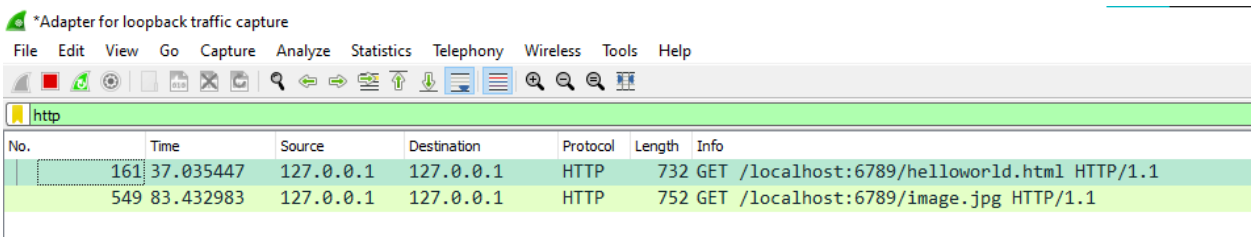
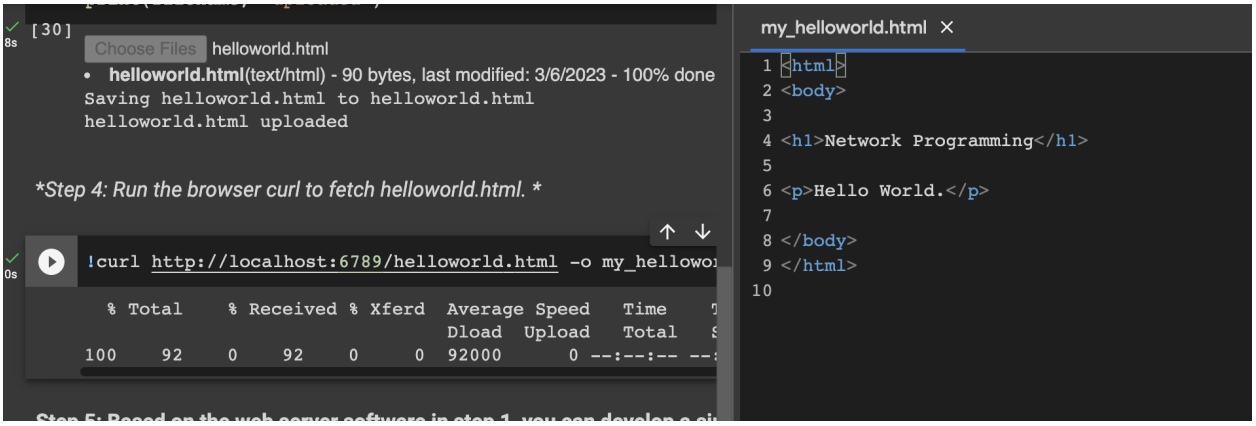
Lab 2: Building a Web Server

1. A brief description of the basic logic of your program.
    a. For our web server, the logic is pretty simple. First, we create a socket and bind to it. After this step, we start listening for any incoming requests. The rest of this program is pretty straightforward in that it parses the packet and sends the contents of the request to the connection socket. In addition to this, there is error handling if the connection socket is able to receive and send the packets.
    b. For our proxy server, the logic is slightly more complicated. It follows the same logic as the web server in creating a socket and binding to it, listening for requests, and sending/receiving packets. However, the difference is that there is a middle man between the client and the web server. This is called the proxy server. In order to do this, we create another socket and bind the existing web server (6789) to the new port (8888). After all of this is done, if the file is successfully read, then we create a file and store it in cache.

2. Sample runs and screenshots as instructed to demonstrate the correctness of your program.
    a. Wireshark capture for proxy:




    b. Google colab photos:



```
[30] filename = next(iter(uploaded))
8s   print(filename, "uploaded")
```

Choose Files helloworld.html

- **helloworld.html**(text/html) - 90 bytes, last modified: 3/6/2023 - 100% done

```
Saving helloworld.html to helloworld.html
helloworld.html uploaded
```

**Step 4: Run the browser curl to fetch helloworld.html. **

```
!curl http://localhost:6789/image.jpg -o my_image.jpg
```

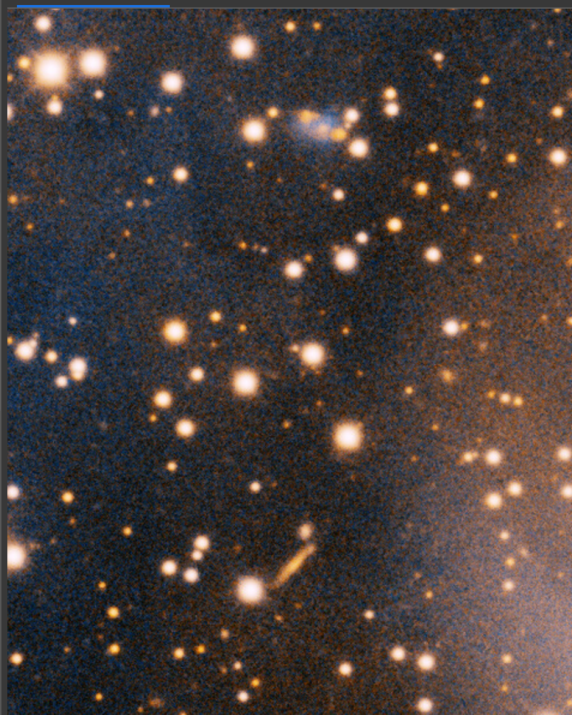
```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 8087k    0 8087k    0     0   717M      0 --:--:-- --:--:-- --:--:--  717M
```

**Step 5: Based on the web server software in step 1, you can develop a si that runs on localhost but at a different port, say 8888. Similar to steps 1 upload your proxy server python file here and run it in the background.**

```
from google.colab import files
print("Upload your proxy server file here")
uploaded = files.upload()
proxyname = next(iter(uploaded))
print(proxyname, "uploaded")
```

```
!nohup python -u $proxyname  >& proxy.log &
```

my_image.jpg



```
!curl http://localhost:6789/helloworld.html -o my_helloworld.html
```

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100    92    0    92    0     0  92000      0 --:--:-- --:--:-- --:--:-- 92000
```

**Step 4: Run the browser curl to fetch helloworld.html. **

```
!curl http://localhost:6789/image.jpg -o my_image.jpg
```

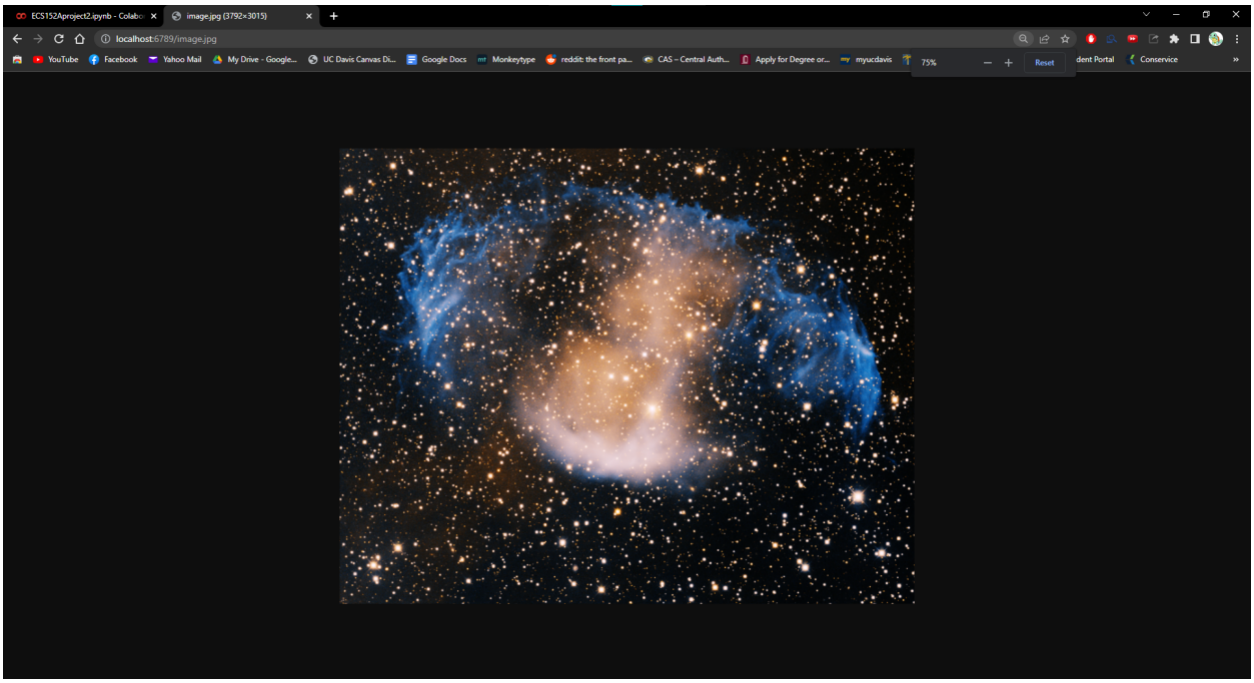
```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 8087k    0 8087k    0     0   717M      0 --:--:-- --:--:-- --:--:--  717M
```

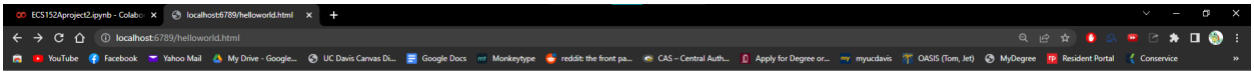
```
!curl http://localhost:8888/localhost:6789/helloworld.html
```

```
<html>
<body>

<h1>Network Programming</h1>

<p>Hello World.</p>

</body>
</html>
```

```
[83] !curl http://localhost:8888/localhost:6789/image.jpg -o myimage.jpg
```

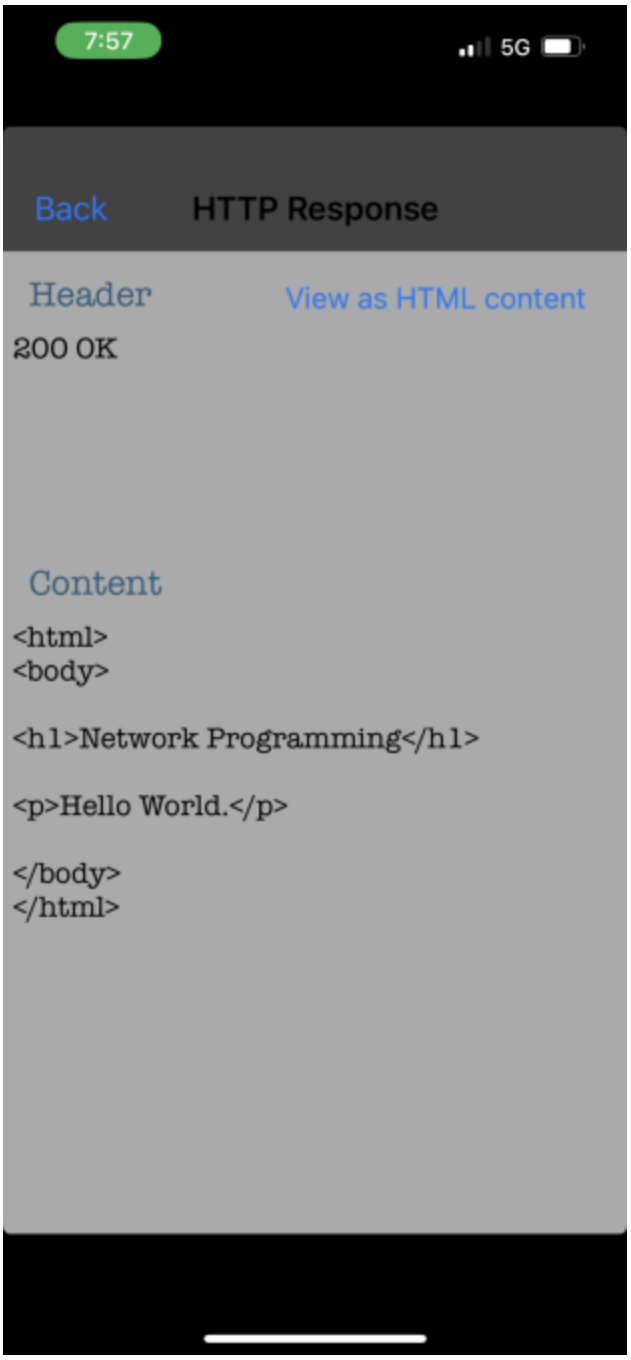
```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time
                                 Dload  Upload   Total   Spent    Left
100 8087k    0 8087k    0     0   394M      0 --:--:-- --:--:-- --:--:-
```



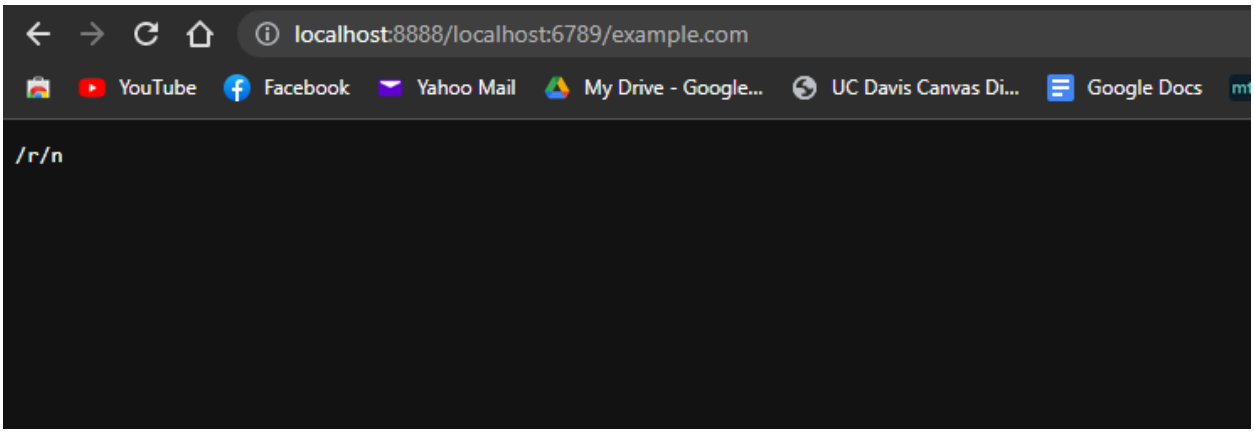



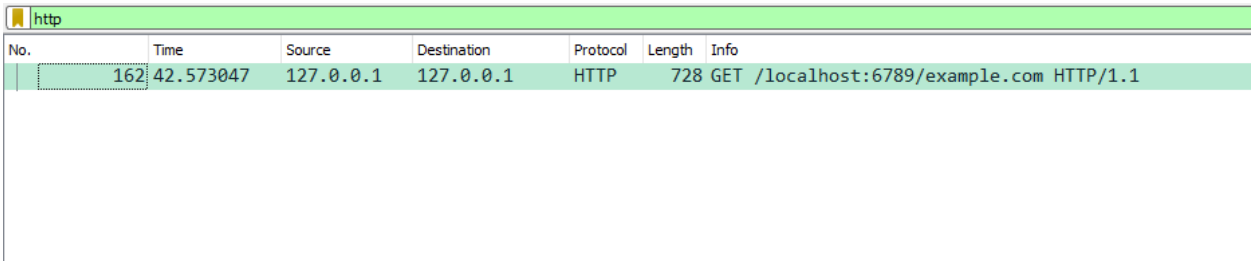

**Network Programming**

Hello World.

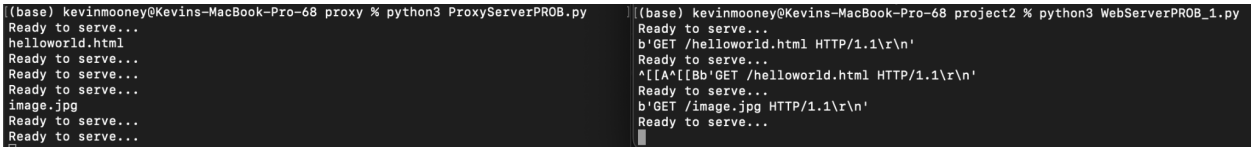

7:57
5G
Back
HTTP Response
Header
View as HTML content
200 OK
Content
<html>
<body>
<h1>Network Programming</h1>
Hello World.
</body>
</html>


7:57
5G
Back
HTTP Response
View as plain text


localhost:8888/localhost:6789/example.com
YouTube
Facebook
Yahoo Mail
My Drive - Google...
UC Davis Canvas Di...
Google Docs
/r/n




c. Terminal output:




3. Answers to questions raised in this assignment.
   a. The performance of our proxy server is faster than the web server because it acts as a caching mechanism. For previously requested content, it recognizes it and speeds up the transaction since it has known info already.
   b. The time required for the proxy server was less than the time required for the web server. This is because the proxy server already had the file in the cache and was able to pull it up faster.
4. Web Server Program

```
# Import socket module
from socket import *
serverPort = 6789

# Create a TCP server socket
#(AF_INET is used for IPv4 protocols)
#(SOCK_STREAM is used for TCP)

serverSocket = socket(AF_INET, SOCK_STREAM)

# Fill in start
serverSocket.bind(("", serverPort))
serverSocket.listen(1)
# Fill in end

# 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() #Fill in start
#Fill in end

    # 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).decode()#Fill in start
#Fill in end

        # 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:], 'rb')

        # Store the entire content of the requested file in a
temporary buffer
        outputdata = f.read()#Fill in start         #Fill in end

        # Send the HTTP response header line to the connection
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 connection
socket
        connectionSocket.sendall(outputdata)
        connectionSocket.send("\r\n".encode())
```

```
        # Close the client connection socket
        connectionSocket.close()

    except IOError:

        # Send HTTP response message for file not found
        # Fill in start
        connectionSocket.send("HTTP/1.1 404 Not
Found\r\n\r\n".encode())
        # Fill in end

        # Close the client connection socket
        # Fill in start
        connectionSocket.close()
        # Fill in end

    serverSocket.close()
```

Proxy Server Program

```
# Import socket module
from socket import *

serverPort = 8888
# Create a TCP server socket
#(AF_INET is used for IPv4 protocols)
#(SOCK_STREAM is used for TCP)

serverSocket = socket(AF_INET, SOCK_STREAM)
webServer = socket(AF_INET, SOCK_STREAM)

# Fill in start
serverSocket.bind(("", serverPort))
serverSocket.listen(1)
# Fill in end

```

```
# 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, clientAddr = serverSocket.accept() #Fill in start             #Fill in end
    print (clientAddr)
    message = connectionSocket.recv(1024)
    filename_whost = message.split()[1]
    filename_b = filename_whost.split(b"/")[2]
    filename = filename_b.decode()
    # 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:
        #if file is in cache
        # Receives the request message from the client
        f = open(filename, 'rb')

        # Store the entire contenet of the requested file in a temporary buffer
        outputdata = f.read()#Fill in start         #Fill in end

        # Send the HTTP response header line to the connection 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 connection socket
```

```
        #for i in range(0, len(outputdata)):
        connectionSocket.sendall(outputdata)
        connectionSocket.send("\r\n".encode())

        # Close the client connection socket
        connectionSocket.close()

    except IOError:
        #if file is not in cache
        print(filename)
        try:
            webServer.connect(('127.0.0.1', 6789))
            tmpstr ="GET "+ "/" + filename + " HTTP/1.1\r\n"
            webServer.send(tmpstr.encode())
            newFile = webServer.makefile('b',0)
        #newFile.write("GET "+ "/" + filename + " HTTP/1.1\r\n")
            tempBuffer = newFile.readlines()

            tempFile= open("./" + filename,"wb")
            for line in tempBuffer:
                tempFile.write(line)
                connectionSocket.send(line)
            connectionSocket.send("/r/n".encode())
        except:

        # Send HTTP response message for file not found
        # Fill in start
            connectionSocket.send("HTTP/1.1 404 Not
Found\r\n\r\n".encode())
        # Fill in end

        # Close the client connection socket
        # Fill in start
```

```
82.             connectionSocket.close()
83.         # Fill in end
84.         serverSocket.close()
```