

### Summary of design and implementation of the server program:

My multi-thread HTTP webserver is written in python with the standard socket and threading library. The *HTTPServer.py* has three major function which are: logging, threading class, and main run function.

#### **Main** function (run):

When the main function starts, it will first initialize the logger by calling the **Logging** function. After that, it will bind the server IP and port and start to listen to that port for HTTP connection. The default IP is local IP (127.0.0.1), and default port is 80 for HTTP protocol. The port can be changed if user input command line argument as port number, but the IP is fixed to local IP and not changeable. Once a HTTP client initiate the connection, a new thread will be created to handle the client. The created thread is created from the **Threading** class and the client IP, port and socket will be passed to the thread for handling the client.

#### **Logging** function (initLogger):

This function will create a logger to write log file named with today's date and other timing information of the logged status.

#### **Threading** class (clientThread):

This threading class serve the most important function to the client request. This class is inherited from the python threading library and override the **initialize** and **run** function to serve the client request. The class **initialize** with three parameters (client IP, port and socket) comes from the main connection and one status flag (self.live: default is true), one timer session flag (self.threadLiveTime: default is 5 seconds). Once it created, it will go to **run** function. In **run** function, it will keep listening to the client until timeout for the session or bad request/ file not found, etc.

During the listening period, the server will decode the client's HTTP request and validate the request by the header value (e.g., not accept the POST request). If the request is not valid, the thread will response 400 Bad request, end the connection and kill the thread. If the request is valid, it will check the requested file/ type and check the server local file status then response the client with 200 OK/ 404 not found/ 304 not modified depends on file status compared to the client and server status. Also, it will extend the session (lifetime) of server and client upon request from client ()

The request status is got from **getRequestedConnectionStatus** function inside the **Threading** class.

This function will accept the client request string as parameter. After is get the client request string, it will decode it to find the key string "Connection" and "Keep-Alive: ". Once it gets the requested status, it will return the connection status Str, keepAliveTime Str and keepAliveTime.

## COMP2322 Project Report

The local file status is got from **getLocalFileStatus** function inside the **Threading** class.

This function will get the local requested file size, last modified time and file type (text/image file).

Once it gets the requested status, it will return the lastModifiedTimeStr, contentLengthStr and fileTypeStr.

Other files beside *HTTPServer.py* attached in the zip file together with the program and report:

*abc.txt*

Testing text file.

*test.png*

Testing image file.

*index.html*

Testing html file.

*TCPClient.py*

Simulation client request program (for bad request).

*readme.md*

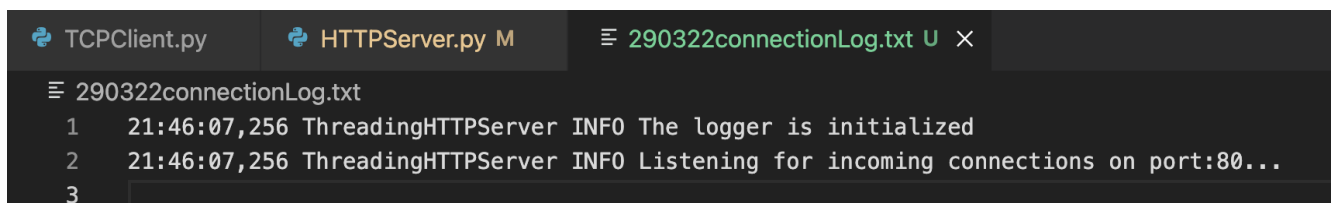
Readme file of instruction to run the program.

*290322connectionLog.txt*

Connection log created when running the server and used in this report.

## Demonstration of executing program & Screen capturing of results of all functions

When the server started, the server will first init the logger by **initLogger()** function and run (port=80) function to open port to listen for requests.



```
⚙ TCPClient.py  ⚙ HTTPServer.py M  ≡ 290322connectionLog.txt U ×
≡ 290322connectionLog.txt
1  21:46:07,256 ThreadingHTTPServer INFO The logger is initialized
2  21:46:07,256 ThreadingHTTPServer INFO Listening for incoming connections on port:80...
3
```

When a client request is initiated, then the server will first get the request sent from browser (clientRequestStr) and determine the response message (the sample response on browser & log status)



## Welcome to the index.html web page.

```
3 21:46:39,728 ThreadingHTTPServer INFO [+] New thread started for 127.0.0.1:63100
4 21:46:39,729 ThreadingHTTPServer INFO clientRequestStr is:
5 GET / HTTP/1.1
6 Host: 127.0.0.1
7 Connection: keep-alive
8 Cache-Control: max-age=0
9 sec-ch-ua: " Not A;Brand";v="99", "Chromium";v="99", "Google Chrome";v="99"
10 sec-ch-ua-mobile: ?0
11 sec-ch-ua-platform: "macOS"
12 Upgrade-Insecure-Requests: 1
13 User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/99.0.4846.83 Safari/537.36
14 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,en-US;q=0.5,en;q=0.7
15 Sec-Fetch-Site: none
16 Sec-Fetch-Mode: navigate
17 Sec-Fetch-User: ?1
18 Sec-Fetch-Dest: document
19 Accept-Encoding: gzip, deflate, br
20 Accept-Language: zh-HK,zh;q=0.9,en-US;q=0.8,en;q=0.7
21
22
23 21:46:39,730 ThreadingHTTPServer INFO Listening for incoming connections on port:80...
24 21:46:39,732 ThreadingHTTPServer INFO Response headerStr is:
25 HTTP/1.1 200 OK
26 Server: Python 2.7
27 Last-Modified: Tue, 29 Mar 2022 20:40:02 GMT
28 Content-Length: 145
29 Keep-Alive: timeout=5, max=100
30 Connection: Keep-Alive
31 Content-Type: text/html
32
```

Browser request

Server response header

## COMP2322 Project Report

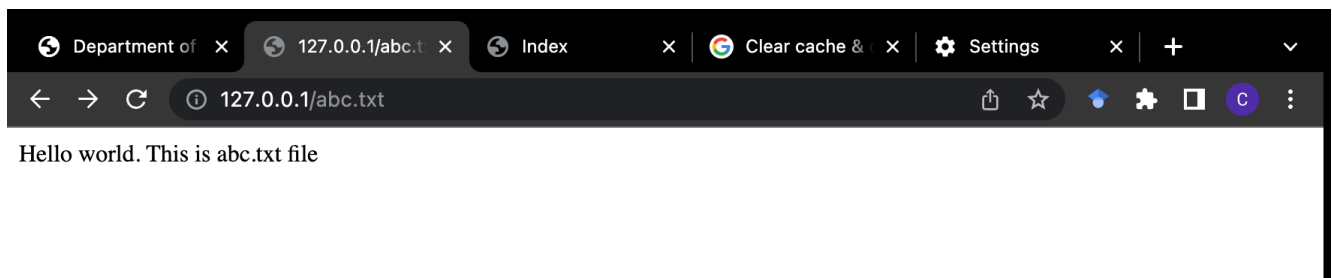
If the user send the same request again, the server will get the last modified time and only response the header, the same situation happen in log below because I have tested the program serval time in browser without clear the browsing history (cached files).

```
34 21:50:07,753 ThreadingHTTPServer INFO clientRequestStr is:
35 GET / HTTP/1.1
36 Host: 127.0.0.1
37 Connection: keep-alive
38 Cache-Control: max-age=0
39 sec-ch-ua: " Not A;Brand";v="99", "Chromium";v="99", "Google Chrome";v="99"
40 sec-ch-ua-mobile: ?0
41 sec-ch-ua-platform: "macOS"
42 Upgrade-Insecure-Requests: 1
43 User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko)
44 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,
45 Sec-Fetch-Site: none
46 Sec-Fetch-Mode: navigate
47 Sec-Fetch-User: ?1
48 Sec-Fetch-Dest: document
49 Accept-Encoding: gzip, deflate, br
50 Accept-Language: zh-HK,zh;q=0.9,en-US;q=0.8,en;q=0.7
51 If-Modified-Since: Tue, 29 Mar 2022 20:40:02 GMT
52
53
54 21:50:07,754 ThreadingHTTPServer INFO Response headerStr is:
55 HTTP/1.1 200 OK
56
```

```
92
93 21:57:03,965 ThreadingHTTPServer INFO [+] New thread started for 127.0.0.1:63375
94 21:57:03,965 ThreadingHTTPServer INFO Listening for incoming connections on port:80...
95 21:57:03,965 ThreadingHTTPServer INFO clientRequestStr is:
96 GET /abc.txt HTTP/1.1
97 Host: 127.0.0.1
98 Connection: keep-alive
99 Cache-Control: max-age=0
100 sec-ch-ua: " Not A;Brand";v="99", "Chromium";v="99", "Google Chrome";v="99"
101 sec-ch-ua-mobile: ?0
102 sec-ch-ua-platform: "macOS"
103 Upgrade-Insecure-Requests: 1
104 User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko)
105 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,
106 Sec-Fetch-Site: none
107 Sec-Fetch-Mode: navigate
108 Sec-Fetch-User: ?1
109 Sec-Fetch-Dest: document
110 Accept-Encoding: gzip, deflate, br
111 Accept-Language: zh-HK,zh;q=0.9,en-US;q=0.8,en;q=0.7
112 If-Modified-Since: Sat, 19 Mar 2022 17:41:04 GMT
113
114
115 21:57:03,965 ThreadingHTTPServer INFO Response headerStr is:
116 HTTP/1.1 200 OK
117
```

## COMP2322 Project Report

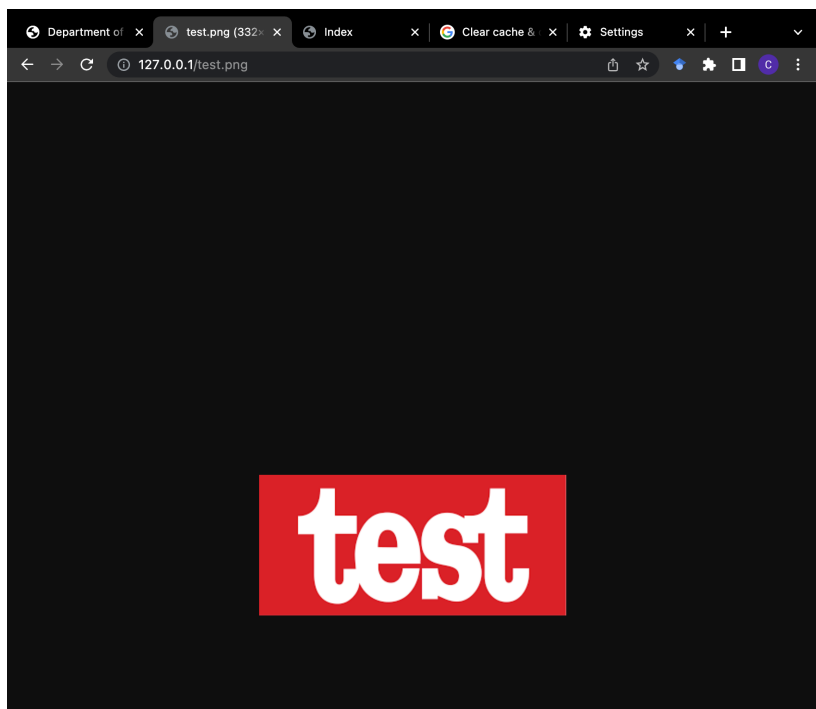
### Another type of Request (text file)



After I manually clear the browser history, the server response normally.

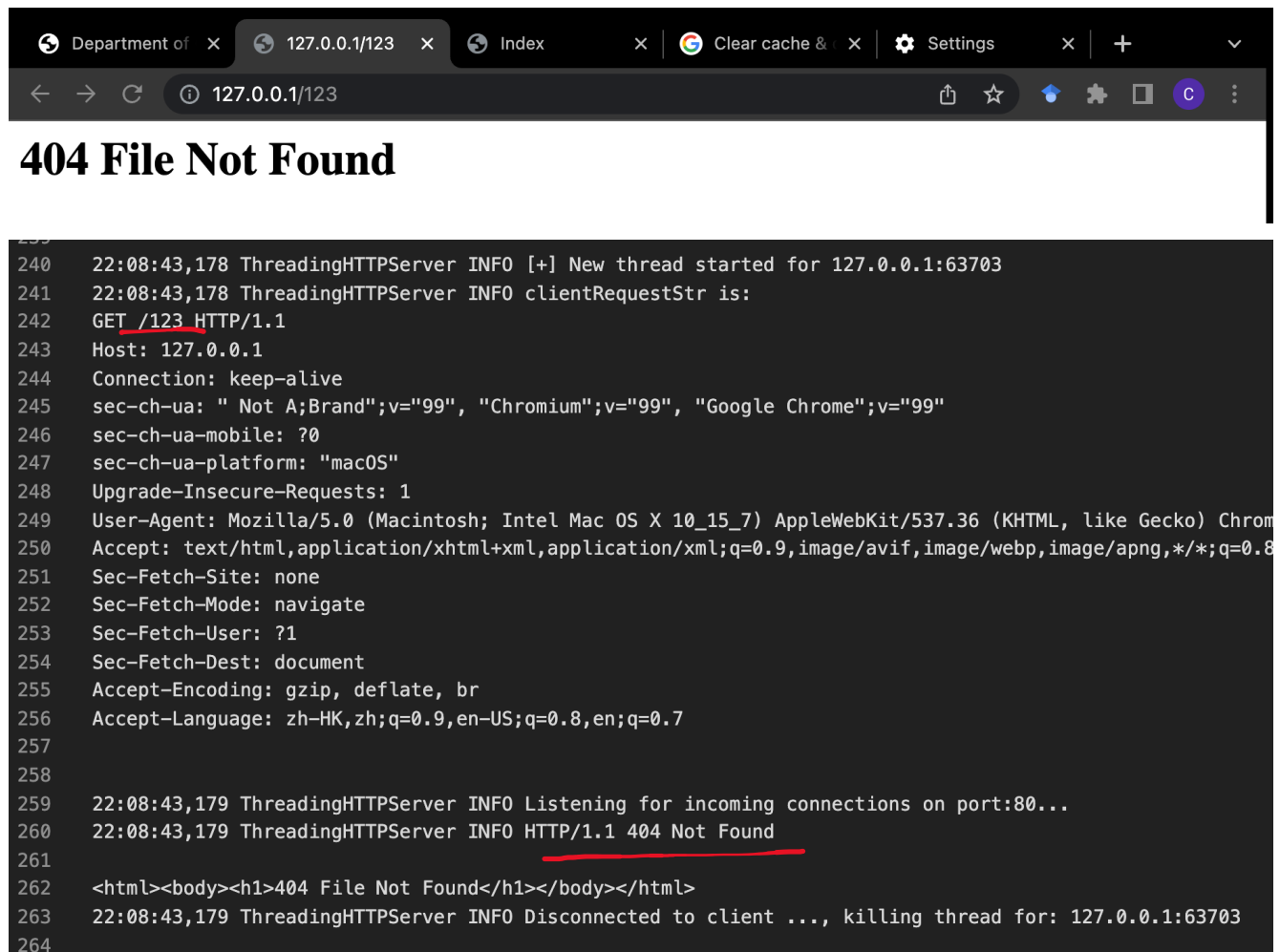
```
119 22:00:13,134 ThreadingHTTPServer INFO clientRequestStr is:
120
121 22:00:13,137 ThreadingHTTPServer INFO [+] New thread started for 127.0.0.1:63467
122 22:00:13,137 ThreadingHTTPServer INFO clientRequestStr is:
123 GET /abc.txt HTTP/1.1
124 Host: 127.0.0.1
125 Connection: keep-alive
126 Cache-Control: max-age=0
127 sec-ch-ua: " Not A;Brand";v="99", "Chromium";v="99", "Google Chrome";v="99"
128 sec-ch-ua-mobile: ?0
129 sec-ch-ua-platform: "macOS"
130 Upgrade-Insecure-Requests: 1
131 User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko)
132 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,
133 Sec-Fetch-Site: none
134 Sec-Fetch-Mode: navigate
135 Sec-Fetch-User: ?1
136 Sec-Fetch-Dest: document
137 Accept-Encoding: gzip, deflate, br
138 Accept-Language: zh-HK,zh;q=0.9,en-US;q=0.8,en;q=0.7
139
140
141 22:00:13,137 ThreadingHTTPServer INFO Listening for incoming connections on port:80...
142 22:00:13,138 ThreadingHTTPServer INFO Response headerStr is:
143 HTTP/1.1 200 OK
144 Server: Python 2.7
145 Last-Modified: Sat, 19 Mar 2022 17:41:04 GMT
146 Content-Length: 33
147 Keep-Alive: timeout=5, max=100
148 Connection: Keep-Alive
149 Content-Type: text/html
150
```

## Another type of Request (image file)



```
181
182 22:07:02,163 ThreadingHTTPServer INFO [+] New thread started for 127.0.0.1:63654
183 22:07:02,163 ThreadingHTTPServer INFO clientRequestStr is:
184 GET /test.png HTTP/1.1
185 Host: 127.0.0.1
186 Connection: keep-alive
187 Cache-Control: max-age=0
188 sec-ch-ua: " Not A;Brand";v="99", "Chromium";v="99", "Google Chrome";v="99"
189 sec-ch-ua-mobile: ?0
190 sec-ch-ua-platform: "macOS"
191 Upgrade-Insecure-Requests: 1
192 User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Ge
193 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng
194 Sec-Fetch-Site: none
195 Sec-Fetch-Mode: navigate
196 Sec-Fetch-User: ?1
197 Sec-Fetch-Dest: document
198 Accept-Encoding: gzip, deflate, br
199 Accept-Language: zh-HK,zh;q=0.9,en-US;q=0.8,en;q=0.7
200
201
202 22:07:02,163 ThreadingHTTPServer INFO Listening for incoming connections on port:80...
203 22:07:02,164 ThreadingHTTPServer INFO Response headerStr is:
204 HTTP/1.1 200 OK
205 Server: Python 2.7
206 Last-Modified: Sat, 12 Mar 2022 19:50:26 GMT
207 Accept-Ranges: bytes
208 Content-Length: 3408
209 Keep-Alive: timeout=5, max=100
210 Connection: Keep-Alive
211 Content-Type: image/png
212
```

Another type of Request (not existing file)



The screenshot displays a web browser window with the address bar showing `127.0.0.1/123`. The page content is a large black rectangle with the text **404 File Not Found** in white. Below this, a terminal log shows the details of the HTTP request and the server's response.

```
240 22:08:43,178 ThreadingHTTPServer INFO [+] New thread started for 127.0.0.1:63703
241 22:08:43,178 ThreadingHTTPServer INFO clientRequestStr is:
242 GET /123 HTTP/1.1
243 Host: 127.0.0.1
244 Connection: keep-alive
245 sec-ch-ua: " Not A;Brand";v="99", "Chromium";v="99", "Google Chrome";v="99"
246 sec-ch-ua-mobile: ?0
247 sec-ch-ua-platform: "macOS"
248 Upgrade-Insecure-Requests: 1
249 User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrom
250 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8
251 Sec-Fetch-Site: none
252 Sec-Fetch-Mode: navigate
253 Sec-Fetch-User: ?1
254 Sec-Fetch-Dest: document
255 Accept-Encoding: gzip, deflate, br
256 Accept-Language: zh-HK,zh;q=0.9,en-US;q=0.8,en;q=0.7
257
258
259 22:08:43,179 ThreadingHTTPServer INFO Listening for incoming connections on port:80...
260 22:08:43,179 ThreadingHTTPServer INFO HTTP/1.1 404 Not Found
261
262 <html><body><h1>404 File Not Found</h1></body></html>
263 22:08:43,179 ThreadingHTTPServer INFO Disconnected to client ..., killing thread for: 127.0.0.1:63703
264
```

## COMP2322 Project Report

### Another type of request (POST) bad request simulation

```
TCPClient.py x HTTPServer.py M 290322connectionLog.txt U
TCPClient.py > ...
1 from socket import *
2 serverName = '127.0.0.1'
3 serverPort = 80
4 serverPath = "/bookstore"
5 clientSocket = socket(AF_INET, SOCK_STREAM)
6 clientSocket.connect((serverName,serverPort))
7
8 request = "POST /test/ HTTP/1.1\r\n\r\n"
9 #getRequest = "GET / HTTP/1.1\r\n" + serverName + ":" + str(serverPort) + serverPath + \
10 #           + "\r\n\r\n"
11 clientSocket.send(request.encode())
12 modifiedSentence = clientSocket.recv(1024)
13 print('From Server (last):', modifiedSentence.decode())
```

```
[Running] python -u "/Users/howingcheng/Documents/COMP2322/Project/TCPClient.py"
From Server (last): HTTP/1.1 400 Bad Request

<html><body><h1>400 Bad Request</h1></body></html>

[Done] exited with code=0 in 0.108 seconds
```

### Server log

```
264 22:10:05,619 ThreadingHTTPServer INFO [+] New thread started for 127.0.0.1:63756
265 22:10:05,620 ThreadingHTTPServer INFO clientRequestStr is:
266 POST /test/ HTTP/1.1
267
268
269 22:10:05,620 ThreadingHTTPServer INFO Listening for incoming connections on port:80...
270 22:10:05,620 ThreadingHTTPServer INFO HTTP/1.1 400 Bad Request
271
272 <html><body><h1>400 Bad Request</h1></body></html>
273 22:10:05,620 ThreadingHTTPServer INFO Disconnected to client ..., killing thread for: 127.0.0.1:63756
274
```

### References:

[https://www.tutorialspoint.com/python/python\\_multithreading.htm](https://www.tutorialspoint.com/python/python_multithreading.htm)

<https://stackoverflow.com/questions/17453212/multi-threaded-tcp-server-in-python>

<https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers>

<https://reqbin.com/Article/HttpHead>

<https://docs.python.org/3/library/socket.html>

<https://stackoverflow.com/questions/57882042/keep-tcp-socket-connection-alive-and-read-write-coordination>

<https://stackoverflow.com/questions/10847157/handling-if-modified-since-header-in-a-php-script>

[https://developer.mozilla.org/en-US/docs/Web/Media/Formats/Image\\_types](https://developer.mozilla.org/en-US/docs/Web/Media/Formats/Image_types)