CSCI 4131 – Spring 2024

Internet Programming Assignment 4

Due Date: Friday, March 22th

Late Submission Deadline: Monday, March 25th

1. Description

The objective of this assignment is for you to bolster your knowledge of the Hyper-Text Transfer Protocol (HTTP) and refactor your HTTP server in Python 3 from the previous assignments so that it can accommodate the addition of new files without requiring changes to the server code. In this assignment, you will need to learn more about response codes, MIME types, query strings, and Python.

An outline of how your refactored server should be different from the server from Programming Homework Assignment 3 is specified below.

- Return the file requested file based on its (Multipurpose Internet Mail Extension (i.e., MIME type) - the long if-else statement goes away.
- When a non-existing file is requested, return a 404 response.
- When a non-accessible file is requested, return a 403 response. See the following discussion for details.
- The server can redirect requests when needed. See the that follows for details.
- The server can process query strings received.
- The server can perform file input and output (I/O), write a log to a local file, and for BONUS points, search for local files.
- All the functionality from previous assignments should still work. (responding to form submissions, etc.)

2. Required Functionality

Download additional files

You will need to download the following files for this assignment:

- 403.html - this file should be sent to the client if permissions do not permit its access (Provided).
- 404.html - this file should be sent to client if the server cannot find the requested file (Provided)
- private.html - this file is the private file that triggers 403 forbidden code, you can use it to test.
- Coffman.html – html file containing an image to use for testing your server's capability to respond to a request for an image (Provided).
- OuttaSpace.html – html file containing an audio controls element to use for testing your server's capability to respond to a request for an audio file (Provided).
- Coffman_N_OuttaSpace.html– contains both an image and audio control element to use for testing your server's capability to respond to a request for an image and an audio file (Provided).
- coffman.png – the image file (a .png file) used by Coffman..html and Coffman_N_OuttaSpace.html (Provided).
- OuttaSpace.mp3 – the audio file (a .mp3 file) used by OuttaSpace.html and Coffman_N_OuttaSpace.html (Provided).

We have provided files listed above in the file named: **Hw4Resources.zip**

IMPORTANT NOTE: YOU WILL HAVE TO SET PERMISSIONS ON ALL THE FILES AND DIRECTORIES IN YOUR DIRECTORY FOLDER STRUCTURE

ALSO – YOU WILL HAVE TO TEST YOUR HOMEWORK ON A UNIX OR LINUX OPERATING SYSTEM. WINDOWS USERS WHO DO NOT USE THE WINDOWS SUBSYSTEM FOR LINUX CAN USE THE CSE LABS MACHINES.

We suggest setting all directory and file permissions to 644 with the chmod command:
For example: `chmod 644 myDirectory_or_myFile`

If that does not work, use 755 instead!!!!!

The following tutorial on file permissions may be helpful:

https://www.tutorialspoint.com/unix/unix-file-permission.htm

Additionally, you will need the files from your previous homework. Include CSS you've written, scripts, and your HTML pages from Homework 3.

I. Updated file structure and file references

As with the previous programming assignments, you should store your files in a standardized directory/folder structure. The changes required for this assignment should be straightforward if you followed the instructions for the previous assignments.

Figure 1: An example of the required directory structure – there is a new directory/folder named audio (to store audio files) and a new directory/folder named: files for the bonus functionality.

Figure 2: An example of the structure of the static folder. An audio folder is added to the directory.

For the static folder, specifically, you should have 5 different subdirectories. Each of these subdirectories can only contain files that match its category. For example, CSS files shouldn't be found in your `images` directory – they should be stored in the CSS directory/folder.

With this new structure, you should update your HTML to properly point to the new location of each image, stylesheet, script, mp3, etc. file to make sure they work properly.

II. GET Request re-implementation for "compatibility"

GET requests are the most commonly used HTTP requests. For example, if you enter the following address in your browser's address bar:

http://localhost:4131/MySchedule.html

the browser will issue a GET request to the server to fetch the MySchedule.html file from the directory in which the server code resides. Below is an example of a GET request received by your server:

```
GET /MySchedule.html HTTP/1.1
Host: localhost:4131
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:33.0) Gecko/20100101 Firefox/33.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

Compatibility:

When grading assignments 1, 2, and 3, we found some compatibility issues. For example, http://localhost:4131/MySchedule.html results in a 404 response (file not found) since in assignments 2 and 3 it was moved to http://localhost:4131/static/html/MySchedule.html.

In this assignment, you need to fix that compatibility issue, so that both http://localhost:4131/MySchedule.html and http://localhost:4131/static/html/MySchedule.html can be processed correctly by the server and return the MySchedule.html file located in the static/html folder.

III. MIME-types-based response:

For the previous assignment, you were given a server that had a long if-else statement like the following:

```
elif url == "/css/style.css":
    return open("css/style.css").read(), "text/css"
elif url == "/css/style_aboutme.css":
    return open("css/style_aboutme.css").read(), "text/css"
elif url == "/img/Lind.jpg":
    return open("img/Lind.jpg", "br").read(), "image/jpeg"
elif url == "/img/Tate.png":
    return open("img/Tate.png", "br").read(), "image/png"
elif url == "/img/zoom.jpg":
    return open("img/zoom.jpg", "br").read(), "image/jpeg"
elif url == "/img/vanCleve.jpg":
    return open("img/vanCleve.jpg", "br").read(), "image/jpeg"
elif url == "/img/breakfast.jpg":
    return open("img/breakfast.jpg", "br").read(), "image/jpeg"
elif url == "/img/track.jpg":
    return open("img/track.jpg", "br").read(), "image/jpeg"
elif url == "/img/AboutMe.png":
    return open("img/AboutMe.png", "br").read(), "image/jpeg"
elif url == "/img/Als.png":
    return open("img/Als.png", "br").read(), "image/jpeg"
elif url == "/img/blegan.jpg":
    return open("img/blegan.jpg", "br").read(), "image/jpeg"
elif url == "/img/cell-blue.jpg":
    return open("img/cell-blue.jpg", "br").read(), "image/jpeg"
elif url == "/img/coffman.jpg":
    return open("img/coffman.jpg", "br").read(), "image/jpeg"
```

Figure 3: An example of the old if-else statement

In this assignment, you need to refactor this long if-else, and instead, make your server return a corresponding file based on its MIME type. Specifically, your server is required to automatically decide if the file needs to be opened in "`br`"`(binary read)` mode, and automatically set the MIME type in the response.

You may still need several if-else statements for 404, 403, and 307(redirect) checking and selecting whether to use "br" (binary read) mode, but when returning an existing and accessible file, you cannot hard-code any file names in this assignment.

Your server should be able to properly return the following types of files:
`html, css, js, png, jpg/jpeg, mp3, txt.`

To ensure you understand the MIME for files, the following link will be helpful:

https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/MIME_types.

IV. Error Checking:

404 Responses

If a requested file doesn't exist, a GET request should return a 404 Response (return the contents of the file ***404.html*** in the body of the response).

403 Responses

If a file cannot be accessed, a GET request should return a 403 Response (return the contents of the file: ***403.html*** in the body of the response).

REMEMBER: You are required to test your solution on a Linux or Unix machine such as the Department of Computer Science Computers running the Ubuntu OS to ensure you test your solution on files with permissions set.

To forbid a file from being accessed, you can remove (minus) the reading permission to the file, for example:

```
chmod 640 private.html
```

If the web server does not have permission to access requested resources (e.g., private.html), your server should create a response message with a 403 error response code and 403.html which is sent to the requesting client.

Figure 4: When requested "private.html" which is not accessible, the server should return the contents of the 403 page (403.html)

V. Add the MyServer Page (and functionality specified below)

Figure 5: An example of an updated MyServer Page

The provided HTML page `**MyServer.html**` should added to your website and be updated to have the following functionality:

1. **A new form that sends a GET request to `/redirect`. This redirection should be handled by the server.**

When a user submits the form, the form's content should be used by the server to create a redirect response. This redirect response should send the user to a YouTube search results page for their query. When testing this functionality using a browser, the server's response should enable the browser to automatically show the results of the YouTube search.

For full credit, you need to allow the user to specify from a selection of possible websites which one they want to search. The client can then choose if they want to search Google or search for YouTube videos.

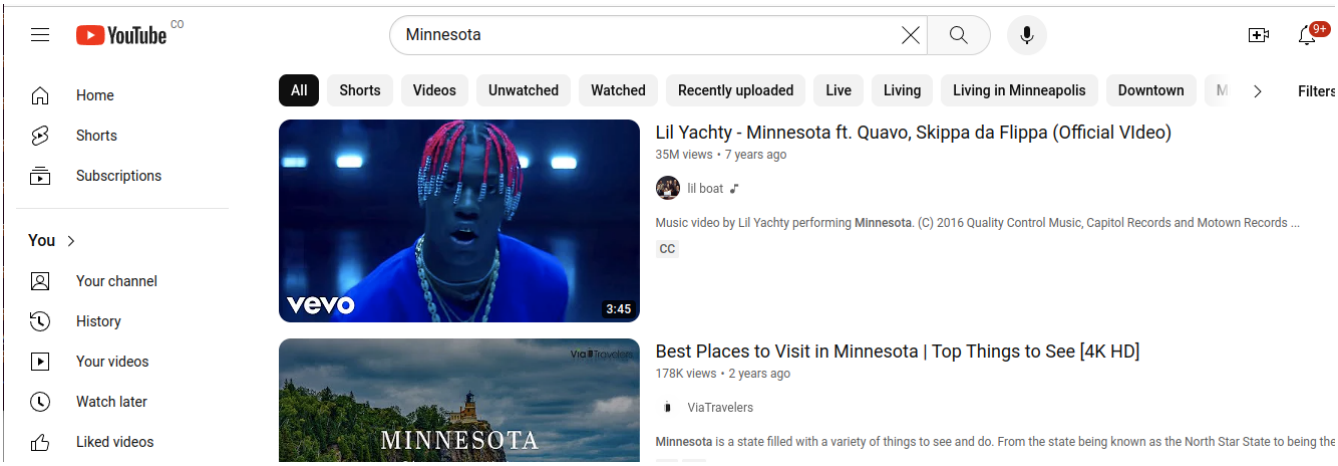


Figure 6: Example, when you search "Minnesota" with YouTube, the server should redirect you to the YouTube search result.

Your server should handle these requests and return a response with an HTTP 307 (Temporary Redirect) status code. You can read more about the status code 307 in section 10.3.8 of RFC 2616. (See: https://www.ietf.org/rfc/rfc2616.txt)

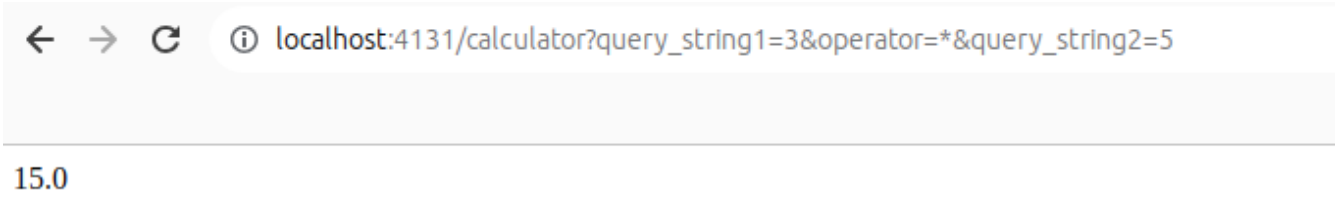
This response will have an empty body. However, in the response headers, you should specify a 'Location' header with a value equal to a YouTube URL. On receiving the response, the browser reads this Location header and navigates to the given YouTube URL.

Note – doing this will involve parsing the query-string part of the URL to extract what the user searched for. You are expected to handle searches with spaces and other "special characters". The function (provided via import at the top of the file) unquote plus can be used to parse the URL encoding used by GET requests.

2. **A new form that sends a GET request to `/calculator` which contains a query string.**

Your server should perform the calculation requested and send the result in a response to the requesting client (your browser) The calculator should have the following 4 operators: `+, -, *, and /`.

Figure 7 and 8: Example, when you select "*" operator, and input 3, and 5 as the parameters, your html will send a request to the server, and the server should respond "15" or "15.0"



ALL pages on your website should have navigation links to your new MyServer page (with the new forms described above), and they should work properly.

Links to navigate to the Coffman.html, OuttaSpace.html, and Coffman_N_OuttaSpace.html pages should work properly.

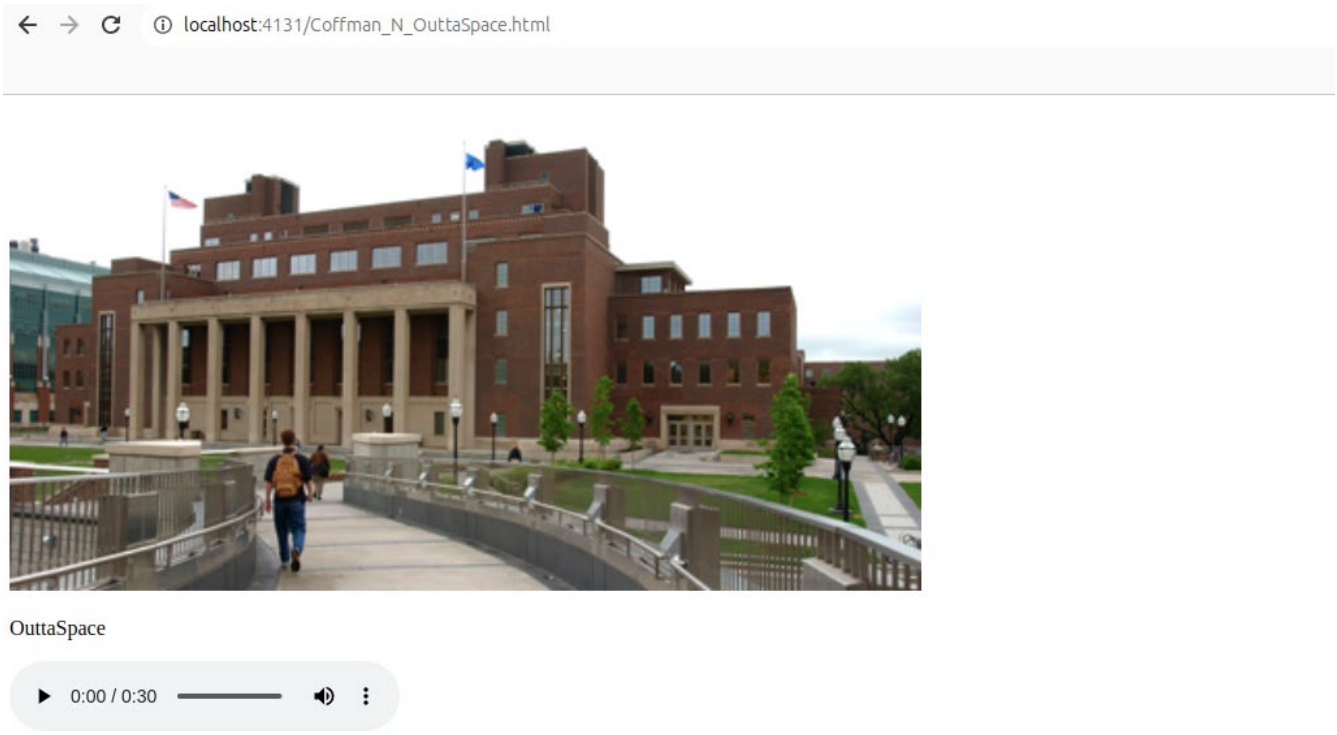


Figure 9: Coffman_N_OuttaSpace.html, by clicking the play button, the mp3 file should be played.

VI. (Bonus Functionality) A file explorer:

Figure 10: Example of the possible contents in the "files" folder

When clicking the “File explorer” link, on your MyServer page, your server should display all supported files in the files folder (See Figure 11 below).

Then, once the files in the files directory are displayed by your browser, if you click on the file name, the corresponding file should be displayed.

Hint: ***To list the files, “glob” in Python might be a possible solution.***

Your file explorer should support all the file types supported by your server, which are:

`html, css, js, png, jpg/jpeg, mp3, txt.`

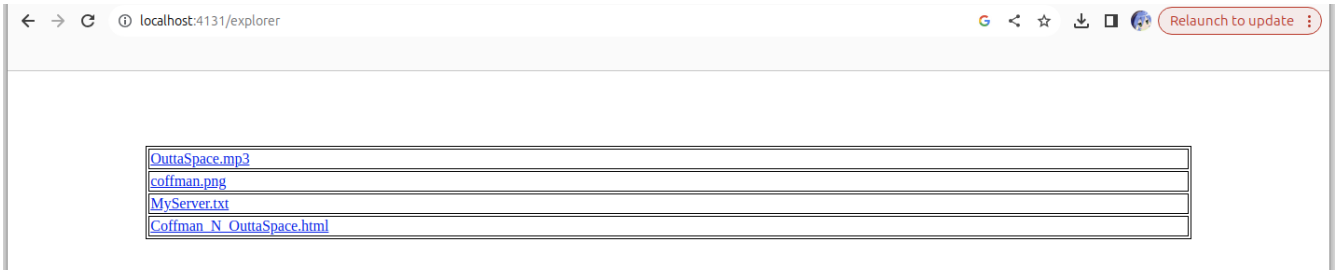


Figure 11: The file explorer should display all supported files in your files folder.

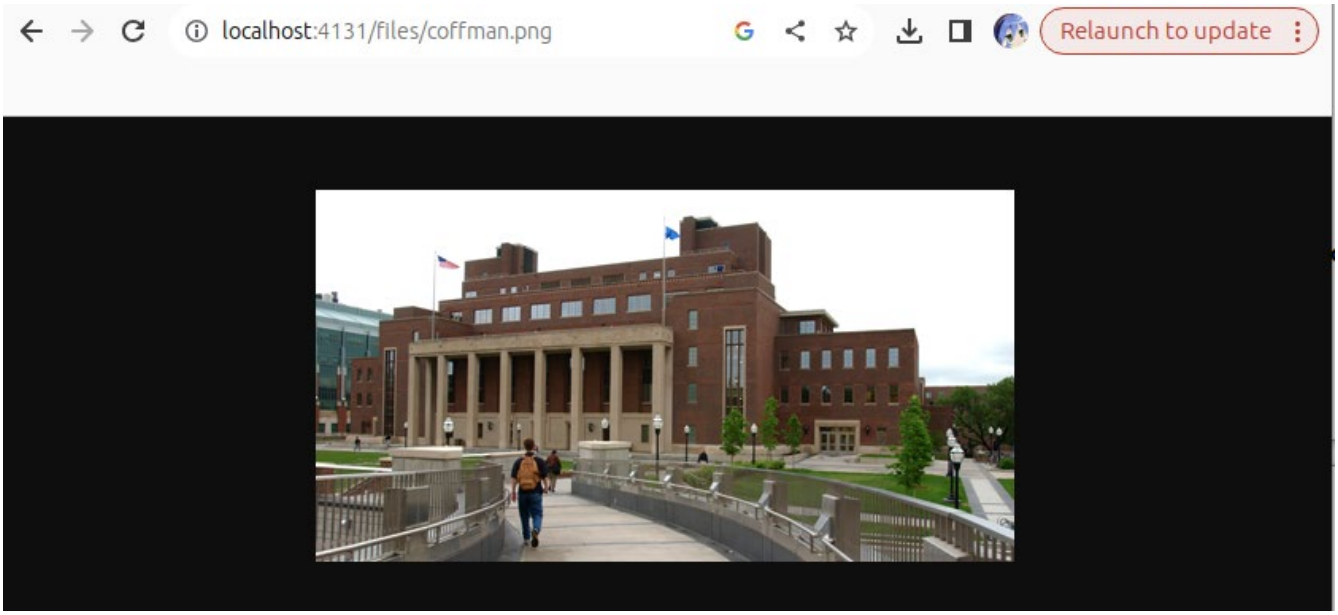


Figure 12: When clicking the “coffman.png”, the image is displayed.

VII. Update the server to log each response to a request

Your server should log its responses to a local file "response.log", which contains the time and information of each response it sends. The information logged should contain, at a minimum, the STATUS CODE and HEADER of the response.

Note: ***Do not log the "message" inside the response, which can be very large.***

```
19:15:27, [200, {'Content-Type': 'text/html; charset=utf-8', 'Content-Length': 165, 'X-Content-Type-Options': 'nosniff'}], /favicon.ico
19:15:30, [200, {'Content-Type': 'text/html; charset=utf-8', 'Content-Length': 165, 'X-Content-Type-Options': 'nosniff'}], /favicon.ico
19:15:31, [200, {'Content-Type': 'text/html', 'Content-Length': 13555, 'X-Content-Type-Options': 'nosniff'}], /html/MySchedule.html
19:15:31, [200, {'Content-Type': 'text/html; charset=utf-8', 'Content-Length': 165, 'X-Content-Type-Options': 'nosniff'}], /html/css/style.css
19:15:31, [200, {'Content-Type': 'text/html; charset=utf-8', 'Content-Length': 165, 'X-Content-Type-Options': 'nosniff'}], /html/js/script.js
19:15:31, [200, {'Content-Type': 'text/html; charset=utf-8', 'Content-Length': 165, 'X-Content-Type-Options': 'nosniff'}], /html/js/map.js
19:15:31, [200, {'Content-Type': 'image/png', 'Content-Length': 3339565, 'X-Content-Type-Options': 'nosniff'}], /img/Tate.png
19:15:31, [200, {'Content-Type': 'image/jpeg', 'Content-Length': 163652, 'X-Content-Type-Options': 'nosniff'}], /img/Lind.jpg
19:15:31, [200, {'Content-Type': 'image/jpeg', 'Content-Length': 37561, 'X-Content-Type-Options': 'nosniff'}], /img/zoom.jpg
19:15:31, [200, {'Content-Type': 'image/jpeg', 'Content-Length': 12318, 'X-Content-Type-Options': 'nosniff'}], /img/vanCleve.jpg
19:15:31, [200, {'Content-Type': 'image/jpeg', 'Content-Length': 11766, 'X-Content-Type-Options': 'nosniff'}], /img/breakfast.jpg
19:15:31, [200, {'Content-Type': 'image/jpeg', 'Content-Length': 166895, 'X-Content-Type-Options': 'nosniff'}], /img/track.jpg
19:15:31, [200, {'Content-Type': 'image/png', 'Content-Length': 82836, 'X-Content-Type-Options': 'nosniff'}], /img/gophers-mascot.png
19:15:31, [200, {'Content-Type': 'image/png', 'Content-Length': 10892, 'X-Content-Type-Options': 'nosniff'}], /img/search.png
19:15:31, [200, {'Content-Type': 'image/png', 'Content-Length': 25240, 'X-Content-Type-Options': 'nosniff'}], /img/direction.png
19:15:31, [200, {'Content-Type': 'text/html; charset=utf-8', 'Content-Length': 165, 'X-Content-Type-Options': 'nosniff'}], /html/css/style.css
19:15:31, [200, {'Content-Type': 'text/html; charset=utf-8', 'Content-Length': 165, 'X-Content-Type-Options': 'nosniff'}], /favicon.ico
09:54:51, [200, {'Content-Type': 'text/html; charset=utf-8', 'Content-Length': 165, 'X-Content-Type-Options': 'nosniff'}], /AboutMe.html
09:55:01, [200, {'Content-Type': 'text/html', 'Content-Length': 1242, 'X-Content-Type-Options': 'nosniff'}], /html/MyServer.html
09:55:01, [200, {'Content-Type': 'text/html; charset=utf-8', 'Content-Length': 165, 'X-Content-Type-Options': 'nosniff'}], /favicon.ico
09:55:07, [200, {'Content-Type': 'text/html', 'Content-Length': 5007, 'X-Content-Type-Options': 'nosniff'}], /html/MyForm.html
09:55:07, [200, {'Content-Type': 'text/html; charset=utf-8', 'Content-Length': 165, 'X-Content-Type-Options': 'nosniff'}], /html/css/style.css
09:55:07, [200, {'Content-Type': 'text/html; charset=utf-8', 'Content-Length': 165, 'X-Content-Type-Options': 'nosniff'}], /html/js/script.js
09:55:07, [200, {'Content-Type': 'text/html; charset=utf-8', 'Content-Length': 165, 'X-Content-Type-Options': 'nosniff'}], /html/js/map.js
09:55:07, [200, {'Content-Type': 'text/html; charset=utf-8', 'Content-Length': 165, 'X-Content-Type-Options': 'nosniff'}], /html/css/style.css
```

Figure 13: Example of contents of the file containing the Logged responses

Grading Criteria

1. All files should be organized in the directory structure requested above **5 points**
2. All HTML files and scripts should be updated to make GET requests for images, styling, scripts, etc. A web page with 4 images should make 5 total GET requests (ignoring styling/scripts) to the web server. **5 points**
3. The HTML files can be accessed in the compatible mode. (That is, both http://localhost:4131/MySchedule.html and http://localhost:4131/static/html/MySchedule.html can be processed correctly by the server and returns the ***MySchedule.html*** file located in the ***static/html folder***) **5 points**
4. GET requests work correctly if a file exists. **20 points**
    i. Images can be requested and returned correctly (2 points)
    ii. JavaScript, CSS, and HTML all can be requested and returned correctly (4 points)
    iii. HTTP status codes and response headers are all correct (2 points)
    iv. MP3 files are properly returned (2 points)
    v. No filenames are hard-coded except 404.html and 403.html when processing the get request. In other words, eliminate the long if-else statement shown in the image. (10 points)
5. 404 Responses are sent if a requested file doesn't exist **5 points**
6. Permission-related requests are handled properly with a 403 response if the file isn't properly accessible **5 points**
7. A redirect request properly routes the user to YouTube **5 points**
8. Redirect also works for Google search. **5 points**
9. Log the responses to a local file **10 points**
10. Calculator works properly. **5 points**
11. Key **NON-GOOGLE MAPS related** Functionality from previous assignments works properly (Remember to include your MySchedule.html and MyForm.html in the submission)
    YOU WILL HAVE TO OBTAIN YOUR OWN KEY FOR GOOGLE MAPS IF YOU WANT THAT FUNCTIONALITY TO REMAIN OPERATIVE – IT IS NOT REQUIRED FOR THIS ASSIGNMENT. **5 points**
12. **(Bonus) File explorer works properly, 10 points**

SUBMISSION INSTRUCTIONS:

Your submission should be packaged in a tar file or zip file. When opened, it must create a directory named: '<Your UMN x.500 ID>' containing ALL of your directories and files (for example server.py, HTML, CSS, Pictures, audio - and any additional external JavaScript and CSS files). Submit your homework via the class Canvas item named Homework 4 Submission Link in the Assignments section of the class Canvas site.