Yap Jas Per

32213835

Global Software

Cloud Computing

**HW1**

# **Introduction**

The given homework is to make a simple web server that opens and listens to port 80. This homework provides a personal refresher on computer network class and some refresher for HTML and Python codes, while also learning new concepts and libraries for python and the essentials for building a web server for Cloud Computing class.

# **Concepts**

The main concept for this homework is computer networks. This homework applies the theory learned in that class and applies it into a working code. Port 80 is required, which is a privileged port for HTTP (Hyper-text Transfer Protocol).

# **Code Design Description**

The local host machine is used as both the server and client, which means that it is not connected to the wider Internet; thus, it cannot be accessed remotely. A ‘web’ directory is created to separate the HTML files from the Python files. The HTML files allow simplicity in responding to GET requests from the client and allows this program to be more malleable in the future if needed. Multi-threading is used to handle multiple client connections concurrently. For certain pages, description on the homework is written onto the HTML files.

## *Importing Required Modules*

* ‘socket’ is required for socket programming.
* ‘threading’ is for multi-threading.
* ‘os’, which stands for operating system, is for file operation and directory listings.

## *Configuration Variables*

Port number is set to 80, while host is set to localhost. These are used to create and open the server.

## *Server Function (‘srvr’)*

This function handles incoming client requests.

* Receiving data: the variable *data* is responsible for reading and storing incoming data from the client.
* Parsing request: received data is split and extracts relevant data.
* GET Requests:
  + The ‘/dynamic’ path dynamically generates a list of files within the current directory.
  + Other paths attempt to retrieve and read the corresponding HTML files in the ‘web’ directory.
  + If the file does not exist of any other errors arise when finding the HTML file, it serves ‘error404.html’ file.
* Get Response: Constructs the HTTP response with appropriate headers and body based on the request.

## *Main Function*

This is the main driver code of the program. It sets the server up and runs it.

* Socket initialization: Creates and initializes the socket and server based on variables given.
* Listening for connections: The server listens for incoming client connections after the server is created and running.
* Accepting connections: Accepts incoming client connections and creates a new thread to handle each client request.

### *Multi-threading*

Multi-threading is used to handle multiple client connections at the same time. Each client c0nnection is handled in its own thread; thus, the server can handle multiple threads simultaneously.

### *Exception Handling*

Keyboard interrupts (Ctrl+C) shut down the server and close the socket.

# **Build Configuration / Environment**

I am using PyCharm along with Sublime Text. The main function is in *socketSrvr.py.* Once the program is up and running, *localhost* is typed into the URL bar of a browser. The following are several paths that are available to test out.

* localhost/test
* localhost/about
* localhost/index
* localhost/error404

If a path is written but the corresponding HTML is not found, it will redirect to the error404 page.

## *Sublime Text*

I use Sublime Text primarily for creating and coding HTML files. In terms of the main functions of this program, it does not play a main part except for making HTML files which are used in response to client requests.

## *PyCharm*

PyCharm is and IDE from JetBrains. In the beginning of doing this homework. Python 3.9 is used as the interpreter for this program.

# **Working Proofs / Screen Captures**

# **A screenshot of a computer Description automatically generated**

Localhost accessing index.html

A screenshot of a computer

Description automatically generated

Server printing out connections

A screenshot of a computer

Description automatically generated

Dynamic path listing files in the directory of socketSrvr.py

A screenshot of a computer

Description automatically generated

Server accepting GET request for dynamic path

A screenshot of a computer

Description automatically generated

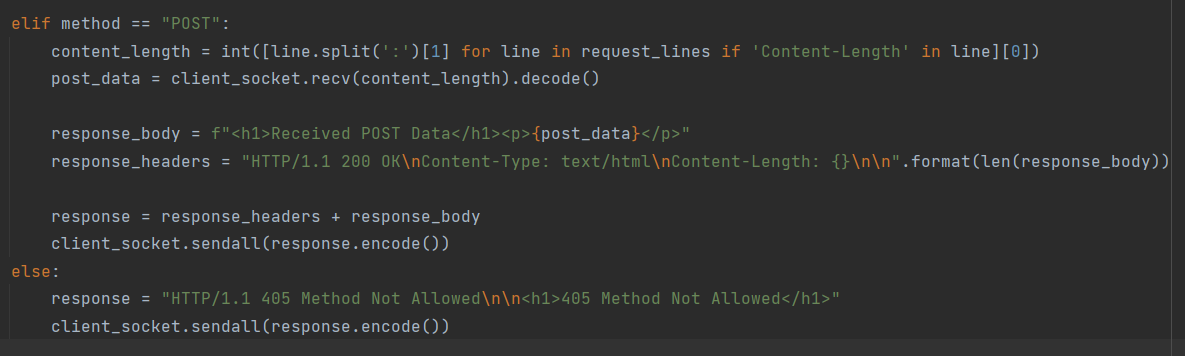
Checking GET request

# **Trials & Errors**

I used PyCharm to run 2 Python files, which are the server and client respectively. With this, I was able to learn the fundamentals of socket programming, with additional help from TutorialsPoint and StackOverflow online. I also used ChatGPT to explain certain libraries and functions I did not fully understand.

I have yet to fully grasp how to check my GET method from the terminal, as I have yet to fully understand the commands and parameters for it; therefore, I am not entirely sure that my GET method is fully functional as I intended.

I attempted to implement POST method; however, I was not fully able to grasp on checking whether it worked. Thus, I decided to remove it from the program as I felt like I was running out of time for this assignment. Below is the removed code and error in checking through the terminal respectively:



A screen shot of a computer

Description automatically generated

# **Conclusion**

Overall, I learned a lot from making this program. I managed to refresh a lot of skills and knowledge that I acquire years ago, specifically HTML. For the sake of simplicity, I chose to only code the web pages in HTML. If given the opportunity or time, I would experiment this program along with JavaScript (.js) files and Cascading Style Sheets (.css), as I think it would be useful in the future. I also refreshed concepts from operating systems to ensure that this program runs only on my local machine.

However, from reading other repositories online, I realised that my program is very rudimentary and only runs the essentials of the requirements of the assignment. There are more exception handlings that should have gone into this program.

In summary, I did enjoy doing this assignment. It was very different and experimental for me. I do not like making web pages, as I feel like I am not artistic enough to make anything of the sort. On the other hand, I did enjoy trying out listening to different ports and attempting my server and client listen to each other in different ways. If given the chance, I would learn more about the specific codes and libraries that go into making a web page functional and safe.