

# CSEN 317: Distributed Systems

## Programming Assignment 1

Naveena Avula  
SCU ID : 07700009231

---

### **Introduction:**

The objective of this programming assignment is to develop a functional web server, focusing on essential distributed programming principles, client-server communication, and the challenges of building web server functionality.

Once a connection is established, the server uses a dedicated thread to handle image file requests, ensuring efficient processing for both HTML and image data. This multithreaded design allows the server to manage the resource-heavy task of image handling without impacting overall server responsiveness.

A key feature of the program is its connection timeout mechanism. Each connection is maintained for a specified period of 10 seconds, ensuring connections are not held open unnecessarily. This timeout approach aids in resource management, preventing bottlenecks and potential resource exhaustion.

The server adheres to HTTP/1.1 specifications to handle incoming requests, while also supporting HTTP/1.0. Adding support for HTTP/1.0 allows the server to work with an earlier version of the Hypertext Transfer Protocol, which uses stateless connections and limited headers, with individual connections per request-response cycle. This standards-compliant approach ensures that the server remains compatible with a wide range of browsers and web clients.

### **Program Specifications:**

- **Functionality:**
  - The program operates as a functional web server in Python.
  - It serves files from a designated directory upon receiving HTTP requests from clients.
- **HTTP Methods:**
  - Primarily handles GET requests as part of basic HTTP method support.
- **HTTP Status Codes:**
  - Implements various HTTP status codes, including:
    - 200 (OK)
    - 400 (Bad Request)

- 403 (Forbidden)
- 404 (File Not Found)
- 501 (Not Supported)
- **Response Headers:**
  - Incorporates key response headers such as:
    - Content-Type
    - Content-Length
    - Date
- **File Types:**
  - Supports a range of file formats, including .pdf, .jpg, .txt, .png, .gif, .mp4 and .html.
- **Concurrency:**
  - Utilizes multi-threading to manage multiple client requests concurrently.
  - Features a default idle timeout of 500 seconds for active connections.

### **Uploaded Files:**

- error (contains 400.html, 403.html, 404.html, 501.html)
- all\_files.html
- collegelogo.jpeg
- error\_page.gif
- index.html
- index1.html
- makefile
- sample\_video.mp4
- server\_def.txt
- server\_image.png
- server.py
- socket\_photo.jpg
- web\_server.pdf

### **Code Execution:**

Python3 <document name>.py -document\_root <path to folder> -port <number btw 8000 to 9999>

- Example: python3 server.py -document\_root /Users/naveenaa/ds\_programming -port 8080

Launch a web browser or client that can connect to the server's designated port and send HTTP requests.

- Example: <http://localhost:8080>.

```

server.py  X  web_server.pdf  collegelogo.jpeg  index.html  index1.html  makefile  all_files.html

server.py > WebServer > initialize
1 # Import required standard Python libraries
2 import time # For adding delays and timing operations
3 import argparse # For parsing command line arguments
4 import mimetypes # For determining file content types
5 import os # For file and path operations
6 import threading # For handling multiple client connections
7 import socket # For network operations
8 from urllib.parse import unquote # For decoding URL-encoded characters
9 from datetime import datetime # For generating timestamps
10
11
12 def detect_content_type(file_location):
13     """
14     Determine the file extension from a given file path
15     Args:
16         file_location: Path to the file
17     Returns:
18         The lowercase file extension including the dot
19     """
20     # Split the file path into name and extension
21     extension = os.path.splitext(file_location)
PROBLEMS 9  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

naveenaa@Naveenas-MacBook-Pro ds_programming % python3 server.py -document_root /Users/naveenaa/ds_programming -port 8080
Web server initialized.
Port: 8080

```

## Multi Threading:

Incorporating multithreading into a web server allows for concurrent handling of multiple client requests through individual threads. However, the Global Interpreter Lock (GIL) in Python restricts true parallelism, making it more suitable for I/O-bound tasks where threads can effectively manage multiple connections at once. While this enhances responsiveness when dealing with numerous concurrent requests, the GIL's effect is more significant in CPU-bound tasks, which may limit performance improvements in parallel processing.

```

naveenaa@Naveenas-MacBook-Pro ds_programming % python3 server.py -document_root /Users/naveenaa/ds_programming -port 8080 --verbose True
Web server initialized.
Port: 8080

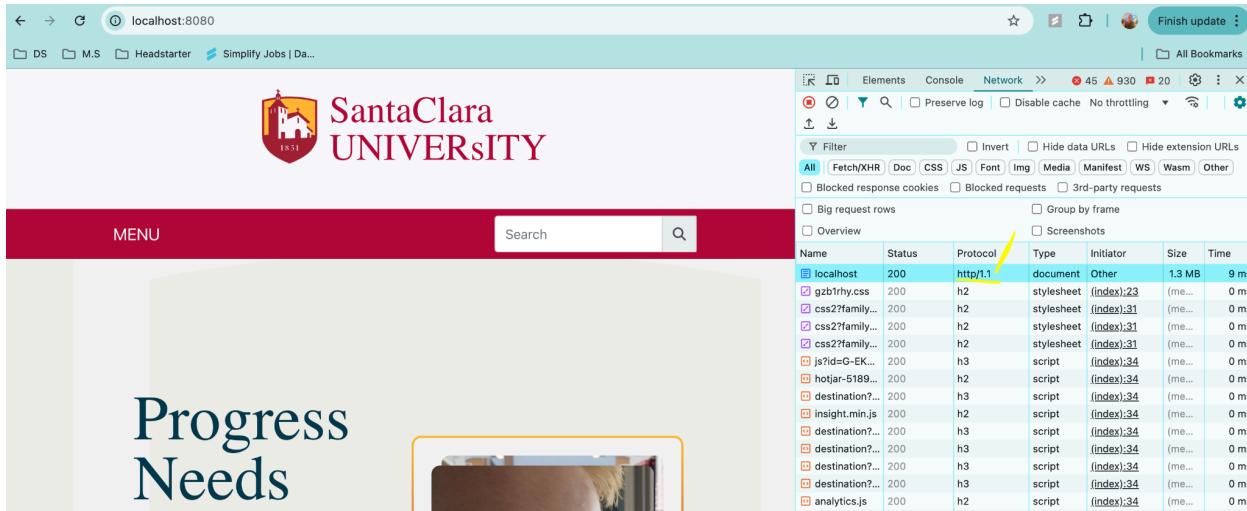
New connection established: 127.0.0.1
Requested resource: /all_files.html
Thread: RequestHandler-1, Serving: /Users/naveenaa/ds_programming/all_files.html
Requested resource: /server_def.txt
Thread: RequestHandler-2, Serving: /Users/naveenaa/ds_programming/server_def.txt
Requested resource: /index1.html
Requested resource: /index.html
Thread: RequestHandler-4, Serving: /Users/naveenaa/ds_programming/index.html
Thread: RequestHandler-3, Serving: /Users/naveenaa/ds_programming/index1.html
Requested resource: /web_server.pdf
Thread: RequestHandler-5, Serving: /Users/naveenaa/ds_programming/web_server.pdf
Requested resource: /socket_photo.jpg
Requested resource: /index.html
Thread: RequestHandler-6, Serving: /Users/naveenaa/ds_programming/socket_photo.jpg
Thread: RequestHandler-7, Serving: /Users/naveenaa/ds_programming/index.html
Requested resource: /server_image.png
Thread: RequestHandler-8, Serving: /Users/naveenaa/ds_programming/server_image.png
Requested resource: /collegelogo.jpeg
Thread: RequestHandler-9, Serving: /Users/naveenaa/ds_programming/collegelogo.jpeg
Requested resource: /error_page.gif
Requested resource: /sample_video.mp4
Thread: RequestHandler-11, Serving: /Users/naveenaa/ds_programming/sample_video.mp4
Thread: RequestHandler-10, Serving: /Users/naveenaa/ds_programming/error_page.gif

```

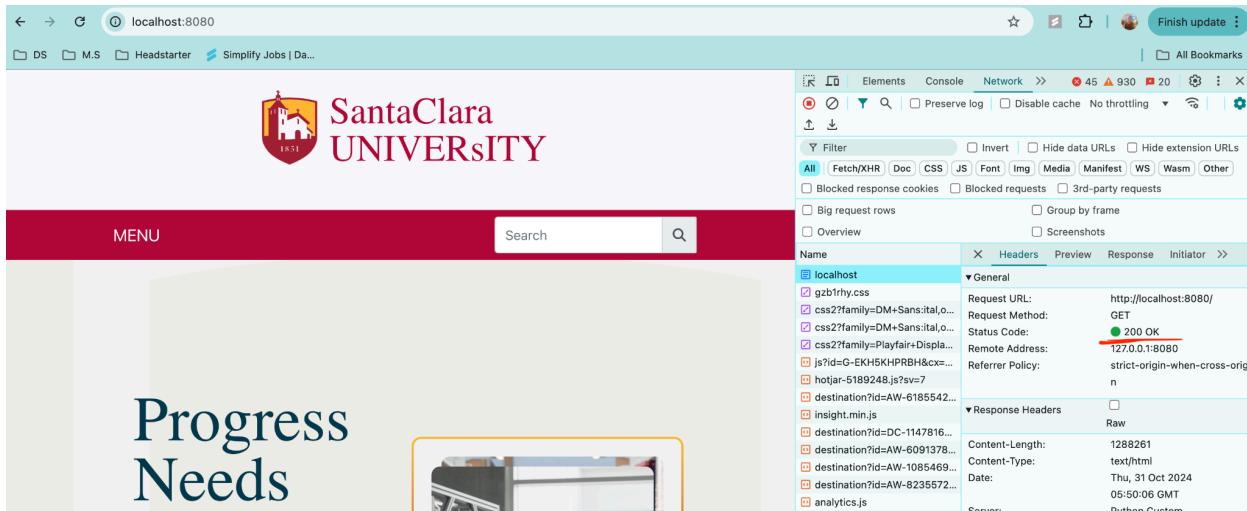
## File formats:

.jpg, .png, .jpeg, .gif, .txt, .html, .pdf, .mp4

1. Launch a browser and input localhost:[port number] (Example: localhost:8080). The server will respond by delivering the content linked to the default endpoint with a status code of 200.



Name	Status	Protocol	Type	Initiator	Size	Time
localhost	200	http/1.1	document	Other	1.3 MB	9 ms
gzbfry.css	200	h2	stylesheet	(index):23	(me...)	0 ms
css2?family=...	200	h2	stylesheet	(index):31	(me...)	0 ms
css2?family=...	200	h2	stylesheet	(index):31	(me...)	0 ms
css2?family=...	200	h2	stylesheet	(index):31	(me...)	0 ms
js?id=G-EK...	200	h3	script	(index):34	(me...)	0 ms
hotjar-5189...	200	h2	script	(index):34	(me...)	0 ms
destination?...	200	h3	script	(index):34	(me...)	0 ms
insight.min.js	200	h2	script	(index):34	(me...)	0 ms
destination?...	200	h3	script	(index):34	(me...)	0 ms
destination?...	200	h3	script	(index):34	(me...)	0 ms
destination?...	200	h3	script	(index):34	(me...)	0 ms
analytics.js	200	h2	script	(index):34	(me...)	0 ms



Name	X	Headers	Preview	Response	Initiator	>>
localhost		▼ General				
gzbfry.css		Request URL: http://localhost:8080/				
css2?family=DM+Sans:ital,o...		Request Method: GET				
css2?family=DM+Sans:ital,o...		Status Code: 200 OK				
css2?family=Playfair+Disp...		Remote Address: 127.0.0.1:8080				
js?id=G-EK5KHPRBH&c=...		Referer Policy: strict-origin-when-cross-origi...				
hotjar-5189248.js?v=7						
destination?id=AW-61685542...						
insight.min.js						
destination?id=DC-1147816...						
destination?id=AW-6091378...						
destination?id=AW-1085469...						
destination?id=AW-8235572...						
analytics.js						
		▼ Response Headers				
		Raw				
		Content-Length: 1288261				
		Content-Type: text/html				
		Date: Thu, 31 Oct 2024				
		05:50:06 GMT				
		Server: Python Custom				

2. Open a browser and type localhost:[port number]/index.html. The server will respond by providing the content associated with the default endpoint (/index.html) from our document root directory, returning a status code of 200.

Name	Status	Protocol	Type	Initiator	Size	Time
index.html	200	http/1.1	document	Other	1.3 MB	21 ms
gz21rhy.css	200	h2	stylesheet	index.html:23	(disk...)	7 ms
css2?family=...	200	h2	stylesheet	index.html:31	(disk...)	6 ms
css2?family=...	200	h2	stylesheet	index.html:31	(disk...)	7 ms
css2?family=...	200	h2	stylesheet	index.html:31	(disk...)	7 ms
895f1e62c1...	200	h2	stylesheet	index.html:33	250 B	126 ms
8fd101ac29.js	200	h2	script	index.html:34	5.0 kB	318 ms
scu.css	200	h2	stylesheet	index.html:88	(disk...)	9 ms
f.css	404	http/1.1	stylesheet	index.html:102	142 B	6 ms
scu.js	200	h2	script	index.html:106	12 B	44 ms
%5Clivewh...	200	h2	stylesheet	index.html:1056	721 B	253 ms
%5Clivewh...	200	h2	script	index.html:1056	925 B	132 ms

Name	Headers	Preview	Response	Initiator
index.html	General Request URL: http://localhost:8080/index.html Request Method: GET Status Code: 200 OK Remote Address: 127.0.0.1:8080 Referrer Policy: strict-origin-when-cross-origin			
f.css				
scu.css				
scu.js				
%5Clivewhale%5Ctheme%5...				
%5Clivewhale%5Cplugins%5...				

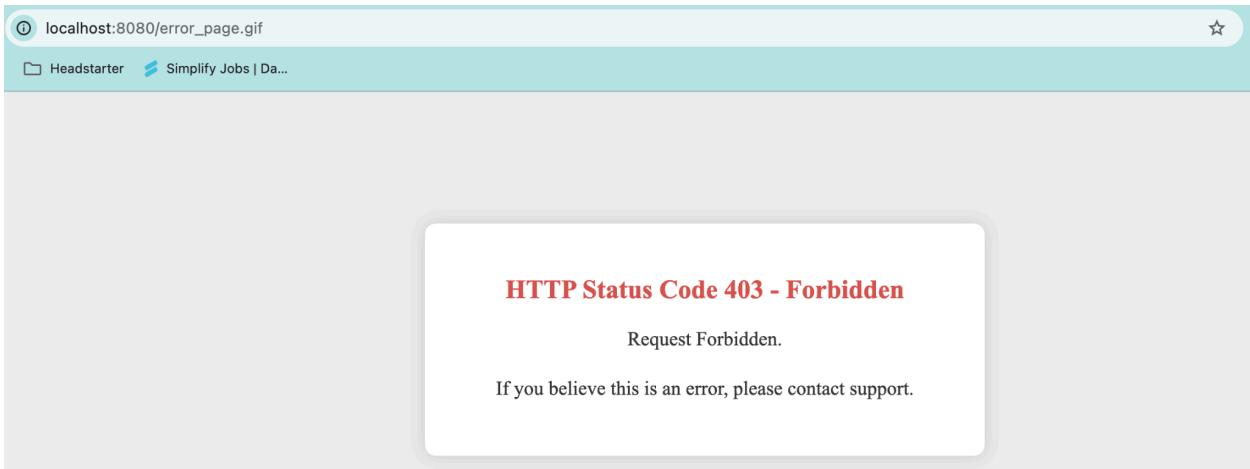
3. After removing read access to the file, the web server will respond with a 403 Forbidden status code, indicating that permissions to access the file are insufficient.

#### Before removing permissions:

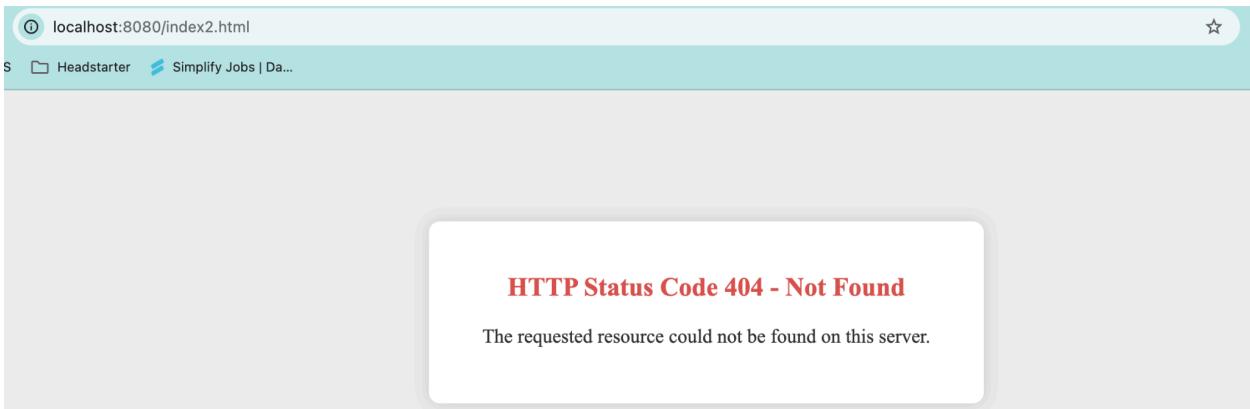
```
naveenaa@Naveenas-MacBook-Pro ds_programming % ls -lart error_page.gif
-rw-r--r--@ 1 naveenaa  staff  759650 Oct 30 15:08 error_page.gif
```

**After removing permissions:**

```
naveenaa@Naveenas-MacBook-Pro ds_programming % ls -lart error_page.gif
--w----- 1 naveenaa  staff  759650 Oct 30 15:08 error_page.gif
```



4. The web server generates a 404 Not Found response because the requested file index2.html (or any file that is not present in the root directory) cannot be found in the specified document root directory.



5. Submitting an invalid POST request to index.html results in a 400 Bad Request response from the web server.

The screenshot shows the Postman application interface. At the top, it says "HTTP http://localhost:8080". Below that is a search bar with "POST" selected and "http://localhost:8080" entered. There are tabs for "Params", "Authorization", "Headers (7)", "Body", "Scripts", and "Settings". On the right, there are "Save" and "Share" buttons. Under "Params", there is a table with one row: "Key" (Value) and "Value" (Description). A "Send" button is at the bottom right.

The screenshot shows the browser's developer tools Network tab. It displays a single request labeled "400 Bad Request" with a status of "3 ms" and "1.27 KB". The "Body" section shows the raw HTML response: "

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="UTF-8">
5     <title>HTTP Error 400 - Bad Request</title>
```

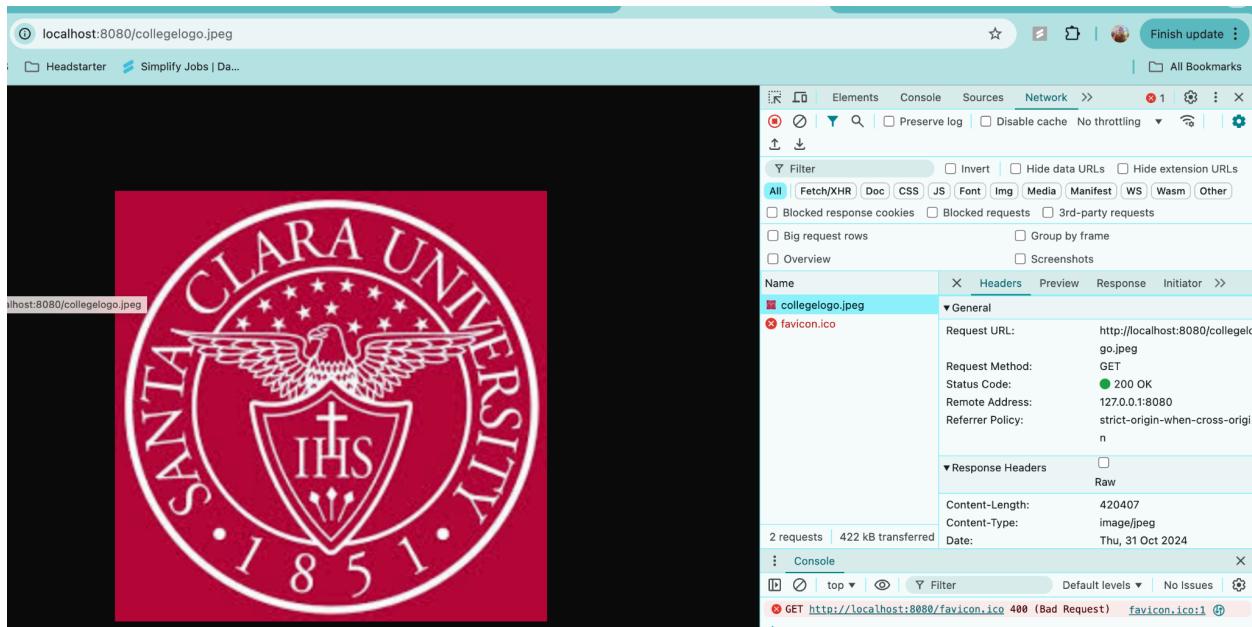
". The "Headers" tab shows four headers: "Content-Type: text/html; charset=UTF-8", "Content-Length: 1121", "Date: Thu, 21 Oct 2021 10:27:20 GMT", and "Server: Apache/2.4.41 (Ubuntu)".

## Client accessing different file contents:

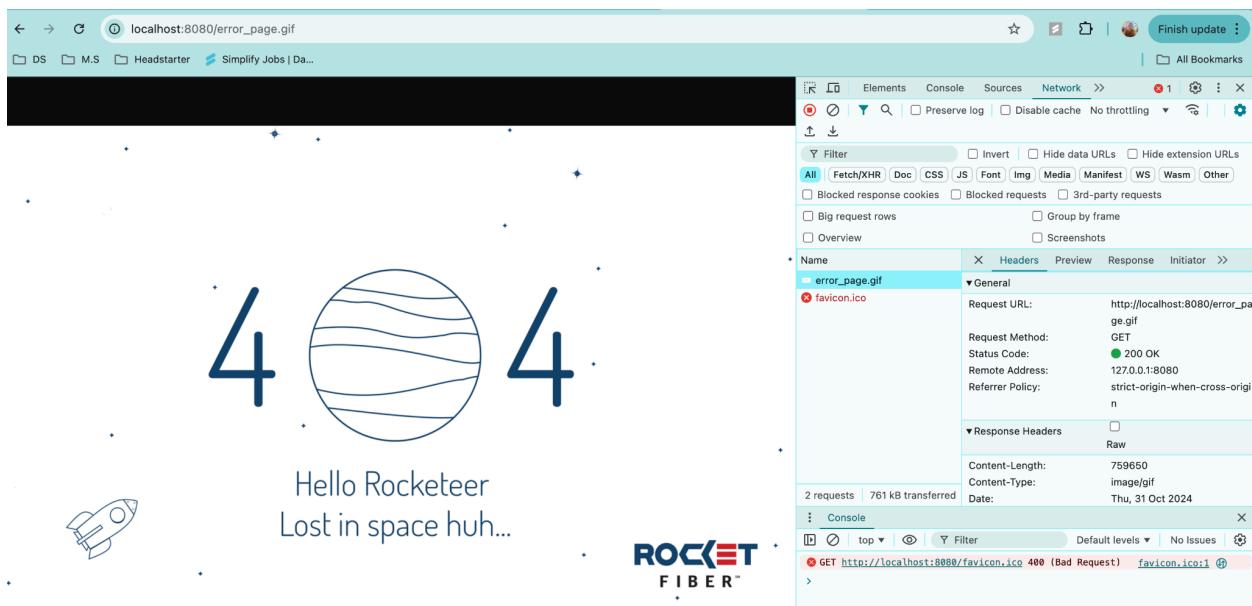
1.The server responds by delivering the content of the existing text file server\_def.txt from the specified document root directory.

The screenshot shows the browser's developer tools Network tab. It displays a request for "server\_def.txt" with a status of "200 OK". The "Headers" tab shows the following headers: "Content-Type: text/plain", "Content-Length: 1121", "Date: Thu, 21 Oct 2021 10:27:20 GMT", and "Server: Apache/2.4.41 (Ubuntu)". The "Response" tab shows the content of the file: "A web server is a software application designed to serve web content to clients over a network, primarily using the HTTP (Hypertext Transfer Protocol). Its main role is to respond to client requests by providing the requested web resources, such as HTML files, images, and other media. When returning a response, a web server cannot be typed. Web servers listen on specific ports, typically port 80 for HTTP and port 443 for HTTPS, and handle multiple requests simultaneously, either by spawning threads or using an event-driven model. They translate URLs requested by clients to file paths on the server's storage, ensuring that only authorized files are served. When the web server receives a request, it checks the file's existence and permissions, sends appropriate HTTP headers (like status codes and content type), and then transmits the file's contents. Modern web servers also support HTTP/1.1 and HTTP/2 for optimized data transfer, with features like persistent connections. Web servers are fundamental to the World Wide Web, enabling browsers to display websites and deliver data across the Internet."

2.The server responds by delivering the content of the existing jpeg file collegelogo.jpeg from the specified document root directory.



3. The server responds by delivering the content of the existing gif file error\_page.gif from the specified document root directory.



4. The server responds by delivering the content of the existing html file index1.html from the specified document root directory.

SANTA CLARA UNIVERSITY  
THE JESUIT UNIVERSITY IN SILICON VALLEY

MENU

Search

Go Anywhere From Here  
Invent the life you want to lead at Santa Clara University.  
[Get Started](#)

Network tab details:

- Request URL: http://localhost:8080/index1.html
- Request Method: GET
- Status Code: 200 OK
- Remote Address: 127.0.0.1:8080
- Referrer Policy: strict-origin-when-cross-origin
- Content-Length: 38935
- Content-Type: text/html
- Date: Thu, 31 Oct 2024

5. The server responds by delivering the content of the existing video file sample\_video.mp4 from the specified document root directory.

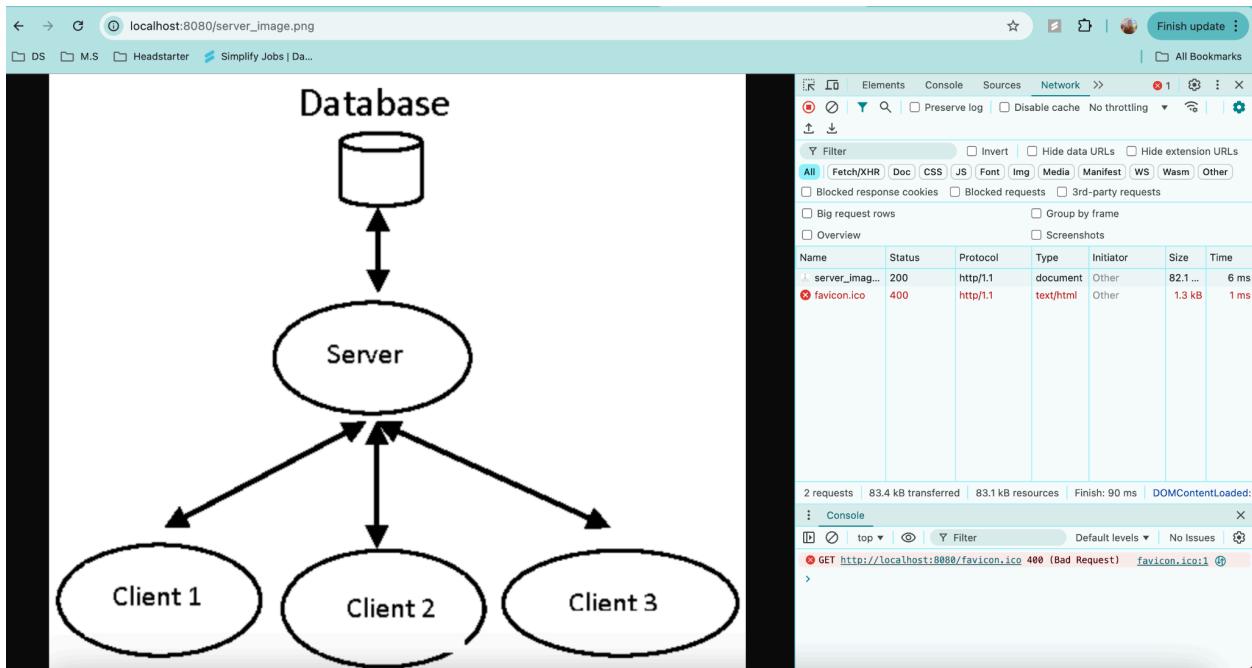
localhost:8080/sample\_video.mp4

Visual Content

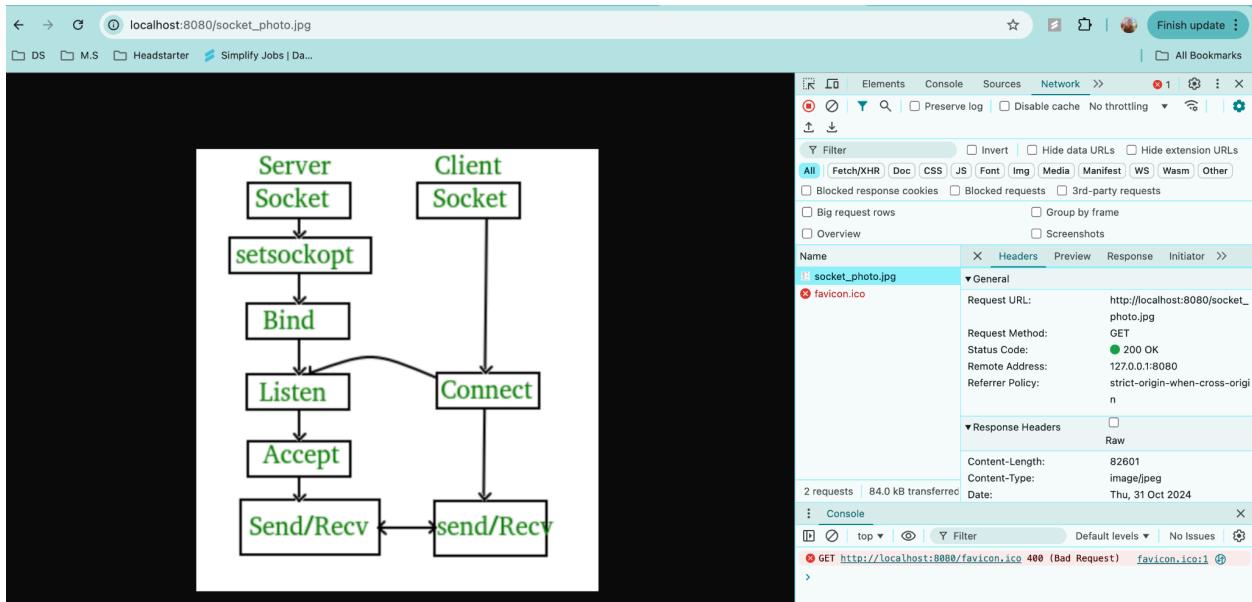
Network tab details:

- Request URL: http://localhost:8080/sample\_video.mp4
- Request Method: GET
- Status Code: 200 OK
- Remote Address: 127.0.0.1:8080
- Referrer Policy: strict-origin-when-cross-origin
- Content-Length: 39975891
- Content-Type: video/mp4
- Date: Thu, 31 Oct 2024

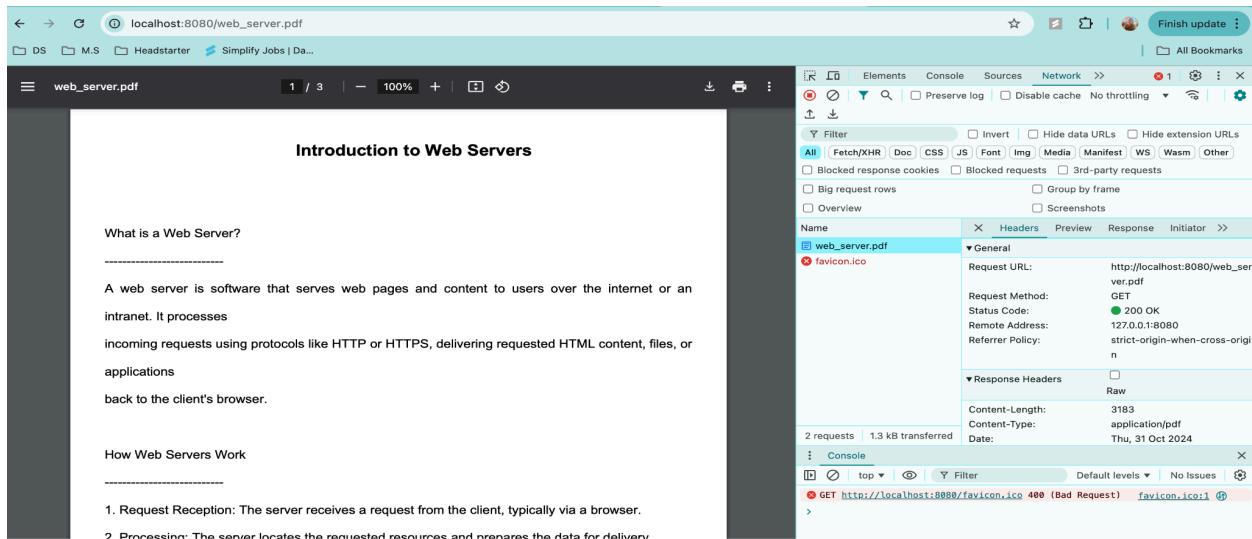
6. The server responds by delivering the content of the existing png file server\_image.png from the specified document root directory.



7. The server responds by delivering the content of the existing jpg file socket\_photo.jpg from the specified document root directory.



8.The server responds by delivering the content of the existing pdf file web\_server.pdf from the specified document root directory.



## Extra Functionality:

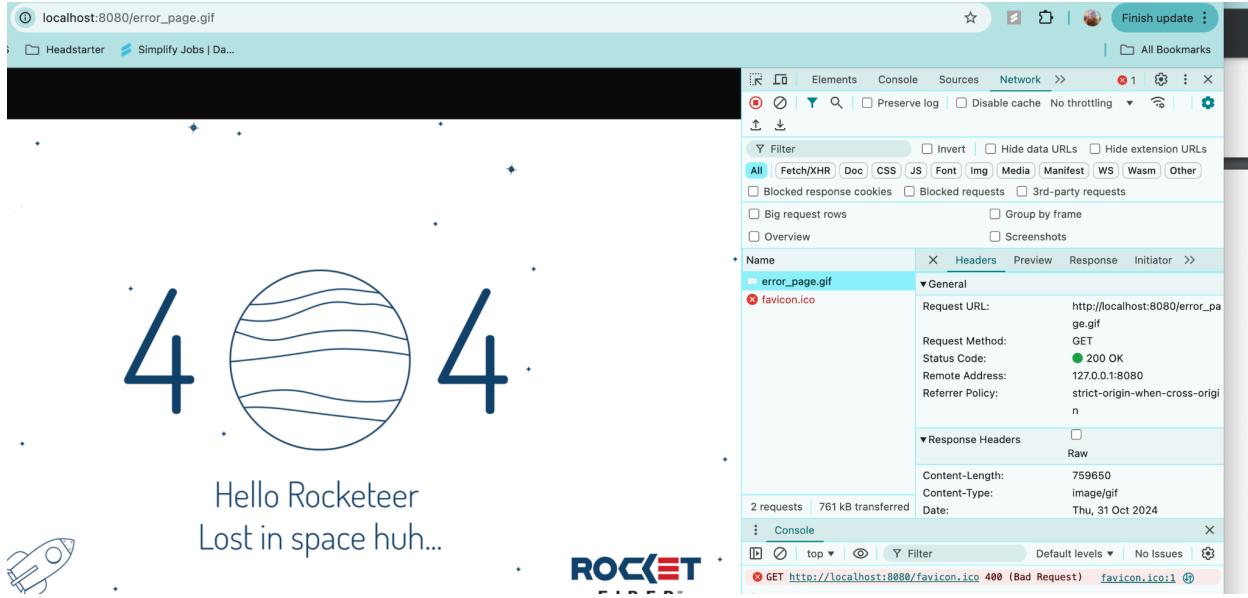
By default, the server.py program utilizes HTTP protocol 1.1 (persistent connection). To enable support for protocol 1.0, use the following optional command-line argument:

- python3 server.py -document\_root /Users/naveenaa/ds\_programming -port 8080 --http\_version 1.0 --verbose True

You can notice that the connection is being closed in the output below:

```
naveenaa@Naveenas-MacBook-Pro ds_programming % python3 server.py -document_root /Users/naveenaa/ds_programming -port 8080 --http_version 1.0
e True
Web server initialized.
Port: 8080

New connection established: 127.0.0.1
Requested resource: /error_page.gif
Thread: RequestHandler-1, Serving: /Users/naveenaa/ds_programming/error_page.gif
Requested resource: /favicon.ico
400 Bad Request: GET
HTTP/1.0 connection terminated
```



## References:

- Reference from class notes/slides.
- <https://docs.python.org/3/library/argparse.html>
- <https://docs.python.org/3/tutorial/errors.html>
- <https://docs.python.org/3/library/mimetypes.html>
- <https://docs.python.org/3/library/socket.html>