

Assignment 2 - Web Server Report

Nick Janota
CSC 138
Assignment 2
Due: 10/23/25

Objective

The goal of this assignment was to **implement a simple TCP web server in Python**. The server receives HTTP GET requests, returns the requested HTML file with an `HTTP 200 OK` response. It also handles missing files with a `404 Not Found` response. **Client.py** was provided to test server functionality.

Overview / Logic

For this assignment, I first implemented the server socket, set it to listen for incoming TCP connections. Next I worked on the handle request function which parses the GET request, opens the file and sends the correct HTTP response. If the HTML file is valid, the page is displayed and `200 OK` is returned. Otherwise, a 404 page is displayed and `404 Not Found` is returned. The server was tested using both a browser (Firefox) and the given client.py.

Documentation

Welcome to CSC 138 Computer Networking

California State University



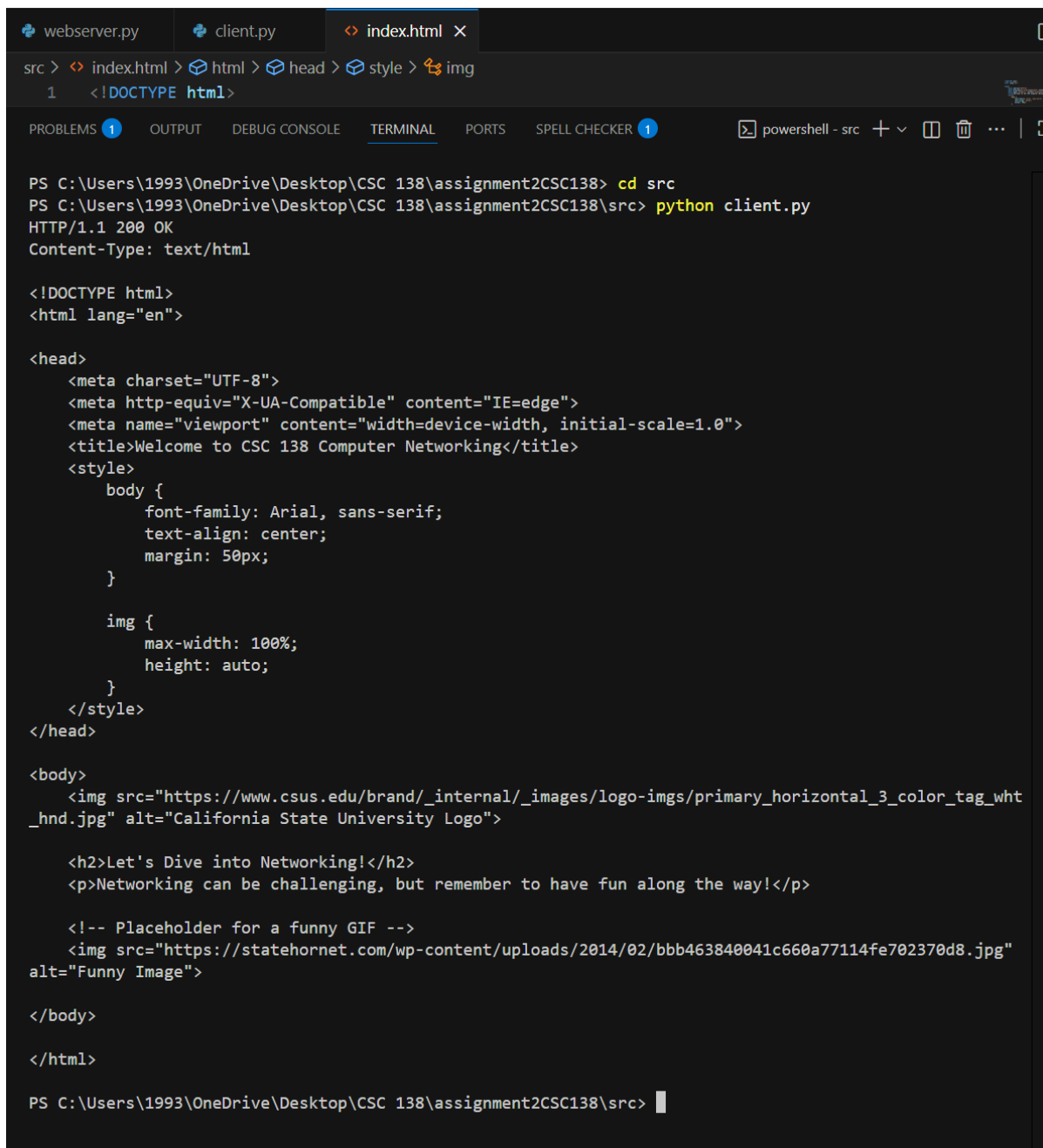
Let's Dive into Networking!

Networking can be challenging, but remember to have fun along the way!



Figure 1: Browser displaying index.html (200 OK)

This screenshot shows the browser successfully loaded index.html from the server indicating that the server correctly parsed the GET request, found the requested file, and returned the content with a 200 OK response.



The screenshot displays a web browser window with the 'index.html' file open. The browser's address bar shows the file path 'src > index.html > html > head > style > img'. The browser's developer tools are open, showing the 'TERMINAL' tab. The terminal output shows the following commands and responses:

```
PS C:\Users\1993\OneDrive\Desktop\CSC 138\assignment2CSC138> cd src
PS C:\Users\1993\OneDrive\Desktop\CSC 138\assignment2CSC138\src> python client.py
HTTP/1.1 200 OK
Content-Type: text/html

<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Welcome to CSC 138 Computer Networking</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      text-align: center;
      margin: 50px;
    }

    img {
      max-width: 100%;
      height: auto;
    }
  </style>
</head>

<body>
  

  <h2>Let's Dive into Networking!</h2>
  <p>Networking can be challenging, but remember to have fun along the way!</p>

  <!-- Placeholder for a funny GIF -->
  

</body>

</html>

PS C:\Users\1993\OneDrive\Desktop\CSC 138\assignment2CSC138\src>
```

Figure 2: Client terminal requesting index.html. 200 OK shows server successfully returned the page that was requested.

This screenshot shows the client.py successfully requesting index.html from the server. The terminal output displays the full HTTP response including the 200 OK header and the pages HTML content. This shows that the server correctly handles requests and returns the expected content if valid.

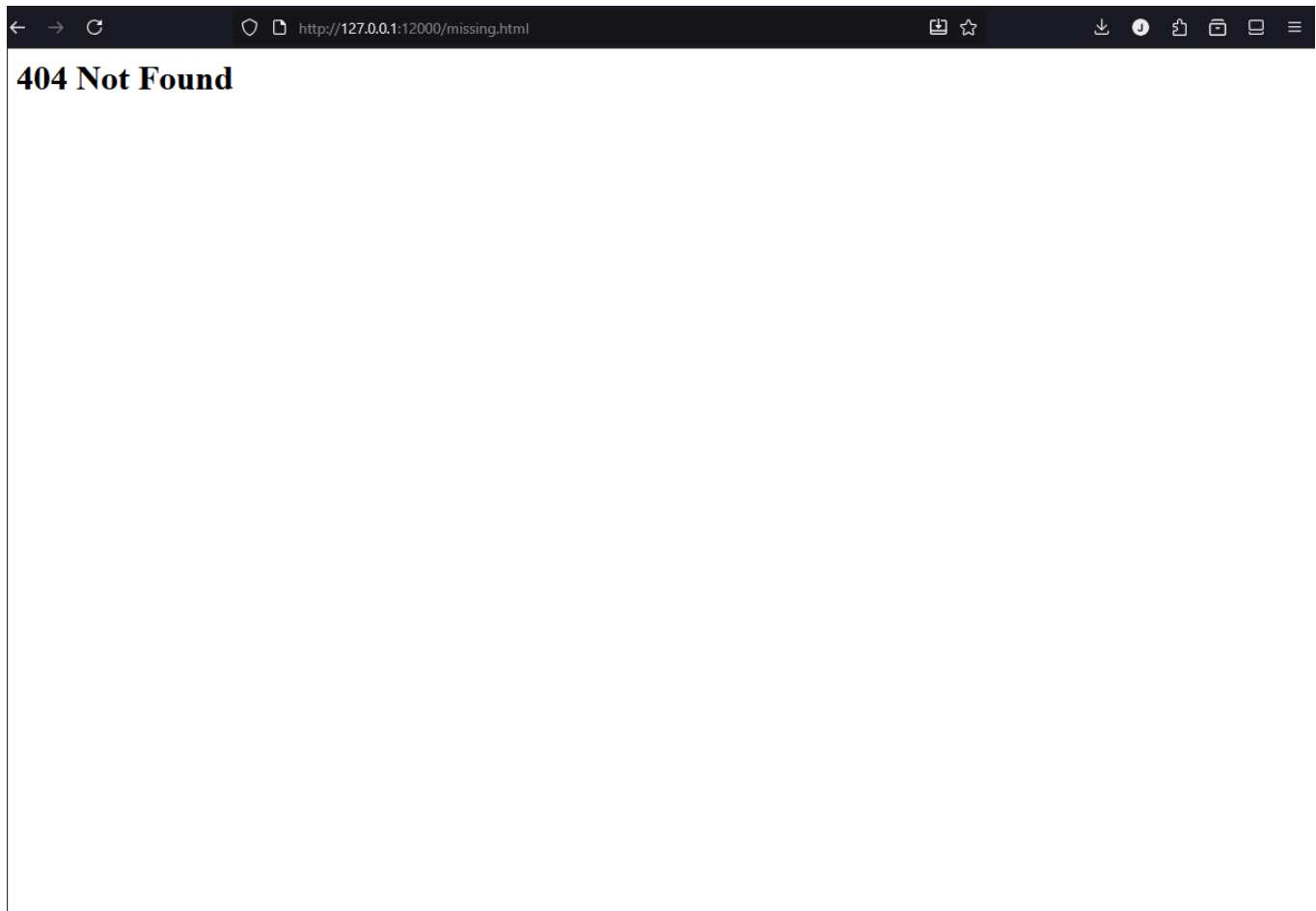
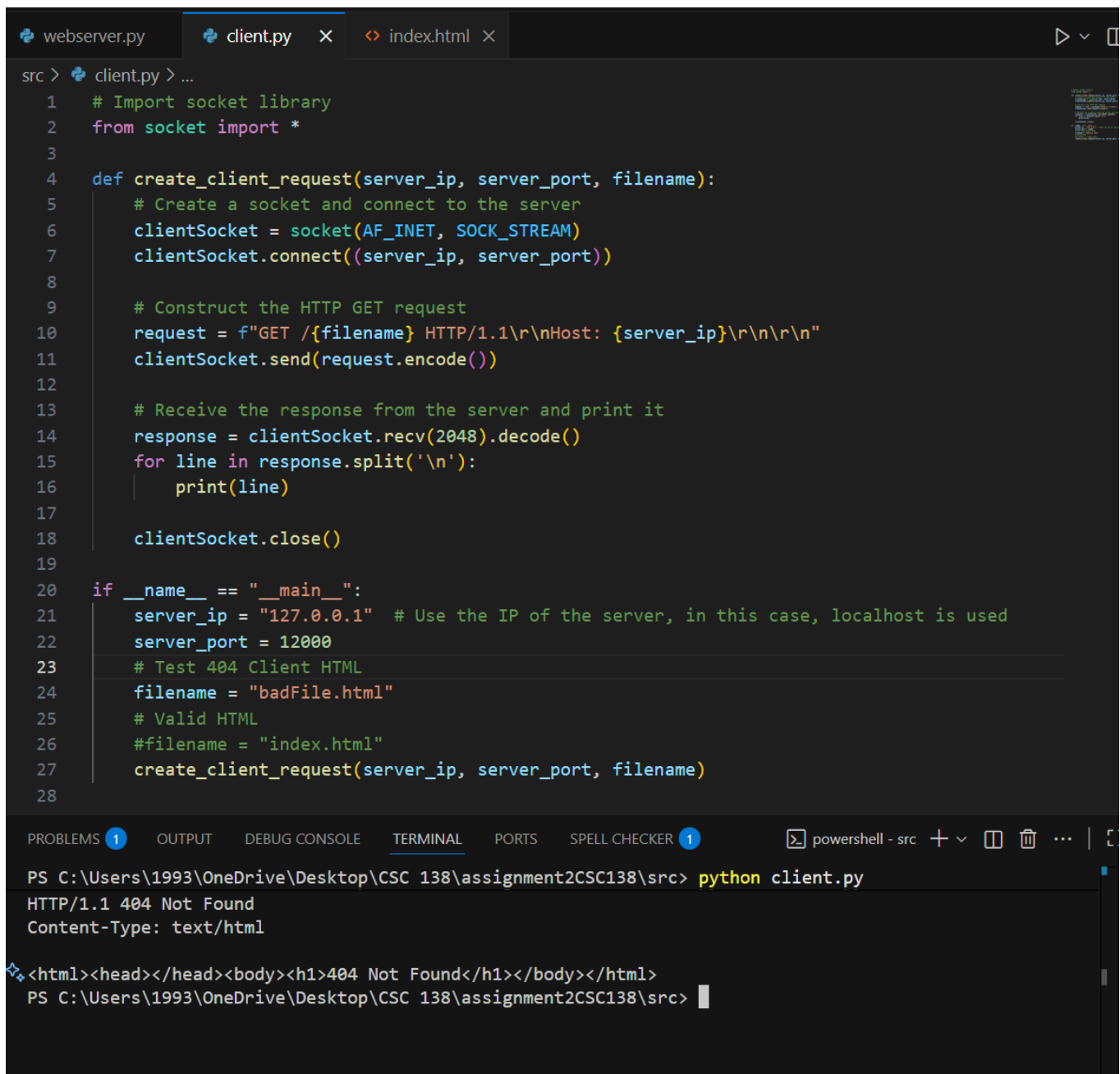


Figure 3: Browser requests a missing file, showing 404 Not Found

This screenshot shows the browser requesting a file that does not exist on the server. The server correctly returned a 404 Not Found page showing that it correctly handles invalid requests and informs the user when the requested file is missing.

The image is a screenshot of a development environment. At the top, there are three tabs: 'webserver.py', 'client.py', and 'index.html'. The 'client.py' tab is active, showing a Python script. The script defines a function 'create_client_request' that takes 'server_ip', 'server_port', and 'filename' as arguments. It creates a socket, connects to the server, sends an HTTP GET request, receives a response, and prints it. Below the function, there is a main block that sets 'server_ip' to '127.0.0.1', 'server_port' to '12000', and 'filename' to 'badFile.html'. It then calls 'create_client_request'. The bottom part of the image shows a terminal window with the command 'python client.py' executed. The terminal output shows the HTTP response: 'HTTP/1.1 404 Not Found' and 'Content-Type: text/html'. Below this, the HTML content is displayed: '<html><head></head><body><h1>404 Not Found</h1></body></html>'.

```
src > client.py > ...
1  # Import socket library
2  from socket import *
3
4  def create_client_request(server_ip, server_port, filename):
5      # Create a socket and connect to the server
6      clientSocket = socket(AF_INET, SOCK_STREAM)
7      clientSocket.connect((server_ip, server_port))
8
9      # Construct the HTTP GET request
10     request = f"GET /{filename} HTTP/1.1\r\nHost: {server_ip}\r\n\r\n"
11     clientSocket.send(request.encode())
12
13     # Receive the response from the server and print it
14     response = clientSocket.recv(2048).decode()
15     for line in response.split('\n'):
16         print(line)
17
18     clientSocket.close()
19
20 if __name__ == "__main__":
21     server_ip = "127.0.0.1" # Use the IP of the server, in this case, localhost is used
22     server_port = 12000
23     # Test 404 Client HTML
24     filename = "badFile.html"
25     # Valid HTML
26     #filename = "index.html"
27     create_client_request(server_ip, server_port, filename)
28
PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS SPELL CHECKER 1 powershell - src + v [ ] [ ] ... | [ ]
PS C:\Users\1993\OneDrive\Desktop\CSC 138\assignment2CSC138\src> python client.py
HTTP/1.1 404 Not Found
Content-Type: text/html

<html><head></head><body><h1>404 Not Found</h1></body></html>
PS C:\Users\1993\OneDrive\Desktop\CSC 138\assignment2CSC138\src>
```

Figure 4: Client terminal requesting a missing file showing 404 Not Found.

This screenshot shows client.py requesting a file that does not exist on the server. The terminal displays the 404 Not Found HTTP header and page content. This shows that the server correctly handles invalid requests from clients and returns the correct error response.

Conclusion

This assignment successfully taught me how to implement a simple TCP web server in Python that handles HTTP GET requests. The server was able to correctly return the requested HTML file with a 200 OK response and handle missing files with a 404 Not Found response. Testing with both a browser and the provided client.py demonstrated how to verify that the server works correctly for both valid and invalid requests.