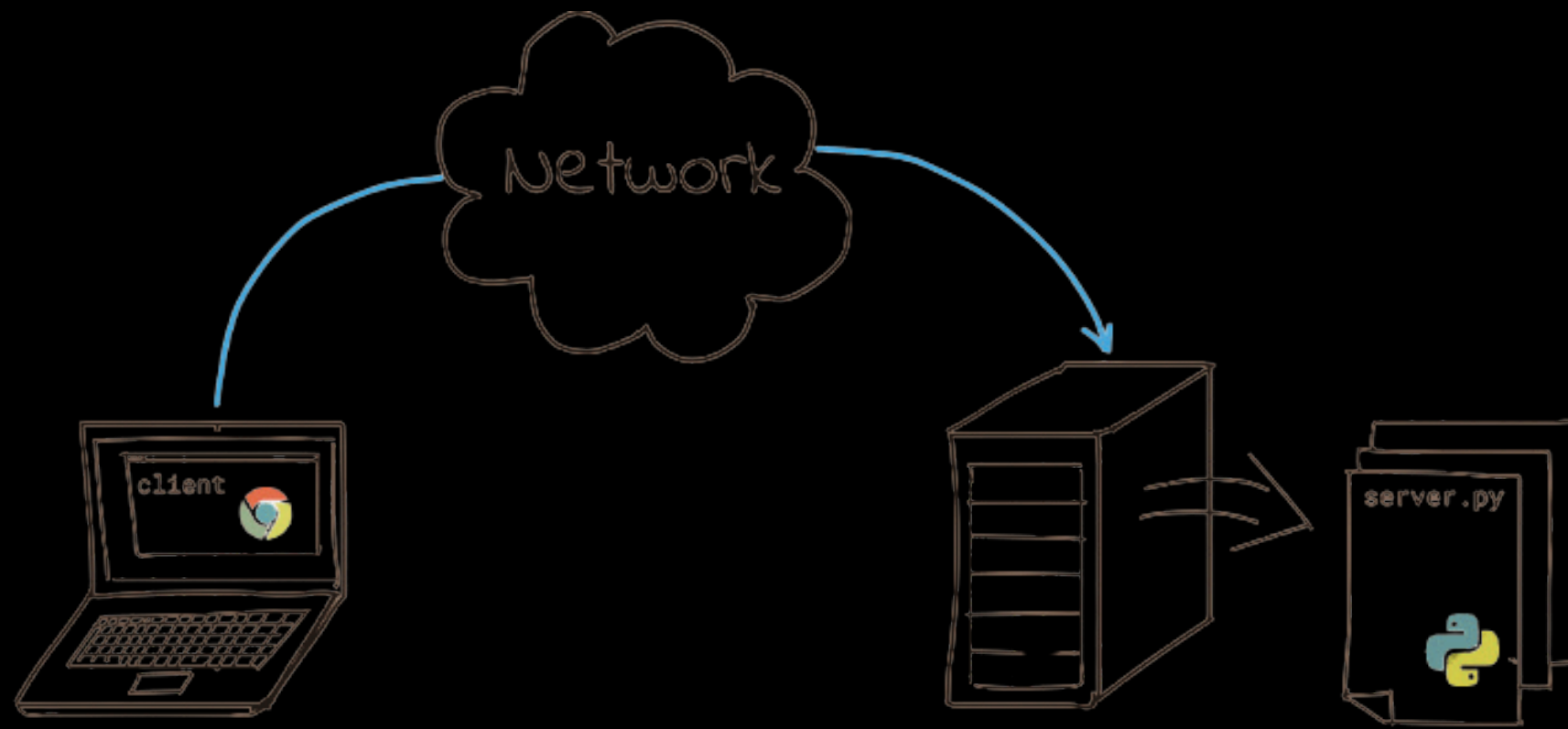

FINAL PROJECT

COMPUTER NETWORK

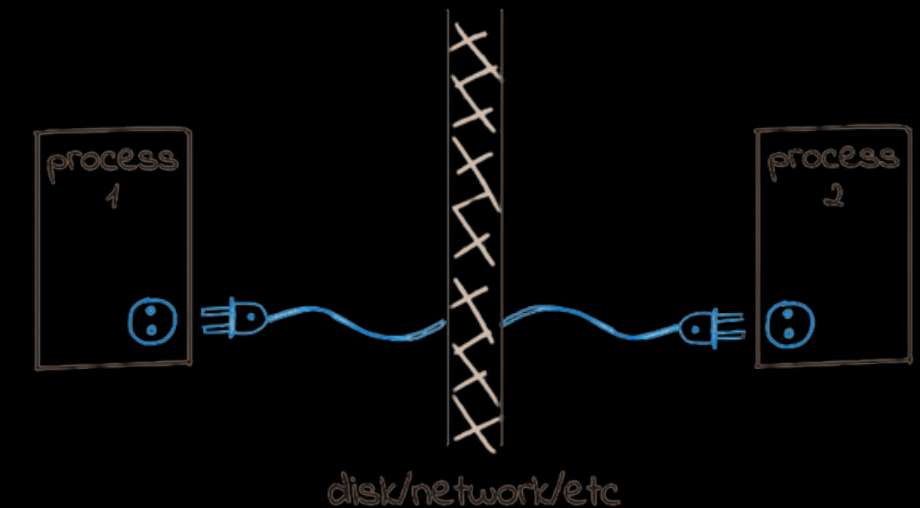
Group Members :

- Romario Viegas Francisco Marcal - 1301225492
 - Ari Ramadhan - 1301224458
 - Laode Muhammad Fathir - 1301224446
-

Web server play a crucial role in the functioning of the World Wide Web. They are software applications that handle client requests over the HTTP protocol, delivering web pages, files, and other resources to users' web browsers. Our servers use TCP (Transmission Control Protocol) to establish reliable connections with clients and HTTP (Hypertext Transfer Protocol) to transmit data over the web.



INTRODUCTION & DETAILS



In this web server we implement two types of server, one is single-threaded and the other one is multi-threaded servers. And we also implement the TCP client for make the request to the servers.

SOURCE CODE

Single-thread server

```
1 import socket
2
3 def handle_client(client_socket, addr):
4     request = client_socket.recv(1024).decode()
5     headers = request.split('\n')
6     if headers and headers[0]:
7         parts = headers[0].split()
8         if len(parts) > 2 and parts[0] == 'GET':
9             filename = parts[1]
10            print(f"Client {addr[0]}:{addr[1]} is requesting file: {filename}")
11            if filename == '/':
12                filename = '/index.html'
13
14            try:
15                file_type = filename.split('.')[-1].lower()
16                if file_type in ["jpg", "gif", "png", "webp", "ico"]:
17                    with open('..' + filename, 'rb') as fin:
18                        content = fin.read()
19                        response = b'HTTP/1.1 200 OK\nContent-Type: image/' + file_type.encode() + b'\nContent-Length: ' + str(len(content)).encode() + b'\n\n' + content
20                elif file_type in ["html", "css", "js"]:
21                    with open('..' + filename, 'r') as fin:
22                        content = fin.read()
23                        response = 'HTTP/1.1 200 OK\nContent-Type: text/' + file_type + '\nContent-Length: ' + str(len(content)) + '\n\n' + content
24                        response = response.encode()
25                else:
26                    with open('..' + filename, 'rb') as fin: # Open in binary mode
27                        content = fin.read()
28                        response = b'HTTP/1.1 200 OK\nContent-Length: ' + str(len(content)).encode() + b'\n\n' + content # Response is bytes
29
30                print(f"Sending 200 OK response to client {addr[0]}:{addr[1]} for file: {filename}")
31
32            except FileNotFoundError:
33                print(f"Client {addr[0]}:{addr[1]} requested a file that was not found: {filename}")
34                response = 'HTTP/1.1 404 NOT FOUND\n\nFile Not Found'
35                response = response.encode() # Encode here
36                print(f"Sending 404 NOT FOUND response to client {addr[0]}:{addr[1]} for file: {filename}")
37
38            else:
39                print(f"Invalid HTTP request line from client {addr[0]}:{addr[1]}: {headers[0]}")
40                response = 'HTTP/1.1 400 BAD REQUEST\n\nInvalid HTTP request line'
41                response = response.encode() # Encode here
42                print(f"Sending 400 BAD REQUEST response to client {addr[0]}:{addr[1]} due to invalid request line: {headers[0]}")
43
44            else:
45                print(f"Empty request received from client {addr[0]}:{addr[1]}")
46                response = 'HTTP/1.1 400 BAD REQUEST\n\nEmpty request received'
47                response = response.encode() # Encode here
48                print(f"Sending 400 BAD REQUEST response to client {addr[0]}:{addr[1]} due to empty request")
49
50            #setelah kirim response, socket ditutup karena ini ciri khas dari HTTP/1.0,
51            #dimana setelah kirim response, socket ditutup, dan jika ada lagi request maka a new TCP connection akan di setup lagi
52            client_socket.send(response)
53            client_socket.close()
54
55
56 def server():
57     ip_address = '127.0.0.2'
58     port = 1234
59     """Creates a server that handles one HTTP request at a time."""
60     server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
61     server_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
62     server_socket.bind((ip_address, port))
63     server_socket.listen(5)
64
65     print(f"Server ready...\nServer is running on port http://{ip_address}:{port}")
66
67     while True:
68         client_socket, addr = server_socket.accept()
69         print(f"Accepted connection from: {addr[0]}:{addr[1]}")
70         handle_client(client_socket, addr) # pass addr to handle_client
71
72
73 if __name__ == "__main__":
74     server()
75
```

Single thread server

6

1

```
1 import socket
```

2

```
1 request = client_socket.recv(1024).decode()
2 headers = request.split('\n')
```

3

```
1 if headers and headers[0]:
2     parts = headers[0].split()
```

4

```
1 if len(parts) > 2 and parts[0] == 'GET':
2     filename = parts[1]
3     print(f"Client {addr[0]}:{addr[1]} is requesting file: {filename}")
```

5

```
1 if filename == '/':
2     filename = '/index.html'
3
```

10

```
1 if __name__ == "__main__":
2     server()
```

8

```
1 client_socket.send(response)
2 client_socket.close()
```

9

```
1
2 def server():
3     ip_address = '127.0.0.2'
4     port = 1234
5     """Creates a server that handles one HTTP request at a time."""
6     server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
7     server_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
8     server_socket.bind((ip_address, port))
9     server_socket.listen(5)
10
11     print(f"Server ready...\nServer is running on port http://{ip_address}:{port}")
12
13     while True:
14         client_socket, addr = server_socket.accept()
15         print(f"Accepted connection from: {addr[0]}:{addr[1]}")
16         handle_client(client_socket, addr) # pass addr to handle_client
17
```

7

```
1 try:
2     file_type = filename.split('.')[-1]
3     if file_type in ["jpg", "gif", "png", "webp", "ico"]:
4         with open('.') + filename, 'rb') as fin:
5             content = fin.read()
6             response = b'HTTP/1.1 200 OK\nContent-Type: image/' + file_type.encode() + b'\nContent-Length: ' + str(len(content)).encode() + b'\n\n' + content
7     elif file_type in ["html", "css", "js"]:
8         with open('.') + filename, 'r') as fin:
9             content = fin.read()
10            response = 'HTTP/1.1 200 OK\nContent-Type: text/' + file_type + '\nContent-Length: ' + str(len(content)) + '\n\n' + content
11            response = response.encode()
12    else:
13        with open('.') + filename, 'rb') as fin: # Open in binary mode
14            content = fin.read()
15            response = b'HTTP/1.1 200 OK\nContent-Length: ' + str(len(content)).encode() + b'\n\n' + content # Response is bytes
16
17    print(f"Sending 200 OK response to client {addr[0]}:{addr[1]} for file: {filename}")
```

```
1 except FileNotFoundError:
2     print(f"Client {addr[0]}:{addr[1]} requested a file that was not found: {filename}")
3     response = 'HTTP/1.1 404 NOT FOUND\n\nFile Not Found'
4     response = response.encode() # Encode here
5     print(f"Sending 404 NOT FOUND response to client {addr[0]}:{addr[1]} for file: {filename}")
6
7 else:
8     print(f"Invalid HTTP request line from client {addr[0]}:{addr[1]}: {headers[0]}")
9     response = 'HTTP/1.1 400 BAD REQUEST\n\nInvalid HTTP request line'
10    response = response.encode() # Encode here
11    print(f"Sending 400 BAD REQUEST response to client {addr[0]}:{addr[1]} due to invalid request line: {headers[0]}")
12
13 else:
14     print(f"Empty request received from client {addr[0]}:{addr[1]}")
15     response = 'HTTP/1.1 400 BAD REQUEST\n\nEmpty request received'
16     response = response.encode() # Encode here
17     print(f"Sending 400 BAD REQUEST response to client {addr[0]}:{addr[1]} due to empty request")
```

SOURCE CODE

Multi-thread server

```
1 import socket
2 import threading
3 def handle_client(client_socket, addr):
4     request = client_socket.recv(1024).decode()
5     headers = request.split('\n')
6     if headers and headers[0]:
7         parts = headers[0].split()
8         if len(parts) > 2 and parts[0] == 'GET':
9             filename = parts[1]
10            print(f"Client {addr[0]}:{addr[1]} is requesting file: {filename}")
11            if filename == '/':
12                filename = '/index.html'
13
14            try:
15                file_type = filename.split('.')[-1]
16                if file_type in [".jpg", "gif", "png", "webp", "ico"]:
17                    with open('.') + filename, 'rb') as fin:
18                        content = fin.read()
19                        response = b'HTTP/1.1 200 OK\nContent-Type: image/' + file_type.encode() + b'\nContent-Length: ' + str(len(content)).encode() + b'\n\n' + content
20                elif file_type in [".html", ".css", ".js"]:
21                    with open('.') + filename, 'r') as fin:
22                        content = fin.read()
23                        response = 'HTTP/1.1 200 OK\nContent-Type: text/' + file_type + '\nContent-Length: ' + str(len(content)) + '\n\n' + content
24                        response = response.encode()
25                else:
26                    with open('.') + filename, 'rb') as fin: # Open in binary mode
27                        content = fin.read()
28                        response = b'HTTP/1.1 200 OK\nContent-Length: ' + str(len(content)).encode() + b'\n\n' + content # Response is bytes
29
30                print(f"Sending 200 OK response to client {addr[0]}:{addr[1]} for file: {filename}")
31
32            except FileNotFoundError:
33                print(f"Client {addr[0]}:{addr[1]} requested a file that was not found: {filename}")
34                response = 'HTTP/1.1 404 NOT FOUND\n\nFile Not Found'
35                response = response.encode() # Encode here
36                print(f"Sending 404 NOT FOUND response to client {addr[0]}:{addr[1]} for file: {filename}")
37
38            else:
39                print(f"Invalid HTTP request line from client {addr[0]}:{addr[1]}: {headers[0]}")
40                response = 'HTTP/1.1 400 BAD REQUEST\n\nInvalid HTTP request line'
41                response = response.encode() # Encode here
42                print(f"Sending 400 BAD REQUEST response to client {addr[0]}:{addr[1]} due to invalid request line: {headers[0]}")
43
44            else:
45                print(f"Empty request received from client {addr[0]}:{addr[1]}")
46                response = 'HTTP/1.1 400 BAD REQUEST\n\nEmpty request received'
47                response = response.encode() # Encode here
48                print(f"Sending 400 BAD REQUEST response to client {addr[0]}:{addr[1]} due to empty request")
49
50            client_socket.send(response)
51            client_socket.close()
52
53 def server():
54     ip_address = '127.0.0.1' # Your server's IP address
55     port = 1234 # Port number to listen on
56
57     server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
58     server_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
59     server_socket.bind((ip_address, port))
60     server_socket.listen(5)
61
62     print(f"Server ready...\nServer is running on port http://{ip_address}:{port}")
63
64     try:
65         while True:
66             client_socket, addr = server_socket.accept()
67             print(f"Accepted connection from: {addr[0]}:{addr[1]}")
68             client_thread = threading.Thread(target=handle_client, args=(client_socket, addr))
69             client_thread.start()
70
71     except KeyboardInterrupt:
72         print("Shutting down server...")
73         server_socket.close()
74
75 if __name__ == "__main__":
76     server()
```


Multi thread server

6

1

```
1 import socket
```

2

```
1 request = client_socket.recv(1024).decode()
2 headers = request.split('\n')
```

3

```
1 if headers and headers[0]:
2     parts = headers[0].split()
```

10

```
1 if __name__ == "__main__":
2     server()
```

8

```
1 client_socket.send(response)
2 client_socket.close()
```

```
1 try:
2     file_type = filename.split('.')[-1]
3     if file_type in ["jpg", "gif", "png", "webp", "ico"]:
4         with open('.') + filename, 'rb') as fin:
5             content = fin.read()
6             response = b'HTTP/1.1 200 OK\nContent-Type: image/' + file_type.encode() + b'\nContent-Length: ' + str(len(content)).encode() + b'\n\n' + content
7     elif file_type in ["html", "css", "js"]:
8         with open('.') + filename, 'r') as fin:
9             content = fin.read()
10            response = 'HTTP/1.1 200 OK\nContent-Type: text/' + file_type + '\nContent-Length: ' + str(len(content)) + '\n\n' + content
11            response = response.encode()
12    else:
13        with open('.') + filename, 'rb') as fin: # Open in binary mode
14            content = fin.read()
15            response = b'HTTP/1.1 200 OK\nContent-Length: ' + str(len(content)).encode() + b'\n\n' + content # Response is bytes
16
17    print(f"Sending 200 OK response to client {addr[0]}:{addr[1]} for file: {filename}")
```

```
1 except FileNotFoundError:
2     print(f"Client {addr[0]}:{addr[1]} requested a file that was not found: {filename}")
3     response = 'HTTP/1.1 404 NOT FOUND\n\nFile Not Found'
4     response = response.encode() # Encode here
5     print(f"Sending 404 NOT FOUND response to client {addr[0]}:{addr[1]} for file: {filename}")
6
7     else:
8         print(f"Invalid HTTP request line from client {addr[0]}:{addr[1]}: {headers[0]}")
9         response = 'HTTP/1.1 400 BAD REQUEST\n\nInvalid HTTP request line'
10        response = response.encode() # Encode here
11        print(f"Sending 400 BAD REQUEST response to client {addr[0]}:{addr[1]} due to invalid request line: {headers[0]}")
12
13    else:
14        print(f"Empty request received from client {addr[0]}:{addr[1]}")
15        response = 'HTTP/1.1 400 BAD REQUEST\n\nEmpty request received'
16        response = response.encode() # Encode here
17        print(f"Sending 400 BAD REQUEST response to client {addr[0]}:{addr[1]} due to empty request")
```

7

9

```
1 def server():
2     ip_address = '127.0.0.1' # Your server's IP address
3     port = 1234 # Port number to listen on
4
5     server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
6     server_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
7     server_socket.bind((ip_address, port))
8     server_socket.listen(5)
9
10    print(f"Server ready...\nServer is running on port http://{ip_address}:{port}")
11
12    try:
13        while True:
14            client_socket, addr = server_socket.accept()
15            print(f"Accepted connection from: {addr[0]}:{addr[1]}")
16            client_thread = threading.Thread(target=handle_client, args=(client_socket, addr))
17            client_thread.start()
18    except KeyboardInterrupt:
19        print("Shutting down server...")
20        server_socket.close()
21
```

4

```
1 if len(parts) > 2 and parts[0] == 'GET':
2     filename = parts[1]
3     print(f"Client {addr[0]}:{addr[1]} is requesting file: {filename}")
```

5

```
1 if filename == '/':
2     filename = '/index.html'
3
```

SOURCE CODE

Client

```
1 #Using terminal for make a request and response between the server and the client
2 import socket
3 import sys
4
5 def http_client(server_host, server_port, filename):
6     """Creates a client that sends an HTTP GET request to the server."""
7     client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
8     client_socket.connect((server_host, server_port))
9
10    request = f"GET /{filename} HTTP/1.1\r\nHost: {server_host}\r\n\r\n"
11    client_socket.send(request.encode())
12
13    response = ''
14    while True:
15        data = client_socket.recv(1024)
16        if not data:
17            break
18        response += data.decode()
19
20    print(response)
21    client_socket.close()
22
23 if __name__ == "__main__":
24     if len(sys.argv) != 4:
25         print("Usage: python client.py server_host server_port filename")
26         sys.exit(1)
27
28     server_host = sys.argv[1]
29     server_port = int(sys.argv[2])
30     filename = sys.argv[3]
31
32     http_client(server_host, server_port, filename)
```

6



```
1 if __name__ == "__main__":
2     if len(sys.argv) != 4:
3         print("Usage: python client.py server_host server_port filename")
4         sys.exit(1)
```

8



```
1 http_client(server_host, server_port, filename)
```

7



```
1 server_host = sys.argv[1]
2 server_port = int(sys.argv[2])
3 filename = sys.argv[3]
```

5



```
1 print(response)
2 client_socket.close()
```

1



```
1 import socket
2 import sys
```

2



```
1 def http_client(server_host, server_port, filename):
2     """Creates a client that sends an HTTP GET request to the server."""
3     client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
4     client_socket.connect((server_host, server_port))
```

3

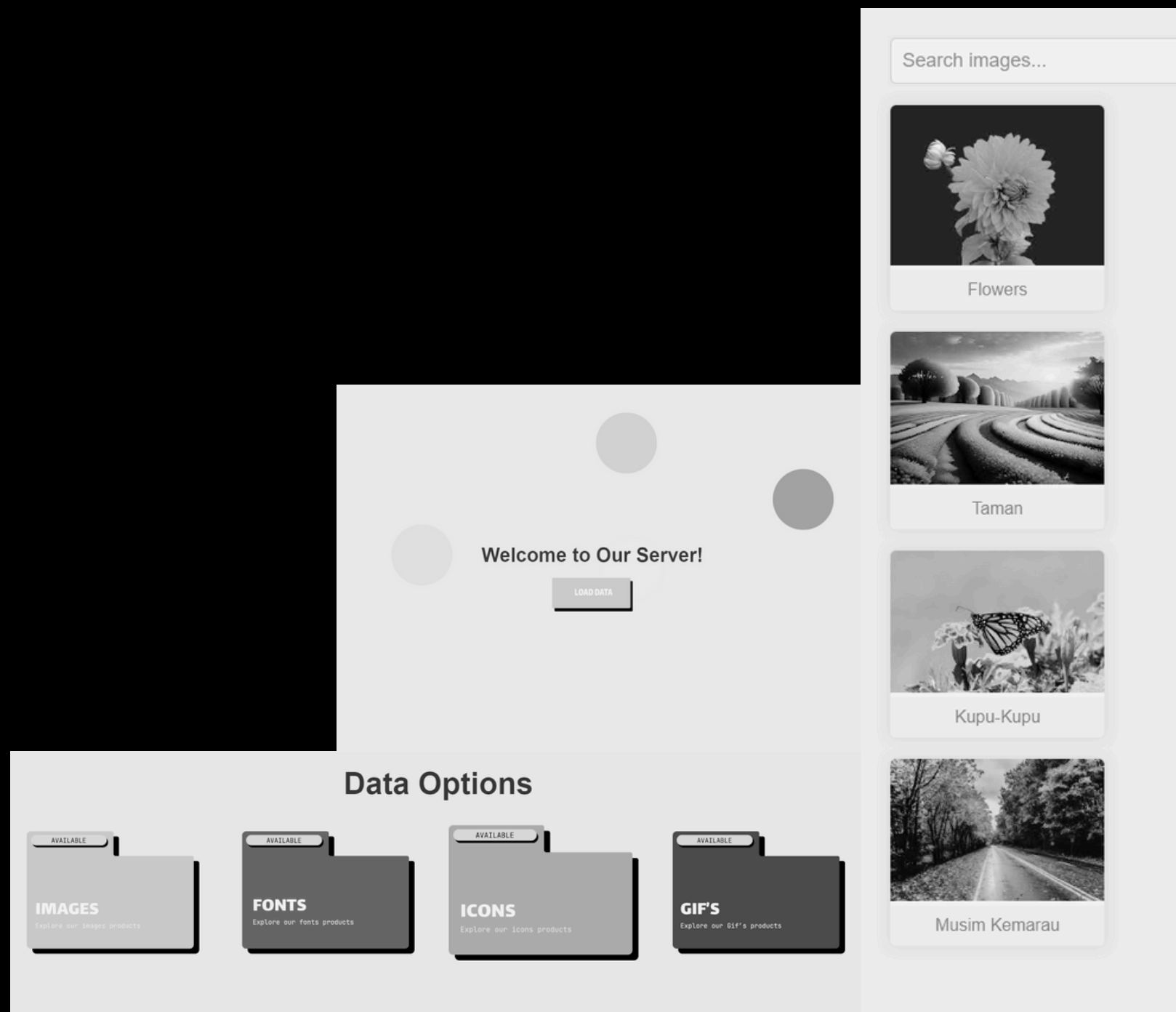


```
1 request = f"GET /{filename} HTTP/1.1\r\nHost: {server_host}\r\n\r\n"
2 client_socket.send(request.encode())
```

4



```
1 while True:
2     data = client_socket.recv(1024)
3     if not data:
4         break
5     response += data.decode()
```

THE RESULTS

The Results of our web server, consist of a main page, that will redirected us to the options page, then we could choose type of data that we want, like images, fonts, icons, and Gif's of specified file.

TESTING

PROBLEMS OUTPUT DEBUG CONSOLE PORTS TERMINAL

```
PS C:\Users\Romario\Documents\TELKOM UNIVERSITY\TUBES JARKOM>  
Server ready...  
Server is running on port http://127.0.0.1:1234
```

```
Administrator: C:\Windows\Sy... x + v  
Microsoft Windows [Version 10.0.22621.3593]  
(c) Microsoft Corporation. All rights reserved.  
  
C:\Users\Romario\Documents\TELKOM UNIVERSITY\TUBES JARKOM>python client.py 127.0.0.2 1234 index.html  
HTTP/1.1 200 OK  
Content-Type: text/html  
Content-Length: 4456  
  
<!DOCTYPE html>  
<html>  
<head>  
  <style>  
    body {  
      background-color: #f0f0f0;  
      font-family: Arial, sans-serif;  
      margin: 0;  
      padding: 0;  
      display: flex;  
      flex-direction: column;  
      justify-content: center;  
      align-items: center;  
      height: 100vh;  
      overflow: hidden; /* Ensure animation doesn't create scrollbars */  
      position: relative; /* Required for absolute positioning of glass */  
    }  
    h1 {  
      color: #333;  
      position: relative;  
      z-index: 1; /* Ensure text is above glass balls */  
    }  
  }  
</head>  
<body>  
  <div>  
    <div>  
      <div>  
        <div>  
          <div>  
            <div>  
              <div>  
                <div>  
                  <div>  
                    <div>  
                      <div>  
                        <div>  
                          <div>  
                        </div>  
                      </div>  
                    </div>  
                  </div>  
                </div>  
              </div>  
            </div>  
          </div>  
        </div>  
      </div>  
    </div>  
  </div>  
</body>  
</html>
```

TESTING

PROBLEMS OUTPUT DEBUG CONSOLE PORTS TERMINAL

```
PS C:\Users\Romario\Documents\TELKOM UNIVERSITY\TUBES JARKOM>  
Server ready...  
Server is running on port http://127.0.0.1:1234
```

```
Administrator: C:\Windows\S... x + v  
Microsoft Windows [Version 10.0.22621.3593]  
(c) Microsoft Corporation. All rights reserved.  
  
C:\Users\Romario\Documents\TELKOM UNIVERSITY\TUBES JARKOM>python client.py 127.0.0.1 1234 index.html  
HTTP/1.1 200 OK  
Content-Type: text/html  
Content-Length: 4456  
  
<!DOCTYPE html>  
<html>  
<head>  
  <style>  
    body {  
      background-color: #f0f0f0;  
      font-family: Arial, sans-serif;  
      margin: 0;  
      padding: 0;  
      display: flex;  
      flex-direction: column;  
      justify-content: center;  
      align-items: center;  
      height: 100vh;  
      overflow: hidden; /* Ensure animation doesn't create scrollbars */  
      position: relative; /* Required for absolute positioning of glass */  
    }  
    h1 {  
      color: #333;  
      position: relative;  
      z-index: 1; /* Ensure text is above glass balls */  
    }  
  }  
</head>  
<body>  
  <h1>HELLO</h1>  
</body>  
</html>
```

How About send multiple request

Apache bench testing

```
Administrator: Command Pro
Microsoft Windows [Version 10.0.22621.3593]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Romario>ab -n 100 -c 10 http://127.0.0.2:1234/
This is ApacheBench, Version 2.3 <$Revision: 1913912 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking 127.0.0.2 (be patient)...
Test aborted after 10 failures

apr_socket_connect(): No connection could be made because the target machine actively refused it. (730061)
Total of 1 requests completed
```

How About send multiple request

```
C:\Users\Romario>ab -n 100 -c 10 http://127.0.0.1:1234/
This is ApacheBench, Version 2.3 <$Revision: 1913912 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking 127.0.0.1 (be patient).....done


Server Software:
Server Hostname:      127.0.0.1
Server Port:          1234

Document Path:        /
Document Length:      4456 bytes

Concurrency Level:    10
Time taken for tests:  5.641 seconds
Complete requests:    100
Failed requests:       0
Total transferred:    451800 bytes
HTML transferred:     445600 bytes
Requests per second:  17.73 [#/sec] (mean)
Time per request:     564.072 [ms] (mean)
Time per request:     56.407 [ms] (mean, across all concurrent requests)
Transfer rate:        78.22 [Kbytes/sec] received
```

Apache bench testing

ANY QUESTION?

JARKOM JAYA JAYA JAYA !

**FOR THE SOURCE
CODE**

VISIT
GITHUB : MR-ROMA

ASPRAK GUIDER

CODE : STO

hello@reallygreatsite.com

THANK YOU

JARKOM JAYA JAYA JAYA !
