FINAL PROJECT

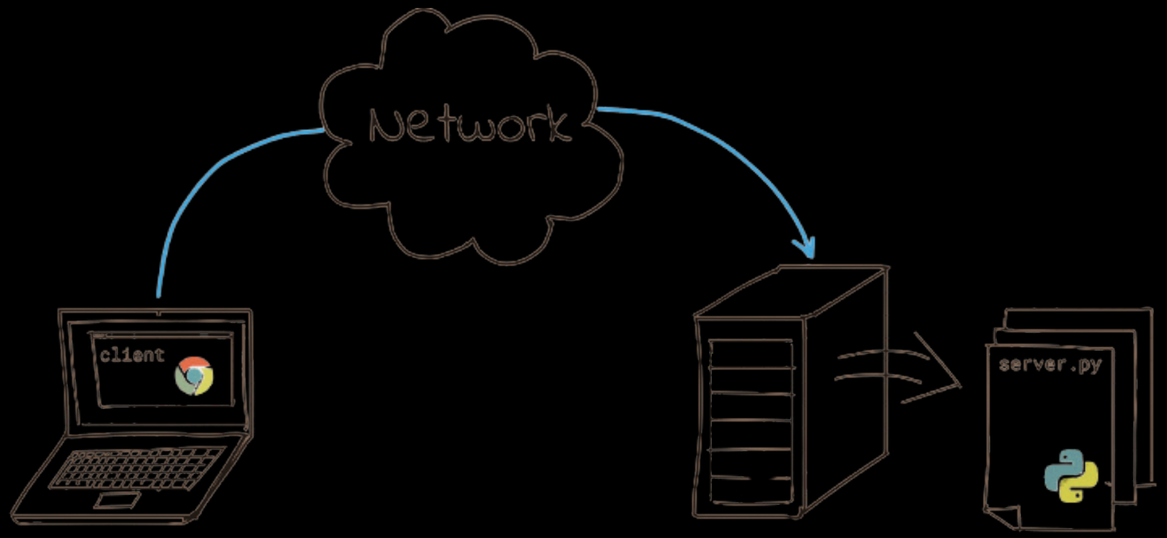
COMPUTER NETWORK

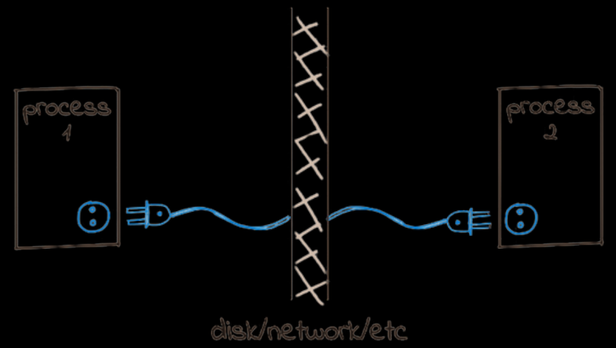
Group Members :

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

INTRODUCTION & DETAILS

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.

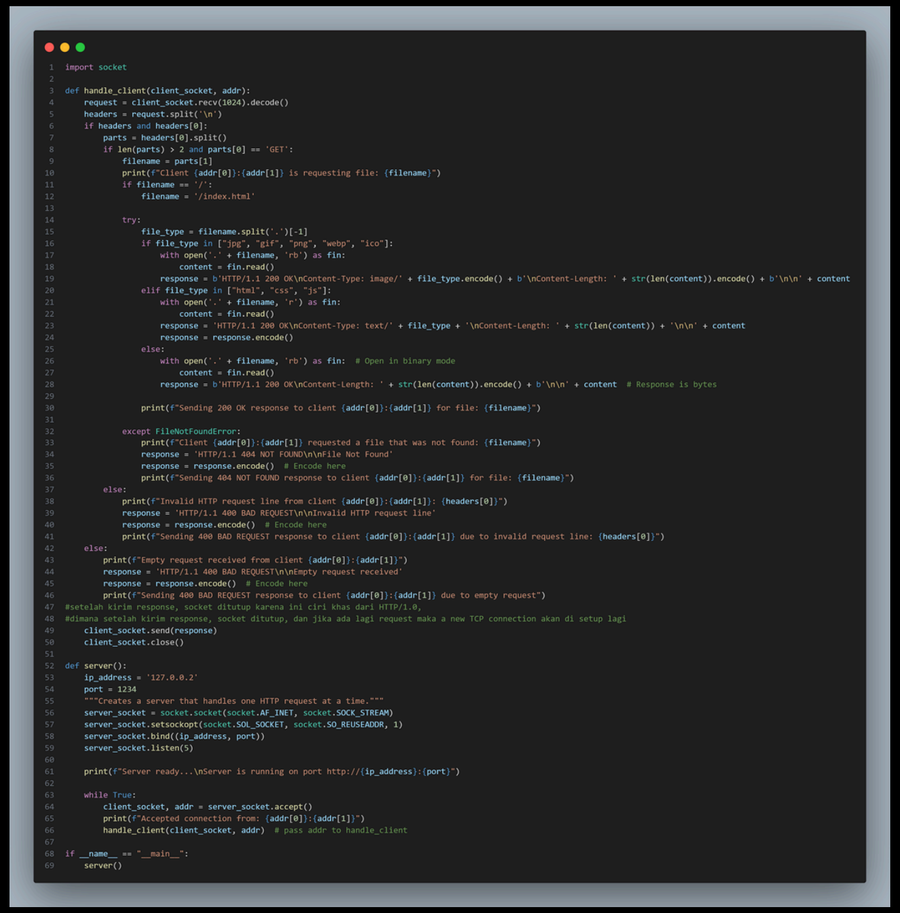




In this web server we implement two types of server, one is single-threaded and the other one is multi-threaded servers. And we also implement the TCP client for make the request to the servers.

SOURCE CODE

Single-thread server



```python
import socket

def handle_client(client_socket, addr):
    request = client_socket.recv(1024).decode()
    headers = request.split('\n')
    if headers and headers[0]:
        parts = headers[0].split()
        if len(parts) > 2 and parts[0] == 'GET':
            filename = parts[1]
            print(f"Client {addr[0]}:{addr[1]} is requesting file: {filename}")
            if filename == '/':
                filename = '/index.html'

            try:
                file_type = filename.split('.')[-1]
                if file_type in ["jpg", "gif", "png", "webp", "ico"]:
                    with open('.' + filename, 'rb') as fin:
                        content = fin.read()
                    response = b'HTTP/1.1 200 OK\nContent-Type: image/' + file_type.encode() + b'\nContent-Length: ' + str(len(content)).encode() + b'\n\n' + content
                elif file_type in ["html", "css", "js"]:
                    with open('.' + filename, 'r') as fin:
                        content = fin.read()
                    response = 'HTTP/1.1 200 OK\nContent-Type: text/' + file_type + '\nContent-Length: ' + str(len(content)) + '\n\n' + content
                    response = response.encode()
                else:
                    with open('.' + filename, 'rb') as fin:  # Open in binary mode
                        content = fin.read()
                    response = b'HTTP/1.1 200 OK\nContent-Length: ' + str(len(content)).encode() + b'\n\n' + content  # Response is bytes

                print(f"Sending 200 OK response to client {addr[0]}:{addr[1]} for file: {filename}")

            except FileNotFoundError:
                print(f"Client {addr[0]}:{addr[1]} requested a file that was not found: {filename}")
                response = 'HTTP/1.1 404 NOT FOUND\n\nFile Not Found'
                response = response.encode()  # Encode here
                print(f"Sending 404 NOT FOUND response to client {addr[0]}:{addr[1]} for file: {filename}")
        else:
            print(f"Invalid HTTP request line from client {addr[0]}:{addr[1]}: {headers[0]}")
            response = 'HTTP/1.1 400 BAD REQUEST\n\nInvalid HTTP request line'
            response = response.encode()  # Encode here
            print(f"Sending 400 BAD REQUEST response to client {addr[0]}:{addr[1]} due to invalid request line: {headers[0]}")
    else:
        print(f"Empty request received from client {addr[0]}:{addr[1]}")
        response = 'HTTP/1.1 400 BAD REQUEST\n\nEmpty request received'
        response = response.encode()  # Encode here
        print(f"Sending 400 BAD REQUEST response to client {addr[0]}:{addr[1]} due to empty request")
#setelah kirim response, socket ditutup karena ini ciri khas dari HTTP/1.0,
#dimana setelah kirim response, socket ditutup, dan jika ada lagi request maka a new TCP connection akan di setup lagi
    client_socket.send(response)
    client_socket.close()

def server():
    ip_address = '127.0.0.2'
    port = 1234
    """Creates a server that handles one HTTP request at a time."""
    server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
    server_socket.bind((ip_address, port))
    server_socket.listen(5)

    print(f"Server ready...\nServer is running on port http://{ip_address}:{port}")

    while True:
        client_socket, addr = server_socket.accept()
        print(f"Accepted connection from: {addr[0]}:{addr[1]}")
        handle_client(client_socket, addr)  # pass addr to handle_client

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

Single thread server

1

```python
import socket
```

2

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

3

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

4

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

5

```python
if filename == '/':
                filename = '/index.html'

```

9

```python

def server():
    ip_address = '127.0.0.2'
    port = 1234
    """Creates a server that handles one HTTP request at a time."""
    server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
    server_socket.bind((ip_address, port))
    server_socket.listen(5)

    print(f"Server ready...\nServer is running on port http://{ip_address}:{port}")

    while True:
        client_socket, addr = server_socket.accept()
        print(f"Accepted connection from: {addr[0]}:{addr[1]}")
        handle_client(client_socket, addr)  # pass addr to handle_client

```

10

```python
if __name__ == "__main__":
    server()
```

8

```python
client_socket.send(response)
client_socket.close()
```

6

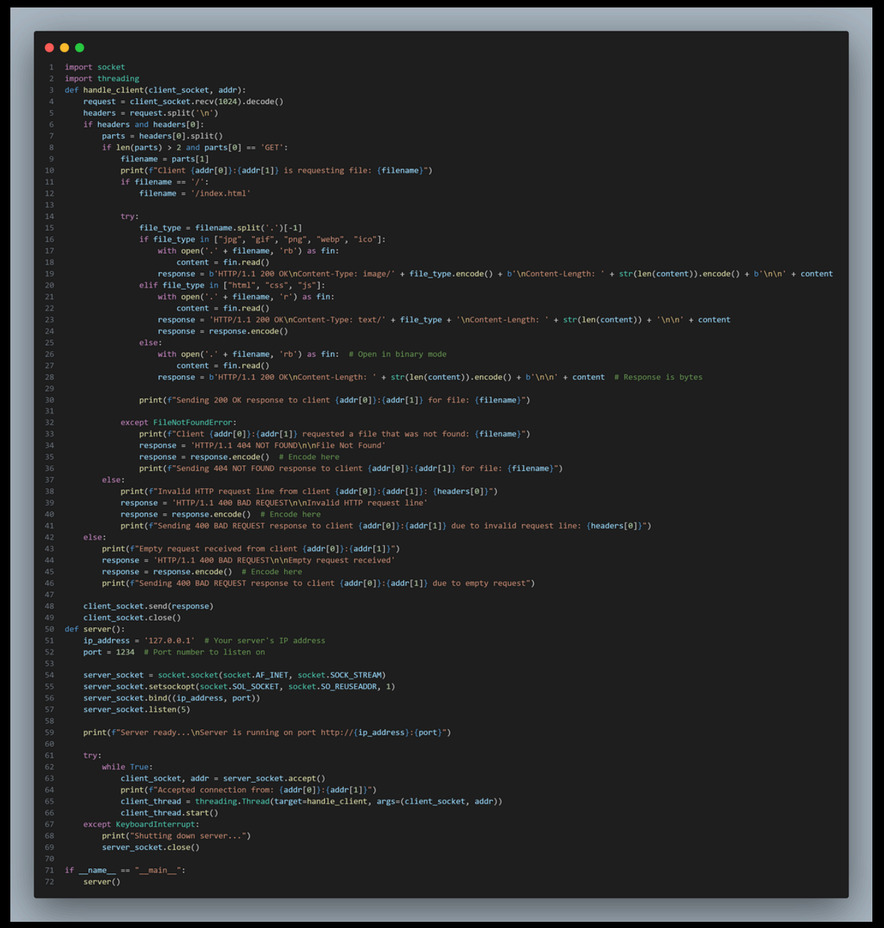
```python
try:
            file_type = filename.split('.')[-1]
            if file_type in ["jpg", "gif", "png", "webp", "ico"]:
                with open('.' + filename, 'rb') as fin:
                    content = fin.read()
                response = b'HTTP/1.1 200 OK\nContent-Type: image/' + file_type.encode() + b'\nContent-Length: ' + str(len(content)).encode() + b'\n\n' + content
            elif file_type in ["html", "css", "js"]:
                with open('.' + filename, 'r') as fin:
                    content = fin.read()
                response = 'HTTP/1.1 200 OK\nContent-Type: text/' + file_type + '\nContent-Length: ' + str(len(content)) + '\n\n' + content
                response = response.encode()
            else:
                with open('.' + filename, 'rb') as fin:  # Open in binary mode
                    content = fin.read()
                response = b'HTTP/1.1 200 OK\nContent-Length: ' + str(len(content)).encode() + b'\n\n' + content  # Response is bytes

            print(f"Sending 200 OK response to client {addr[0]}:{addr[1]} for file: {filename}")
```

7

```python
except FileNotFoundError:
                print(f"Client {addr[0]}:{addr[1]} requested a file that was not found: {filename}")
                response = 'HTTP/1.1 404 NOT FOUND\n\nFile Not Found'
                response = response.encode()  # Encode here
                print(f"Sending 404 NOT FOUND response to client {addr[0]}:{addr[1]} for file: {filename}")
        else:
            print(f"Invalid HTTP request line from client {addr[0]}:{addr[1]}: {headers[0]}")
            response = 'HTTP/1.1 400 BAD REQUEST\n\nInvalid HTTP request line'
            response = response.encode()  # Encode here
            print(f"Sending 400 BAD REQUEST response to client {addr[0]}:{addr[1]} due to invalid request line: {headers[0]}")
    else:
        print(f"Empty request received from client {addr[0]}:{addr[1]}")
        response = 'HTTP/1.1 400 BAD REQUEST\n\nEmpty request received'
        response = response.encode()  # Encode here
        print(f"Sending 400 BAD REQUEST response to client {addr[0]}:{addr[1]} due to empty request")
```

SOURCE CODE

Multi-thread server



```python
import socket
import threading
def handle_client(client_socket, addr):
    request = client_socket.recv(1024).decode()
    headers = request.split('\n')
    if headers and headers[0]:
        parts = headers[0].split()
        if len(parts) > 2 and parts[0] == 'GET':
            filename = parts[1]
            print(f"Client {addr[0]}:{addr[1]} is requesting file: {filename}")
            if filename == '/':
                filename = '/index.html'

            try:
                file_type = filename.split('.')[-1]
                if file_type in ["jpg", "gif", "png", "webp", "ico"]:
                    with open('.' + filename, 'rb') as fin:
                        content = fin.read()
                    response = b'HTTP/1.1 200 OK\nContent-Type: image/' + file_type.encode() + b'\nContent-Length: ' + str(len(content)).encode() + b'\n\n' + content
                elif file_type in ["html", "css", "js"]:
                    with open('.' + filename, 'r') as fin:
                        content = fin.read()
                    response = 'HTTP/1.1 200 OK\nContent-Type: text/' + file_type + '\nContent-Length: ' + str(len(content)) + '\n\n' + content
                    response = response.encode()
                else:
                    with open('.' + filename, 'rb') as fin:  # Open in binary mode
                        content = fin.read()
                    response = b'HTTP/1.1 200 OK\nContent-Length: ' + str(len(content)).encode() + b'\n\n' + content  # Response is bytes

                print(f"Sending 200 OK response to client {addr[0]}:{addr[1]} for file: {filename}")

            except FileNotFoundError:
                print(f"Client {addr[0]}:{addr[1]} requested a file that was not found: {filename}")
                response = 'HTTP/1.1 404 NOT FOUND\n\nFile Not Found'
                response = response.encode()  # Encode here
                print(f"Sending 404 NOT FOUND response to client {addr[0]}:{addr[1]} for file: {filename}")
        else:
            print(f"Invalid HTTP request line from client {addr[0]}:{addr[1]}: {headers[0]}")
            response = 'HTTP/1.1 400 BAD REQUEST\n\nInvalid HTTP request line'
            response = response.encode()  # Encode here
            print(f"Sending 400 BAD REQUEST response to client {addr[0]}:{addr[1]} due to invalid request line: {headers[0]}")
    else:
        print(f"Empty request received from client {addr[0]}:{addr[1]}")
        response = 'HTTP/1.1 400 BAD REQUEST\n\nEmpty request received'
        response = response.encode()  # Encode here
        print(f"Sending 400 BAD REQUEST response to client {addr[0]}:{addr[1]} due to empty request")

    client_socket.send(response)
    client_socket.close()
def server():
    ip_address = '127.0.0.1'  # Your server's IP address
    port = 1234  # Port number to listen on

    server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
    server_socket.bind((ip_address, port))
    server_socket.listen(5)

    print(f"Server ready...\nServer is running on port http://{ip_address}:{port}")

    try:
        while True:
            client_socket, addr = server_socket.accept()
            print(f"Accepted connection from: {addr[0]}:{addr[1]}")
            client_thread = threading.Thread(target=handle_client, args=(client_socket, addr))
            client_thread.start()
    except KeyboardInterrupt:
        print("Shutting down server...")
        server_socket.close()

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

Multi thread server

1

```python
import socket
```

2

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

3

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

4

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

5

```python
if filename == '/':
                filename = '/index.html'

```

9

```python
def server():
    ip_address = '127.0.0.1'  # Your server's IP address
    port = 1234  # Port number to listen on

    server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
    server_socket.bind((ip_address, port))
    server_socket.listen(5)

    print(f"Server ready...\nServer is running on port http://{ip_address}:{port}")

    try:
        while True:
            client_socket, addr = server_socket.accept()
            print(f"Accepted connection from: {addr[0]}:{addr[1]}")
            client_thread = threading.Thread(target=handle_client, args=(client_socket, addr))
            client_thread.start()
    except KeyboardInterrupt:
        print("Shutting down server...")
        server_socket.close()

```

10

```python
if __name__ == "__main__":
    server()
```

8

```python
client_socket.send(response)
client_socket.close()
```

6

```python
try:
            file_type = filename.split('.')[-1]
            if file_type in ["jpg", "gif", "png", "webp", "ico"]:
                with open('.' + filename, 'rb') as fin:
                    content = fin.read()
                response = b'HTTP/1.1 200 OK\nContent-Type: image/' + file_type.encode() + b'\nContent-Length: ' + str(len(content)).encode() + b'\n\n' + content
            elif file_type in ["html", "css", "js"]:
                with open('.' + filename, 'r') as fin:
                    content = fin.read()
                response = 'HTTP/1.1 200 OK\nContent-Type: text/' + file_type + '\nContent-Length: ' + str(len(content)) + '\n\n' + content
                response = response.encode()
            else:
                with open('.' + filename, 'rb') as fin:  # Open in binary mode
                    content = fin.read()
                response = b'HTTP/1.1 200 OK\nContent-Length: ' + str(len(content)).encode() + b'\n\n' + content  # Response is bytes

            print(f"Sending 200 OK response to client {addr[0]}:{addr[1]} for file: {filename}")
```

7

```python
except FileNotFoundError:
                print(f"Client {addr[0]}:{addr[1]} requested a file that was not found: {filename}")
                response = 'HTTP/1.1 404 NOT FOUND\n\nFile Not Found'
                response = response.encode()  # Encode here
                print(f"Sending 404 NOT FOUND response to client {addr[0]}:{addr[1]} for file: {filename}")
        else:
            print(f"Invalid HTTP request line from client {addr[0]}:{addr[1]}: {headers[0]}")
            response = 'HTTP/1.1 400 BAD REQUEST\n\nInvalid HTTP request line'
            response = response.encode()  # Encode here
            print(f"Sending 400 BAD REQUEST response to client {addr[0]}:{addr[1]} due to invalid request line: {headers[0]}")
    else:
        print(f"Empty request received from client {addr[0]}:{addr[1]}")
        response = 'HTTP/1.1 400 BAD REQUEST\n\nEmpty request received'
        response = response.encode()  # Encode here
        print(f"Sending 400 BAD REQUEST response to client {addr[0]}:{addr[1]} due to empty request")
```

SOURCE CODE

Client

```python
#Using terminal for make a request and response between the server and the client
import socket
import sys

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

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

    response = ''
    while True:
        data = client_socket.recv(1024)
        if not data:
            break
        response += data.decode()

    print(response)
    client_socket.close()

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

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

    http_client(server_host, server_port, filename)
```

Client

1

```python
import socket
import sys
```

2

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

3

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

4

```python
while True:
    data = client_socket.recv(1024)
    if not data:
        break
    response += data.decode()
```

5

```python
print(response)
client_socket.close()
```

6

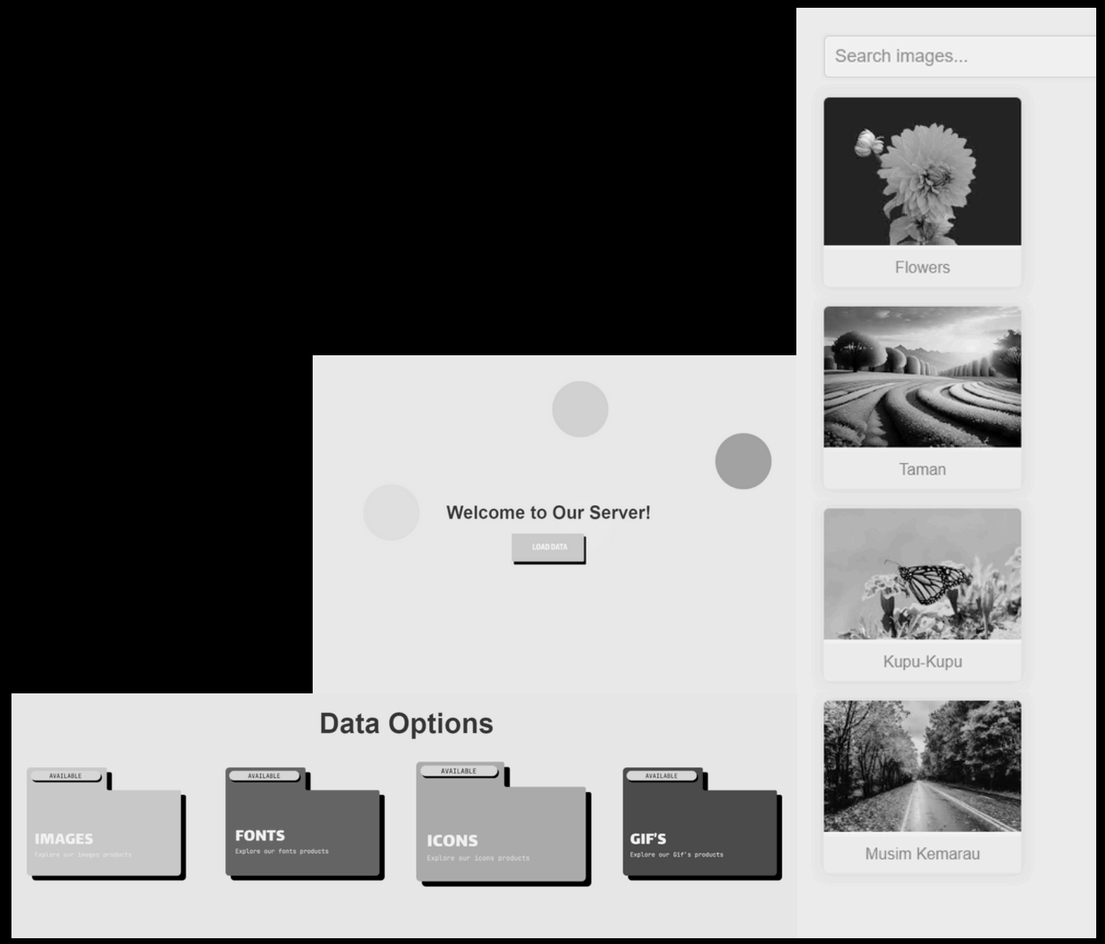
```python
if __name__ == "__main__":
    if len(sys.argv) != 4:
        print("Usage: python client.py server_host server_port filename")
        sys.exit(1)
```

7

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

8

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



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.

Single-thread server

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

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 */
        }
```

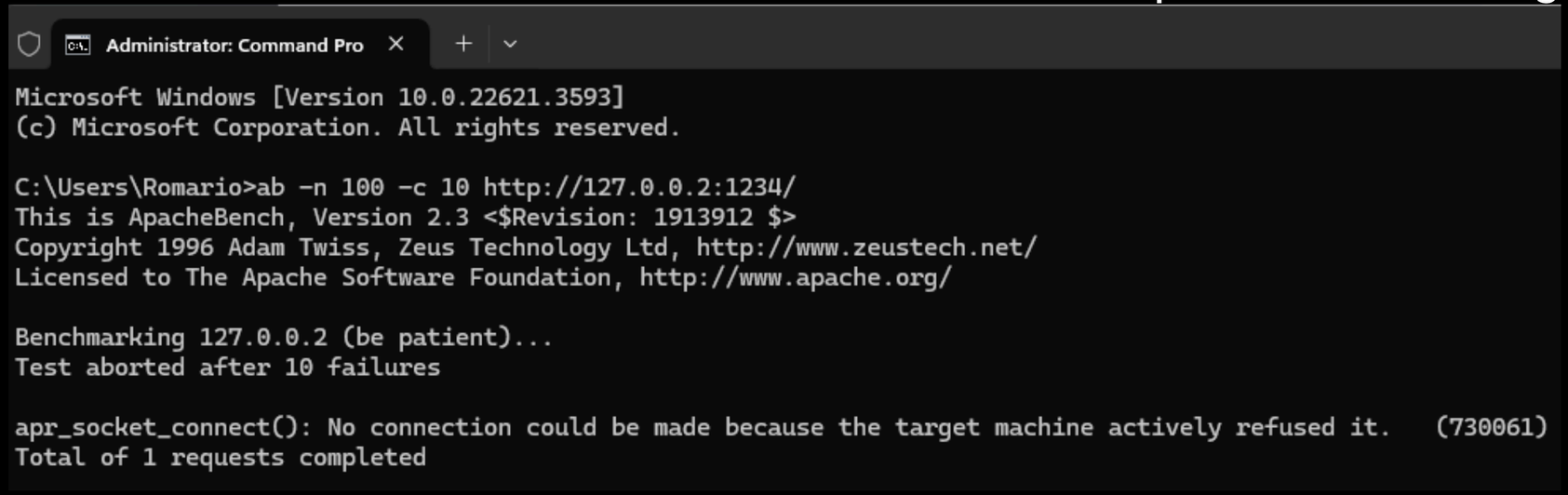
TESTING

```
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
```

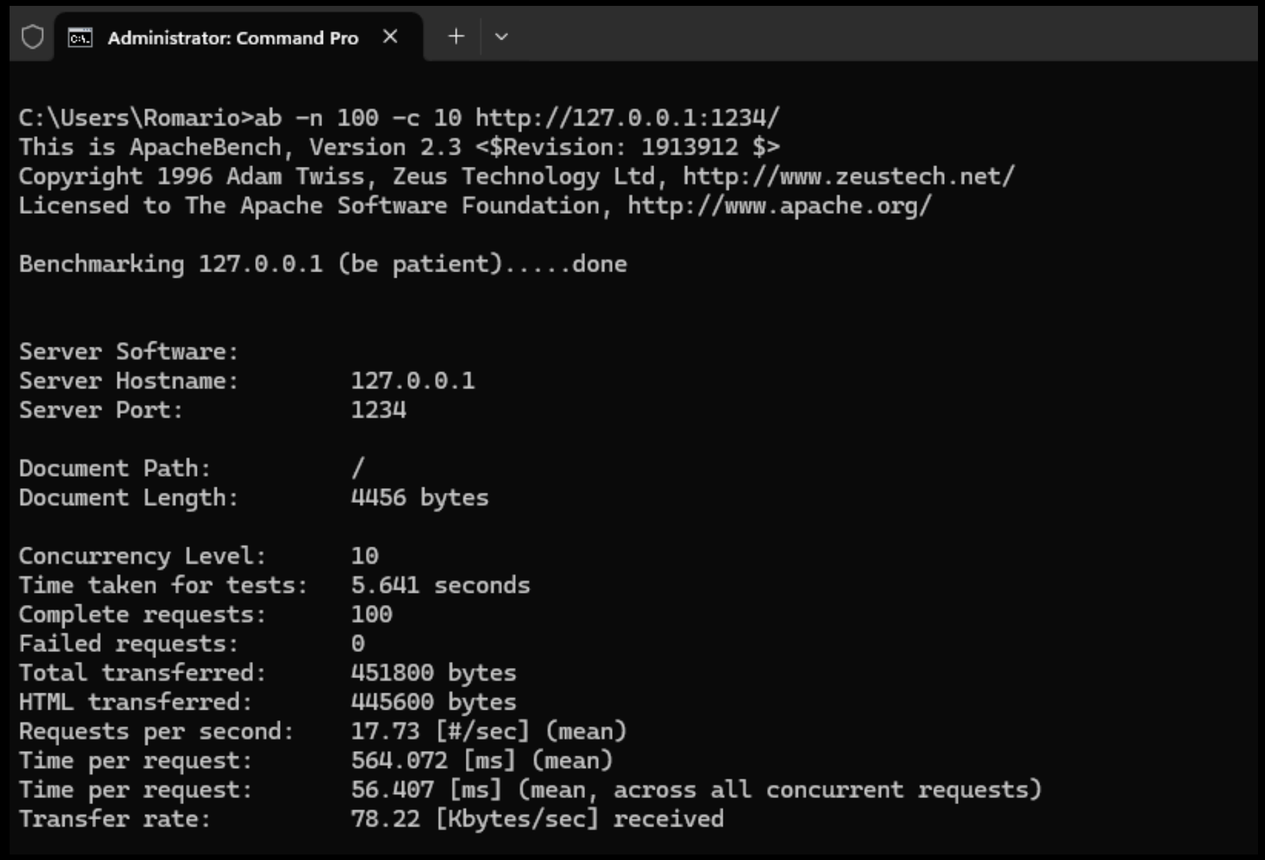
```
Administrator: C:\Windows\S

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 */
        }
```

How About send mulitple request

Apache bench testing



How About send mulitple request



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 !