**FINAL REPORT**

**WEB SERVER USING SOCKET PROGRAMMING**



**Prepared by:**

**Romario Viegas Francisco Marcal 1301225492**

**Ari Ramadhan 1301224458**

**Laode Muhammad Fathir 1301224446**

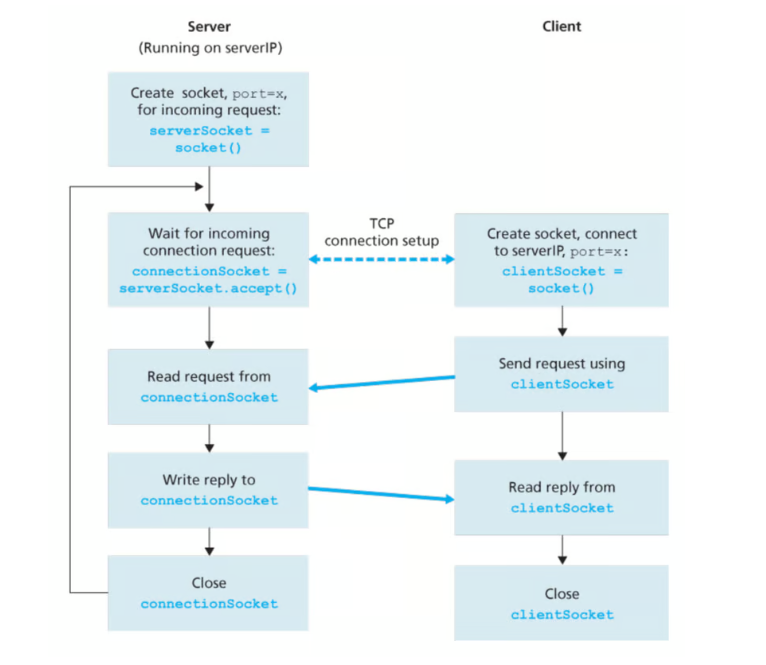
**Telkom University**

**2024/2025**

## Socket Programming

Socket programming is programming used to perform process-to-process communication within a network. Besides process communication, computers in a network also perform host-to-host communication. Socket programming testing can be conducted using the sock\_stream networking socket type from client to server using process addresses (IP and Port) on the same computer with the same directory, different directories, and on different computers. For testing, a server and client will be created to perform process communication. In both server and client, the sockets used are AF\_INET. In the server's process communication, first, the socket creation process is done with the socket() function call, then the server binds the created socket to the network based on the local address and port number with the bind() function call.

The flow of how socket works :



1. **SPESIFICATIONS**

* **Single Thread**

A single-threaded program contains only one main thread of execution. It processes tasks sequentially, one after the other. Single-threaded programs are straightforward and easy to reason about, making them suitable for simple applications with limited concurrency requirements.

Single-threaded programs are suitable for applications with low computational complexity or minimal I/O operations. They are easy to debug and maintain due to their sequential nature.

* **Multi-Thread**

Multi-threaded programs utilize multiple threads to handle tasks concurrently, potentially improving performance and responsiveness. They are well-suited for applications that involve complex computations, I/O operations, or tasks that can run independently.

Multi-threaded programs excel in scenarios with computationally intensive tasks or applications that can benefit from parallel processing. They can enhance overall system performance and responsiveness.

1. **THREAD SCHEME**

* **Single Thread**

Skema pada web server single thread ini adalah sebagai berikut:

Fungsi handle\_client: Ini adalah fungsi yang menangani permintaan dari satu klien pada satu waktu. Fungsi ini mirip dengan yang di skema multi-thread, tetapi di sini tidak ada pembuatan thread baru. Fungsi ini menerima soket klien dan alamatnya sebagai argumen. Setelah menerima permintaan HTTP dari klien, itu memproses permintaan tersebut sesuai dengan kode yang telah diimplementasikan. Setelah menangani permintaan, tanggapan HTTP dikirim ke klien dan soket klien ditutup.

Fungsi server: Ini adalah fungsi utama yang bertanggung jawab untuk membuat server socket, mengikatnya ke alamat IP dan port tertentu, dan mulai mendengarkan koneksi masuk. Setiap kali koneksi baru diterima, fungsi ini memanggil handle\_client untuk menangani permintaan dari klien tersebut.

Loop Utama: Program berada dalam loop tak terbatas yang terus menerima koneksi baru dari klien. Setiap koneksi diterima, server menangani permintaan tersebut menggunakan fungsi handle\_client. Karena hanya ada satu thread yang menangani semua koneksi, server akan menangani koneksi satu per satu secara berurutan.

Dengan pendekatan ini, server hanya dapat menangani satu koneksi pada satu waktu. Karena tidak ada multi-threading, kinerja server dapat terpengaruh jika ada banyak koneksi yang harus ditangani secara bersamaan. Namun, pendekatan ini sederhana dan cocok untuk aplikasi yang tidak memerlukan kinerja yang sangat tinggi atau tidak memiliki persyaratan khusus terkait skala atau responsifitas tinggi.

* **Multi-Thread**

Skema thread pada web server multi thread ini menggunakan konsep multi-threading untuk mengelola koneksi dari beberapa klien secara bersamaan. Mari kita bahas langkah-langkahnya:

Import Libraries: Program mengimpor modul socket untuk membuat soket dan modul threading untuk mendukung multi-threading.

Fungsi handle\_client: Ini adalah fungsi yang akan dijalankan oleh setiap thread untuk menangani permintaan dari klien. Fungsi ini menerima soket klien dan alamatnya sebagai argumen. Ini kemudian membaca permintaan HTTP dari klien, mengidentifikasi file yang diminta, membaca isi file, dan mengirimkan tanggapan HTTP ke klien. Setelah selesai menangani permintaan, soket klien ditutup.

Fungsi server: Ini adalah fungsi utama yang bertanggung jawab untuk membuat server socket, mengikatnya ke alamat IP dan port tertentu, dan mulai mendengarkan koneksi masuk. Ketika koneksi diterima, fungsi ini membuat thread baru untuk menangani klien tersebut, menggunakan fungsi handle\_client.

Loop Utama: Program berada dalam loop tak terbatas yang terus menerima koneksi baru dari klien. Setiap koneksi baru diterima, server membuat thread baru untuk menangani klien tersebut, sehingga mengizinkan server untuk melayani beberapa klien secara bersamaan.

Penanganan KeyboardInterrupt: Jika pengguna menekan Ctrl+C, server akan menangkap pengecualian KeyboardInterrupt dan menutup soket server sebelum keluar.

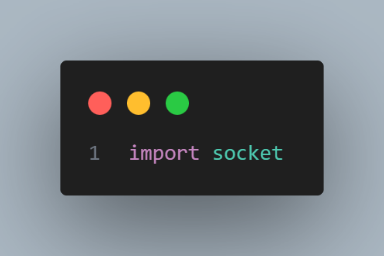
Dengan menggunakan skema multi-threading seperti ini, server dapat mengatasi banyak koneksi klien secara efisien dan secara bersamaan, meningkatkan kinerja dan responsivitasnya.

1. **SOURCE CODE**

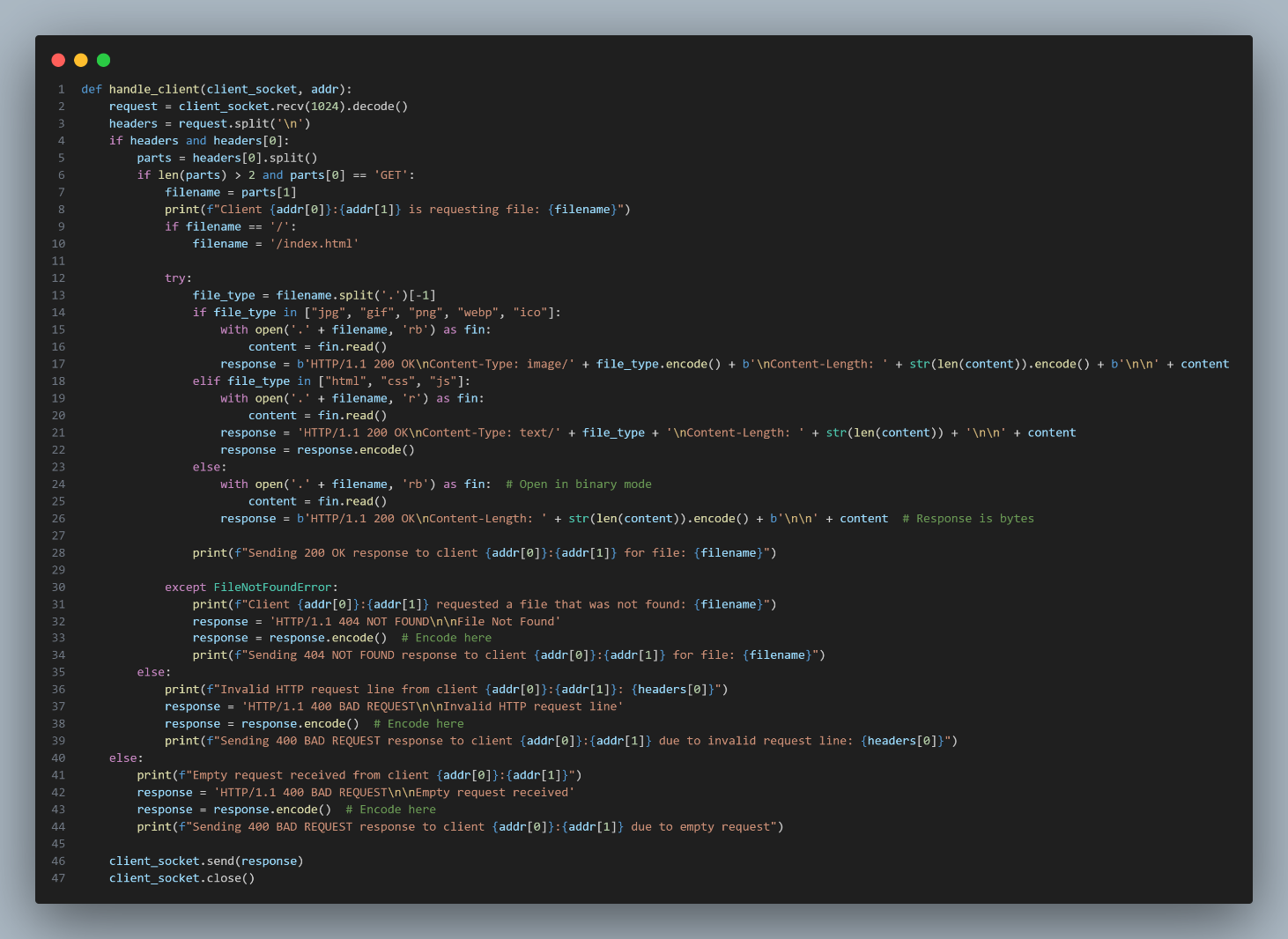
* **Single Thread**

This Python script is a simple HTTP server that listens for connections, accepts HTTP GET requests, and sends back the requested file if it exists.

import socket: This line imports the socket module, which provides low-level networking interface.



def handle\_client(client\_socket, addr): This function handles the client request. It takes two parameters: the client socket and the client address.



request = client\_socket.recv(1024).decode(): This line receives data from the client. The recv function reads at most 1024 bytes.



headers = request.split('\n'): This line splits the request into lines.



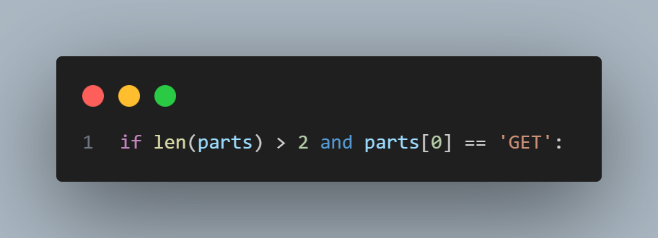
if headers and headers[0]: This checks if the request is not empty.



parts = headers[0].split(): This line splits the first line of the request into words.



if len(parts) > 2 and parts[0] == 'GET': This checks if the request is a valid HTTP GET request.



filename = parts[1]: This line gets the requested filename from the request.



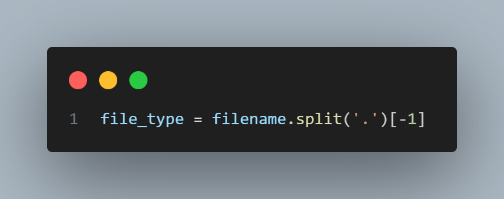
if filename == '/': filename = '/index.html': This line sets the default file to 'index.html' if the request is for the root directory.



The block of code inside the try block are responsible for handling the requested file and preparing the HTTP response.



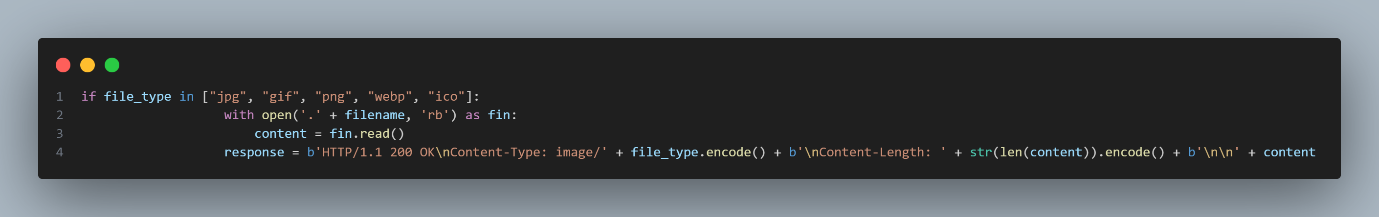
file\_type = filename.split('.')[-1]: This line gets the file extension by splitting the filename on the period and taking the last element.



if file\_type in ["jpg", "gif", "png", "webp", "ico"]::

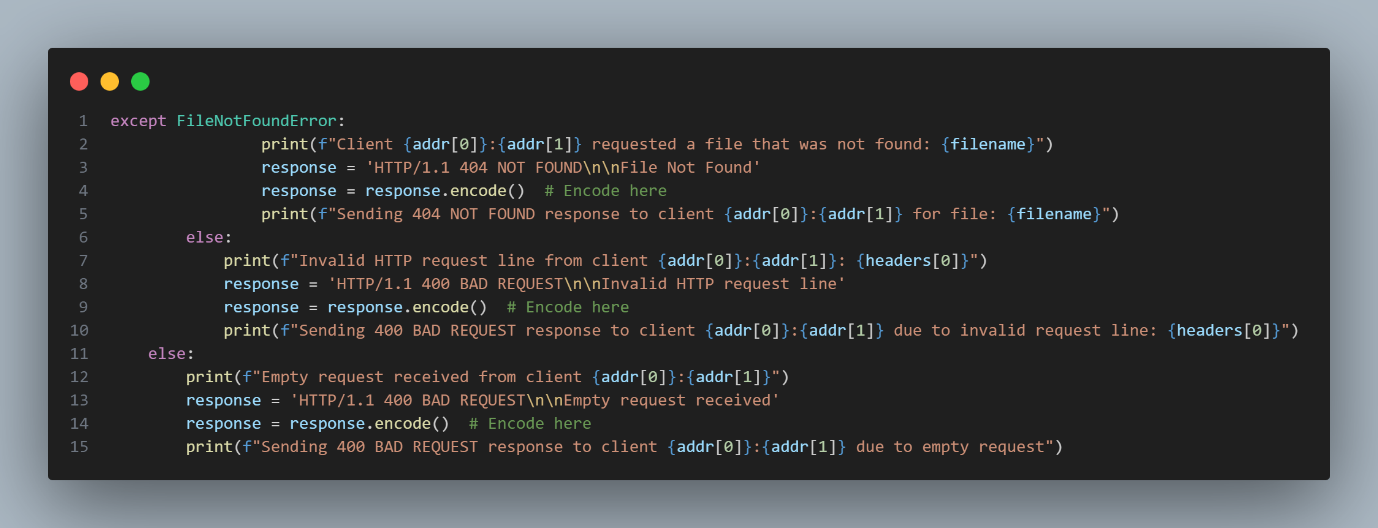
with open('.' + filename, 'rb') as fin::

content = fin.read():

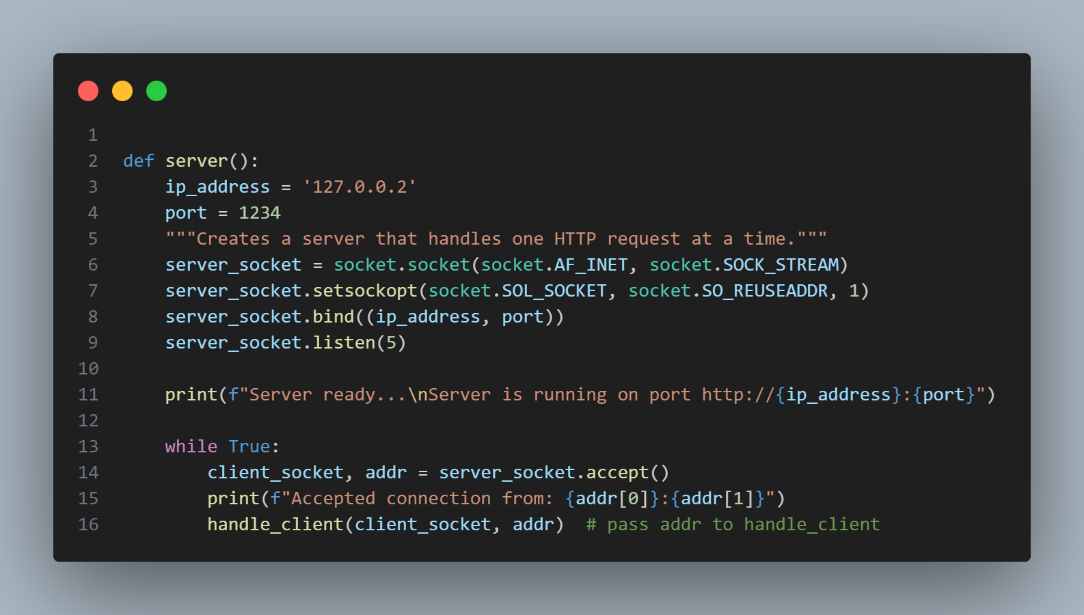


This checks if the file type is an image. And open the file in binary mode for reading. The file path is the current directory plus the filename. Reads the entire file content. Then prepares the HTTP response. It includes the status line, Content-Type header, Content-Length header, and the file content.

try: ... except FileNotFoundError: ...: This block tries to open the requested file and send it back to the client. If the file does not exist, it sends a 404 error.



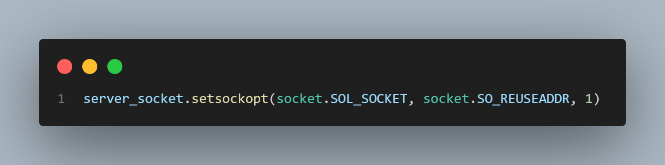
def server(): ...: This function sets up the server.



server\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM): This line creates a new socket using the given address family (IPv4) and socket type (TCP).



server\_socket.setsockopt(socket.SOL\_SOCKET, socket.SO\_REUSEADDR, 1): This line sets the socket option to reuse the address.



server\_socket.bind((ip\_address, port)):

server\_socket.listen(5):

This line makes the server listen for incoming connections, with a backlog of 5 connections.

And binds the socket to the given IP address and port.

while True: ...: This infinite loop accepts new connections and handles them.



* **Multi-Thread**

In the Multi-Thread the overall code is similar, but what makes different is, it’s using threading package package in order to handling for multiple request. Here is the code :



The client\_t hread = threading. This line creates a new thread object. The target parameter is the function to be executed by the thread, in this case handle\_client. The args parameter is a tuple of arguments to pass to the target function. Here, it's passing the client socket and the client address. Then it starts the execution of the thread. This means that the handle\_client function will be called in a separate thread of execution. This allows the server to handle multiple client connections concurrently, as each connection is managed in its own thread.

if \_\_name\_\_ == "\_\_main\_\_": server(): This line runs the server function if the script is run directly.

* **Client**

import socket, sys: This line imports the socket and sys modules. The socket module is for network communications, and the sys module is for accessing command-line arguments.

def http\_client(server\_host, server\_port, filename):: This function creates a client that sends an HTTP GET request to the server. It takes three parameters: the server host, the server port, and the filename to request.



client\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM): This line creates a new socket using the given address family (IPv4) and socket type (TCP). client\_socket.connect((server\_host, server\_port)): This line connects the client socket to the server at the given host and port. request = f"GET /{filename} HTTP/1.1\r\nHost: {server\_host}\r\n\r\n": This line prepares the HTTP GET request. client\_socket.send(request.encode()): This line sends the HTTP request to the server.

while True: ...: This loop receives the server's response. It reads data from the socket in chunks of 1024 bytes until there's no more data.



print(response): This line prints the server's response.

client\_socket.close(): This line closes the socket.

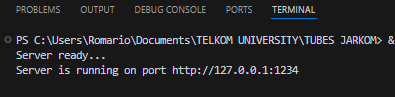
if \_\_name\_\_ == "\_\_main\_\_": ...: This block runs the client function if the script is run directly. It checks if the correct number of command-line arguments are provided, and if so, it extracts the server host, server port, and filename from the arguments and calls the http\_client function. If the correct number of arguments are not provided, it prints a usage message and exits.



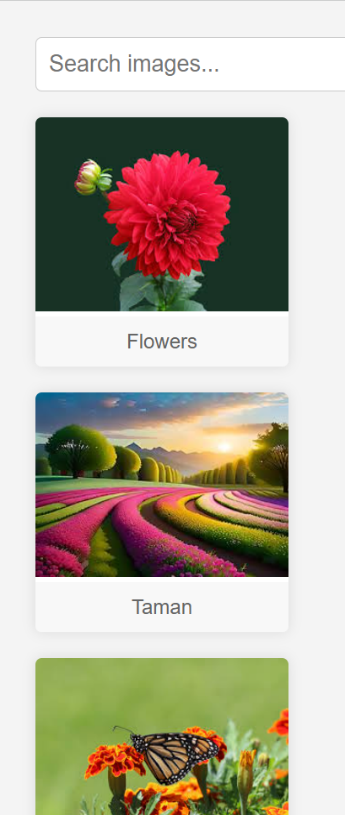
1. RESULTS

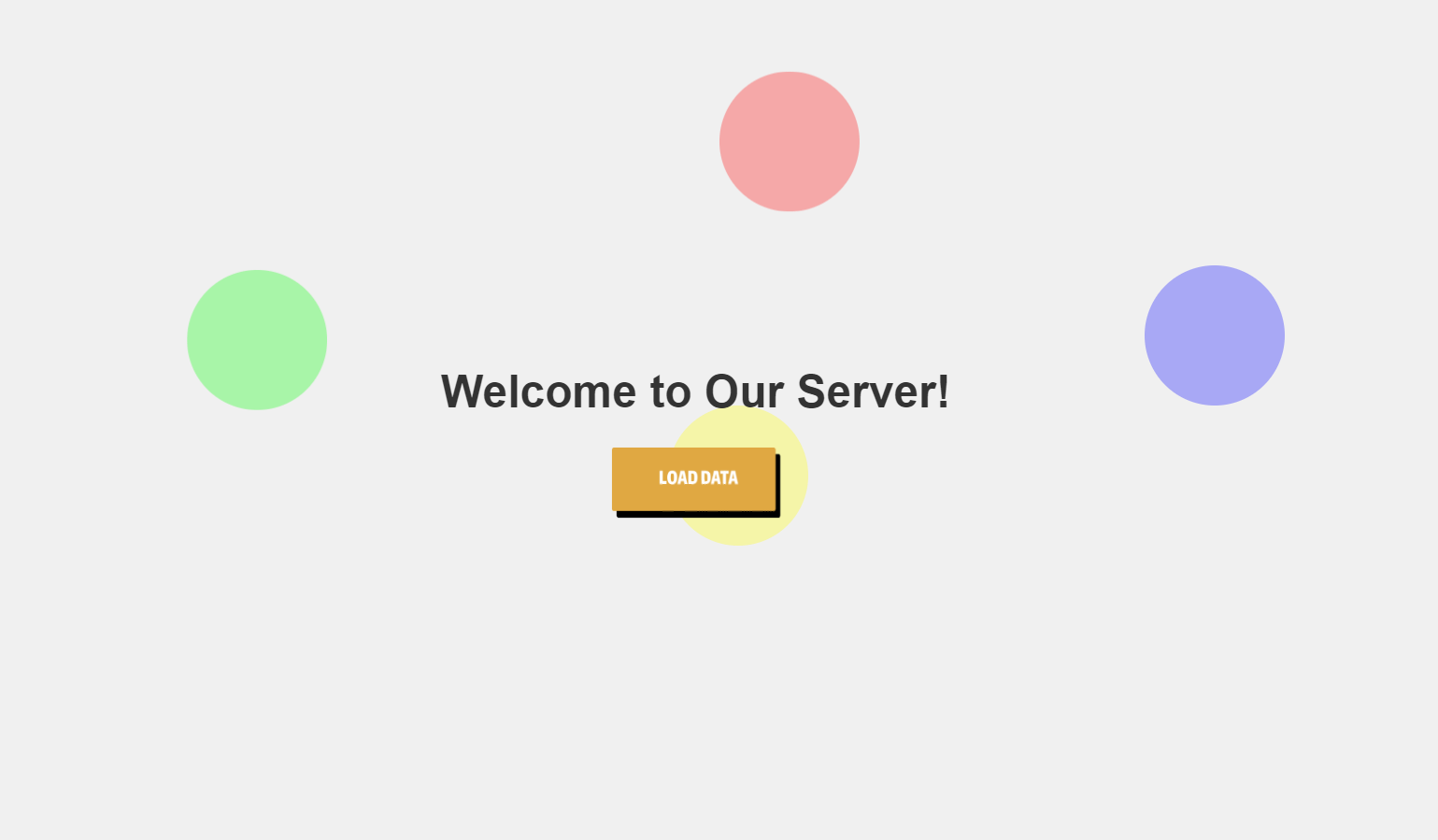
The Results will be seen in the steps below:

When the server is running :

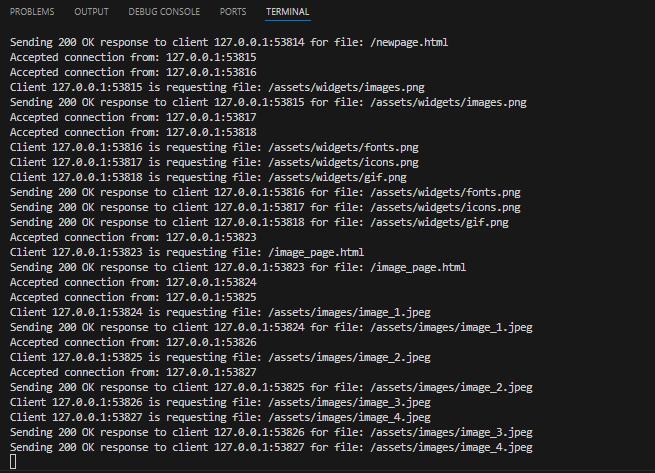


When the Web Page is appears like in the image below, the terminal show informations from the server:



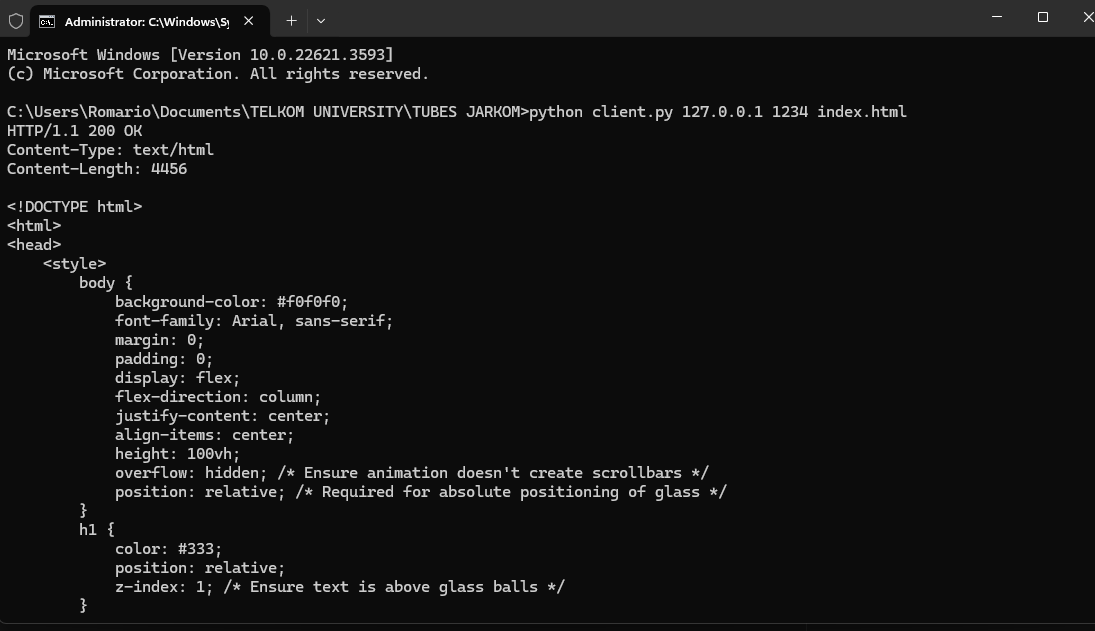
The response information from the server that appears in the terminal :



1. **TESTING**

In this stage we are trying to make testing regarding to the socket connecttion using web browser and the terminal :

Requeest for the index.html file in the terminal :

****

In the VS CODE terminal :



And working for other file type as well like jpeg, png, gif, text and others. Here is the examples :







1. **CONCLUSIONS**

In this final project, we have developed a web server using socket programming, along with the implementation of several protocols such as TCP and HTTP. In building the server, we created various types of server connections ranging from single-threaded to multi-threaded, each serving the same functional purpose but differing in how they handle requests from clients. All source code was written in Python. From this experience, we gained valuable insights into understanding the technical aspects of a connection between a server and client, including socket programming and various commonly used protocols. We hope that in the future, we can delve deeper into exploring what goes on within internet networks.