# Laporan Tugas Besar Mata Kuliah Jaringan Komputer
# Web Server TCP Socket Programming


**Disusun Oleh:**

Fathan Arya Maulana 103012300083

Dzaky Alfaris 103012300391

M. Rifqi Dzaky Azhad 103012330009

**Program Studi S1 Informatika**
**Fakultas Informatika**
**Universitas Telkom**
**Bandung, Indonesia**
**2025**

# I. Pendahuluan

Berikut adalah pendahuluan untuk laporan ini, terdiri ata latar belakang, rumusan masalah, dan tujuan.

## A. Latar Belakang

Dalam era digital saat ini, komunikasi antar perangkat dalam jaringan komputer menjadi sangat penting, terutama melalui protokol HTTP yang digunakan pada layanan web. Untuk memahami cara kerja dasar dari web server dan klien, mahasiswa perlu mempelajari dan mengimplementasikan komunikasi berbasis socket programming dengan protokol TCP.

## B. Rumusan Masalah

1. Bagaimana cara membangun web server berbasis TCP yang dapat menerima dan memproses request HTTP dari klien?
2. Bagaimana mengimplementasikan server yang mampu menangani banyak permintaan secara simultan menggunakan multithreading?

## C. Tujuan

1. Mengembangkan web server sederhana berbasis TCP yang dapat menangani request HTTP.
2. Mengimplementasikan multithreading pada server untuk melayani beberapa klien secara bersamaan.
3. Meningkatkan pemahaman tentang komunikasi jaringan dan pemrograman socket.

# II. Pembahasan

Berikut adalah pembahasan terkait dengan pendekatan dalam menyelesai masalah, dokumentasi code, evaluasi, dan analisa.

## A. Pendekatan

Pendekatan yang digunakan dalam tugas ini dimulai dengan membangun server web sederhana berbasis TCP menggunakan socket programming. Server dikembangkan untuk menerima dan memproses permintaan HTTP metode GET, mengambil file dari direktori lokal, dan mengirimkan respons sesuai format HTTP.

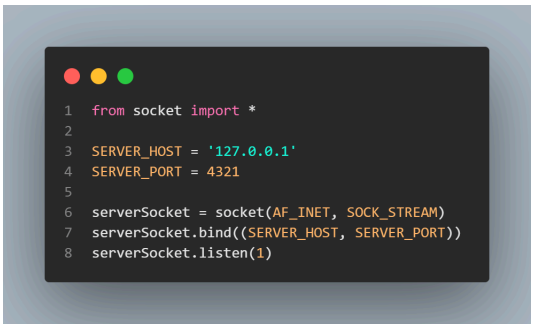
Setelah server dasar berfungsi, pendekatan berikutnya adalah mengimplementasikan fitur multithreading. Setiap koneksi dari klien akan dilayani oleh thread terpisah agar server dapat menangani beberapa request secara bersamaan. Pengujian dilakukan melalui browser dan klien mandiri yang dibuat menggunakan bahasa pemrograman Python.

## B. Code

Berikut adalah implementasi code untuk server single, server multi, dan client.

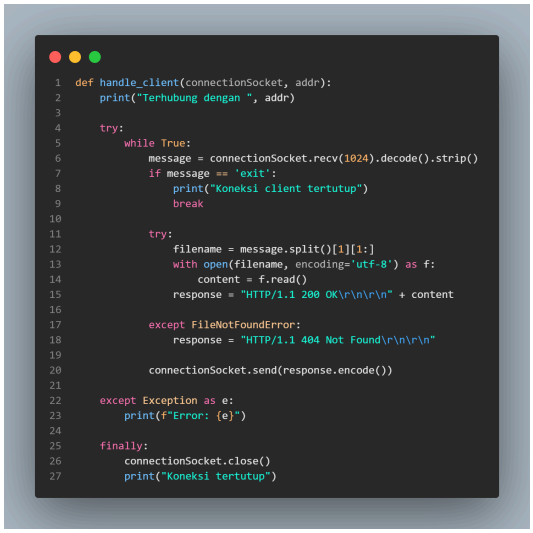
### 1. Server Single

Code server single diawali dengan import library socket dengan seluruh method dan atributnya. Dilanjutkan dengan inisialisasi SERVER_HOST dan SERVER_PORT. serverSocket dibentuk dan melakukan bind dan listen dengan nilai 1.



```python
from socket import *

SERVER_HOST = '127.0.0.1'
SERVER_PORT = 4321

serverSocket = socket(AF_INET, SOCK_STREAM)
serverSocket.bind((SERVER_HOST, SERVER_PORT))
serverSocket.listen(1)
```

Dilanjutkan dengan mendefinisikan prosedur ‘handle_client’ dengan parameter socket dan address user, prosedur ini melakukan loop untuk menerima pesan dari client dan mengembalikan file yang direquest, terdapat penanganan error 404 apabila tidak ada file yang diminta.



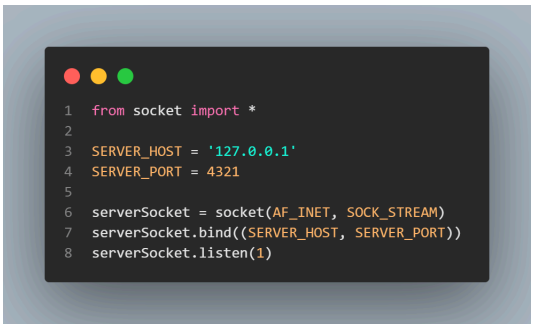
```python
def handle_client(connectionSocket, addr):
    print("Terhubung dengan ", addr)

    try:
        while True:
            message = connectionSocket.recv(1024).decode().strip()
            if message == 'exit':
                print("Koneksi client tertutup")
                break

            try:
                filename = message.split()[1][1:]
                with open(filename, encoding='utf-8') as f:
                    content = f.read()
                response = "HTTP/1.1 200 OK\r\n\r\n" + content

            except FileNotFoundError:
                response = "HTTP/1.1 404 Not Found\r\n\r\n"

            connectionSocket.send(response.encode())

    except Exception as e:
        print(f"Error: {e}")

    finally:
        connectionSocket.close()
        print("Koneksi tertutup")
```

Akhir code adalah loop untuk menerima connectionSocket dan alamat client dan menjalankan ‘handle_client’.

```python
while True:
    connectionSocket, addr = serverSocket.accept()
    handle_client(connectionSocket, addr)
```

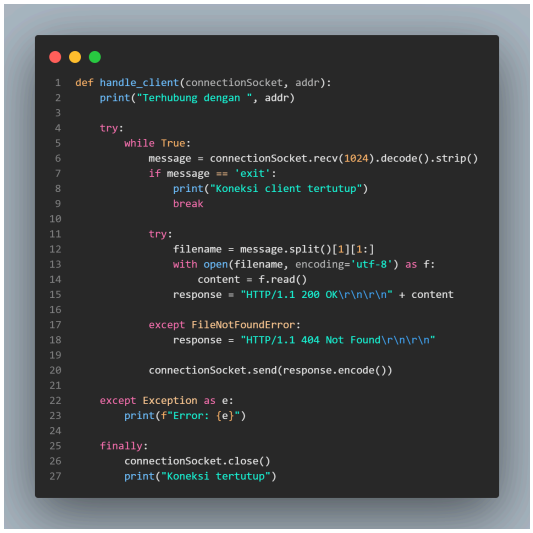
2. **Server Multi / Threading**

Kode server diawali dengan import library socket dan threading. Selanjutnya, dilakukan inisialisasi SERVER_HOST dan SERVER_PORT dengan alamat IP lokal dan port tertentu. Objek serverSocket dibentuk menggunakan socket(AF_INET, SOCK_STREAM) untuk koneksi TCP, lalu dilakukan proses bind dan listen dengan nilai 5, yang berarti server dapat mengantre hingga 5 koneksi masuk secara bersamaan.



```python
from socket import *

SERVER_HOST = '127.0.0.1'
SERVER_PORT = 4321

serverSocket = socket(AF_INET, SOCK_STREAM)
serverSocket.bind((SERVER_HOST, SERVER_PORT))
serverSocket.listen(1)
```

Dilanjutkan dengan mendefinisikan prosedur 'handle_client' dengan parameter socket dan address user, prosedur ini mirip dengan server single, yaitu dengan melakukan loop untuk menerima pesan dari client dan mengembalikan file yang direquest, terdapat penanganan error 404 apabila tidak ada file yang diminta.



```python
def handle_client(connectionSocket, addr):
    print("Terhubung dengan ", addr)

    try:
        while True:
            message = connectionSocket.recv(1024).decode().strip()
            if message == 'exit':
                print("Koneksi client tertutup")
                break

            try:
                filename = message.split()[1][1:]
                with open(filename, encoding='utf-8') as f:
                    content = f.read()
                response = "HTTP/1.1 200 OK\r\n\r\n" + content

            except FileNotFoundError:
                response = "HTTP/1.1 404 Not Found\r\n\r\n"

            connectionSocket.send(response.encode())

    except Exception as e:
        print(f"Error: {e}")

    finally:
        connectionSocket.close()
        print("Koneksi tertutup")
```

Bagian akhir kode menggunakan loop while True untuk menerima koneksi klien. Setiap koneksi diproses dalam thread baru dengan memanggil fungsi handle_client, sehingga server dapat menangani banyak klien secara bersamaan.

```python
while True:
    connectionSocket, addr = serverSocket.accept()
    thread = threading.Thread(target=handle_client, args=(connectionSocket, addr))
    thread.start()
```

3. **Client**

Kode ini mengimpor seluruh fungsi dari modul socket dan sys. Kemudian, program memeriksa apakah argumen yang dimasukkan dari command line berjumlah dua (alamat host dan port server). Jika tidak sesuai, program akan keluar. Nilai SERVER_HOST dan SERVER_PORT diambil dari argumen tersebut. Selanjutnya, socket TCP dibuat dengan socket(AF_INET, SOCK_STREAM) dan diberi batas waktu (timeout) selama 10 detik untuk mencegah program menunggu respons terlalu lama dari server.

```python
from socket import *
import sys

if len(sys.argv) != 3:
    print("Penggunaan: <client.py> <server_host> <server_port>")
    sys.exit(1)

SERVER_HOST = sys.argv[1]
SERVER_PORT = int(sys.argv[2])

clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.settimeout(10)
```

Bagian ini mencoba menghubungkan klien ke server menggunakan connect(). Jika berhasil, pengguna dapat memasukkan nama file yang ingin diminta. Permintaan dikirim dalam format HTTP GET, dan server diharapkan merespons dengan isi file. Data diterima secara bertahap (chunk) menggunakan recv(). Jika tidak ada data atau koneksi berhenti, pengambilan dihentikan. Jika terjadi kesalahan selama koneksi, program akan menangkap dan mencetak pesan error. Akhirnya, socket ditutup di blok finally.

```python
try:
    clientSocket.connect((SERVER_HOST, SERVER_PORT))
    print("Terhubung dengan Server")

    while True:
        filename = input("Isi nama file (atau 'exit'): ")

        if not filename:
            continue
        if filename == "exit":
            break

        request = f"GET /{filename} HTTP/1.0\r\nHost: {SERVER_HOST}\r\n\r\n"
        clientSocket.send(request.encode())

        response = ""
        while True:
            try:
                chunk = clientSocket.recv(4096).decode()
                if not chunk:
                    break
                response += chunk
            except timeout:
                break

        print(response)

except Exception as e:
    print(f"Error: {e}")

finally:
    clientSocket.close()
```

## C. Running

### 1. Running Server Single

#### a) Running Code Server

Server single berhasil running dan terhubung dengan 1 client.

```
D:\Kuliah\Matkul\Semester 4\JARINGAN KOMPUTER (JARKOM)\[2] Tugas\Tugas Besar\Jaringan-Komputer-LAB_Tugas-Besar>python server_single.py
Terhubung dengan  ('127.0.0.1', 56601)
```

#### b) 1 Command Prompt

Server single berhasil terhubung dengan client dan mengirimkan respons ke client. Selanjutnya, client menampilkan respons dari server single.

```
D:\Kuliah\Matkul\Semester 4\JARINGAN KOMPUTER (JARKOM)\[2] Tugas\Tugas Besar\Jaringan-Komputer-LAB_Tugas-Besar>python client.py 127.0.0.1 4321
Terhubung dengan Server
Isi nama file (atau 'exit'): 1.html
HTTP/1.1 200 OK
```

```
            <div class="container">
                <h1 class="flash-success">SELAMAT WEB SERVER ANDA BISA JALAN 🎉👏</h1>
                <p>- STO</p>
                <div class="emoji">😊</div>
            </div>
        `;
    }
</script>
</body>
</html>
Isi nama file (atau 'exit'):
```

#### c) 2 Command Prompt

Server single hanya berhasil terhubung dengan satu client, sehingga client 1 menerima respons dari server, sedangkan client 2 berhasil terhubung akan tetapi tidak menerima respons.

```
D:\Kuliah\Matkul\Semester 4\JARINGAN KOMPUTER (JARKOM)\[2] Tugas\Tugas Besar\Jaringan-Komputer-LAB_Tugas-Besar>python client.py 127.0.0.1 4321
Terhubung dengan Server
Isi nama file (atau 'exit'): 1.html
HTTP/1.1 200 OK
```

```
            <div class="container">
                <h1 class="flash-success">SELAMAT WEB SERVER ANDA BISA JALAN 🎉👏</h1>
                <p>- STO</p>
                <div class="emoji">😊</div>
            </div>
        `;
    }
</script>
</body>
</html>
Isi nama file (atau 'exit'):
```

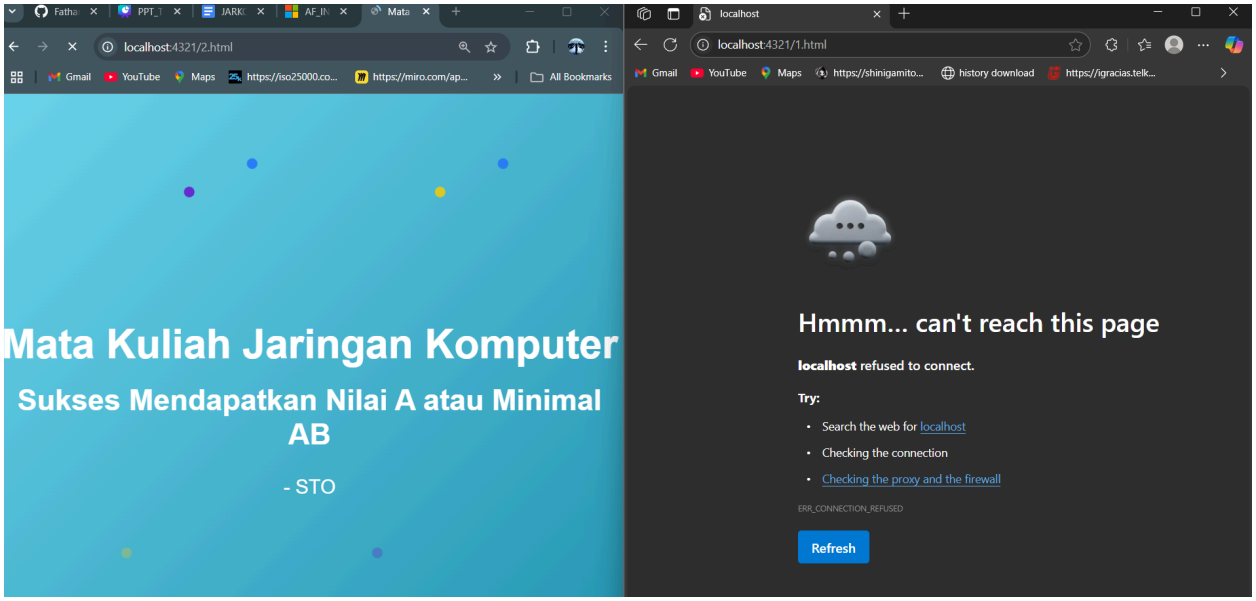
```
D:\Kuliah\Matkul\Semester 4\JARINGAN KOMPUTER (JARKOM)\[2] Tugas\Tugas Besar\Jaringan-Komputer-LAB_Tugas-Besar>python client.py 127.0.0.1 4321
Terhubung dengan Server
Isi nama file (atau 'exit'): 1.html

Isi nama file (atau 'exit'):
```

**d) 1 Website**

Server single berhasil terhubung dengan client dan mengirimkan respons ke client. Selanjutnya, client menampilkan respons dari server single pada website chrome.

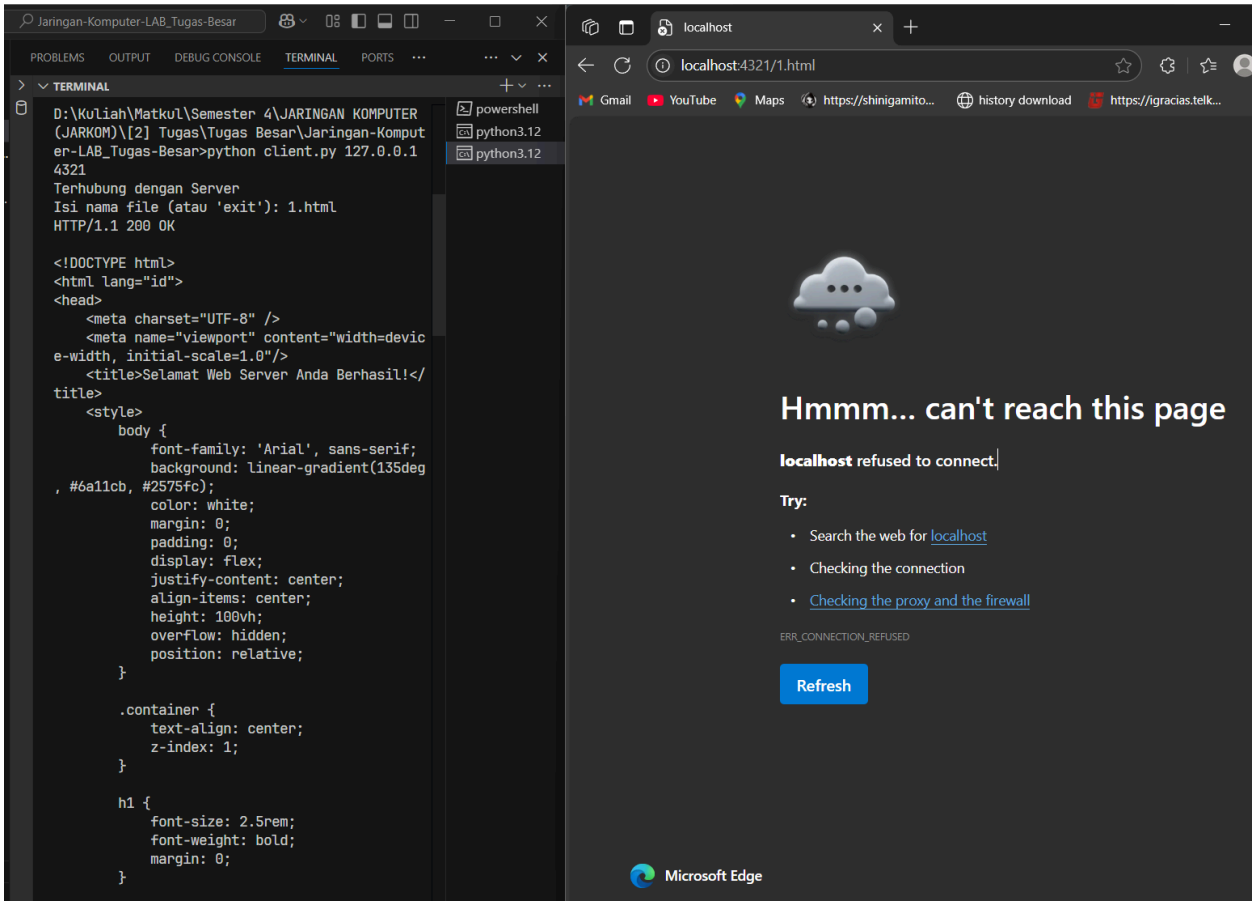

**e) 2 Website**

Server single hanya berhasil terhubung dengan satu client, sehingga client 1 menerima respons dari server dan dapat ditampilkan pada website chrome, sedangkan client 2 berhasil terhubung akan tetapi tidak menerima respons sehingga tidak dapat ditampilkan pada website edge.



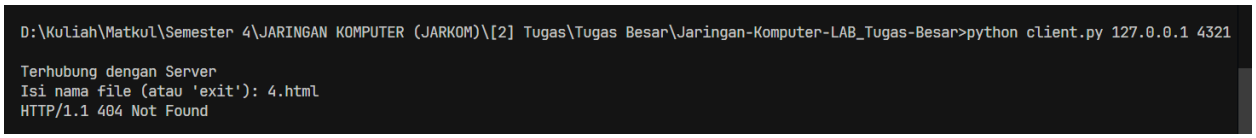
**f) 1 Command Prompt dan 1 Website**

Server single berhasil terhubung dengan client melalui Command Prompt dan menampilkan file 1.html. Namun, website tidak dapat menampilkan file tersebut karena server hanya memberikan respons ke client yang terhubung di Command Prompt.




**g) File Tidak ditemukan**

Server single mengembalikan response "HTTP/1.1 404 Not Found" pada client karena file 4.html tidak ada ditemukan.



2. **Running Server Multi**

a) **Running Code Server**

Server berhasil hidup dan bisa menerima pesan dari client.

```
PS D:\github\Jarkom\Jaringan-Komputer-LAB_Tugas-Besar> python server_multi.py
Terhubung dengan  ('127.0.0.1', 52940)
Terhubung dengan  ('127.0.0.1', 52941)
```

b) **1 Command Prompt**

Server single berhasil terhubung dengan client dan mengirimkan respons ke client.

```
PS D:\github\Jarkom\Jaringan-Komputer-LAB_Tugas-Besar> python client.py 127.0.0.1 1234
Terhubung dengan Server
Isi nama file (atau 'exit'): 1.html
HTTP/1.1 200 OK

<!DOCTYPE html>
```

c) **2 Command Prompt**

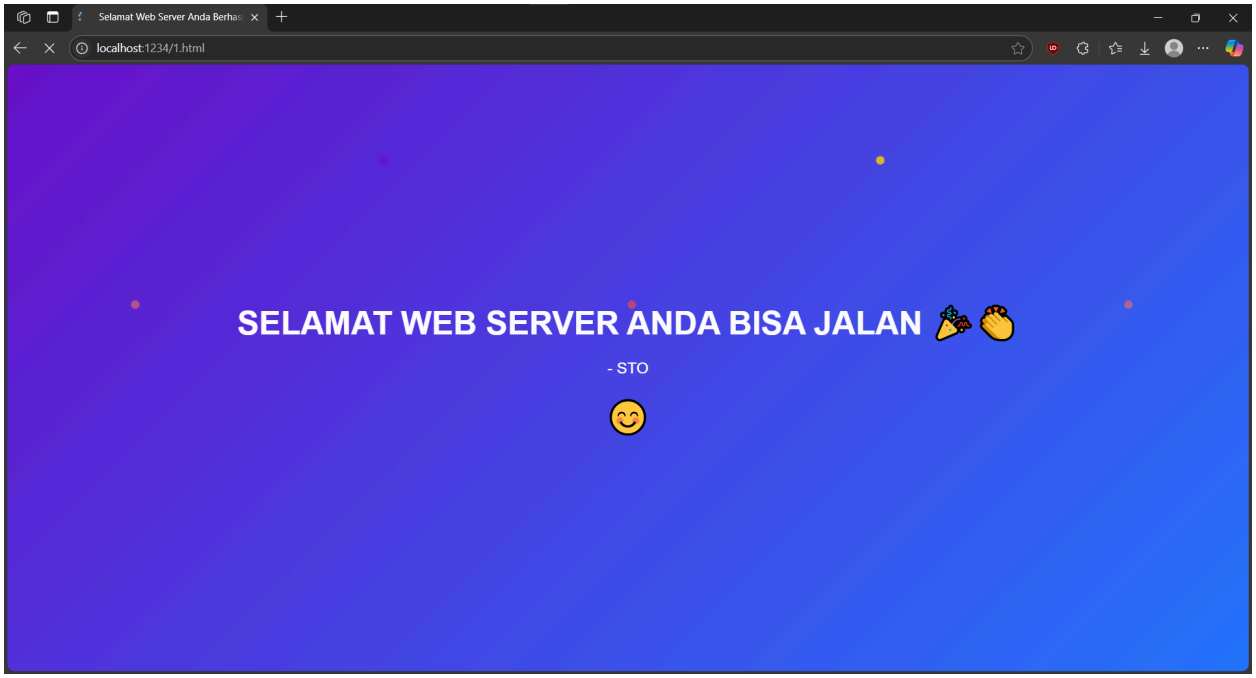
Server multi berhasil terhubung ke 2 client secara parallel tanpa perlu mematikan salah satu client.

```
PS D:\github\Jarkom\Jaringan-Komputer-LAB_Tugas-Besar> python client.py 127.
Terhubung dengan Server
Isi nama file (atau 'exit'): 1.html
HTTP/1.1 200 OK

<!DOCTYPE html>
<html lang="id">
```

```
PS D:\github\Jarkom\Jaringan-Komputer-LAB_Tugas-Besar> python client.py 127.
0.0.1 1234
Terhubung dengan Server
Isi nama file (atau 'exit'): 1.html
HTTP/1.1 200 OK

<!DOCTYPE html>
```

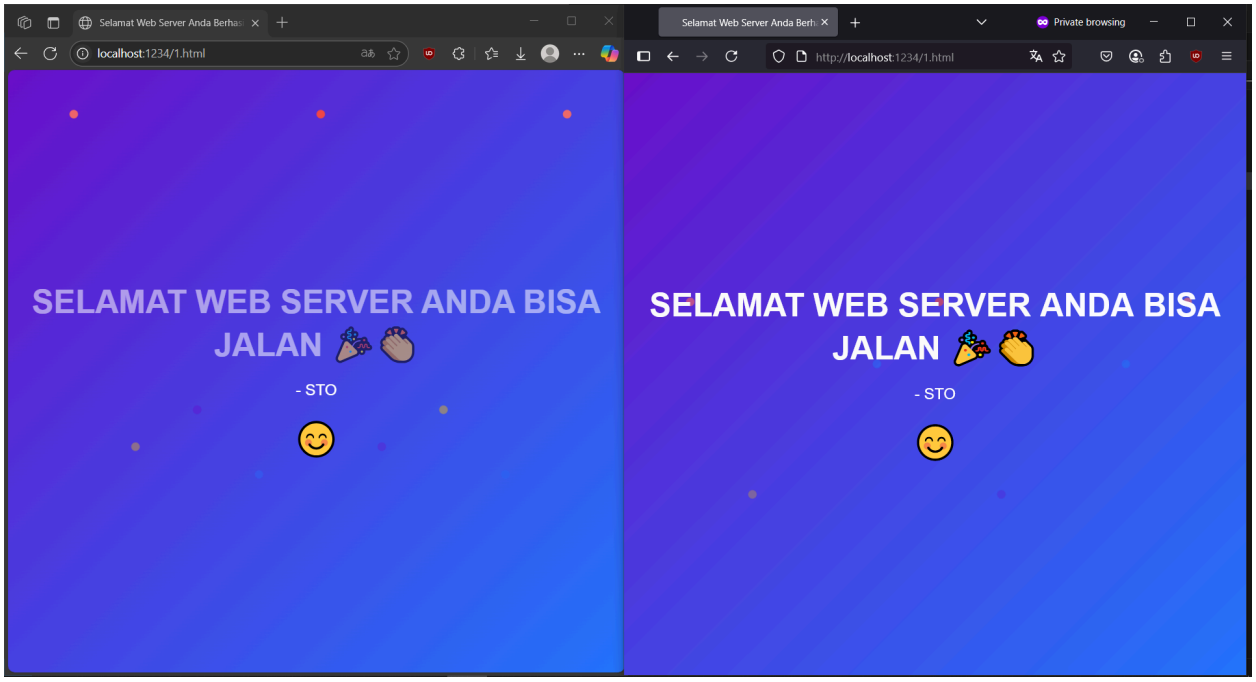
d) **1 Website**

Ketika menggunakan browser server multi berhasil terhubung dan merespon pesan ke client.



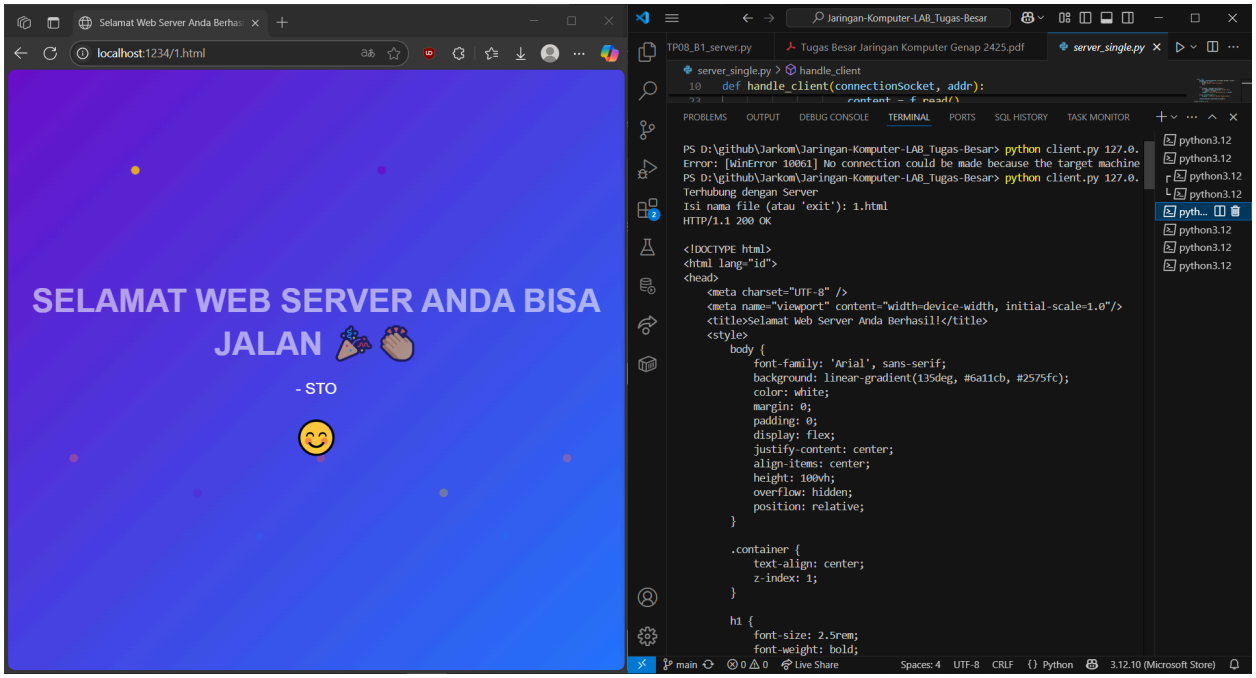

**e) 2 Website**

Ketika menggunakan browser 2 client server multi berhasil berjalan secara parallel tanpa perlu menutup 1 browser.




**f) 1 Command Prompt dan 1 Website**

Ketika menggunakan command prompt dan browser secara bersamaan server multi tetap bisa merespon ke 2 client secara parallel.




**g) File Tidak Ditemukan**

Server multi berhasil mengembalikan pesan “”HTTP/1.1 404 Not Found” ketika client meminta file yang tidak ada.

```
PS D:\github\Jarkom\Jaringan-Komputer-LAB_Tugas-Besar> python client.py 127.0.
0.1 1234
Terhubung dengan Server
0.1 1234
Terhubung dengan Server
Terhubung dengan Server
Isi nama file (atau 'exit'): 10.html
HTTP/1.1 404 Not Found


Isi nama file (atau 'exit'):
```

## III. Kesimpulan

Server single hanya mampu menangani satu klien dalam satu waktu, sedangkan server multi (threading) dapat melayani banyak klien secara bersamaan. Laporan ini mengeksplorasi perbedaan server dengan dan tanpa threading, serta mengetahui bahwa socket programming menggunakan protokol TCP dapat memenuhi kebutuhan GET, baik melalui Command Prompt maupun browser, serta mampu menangani permintaan file yang tidak ditemukan dengan mengembalikan response error 404 secara tepat.

## IV. Referensi

Informatics Lab, School of Computing, Telkom University (2025). Modul Praktikum Jaringan Komputer S1 Informatika.

Kurose, J. F., & Ross, K. W. (2022). Computer networking: A top-down approach (8th ed.). Pearson Education.