LAPORAN TUGAS BESAR
JARINGAN KOMPUTER

Disusun Oleh Kelompok 13 :

Annisaa Alya Hafiza (1301213058)

Rexcel Pabian (1301210144)

Muhamad Khoir Fahni Nur Islami (1301213107)

PROGRAM STUDI INFORMATIKA

UNIVERSITAS TELKOM

KAB. BANDUNG

2023

Link Video Pemaparan

https://drive.google.com/drive/folders/1ushZ8Cf0IiguOYSNj3Xv5fNEaZ5LnYDE?usp=share_link

Didalamnya termasuk video penjelasan codingan dan video demo aplikasi

A. Latar Belakang

Dalam era digital yang terus berkembang, pengembangan aplikasi web menjadi sangat penting karena memberikan kemampuan bagi pengguna untuk berkomunikasi, berbagi informasi, dan berinteraksi melalui internet. Salah satu komponen kunci dalam pengembangan aplikasi web adalah web server.

Web server merupakan perangkat lunak yang berjalan di server dan bertugas melayani permintaan HTTP dari klien. Dalam prosesnya, web server menerima permintaan dari klien melalui protokol TCP/IP dan mengirimkan respons yang sesuai. Dalam pengembangan web server, salah satu pendekatan yang umum digunakan adalah TCP socket programming.

TCP socket programming adalah teknik yang memungkinkan aplikasi untuk berkomunikasi melalui jaringan menggunakan protokol TCP. Dalam konteks web server, TCP socket programming memungkinkan pengembang untuk membuat server yang dapat menerima permintaan HTTP dari klien dan memberikan respons yang tepat.

Membangun program web server sederhana berbasis TCP socket programming memiliki beberapa keuntungan. Pertama, pemahaman tentang TCP socket programming memungkinkan pengembang memiliki kontrol penuh terhadap proses komunikasi antara server dan klien. Kedua, penggunaan protokol TCP memastikan bahwa data yang dikirimkan antara server dan klien tiba secara aman dan integritasnya terjaga.

B. Batasan Masalah

Dalam pengembangan program web server sederhana berbasis TCP socket programming, terdapat beberapa batasan yang perlu diperhatikan. Batasan-batasan ini mempengaruhi cakupan dan fokus pengembangan program, serta memberikan kerangka kerja yang jelas untuk mencapai tujuan yang telah ditetapkan. Berikut adalah batasan-batasan masalah yang relevan:

1. Implementasi pembuatan TCP socket dan menghubungkannya dengan alamat dan port tertentu:

   Dalam persyaratan ini, program web server harus dapat mengimplementasikan proses pembuatan TCP socket yang sukses dan kemudian menghubungkannya dengan alamat dan port yang telah ditentukan. Hal ini melibatkan penggunaan

metode dan fungsi yang sesuai untuk membuka socket, mengaitkannya dengan alamat dan port yang benar, dan memastikan koneksi yang stabil dan dapat beroperasi.

2. Kemampuan program web server untuk menerima dan memparsing HTTP request yang dikirimkan oleh browser:

   Persyaratan ini mencakup kemampuan program web server untuk menerima permintaan HTTP yang dikirimkan oleh browser klien. Program harus dapat membaca data yang dikirimkan dalam permintaan, seperti metode HTTP, URL, header, dan data lainnya. Selanjutnya, program harus dapat memparsing data tersebut dengan benar dan mengidentifikasi informasi yang diperlukan untuk merespons permintaan dengan tepat.

3. Web server dapat mencari dan mengambil file yang diminta oleh klien dari sistem file:

   Persyaratan ini mengharuskan web server memiliki kemampuan untuk mencari file yang diminta oleh klien dalam sistem file server. Program harus dapat menentukan lokasi file yang diminta berdasarkan URL yang diterima, membaca file dari sistem file dengan benar, dan memastikan bahwa file yang diminta siap untuk dikirimkan sebagai respons kepada klien.

4. Web server dapat membuat pesan HTTP response yang terdiri dari header dan konten file yang diminta:

   Persyaratan ini melibatkan kemampuan web server untuk membuat pesan respons HTTP yang sesuai dengan spesifikasi protokol HTTP. Pesan respons harus terdiri dari header yang lengkap, termasuk kode status HTTP yang relevan, informasi tambahan jika diperlukan, dan konten file yang diminta oleh klien. Program harus mampu menyusun pesan respons dengan benar dan memastikan kelengkapan dan ketepatan header serta konten yang dikirimkan.

5. Web server dapat mengirimkan pesan response yang telah dibuat ke browser klien dan menampilkannya dengan benar di sisi klien:

   Setelah pesan respons HTTP dibuat, program web server harus dapat mengirimkannya kepada browser klien melalui koneksi TCP yang telah terbentuk. Pesan respons harus dikirim dengan sukses dan diterima oleh klien. Selain itu,

klien juga harus dapat menampilkan pesan respons dengan benar di sisi pengguna, termasuk menampilkan konten file yang diminta secara sesuai.

6. Jika file yang diminta oleh klien tidak tersedia, web server dapat mengirimkan pesan "404 Not Found" dan menampilkannya dengan benar di sisi klien:

   Persyaratan ini berhubungan dengan penanganan situasi ketika web server tidak dapat menemukan file yang diminta oleh klien. Program harus memiliki mekanisme yang dapat menghasilkan pesan "404 Not Found" yang ses

C. Sistem yang dibangun

Sistem yang dibangun adalah program web server sederhana berbasis TCP socket programming. Tujuannya adalah menerima permintaan HTTP dari browser klien, mencari dan mengambil file yang diminta, serta mengirimkan respons HTTP yang tepat kembali kepada klien.

Pada tahap implementasi, program membuat TCP socket dan mengaitkannya dengan alamat dan port tertentu. Ini memungkinkan program untuk menerima koneksi dari browser klien melalui protokol TCP/IP.

Ketika menerima permintaan HTTP dari klien, program memparsing permintaan tersebut untuk mendapatkan informasi seperti metode HTTP, URL, dan header. Informasi ini digunakan untuk menentukan file yang diminta dari sistem file server.

Selanjutnya, program mencari dan mengambil file yang diminta. Jika file ditemukan, program membuat respons HTTP yang terdiri dari header dan konten file.

Setelah respons dibuat, program mengirimkannya kembali kepada klien melalui koneksi TCP yang sudah terbentuk. Klien menerima respons dan menampilkan konten file yang diminta.

Jika file tidak ditemukan, program mengirim pesan "404 Not Found" kepada klien, yang ditampilkan sesuai protokol HTTP.

Sistem ini memungkinkan program web server sederhana untuk menerima permintaan HTTP, mencari dan mengambil file, serta mengirimkan respons yang tepat kepada klien.

Berikut adalah alur kerja dari program web server sederhana berbasis TCP socket programming yang dibangun:

1. Program web server dimulai dengan membuat TCP socket dan mengaitkannya dengan alamat dan port tertentu.
2. Program web server siap menerima koneksi dari browser klien melalui protokol TCP/IP.
3. Ketika browser klien mengirim permintaan HTTP, program web server menerima permintaan tersebut melalui koneksi TCP yang terbentuk.
4. Program web server memparsing permintaan HTTP untuk mendapatkan informasi seperti metode HTTP, URL, dan header.
5. Berdasarkan informasi yang diperoleh, program web server menentukan file yang diminta oleh klien dari sistem file server.
6. Jika file ditemukan, program web server membaca isi file dari sistem file server.
7. Program web server membuat pesan HTTP response yang terdiri dari header dan konten file yang diminta.
8. Pesan response yang telah dibuat dikirimkan kembali kepada browser klien melalui koneksi TCP.
9. Browser klien menerima respons HTTP dari web server.
10. Jika respons HTTP berisi konten file, browser klien menampilkan konten file tersebut di antarmuka pengguna.
11. Jika file yang diminta tidak ditemukan, program web server menghasilkan pesan "404 Not Found" dan mengirimkannya kembali kepada browser klien.
12. Browser klien menampilkan pesan "404 Not Found" sesuai dengan spesifikasi protokol HTTP.

Dengan alur kerja ini, program web server sederhana dapat berfungsi untuk menerima permintaan HTTP, mencari dan mengambil file yang diminta, serta mengirimkan respons yang sesuai kepada browser klien.

D. Hasil program dan analisis

Berikut adalah program yang telah kami buat :

```
1  import socket
2  import threading
```

Potongan kode diatas adalah proses import library socket dan os

```
5  def handle_connection(connectionSocket):
6      message = connectionSocket.recv(1024).decode()
7      print(message)
8
9      # melakukan parsing request untuk klien
10     request_method = message.split()[0]
11     file_requested = message.split()[1]
```

Potongan kode diatas adalah proses pembuatan fungsi handle_connection dan melakukan parsing request untuk client.

```
13     # file requestnya adalah indefix.html
14     if file_requested == '/':
15         file_requested = '/indexfix.html'
```

Mengkondisikan file_requested dengan indexfix.html agar indexfix.html menjadi deafult ketika server berjalan.

```
17     # penggunaan content type untuk mapping file extension
18     content_types = {
19         'png': 'image/png',
20         'txt': 'text/plain',
21         'jpeg': 'image/jpeg',
22         'js': 'application/javascript',
23         'video': 'video/mp4',
24         'scss': 'text/css',
25         'html': 'text/html',
26         'css': 'text/css',
27         'jpg': 'image/jpeg',
28     }
```

Membuat dictionary untuk memetakan ekstensi file menggunakan content_type

```python
    # parsing (membuka file requestan client)
    try:
        file_extension = file_requested.split('.')[-1]
        with open(file_requested[1:], 'rb') as file:
            file_content = file.read()

        # pencarian content type sesuai file ekstensinya
        content_type = content_types.get(
            file_extension, 'application/octet-stream')

        # pencarian http response berdasarkan header dan file yang diperlukan
        response_header = f'HTTP/1.1 200 OK\r\nContent-Type: {content_type}\r\n\r\n'
        print(response_header)
        response_content = file_content

        # pengiriman http response ke client
        connectionSocket.send(response_header.encode())
        connectionSocket.send(response_content)
```

Potongan kode diatas adalah kode untuk melakukan parsing (membuka file yang diminta oleh client). Dimana file_requested akan di split/partisi menjadi beberapa bagian sederhana, agar tidak mengalami kesulitan saat melakukan pencarian di file library.
Ketika server telah menemukan apa yang dibutuhkan oleh klien, selanjutnya server akan membuka file_requested tersebut dan nantinya file tersebut akan ditinjau ulang oleh server dibagian konten type yang tersedia.

Selanjutnya dengan content_type.get server akan melakukan pencarian konten type berdasarkan file ekstensinya. Ketika server tekah melakukan respons terhadap klien, bukti response tersebut nantinya akan dialokasikan sebagai response header dan response konten. Selanjutnya response header dan response konten akan dikirimkan dengan menggunakan connection socket.

```
    # pengiriman pesan 404 not found
    except IOError as error:
        with open('404.html', 'rb') as file:
            file_content = file.read()

        response_header = 'HTTP/1.1 404 Not Found\r\nContent-Type: text/html\r\n\r\n'
        response_content = file_content
        print(error)

        # pengiriman pesan 404 not found
        connectionSocket.send(response_header.encode())
        connectionSocket.send(response_content)
    # tutup koneksi soket
    connectionSocket.close()
```

jika yang diminta oleh client tidak tersedia dialam file library, maka akan diarahkan ke 404.html dan response dari web server ke client yaitu menyatakan bahawa inputan tersebut not found atau yang bisa dilihat 404 not found dan bukti dari response dari web server tersbut juga akan dikirmkan ke client melalui conection socket

```
# buat TCP socket
serverSocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# aktivasi SO_REUSEADDR
serverSocket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)

# aktivasi SO_KEEPALIVE
serverSocket.setsockopt(socket.SOL_SOCKET, socket.SO_KEEPALIVE, 1)

# menghubungkan soket ke lokalhost dan port 80
serverHost = 'localhost'  # alamat IP lokal
serverPort = 80  # port yang digunakan

# menggabungkan host dan port (binding)
serverSocket.bind((serverHost, serverPort))

# digunakan untuk menerima koneksi dari client
serverSocket.listen(1)
```

Potongan kode diatas adalah kode untuk Membuat TCP socket dengan parameter AF_INET dan SOCK_STREAM, aktivasi SO_REUSEADOR dan SO_KEEPALIVE.
Selanjutnya kita akan menghubungkan soket ke lokalhost dan port 80.
Gunakan **serverSocket.listen(1)** untuk menerima koneksi internet dari klien.

```
print(f'Klik disini http://{serverHost}:{serverPort}/')
while True:
    # digunakan untuk nerima koneksi klien
    connectionSocket, addr = serverSocket.accept()

    # buat thread baru
    t = threading.Thread(target=handle_connection, args=(connectionSocket,))
    t.start()

```

Untuk web server ini kami mengusahakan untuk bisa multithreading, kami mengusung ide untuk membuat follow link yang kami rencanakan akan dapat mempermudah untuk mencapai tujuan multithreading.

E. Kesimpulan

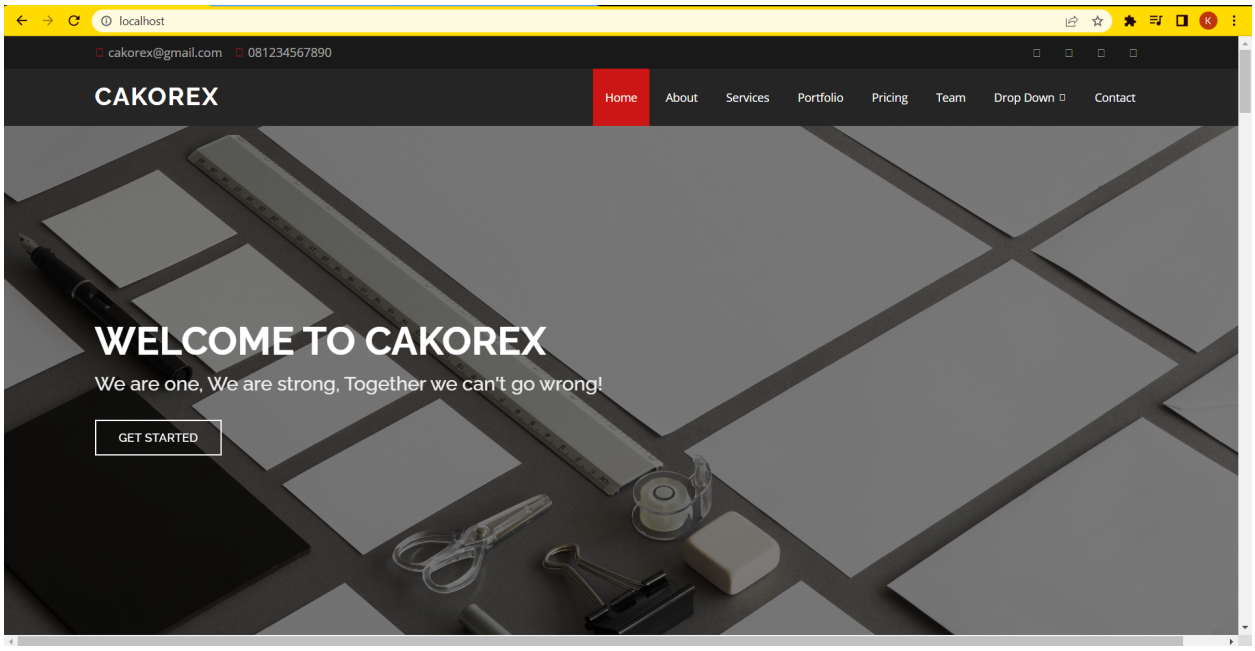
```
PS C:\belajar woi\yok ah\TUBESJARKOM ICA KORI REXCEL\Day (1)\Day> py cakorex.py
Klik disini http://localhost:80/
GET / HTTP/1.1
Host: localhost
Connection: keep-alive
Cache-Control: max-age=0
sec-ch-ua: "Google Chrome";v="113", "Chromium";v="113", "Not-A.Brand";v="24"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Windows"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/113.0.0.0 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9,es;q=0.8


HTTP/1.1 200 OK
Content-Type: text/html


GET /assets/vendor/aos/aos.css HTTP/1.1
Host: localhost
Connection: keep-alive
sec-ch-ua: "Google Chrome";v="113", "Chromium";v="113", "Not-A.Brand";v="24"
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/113.0.0.0 Safari/537.36
sec-ch-ua-platform: "Windows"
Accept: text/css,*/*;q=0.1
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: no-cors
Sec-Fetch-Dest: style
Referer: http://localhost/
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9,es;q=0.8


HTTP/1.1 200 OK
Content-Type: text/css


GET /assets/vendor/bootstrap-icons/bootstrap-icons.css HTTP/1.1
Host: localhost
Connection: keep-alive
sec-ch-ua: "Google Chrome";v="113", "Chromium";v="113", "Not-A.Brand";v="24"
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/113.0.0.0 Safari/537.36
sec-ch-ua-platform: "Windows"
```



404 Not Found

Referensi

https://www.youtube.com/watch?v=hFNZ6kdBgO0

https://www.youtube.com/watch?v=kogOfxg1c_g

Modul 8 Jaringan Komputer

Chat GPT dengan keyword “content type mapping dengan python”

Chat GPT dengan keyword “penjelasan web server”

Chat GPT dengan keyword “aktivasi SO_REUSEADDR pada web server python”

Chat GPT dengan keyword “aktivasi SO_KEEPALIVE pada web server python”

Kontribusi Anggota

Nama	NIM	Kontribusi
Annisaa Alya Hafiza	1301213058	• Membuat web server • Membuat HTML • Membuat laporan
Rexcel Pabian	1301210144	• Membuat web server • Membuat HTML • Membuat laporan
Muhamad Khoir Fahni Nur Islami	1301213107	• Membuat web server • Membuat HTML • Membuat laporan