LAPORAN PEMBUATAN APLIKASI CHATTING

|  |  |  |
| --- | --- | --- |
| NAMA | : | TEDY WIBISONO |
| KELAS | : | B |
| NIM | : | 202151063 |
| MATKUL | : | PRATIKUM PEMROGAMAN JARINGAN |

**Deskripsi aplikasi**

Laporan ini membahas penggunaan socket dan threading untuk menciptakan aplikasi chatting real-time sederhana antara server dan klien-klien. Setiap klien dapat mengirim dan menerima pesan secara bersamaan, memberikan pengalaman pengguna yang interaktif. Penggunaan tkinter memperindah antarmuka pengguna dan meningkatkan daya interaktif aplikasi. Aplikasi ini cocok sebagai dasar untuk pengembangan aplikasi komunikasi real-time yang lebih kompleks.

**Library yang di gunakan**

Dalam pembuatan aplikasi chatting ini kita harus mengimport dua modul utama yaitu ‘socket’ dan ‘threading’. socket digunakan untuk berkomunikasi melalui jaringan, dan threading digunakan untuk menangani multiple klien secara bersamaan. Tkinter di gunakan untuk membuat gui di chat.

**Pembuatan Aplikasi**

Pembuatan aplikasi chatting realtime menggunakan library socket, tkinter dan threading

1. **File Server.py**
2. Import Libraries

*import* socket

*import* threading

1. Inisialisasi Server Socket

server\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

Sebuah objek socket dibuat dengan menggunakan kode di atas ini mendefinisikan socket sebagai IPv4 dan menggunakan protokol TCP.

1. Menentukan Alamat dan Port Server

server\_address = ('localhost', 12345)

Alamat dan port server ditentukan dengan mendefinisikan server\_address sebagai tupel yang berisi nama host (localhost) dan nomor port (12345).

1. Mengikat Server ke Alamat dan Port

Server socket diikat ke alamat dan port yang telah ditentukan menggunakan bind().

server\_socket.bind(server\_address)

1. Mendengarkan Koneksi Masuk

Server socket mulai mendengarkan koneksi masuk dengan menggunakan listen(5). Angka 5 menunjukkan jumlah koneksi yang diizinkan dalam antrian.

server\_socket.listen(5)

1. Membuat daftar client

Sebuah dictionary connected\_clients dibuat untuk melacak klien yang terhubung bersama dengan nama file mereka.

connected\_clients = {}

1. Fungsi handle\_client:

* Fungsi ini digunakan untuk menangani koneksi dari setiap klien.

handle\_client(client\_socket)

bertanggung jawab untuk menangani koneksi dari setiap klien yang terhubung ke server.

* Menerima nama file dari klien.

Fungsi ini pertama-tama menerima nama file dari klien dengan menggunakan recv(1024) untuk menerima data dengan ukuran buffer 1024 bytes.

Nama file klien kemudian di-decode menggunakan decode().

client\_name = client\_socket.recv(1024).decode()

* Menambahkan klien ke daftar klien yang terhubung.

Setelah menerima nama file, fungsi menambahkan informasi klien (nama file) ke dalam dictionary connected\_clients, dengan menggunakan socket klien sebagai kunci.

connected\_clients[client\_socket] = client\_name

* Loop Menerima dan Mengirim Pesan

Fungsi memasuki loop tak terbatas untuk terus menerima pesan dari klien. Jika pesan yang diterima tidak kosong, fungsi akan melanjutkan proses berikutnya.

*while* True:

message = client\_socket.recv(1024).decode()

*if* not message:

*break*

*for* client, \_ *in* connected\_clients.items():

*if* client != client\_socket:

sender\_name = connected\_clients[client\_socket]

client.send("{}: {}".format(sender\_name, message).encode())

* Mengirim Pesan ke Klien Lain

Fungsi menggunakan loop for untuk mengiterasi melalui daftar klien yang terhubung.

Pesan yang diterima dari satu klien dikirimkan kepada semua klien lainnya, termasuk informasi pengirim (nama file klien).

*for* client, \_ *in* connected\_clients.items():

*if* client != client\_socket:

sender\_name = connected\_clients[client\_socket]

client.send("{}: {}".format(sender\_name, message).encode())

* Penanganan Eksepsi

Terdapat blok try...except untuk menangani pengecualian yang mungkin terjadi selama proses. Jika terjadi kesalahan atau koneksi terputus, pesan kesalahan dicetak, dan koneksi dengan klien dianggap terputus.

*except* Exception *as* e:

print("Koneksi klien terputus: {}".format(e))

*finally*:

*del* connected\_clients[client\_socket]

client\_socket.close()

* Menghapus Klien dari Daftar Terhubung

Di bagian finally, klien dihapus dari daftar connected\_clients.Socket klien ditutup menggunakan client\_socket.close().

*finally*:

*del* connected\_clients[client\_socket]

client\_socket.close()

1. **File seller.py & customer.py**
2. Inisialisasi Koneksi

client\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

server\_address = ('localhost', 12345)

Membuat socket klien dan menghubungkannya ke alamat dan port server yang ditentukan.

1. Fungsi send\_message

def send\_message():

message = message\_entry.get()

*if* message:

*try*:

client\_socket.send(message.encode())

message\_entry.delete(0, END)

chat\_history.insert(tk.END, "Anda: " + message + '\n')

chat\_history.see(tk.END)

Mengambil pesan dari input teks dan mengirimkannya ke server menggunakan client\_socket.send().Menampilkan pesan yang dikirim ke area chat.

1. Fungsi receive\_message

def receive\_message():

*while* True:

*try*:

data = client\_socket.recv(1024).decode()

*if* not data:

*break*

chat\_history.insert(tk.END, data + '\n')

chat\_history.see(tk.END)

*except* ConnectionError:

print("Koneksi dengan server terputus.")

*break*

Menerima pesan dari server menggunakan client\_socket.recv() dalam loop tak terbatas.Menampilkan pesan yang diterima ke area chat.

1. Fungsi send\_client\_name

def send\_client\_name():

client\_name = os.path.basename(\_\_file\_\_)

client\_socket.send(client\_name.encode())

Mengirim nama file klien ke server menggunakan client\_socket.send().

1. Antarmuka GUI dengan tkinter

root = tk.Tk()

root.title("Cromboloni Cake")

root.geometry("550x650")

root.configure(bg='#FFC0CB')

client\_name\_label = tk.Label(

root, text="Tedy Wibisono: " + os.path.basename(\_\_file\_\_), font=("Arial", 12))

client\_name\_label.pack(pady=10)

chat\_history = scrolledtext.ScrolledText(

root, width=50, height=20, font=("Arial", 12))

chat\_history.pack(padx=10, pady=10)

message\_entry = tk.Entry(root, width=30, font=("Arial", 12))

message\_entry.pack(pady=10)

send\_button = tk.Button(

root, text="Kirim", command=send\_message, font=("Arial", 12))

send\_button.pack()

name\_thread = threading.Thread(target=send\_client\_name)

name\_thread.start()

receive\_thread = threading.Thread(target=receive\_message)

receive\_thread.start()

Membuat antarmuka pengguna dengan tkinter yang terdiri dari label, textarea, input teks, dan tombol untuk mengirim pesan.Menggunakan thread terpisah (name\_thread dan receive\_thread) untuk mengirim nama file klien dan menerima pesan dari server.

1. Loop utama

root.mainloop()

Memulai loop utama untuk menjalankan aplikasi GUI.