Project

Ujian Tengah Semester

Nama   : Abrar Ramadhan

Kelas   : 4IA10

Npm   : 50421015

Matkul : Pemrograman Jaringan

Source Code

➢ **fileclient.py**

```
import socket
import threading
import os
import tkinter as tk
from tkinter import ttk, filedialog, messagebox
from PIL import Image, ImageTk
import webbrowser
import subprocess
import sys

PORT = 5050
CHUNKSIZE = 4096
SEPARATOR = '<SEPARATOR>'

class FileClientApp:
    def __init__(self, master):
        self.master = master
        master.title('Client-Server File Transfer - Client')

        frame = ttk.Frame(master)
        frame.pack(padx=10, pady=5)
        ttk.Label(frame, text='Server IP:').grid(row=0, column=0)
        self.ip_entry = ttk.Entry(frame)
        self.ip_entry.insert(0, socket.gethostbyname(socket.gethostname()))
        self.ip_entry.grid(row=0, column=1, padx=5)
        ttk.Label(frame, text='Port:').grid(row=0, column=2)
        self.port_entry = ttk.Entry(frame, width=6)
        self.port_entry.insert(0, str(PORT))
        self.port_entry.grid(row=0, column=3, padx=5)
```

```python
        send_frame = ttk.LabelFrame(master, text='Send File to Server')
        send_frame.pack(fill='x', padx=10, pady=5)
        ttk.Button(send_frame, text='Choose File',
command=self.choose_file).pack(side='left', padx=5)
        self.file_label = ttk.Label(send_frame, text='No file selected')
        self.file_label.pack(side='left', padx=5)
        ttk.Button(send_frame, text='Send',
command=self.gui_send).pack(side='left', padx=5)

        recv_frame = ttk.LabelFrame(master, text='Request File from Server')
        recv_frame.pack(fill='x', padx=10, pady=5)
        ttk.Button(recv_frame, text='Refresh File List',
command=self.refresh_list).pack(side='left', padx=5)
        self.file_combo = ttk.Combobox(recv_frame)
        self.file_combo.pack(side='left', padx=5)
        ttk.Button(recv_frame, text='Receive',
command=self.gui_receive).pack(side='left', padx=5)

        self.progress = ttk.Progressbar(master, length=300)
        self.progress.pack(pady=10)
        self.status = ttk.Label(master, text='')
        self.status.pack()

        self.filename = None

    def choose_file(self):
        path = filedialog.askopenfilename()
        if path:
            self.filename = path
            self.file_label.config(text=os.path.basename(path))

    def gui_send(self):
        addr = (self.ip_entry.get(), int(self.port_entry.get()))
        threading.Thread(target=self.send_file, args=(addr, self.filename),
daemon=True).start()

    def send_file(self, addr, filepath):
        try:
            with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
                s.connect(addr)
                s.send(b'SEND')
                if s.recv(1024).decode().strip() == 'OK':
                    fname = os.path.basename(filepath)
                    size = os.path.getsize(filepath)
                    s.send(fname.encode())
                    s.send(f"{fname}{SEPARATOR}{size}".encode())
                    sent = 0
                    with open(filepath, 'rb') as f:
                        while chunk := f.read(CHUNKSIZE):
                            s.sendall(chunk)
                            sent += len(chunk)
                            self.progress['value'] = (sent/size)*100
                            self.status.config(text=f"Sent {sent}/{size}
bytes")
                    self.status.config(text='File sent successfully')
                    messagebox.showinfo('Success', f'File \"{fname}\" sent
successfully.')
        except Exception as e:
            messagebox.showerror('Error', str(e))

    def refresh_list(self):
        addr = (self.ip_entry.get(), int(self.port_entry.get()))
        try:
            with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
                s.connect(addr)
                s.send(b'LIST')
                data = s.recv(8192).decode()
                files = data.split('|') if data else []
                self.file_combo['values'] = files
                if files:
                    self.file_combo.set(files[0])
        except Exception as e:
            messagebox.showerror('Error', f'Failed to fetch file list: {e}')

    def gui_receive(self):
        addr = (self.ip_entry.get(), int(self.port_entry.get()))
        fname = self.file_combo.get().strip()
        threading.Thread(target=self.receive_file, args=(addr, fname),
daemon=True).start()

    def receive_file(self, addr, fname):
        try:
            with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
                s.connect(addr)
                s.send(b'RECEIVE')
                if s.recv(1024).decode().strip() == 'OK':
                    s.send(fname.encode())
                    resp = s.recv(1024).decode()
                    if not resp or resp.startswith('ERROR'):
                        messagebox.showerror('Error', resp or 'No response
from server')
                        return
                    name, size = resp.split(SEPARATOR)
                    size = int(size)
                    received = 0
```

```python
                    with open(name, 'wb') as f:
                        while received < size:
                            chunk = s.recv(CHUNKSIZE)
                            if not chunk: break
                            f.write(chunk)
                            received += len(chunk)
                            self.progress['value'] = (received/size)*100
                            self.status.config(text=f"Received
{received}/{size} bytes")
                    self.status.config(text='File received successfully')
                    messagebox.showinfo('Success', f'File \"{name}\" received
successfully.')
                    self.preview_file(name)
        except Exception as e:
            messagebox.showerror('Error', str(e))

    def preview_file(self, filepath):
        ext = os.path.splitext(filepath)[1].lower()
        try:
            if ext in ('.txt', '.py', '.csv', '.log'):
                with open(filepath, 'r', encoding='utf-8') as f:
                    content = f.read()
                win = tk.Toplevel(self.master)
                win.title(f"Preview - {filepath}")
                text_widget = tk.Text(win, wrap='word')
                text_widget.insert('1.0', content)
                text_widget.pack(expand=True, fill='both')

            elif ext in ('.png', '.jpg', '.jpeg', '.bmp', '.gif'):
                win = tk.Toplevel(self.master)
                win.title(f"Image Preview - {filepath}")
                img = Image.open(filepath)
                photo = ImageTk.PhotoImage(img)
                label = ttk.Label(win, image=photo)
                label.image = photo
                label.pack()

            elif ext == '.html':
                webbrowser.open(f"file://{os.path.abspath(filepath)}")

            elif ext == '.pdf':
                if sys.platform == 'win32':
                    os.startfile(filepath)
                elif sys.platform == 'darwin':
                    subprocess.call(('open', filepath))
                else:
                    subprocess.call(('xdg-open', filepath))
            else:
                messagebox.showinfo('Preview', 'Preview tidak tersedia untuk 
file ini.')
        except Exception as e:
            messagebox.showerror('Preview Error', f'Gagal menampilkan preview: 
{e}')

if __name__ == '__main__':
    root = tk.Tk()
    app = FileClientApp(root)
    root.mainloop()
```

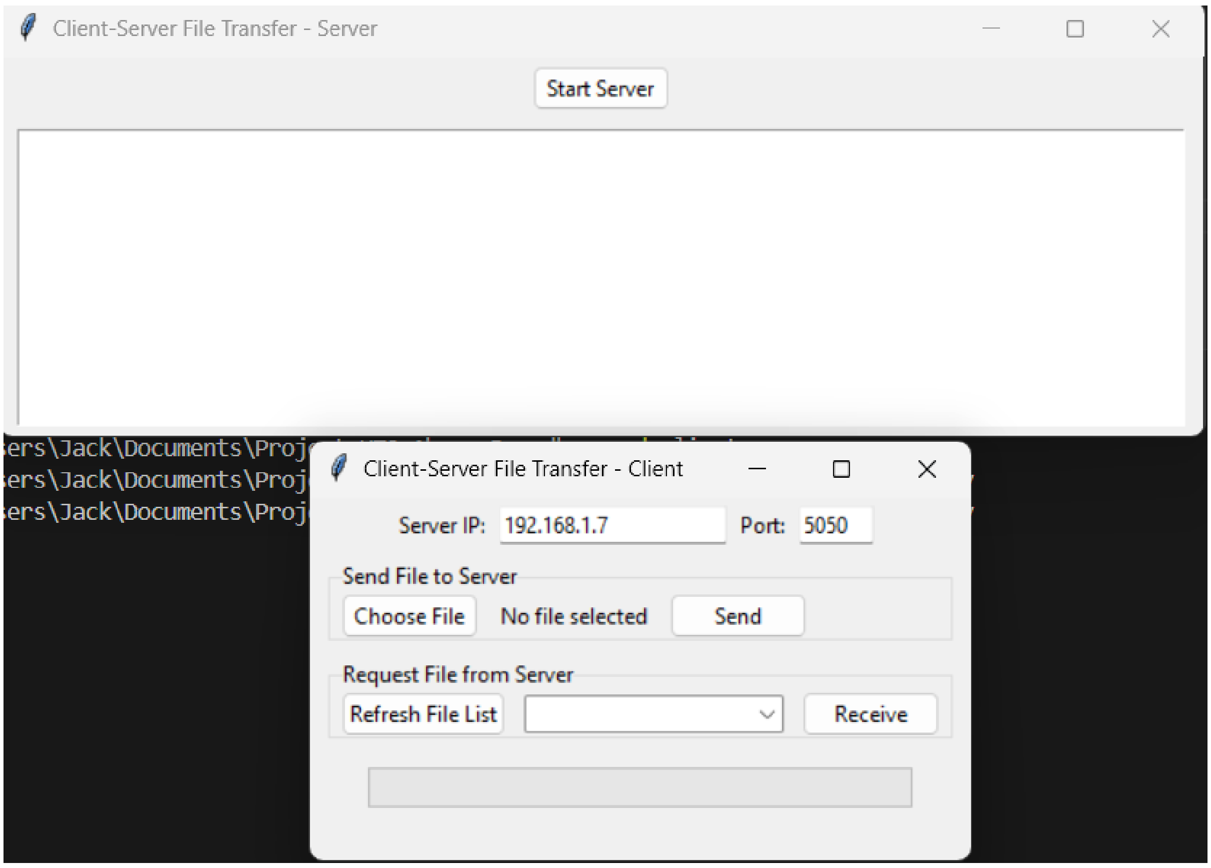
➢ **fileserver.py**

```
import socket
import threading
import os
import tkinter as tk
from tkinter import ttk, messagebox

PORT = 5050
CHUNKSIZE = 4096
SEPARATOR = '<SEPARATOR>'

class FileServerApp:
    def __init__(self, master):
        self.master = master
        master.title('Client-Server File Transfer - Server')

        ttk.Button(master, text='Start Server', 
command=self.start_server).pack(pady=5)

        self.log = tk.Text(master, height=10)
        self.log.pack(fill='both', expand=True, padx=10, pady=5)

    def log_message(self, msg):
        self.log.insert('end', msg + '\n')
        self.log.see('end')

    def start_server(self):
        threading.Thread(target=self.run_server, daemon=True).start()
        self.log_message(f'[LISTENING] on port {PORT}')

    def run_server(self):
        srv = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        srv.bind(('', PORT))
        srv.listen()
```

```python
        while True:
            conn, addr = srv.accept()
            threading.Thread(target=self.handle_client, args=(conn, addr),
daemon=True).start()

    def handle_client(self, conn, addr):
        self.log_message(f'[NEW] Connection from {addr}')
        try:
            cmd = conn.recv(1024).decode().strip()
            if cmd == 'SEND':
                conn.send(b'OK')
                fname = conn.recv(1024).decode().strip()
                meta = conn.recv(1024).decode()
                name, size = meta.split(SEPARATOR)
                size = int(size)
                with open(name, 'wb') as f:
                    received = 0
                    while received < size:
                        chunk = conn.recv(CHUNKSIZE)
                        if not chunk: break
                        f.write(chunk)
                        received += len(chunk)
                        self.log_message(f'[RECV] {received}/{size} bytes')
                self.log_message(f'[DONE] Received {name}')

            elif cmd == 'RECEIVE':
                conn.send(b'OK')
                fname = conn.recv(1024).decode().strip()
                if os.path.exists(fname):
                    size = os.path.getsize(fname)
                    conn.send(f"{fname}{SEPARATOR}{size}".encode())
                    with open(fname, 'rb') as f:
                        sent = 0
                        while chunk := f.read(CHUNKSIZE):
                            conn.sendall(chunk)
                            sent += len(chunk)
                            self.log_message(f'[SEND] {sent}/{size} bytes')
                    self.log_message(f'[DONE] Sent {fname}')
                else:
                    conn.send(b'ERROR: File not found')
                    self.log_message(f'[ERROR] {fname} not found')

            elif cmd == 'LIST':
                files = os.listdir('.')
                file_list = '|'.join(files)
                conn.send(file_list.encode())

        except Exception as e:
            self.log_message(f'[EXC] {e}')
        finally:
            conn.close()
            self.log_message(f'[CLOSED] {addr}')

if __name__ == '__main__':
    root = tk.Tk()
    app = FileServerApp(root)
    root.mainloop()
```

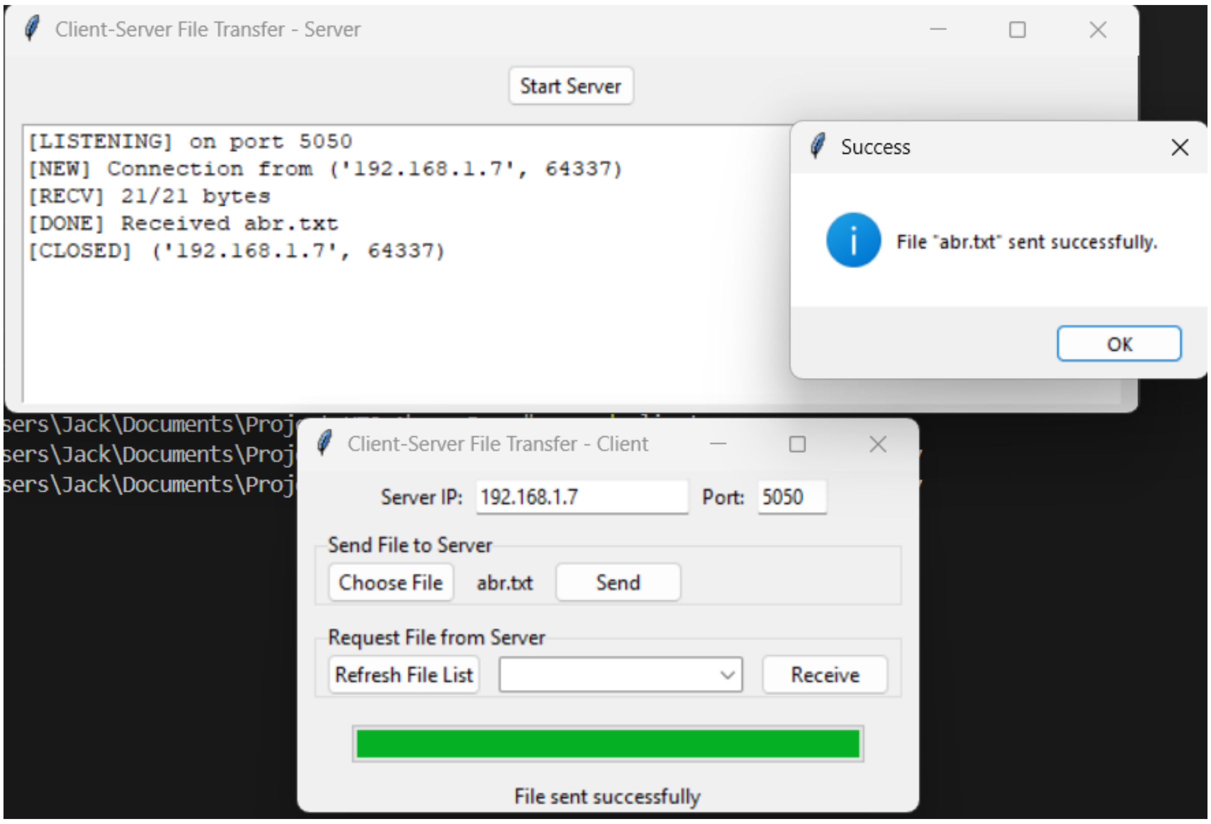
Output

1. **Server dan Client**



```
Client-Server File Transfer - Server
Start Server

ers\Jack\Documents\Proj
ers\Jack\Documents\Proj
ers\Jack\Documents\Proj

Client-Server File Transfer - Client
Server IP: 192.168.1.7    Port: 5050
Send File to Server
Choose File   No file selected   Send
Request File from Server
Refresh File List   [          ]   Receive
```

2. Kirim File



```
[LISTENING] on port 5050
[NEW] Connection from ('192.168.1.7', 64337)
[RECV] 21/21 bytes
[DONE] Received abr.txt
[CLOSED] ('192.168.1.7', 64337)
```

Success

File "abr.txt" sent successfully.

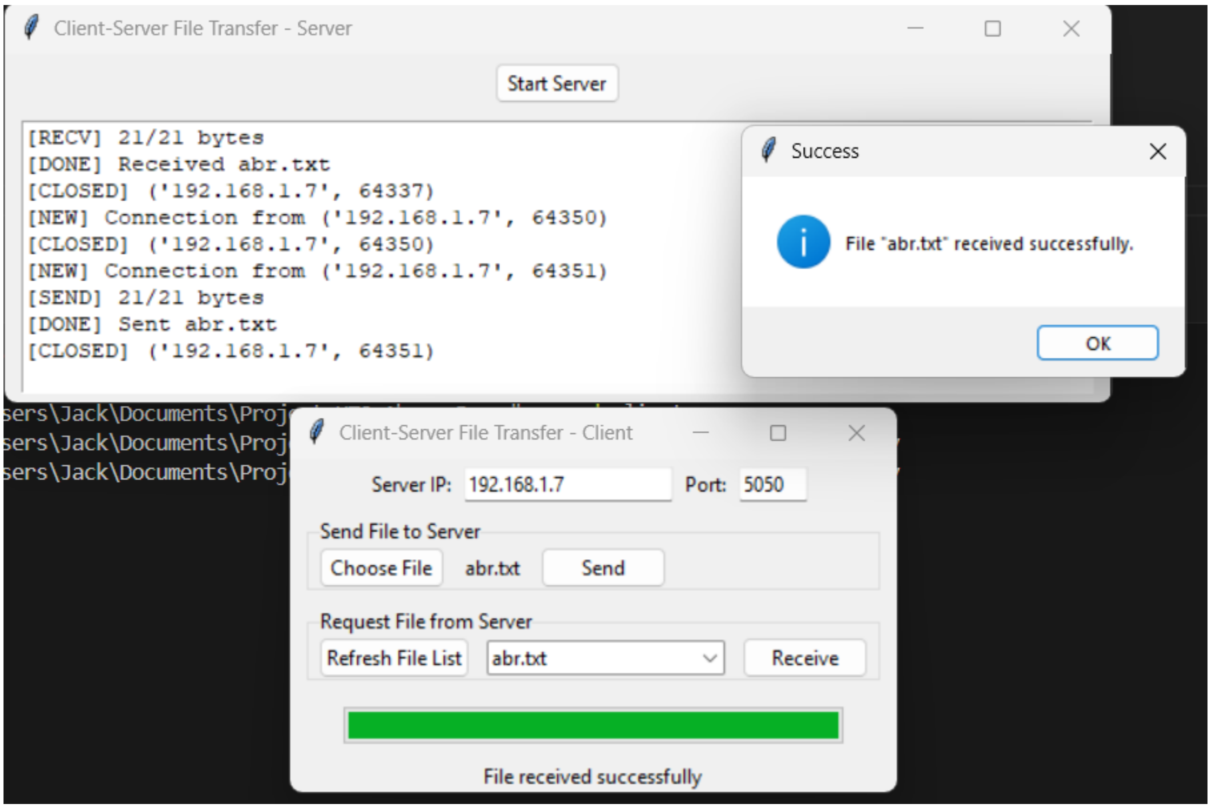
OK

```
sers\Jack\Documents\Proj
sers\Jack\Documents\Proj
sers\Jack\Documents\Proj
```

Server IP: 192.168.1.7 Port: 5050

File sent successfully

3. Terima File


Client-Server File Transfer - Server
Start Server

```
[RECV] 21/21 bytes
[DONE] Received abr.txt
[CLOSED] ('192.168.1.7', 64337)
[NEW] Connection from ('192.168.1.7', 64350)
[CLOSED] ('192.168.1.7', 64350)
[NEW] Connection from ('192.168.1.7', 64351)
[SEND] 21/21 bytes
[DONE] Sent abr.txt
[CLOSED] ('192.168.1.7', 64351)
```

Success

File "abr.txt" received successfully.

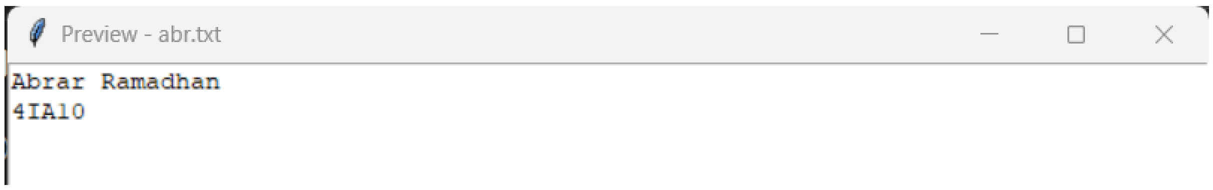
OK

```
sers\Jack\Documents\Proj
sers\Jack\Documents\Proj
sers\Jack\Documents\Proj
```

Client-Server File Transfer - Client

Server IP: 192.168.1.7 Port: 5050

File received successfully



```
Abrar Ramadhan
4IA10
```

Penjelasan :

➢ **fileserver.py**

1. Startup & UI
   - Aplikasi membuat jendela Tkinter dengan judul “P2P File Sharing – Server”.
   - Terdapat tombol Start Server dan area log (widget Text) untuk mencatat event.
2. Menyalakan Server
   - Klik Start Server → start_server() → spawn thread yang menjalankan run_server() → log [LISTENING] on port 5050.
3. Menerima Koneksi
   - run_server() membuka socket di port 5050, listen(), lalu terus-menerus accept().
   - Untuk setiap koneksi, dibuat thread baru handle_client(conn, addr).
4. Protokol ‘SEND’
   - Client kirim string 'SEND'.
   - Server balas 'OK', lalu terima:
     1. Nama file asli (fname),
     2. Metadata "<nama_file><SEPARATOR><size>".
   - Server membuka file baru dengan nama tersebut, membaca byte-by-byte hingga mencapai ukuran, menulis ke disk, sekaligus mencatat progres di log.
   - Setelah selesai, log [DONE] Received <nama_file>.
5. Protokol ‘RECEIVE’
   - Client kirim string 'RECEIVE'.
   - Server balas 'OK', lalu terima nama file yang diminta (fname).
   - Jika file ada di server:
     1. Hitung ukurannya, kirim "<nama_file><SEPARATOR><size>".
     2. Baca file byte-by-byte dan kirim ke client sambil log progres.
   - Jika tidak ada, kirim pesan error ("ERROR: File not found").

6. Penanganan Error & Cleanup
   - Exception di-handle dan dicatat di log.
   - Koneksi selalu ditutup (conn.close()) dalam blok finally.

- **fileclient.py**

1. Startup & UI
   - Aplikasi membuat jendela Tkinter “P2P File Sharing – Client”.
   - Ada input Server IP & Port, radio button untuk pilih mode Send File atau Receive File, progress bar, dan status label.
2. Memilih Mode Aksi
   - switch_action() menampilkan panel “Choose File + Send” jika mode send, atau panel “Filename + Receive” jika mode receive.
3. Mengirim File
   - Klik Send → gui_send() spawn thread send_file(addr, filepath).
   - send_file melakukan:
     1. connect() ke server → kirim b'SEND'.
     2. Tunggu balasan OK.
     3. Kirim nama file dan metadata "<nama><SEPARATOR><size>".
     4. Loop baca file dalam chunk, kirim, update progress bar & status (“Sent X/Y bytes”).
     5. Setelah selesai, update status jadi “File sent successfully” dan munculkan dialog messagebox.showinfo.
4. Refresh List
   - Meminta daftar file yang tersedia di server

5. Menerima File
   - Klik Receive → gui_receive() spawn thread receive_file(addr, fname).
   - receive_file melakukan:
     1. connect() ke server → kirim b'RECEIVE'.
     2. Tunggu balasan OK.
     3. Kirim nama file yang diminta.
     4. Terima respons metadata; jika error atau format tak terduga → tampilkan messagebox.showerror.

5. Jika valid, parse name, size, buka file baru, baca stream hingga mencapai size, tulis ke disk sambil update progress bar & status (“Received X/Y bytes”).
6. Setelah selesai, status jadi “File received successfully” dan munculkan dialog sukses.

6. Preview Isi File

- Setelah transfer selesai, receive_file memanggil show_file_content(name).
- show_file_content mencoba buka file sebagai teks UTF-8:
    1. Jika berhasil, buka jendela baru (Toplevel) dengan widget Text ber-scrollbar, lalu tampilkan seluruh konten.
    2. Jika gagal (binary atau encoding error), munculkan messagebox yang mengatakan konten tidak dapat ditampilkan.