Exp.No: 09 Pavipraja.P

231701037

**Create an application that allows users to send and receive multimedia messages including text, image, and audio.**

**Aim:**

To develop a Python application that allows users to **send and receive multimedia messages** such as **text, images, and audio** using socket programming and GUI (Tkinter).

**Procedure:**

* 1. Import required Python libraries (socket, threading, tkinter, etc.).
  2. Design a **server** that:
     + Listens for incoming client connections.
     + Receives messages (text, image, audio).
     + Broadcasts messages to all connected clients.
  3. Design a **client application** that:
     + Connects to the server via TCP socket.
     + Has GUI buttons to send text, image, or audio.
     + Displays received text, shows images, and plays audio files.
  4. Run the server and then start multiple client instances.
  5. Test sending and receiving multimedia messages between clients.

**Code:**

**Server.py**

import socket, threading, json

HOST = '0.0.0.0'

PORT = 5001

clients = []

lock = threading.Lock()

def broadcast(sender, header, data):

for c in list(clients):

if c is sender:

continue

try:

c.sendall(len(header).to\_bytes(4,'big') + header + data)

except:

with lock:

clients.remove(c)

c.close()

def handle\_client(conn, addr):

print(f"Connected: {addr}")

try:

while True:

hlen = conn.recv(4)

if not hlen: break

hlen = int.from\_bytes(hlen,'big')

header = conn.recv(hlen)

meta = json.loads(header.decode())

size = meta['size']

data = b''

while len(data) < size:

chunk = conn.recv(min(4096, size-len(data)))

if not chunk: break

data += chunk

broadcast(conn, header, data)

except:

pass

with lock:

if conn in clients: clients.remove(conn)

conn.close()

s = socket.socket()

s.bind((HOST, PORT))

s.listen(5)

print("Server running on port", PORT)

while True:

conn, addr = s.accept()

with lock: clients.append(conn)

threading.Thread(target=handle\_client,args=(conn,addr),daemon=True).start()

**Client.py**

import socket, threading, json, os, tempfile

import tkinter as tk

from tkinter import filedialog, scrolledtext

from PIL import Image, ImageTk

from playsound import playsound

SERVER\_IP = '127.0.0.1'

PORT = 5001

def send(sock, type\_, name, data):

h = {'type':type\_, 'filename':name, 'size':len(data)}

hb = json.dumps(h).encode()

sock.sendall(len(hb).to\_bytes(4,'big') + hb + data)

class ChatApp:

def \_\_init\_\_(self, root):

self.sock = socket.socket()

self.sock.connect((SERVER\_IP, PORT))

self.txt = scrolledtext.ScrolledText(root, width=60, height=15)

self.txt.pack()

self.msg = tk.Entry(root, width=45)

self.msg.pack(side=tk.LEFT, padx=3, pady=5)

tk.Button(root, text="Send Text", command=self.send\_text).pack(side=tk.LEFT)

tk.Button(root, text="Image", command=self.send\_img).pack(side=tk.LEFT)

tk.Button(root, text="Audio", command=self.send\_audio).pack(side=tk.LEFT)

self.imglbl = tk.Label(root)

self.imglbl.pack(pady=5)

threading.Thread(target=self.receive, daemon=True).start()

def send\_text(self):

text = self.msg.get()

if text:

send(self.sock, 'text', '', text.encode())

self.msg.delete(0, tk.END)

self.append(f"You: {text}")

def send\_img(self):

p = filedialog.askopenfilename(filetypes=[("Images","\*.png;\*.jpg;\*.jpeg")])

if not p: return

b = open(p,'rb').read()

send(self.sock, 'image', os.path.basename(p), b)

self.append("You sent image: " + os.path.basename(p))

def send\_audio(self):

p = filedialog.askopenfilename(filetypes=[("Audio","\*.mp3;\*.wav")])

if not p: return

b = open(p,'rb').read()

send(self.sock, 'audio', os.path.basename(p), b)

self.append("You sent audio: " + os.path.basename(p))

def append(self, msg):

self.txt.insert(tk.END, msg + "\n")

self.txt.see(tk.END)

def receive(self):

while True:

hlen = self.sock.recv(4)

if not hlen: break

hlen = int.from\_bytes(hlen,'big')

header = self.sock.recv(hlen)

meta = json.loads(header.decode())

data = b''

while len(data) < meta['size']:

chunk = self.sock.recv(min(4096, meta['size']-len(data)))

if not chunk: break

data += chunk

typ = meta['type']

if typ == 'text':

self.append("Peer: " + data.decode())

elif typ == 'image':

tmp = tempfile.NamedTemporaryFile(delete=False, suffix='.jpg')

tmp.write(data)

tmp.close()

img = Image.open(tmp.name)

img.thumbnail((250,250))

tkimg = ImageTk.PhotoImage(img)

self.imglbl.configure(image=tkimg)

self.imglbl.image = tkimg

self.append("Peer sent image: " + meta['filename'])

elif typ == 'audio':

tmp = tempfile.NamedTemporaryFile(delete=False, suffix='.wav')

tmp.write(data)

tmp.close()

threading.Thread(target=lambda: playsound(tmp.name), daemon=True).start()

self.append("Peer sent audio: " + meta['filename'])

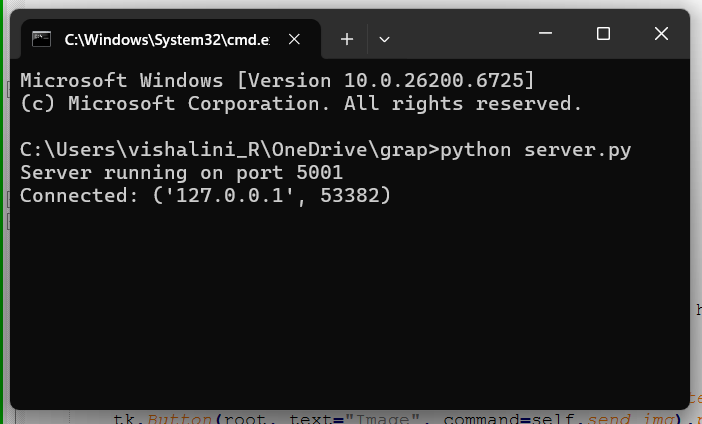
root = tk.Tk()

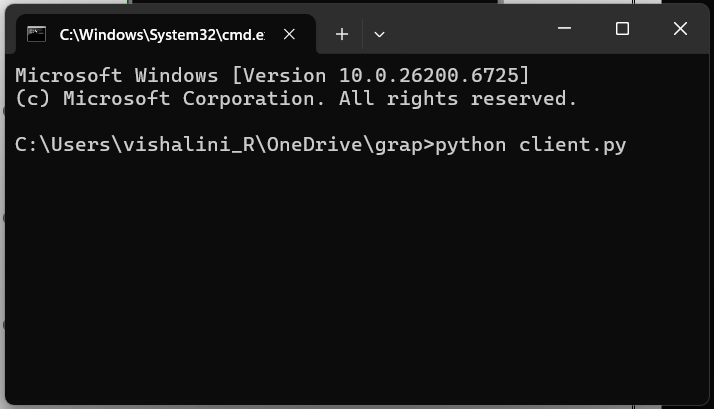
root.title("Multimedia Messenger")

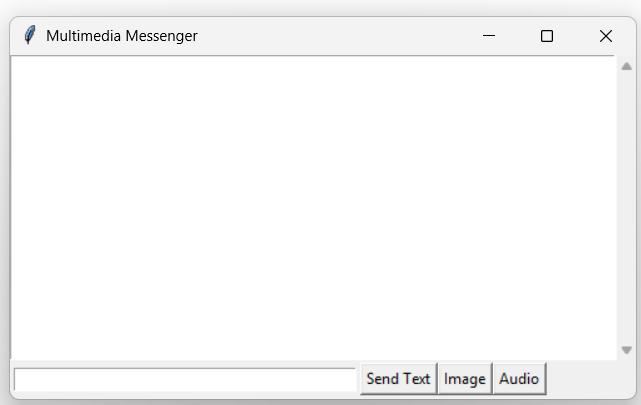
ChatApp(root)

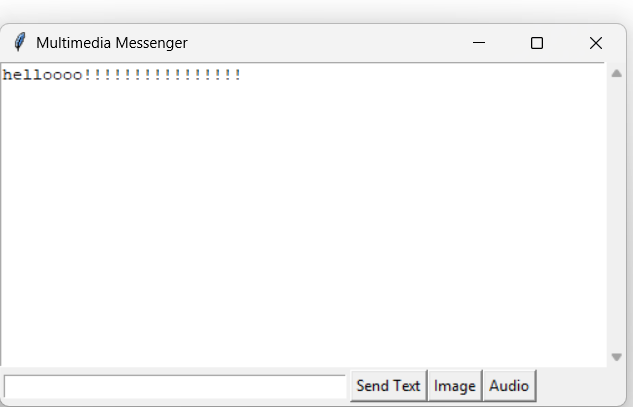
root.mainloop()

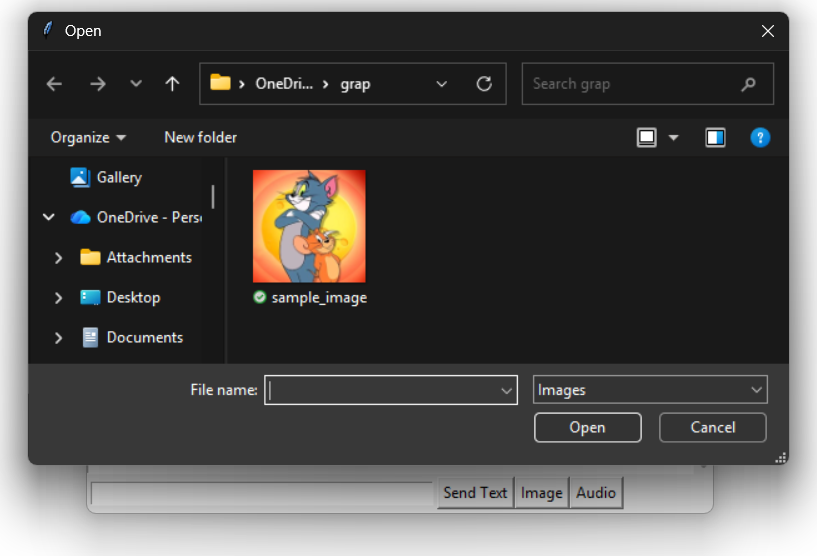
**Output:**

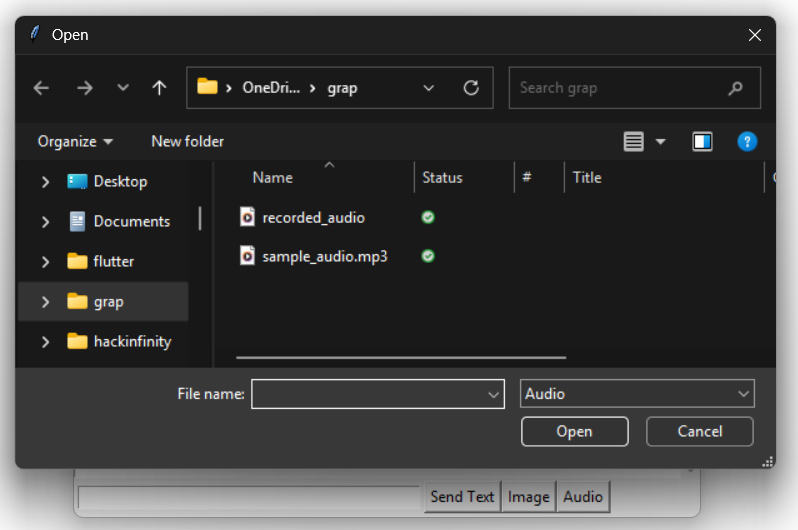


****

****

****

****

****

**Result:**

The **Multimedia Messaging Application** was successfully implemented.  
The system enables multiple users to:

* Send and receive **text messages** instantly.
* Exchange **image files**, which are displayed in the chat window.
* Send and automatically play **audio messages**.