NETWORK PROGRAMMING

MEDIA-SHARING CHAT APPLICATION

Project Overview

1. A real-time chat application with media sharing.
2. Features secure user login, public/private messaging, and file transfer
3. Built with Python using a client-server model.
4. GUI created with tkinter for ease of use.

System Architecture

Client-Server model

- Server handles authentication, messaging, file transfer, and logging.
- Clients interact with GUI, send messages/files, and download shared media.
- Server supports multiple clients via threading.

KEY FEATURES

Authentication

Passwords hashed with SHA-256;
stored in creds.txt.

Text Messaging

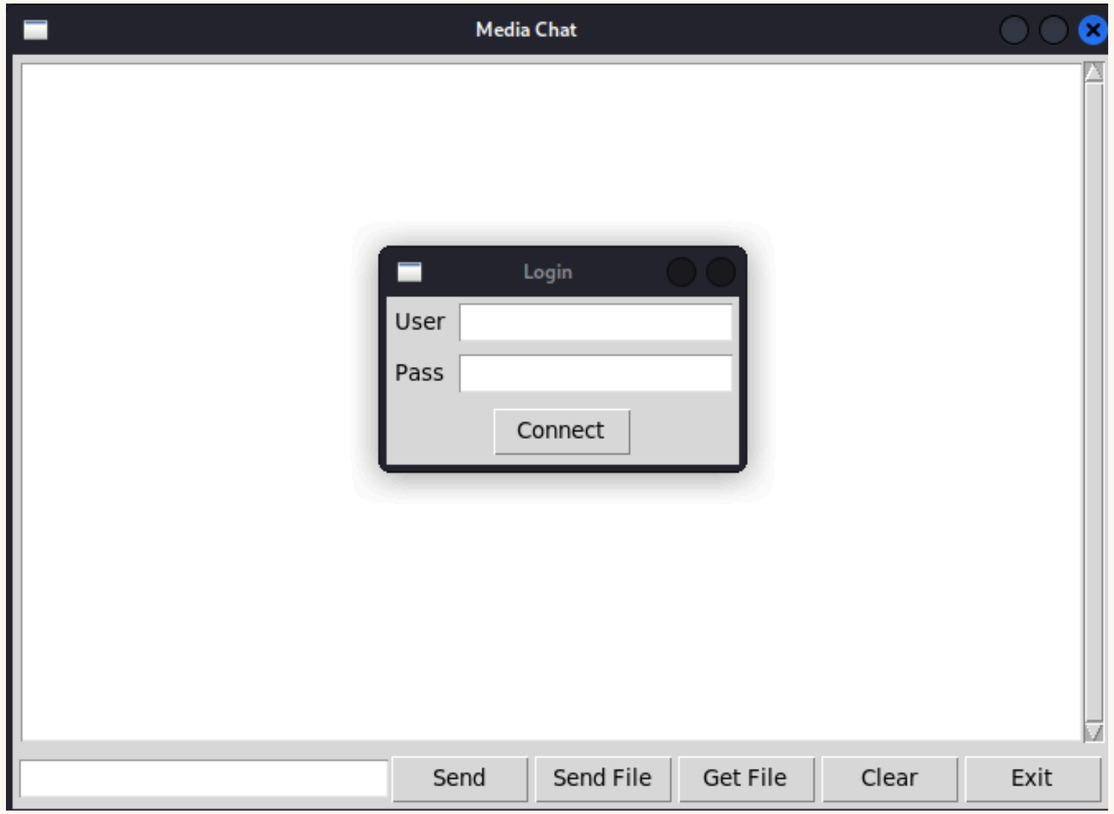
- Public messages by default.
- Private messages using @username.

File Sharing

- File size ⩽ 10MB.
- SHA-256 used for integrity verification.

Chat Logging and message preview

Authentication



Login feature with GUI-based user notification

Verifies user credentials

Implemented with Tkinter

Authentication

```python
def authenticate(conn):
    data=recv_line(conn)
    if '|' not in data:
        conn.sendall(b"ERROR|Bad login\n"); return None
    user,pwd=data.split('|',1)
    h=hashlib.sha256(pwd.encode()).hexdigest()

    lines=USER_DB.read_text().splitlines()
    for ln in lines:
        u,hh=ln.split('|')
        if u==user:
            if hh==h:
                conn.sendall(b"OK|Login successful\n"); return user
            conn.sendall(b"ERROR|Wrong password\n"); return None

    USER_DB.write_text(USER_DB.read_text()+f"{user}|{h}\n")
    conn.sendall(b"OK|New user registered\n"); return user
```

Each username has one correct password, saved in a text file

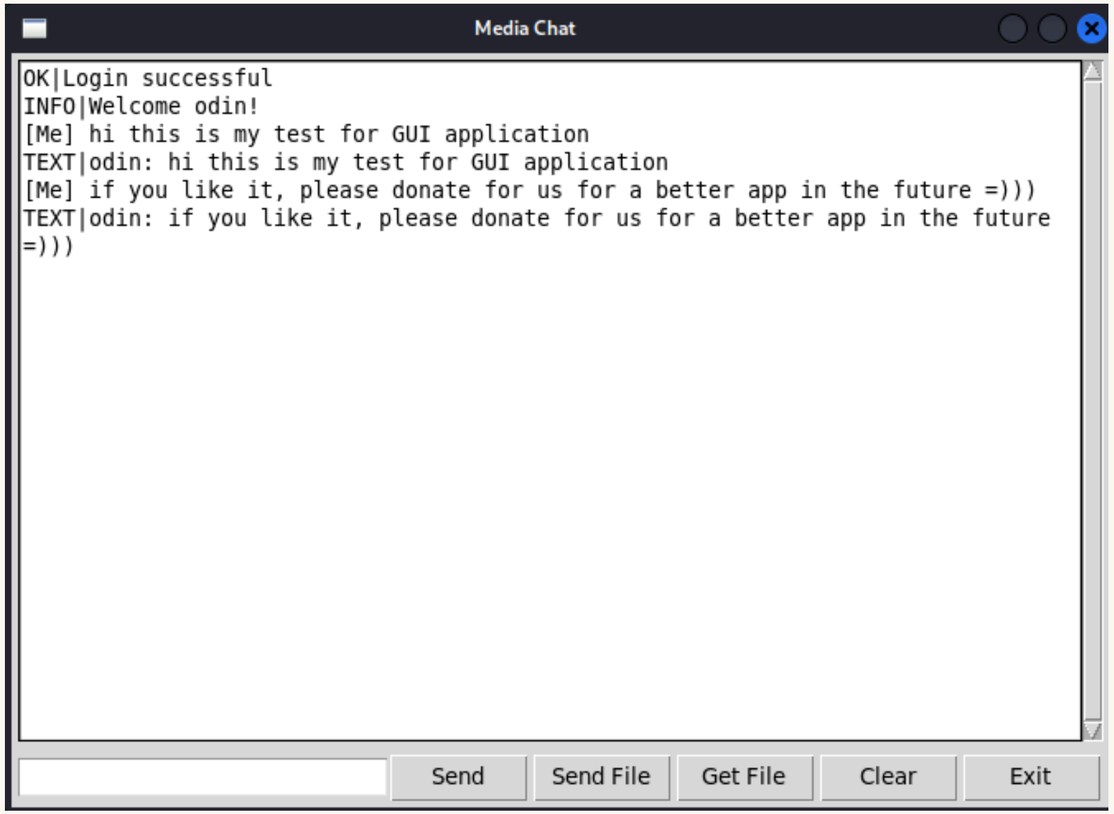
Authentication

```python
def sha(path):
    h=hashlib.sha256()
    with open(path,'rb') as f:
        for ch in iter(lambda:f.read(4096),b''): h.update(ch)
    return h.hexdigest()
```

```
buidat@buidat:~/networkforfinal$ cat user_credentials.txt
dat|0ebe2eca800cf7bd9d9d9f9f4aafbc0c77ae155f43bbbeca69cb256a24c7f9bb
vinh|e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
cuong|5994471abb01112afcc18159f6cc74b4f511b99806da59b3caf5a9c173cacfc5
ton|03ac674216f3e15c761ee1a5e255f067953623c8b388b4459e13f978d7c846f4
join|0ebe2eca800cf7bd9d9d9f9f4aafbc0c77ae155f43bbbeca69cb256a24c7f9bb
minh|be9688998f08509844f215f9397237b2f1147c69989ae9d94e8fd43351a8a60e
(r|de47c9b27eb8d300dbb5f2c353e632c393262cf06340c4fa7f1b40c4cbd36f90
hong|0ebe2eca800cf7bd9d9d9f9f4aafbc0c77ae155f43bbbeca69cb256a24c7f9bb
odin|52f3141f5cb06025c96c6794e35f5611a578919d0deb4d934136acaba06d4b82
```

Password is stored as a hash for security

Authentication



Tada!!

Text Messaging

```python
# ---------- send ----------
def send_text(self):
    if not self.sock: return
    raw=self.e.get().strip()
    if not raw: return
    # ---- detect private ----
    if raw.startswith('@') or raw.lower().startswith('/dm '):
        if raw.startswith('@'): split=raw[1:].split(' ',1)
        else:                   split=raw[4:].split(' ',1)
        if len(split)<2:
            messagebox.showwarning("DM","Nhập dạng @user nội_dung"); return
        to,msg=split
        self.sock.sendall(f"DM|{to}|{msg}\n".encode())
        self._append(f"[PM → {to}] {msg}")
    else:
        self.sock.sendall(f"TEXT|{raw}\n".encode())
        self._append(f"[Me] {raw}")
    self.e.delete(0,tk.END)
```

Function to send message

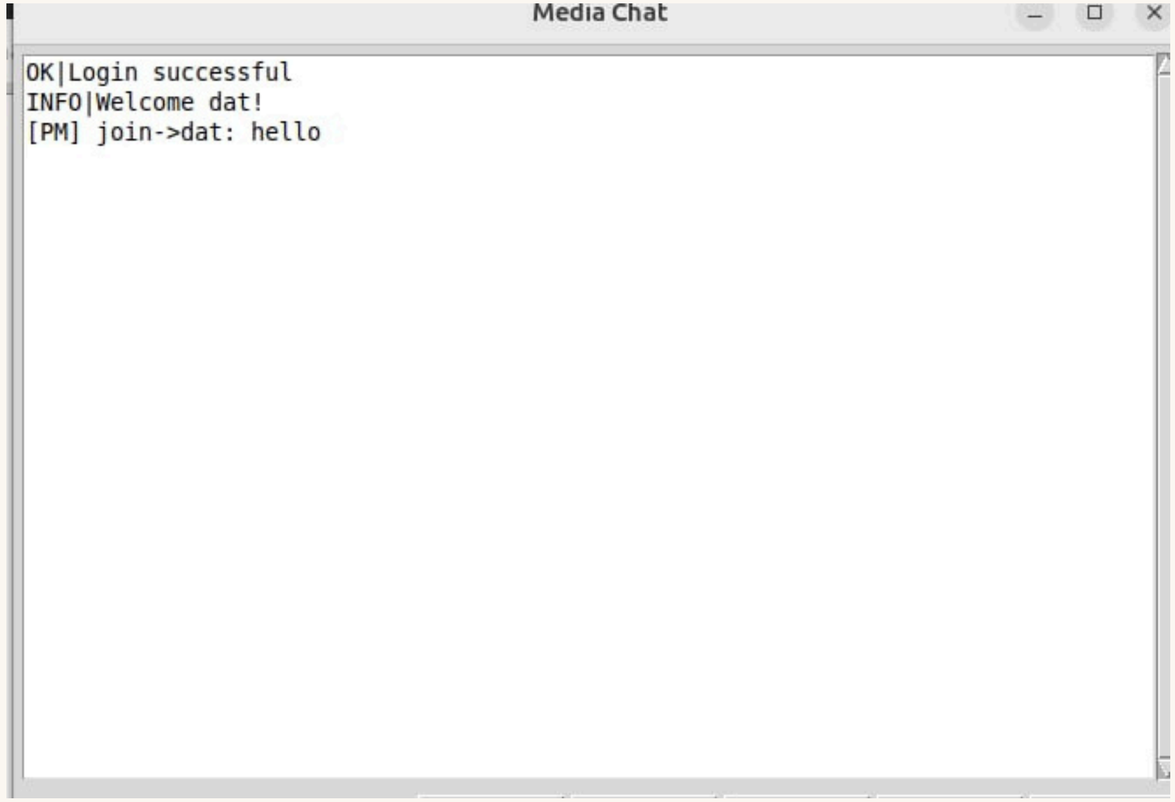
```python
# ---------- send ----------
def send_text(self):
    if not self.sock: return
    raw=self.e.get().strip()
    if not raw: return
    # ---- detect private ----
    if raw.startswith('@') or raw.lower().startswith('/dm '):
        if raw.startswith('@'): split=raw[1:].split(' ',1)
        else:                   split=raw[4:].split(' ',1)
        if len(split)<2:
            messagebox.showwarning("DM","Nhập dạng @user nội_dung"); return
        to,msg=split
        self.sock.sendall(f"DM|{to}|{msg}\n".encode())
        self._append(f"[PM → {to}] {msg}")
```

✅ Gets the user input from the text field
📩 If it starts with @ or /dm, treats it as a private message
🔄 Sends private messages as DM|user|message

```
else:
    self.sock.sendall(f"TEXT|{raw}\n".encode())
    self._append(f"[Me] {raw}")
self.e.delete(0,tk.END)
```

💬 Otherwise, sends as TEXT|message for public chat
🧹 Clears the input field after sending

The result:


Media Chat
OK|Login successful
INFO|Welcome dat!
[PM] join->dat: hello

File Sharing

```python
def send_file(self):
    if not self.sock: return
    p=filedialog.askopenfilename(); 0
    if not p: return
    fname=os.path.basename(p); size=os.path.getsize(p); h=sha256(p)
    try:
        self.sock.sendall(f"FILE|{fname}|{size}|{h}\n".encode())
        with open(p,'rb') as f:
            for ch in iter(lambda:f.read(4096),b''): self.sock.sendall(ch)
        self.sock.sendall(b'\n')
        self._append(f"[Me] sent {fname} ({size} B)")
    except Exception as e: messagebox.showerror("File",e)
# ---------- helpers ----------
def sha256(path):
    h=hashlib.sha256()
    with open(path,'rb') as f:
        for ch in iter(lambda:f.read(4096),b''): h.update(ch)
```

💬 Users share files with metadata (FILE|...), then content.

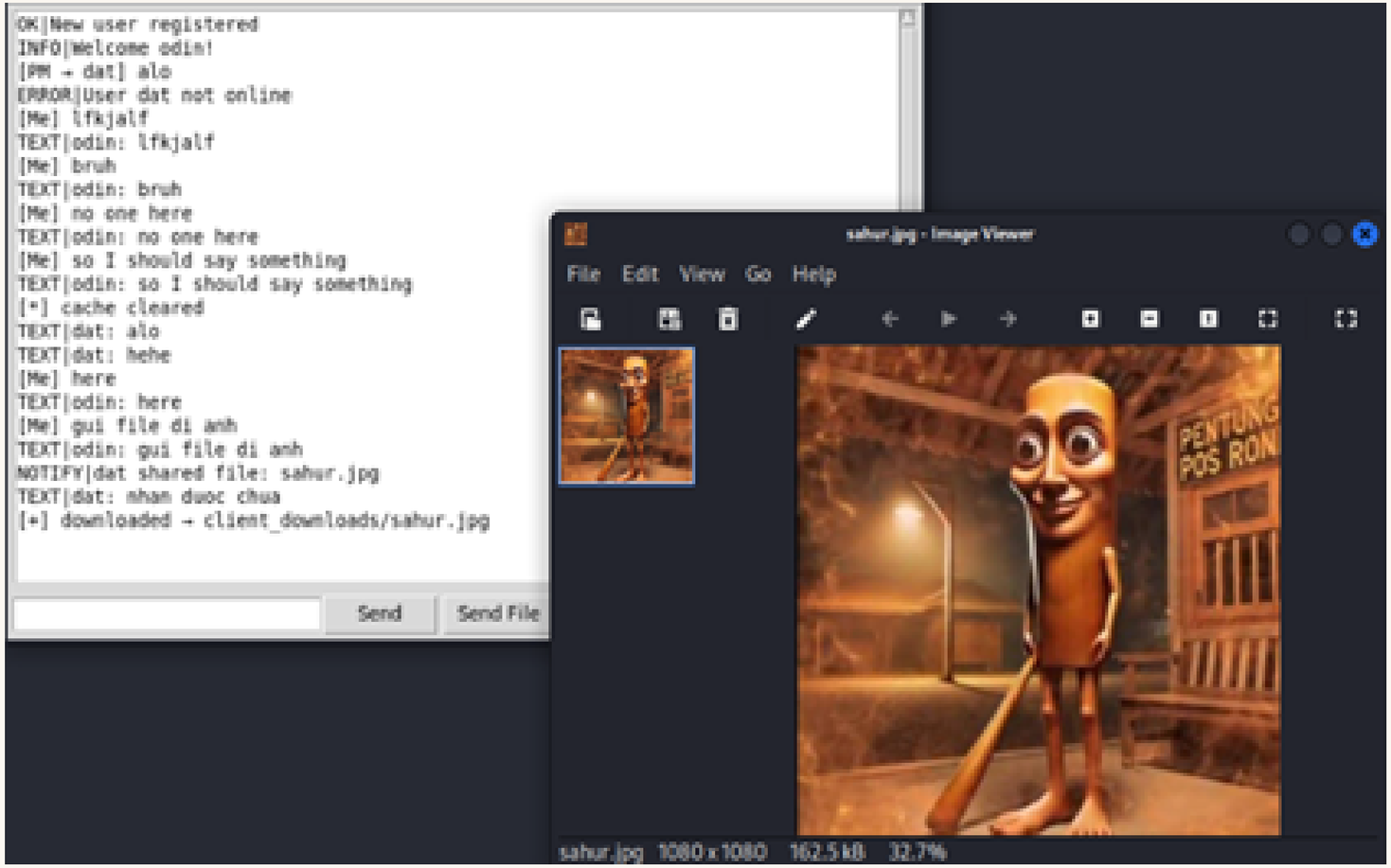
```python
"""
Media-Chat - server (broadcast + private DM)
"""
import os, socket, threading, hashlib, pathlib

HOST, PORT  = '0.0.0.0', 5000
MEDIA_DIR   = pathlib.Path('received_media')
USER_DB     = pathlib.Path('user_credentials.txt')
LOG_FILE    = pathlib.Path('message_history.txt')
MAX_SIZE    = 10 * 1024 * 1024
ALLOWED_EXT = {'.jpg','.png','.gif','.mp3','.wav','.txt','.pdf'}

MEDIA_DIR.mkdir(exist_ok=True)
USER_DB.touch(); LOG_FILE.touch()

clients_lock           = threading.Lock()
sock_by_user:dict[str,socket.socket] = {}   #  <— NEW
user_by_sock:dict[socket.socket,str] = {}
```

Setup: Files, Limits, and Media Settings



💬 A notify message appears; others click "Get File" to download.

Security Considerations

1. Password Security:
2. Upload Limitation:
3. File Transfer Integrity:

SHA-256 hashing; no plaintext storage.

File hash checked after download.

Files over 10MB are rejected (future enforcement structure in place).

Limitations & Future Work

ICE BREAKER

CURRENT LIMITATIONS

❌ No end-to-end encryption yet.
📁 Media type support is limited.
💬 Lacks group chat and emoji reactions.

FUTURE ENHANCEMENTS

✅ ADD ENCRYPTION FOR SECURE MESSAGES.
🖼️ PREVIEW MEDIA FILES DIRECTLY IN CHAT.
👥 ENABLE GROUP CHAT & INTERACTIVE FEATURES (EMOJI, REACTIONS).

CONCLUSIONS

CONCLUSION 1

Successfully implemented a media-sharing chat system.

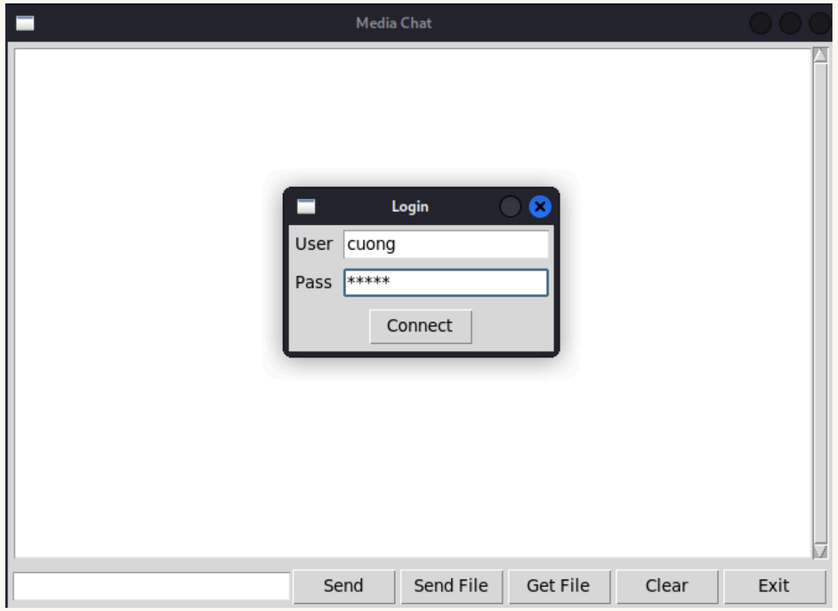
CONCLUSION 2

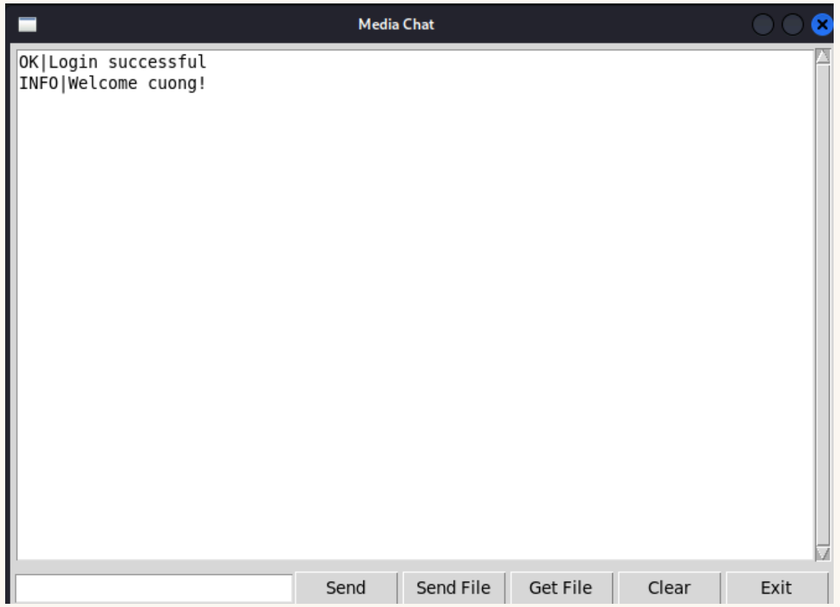
Secure, user-friendly interface.

CONCLUSION 3

Strong foundation for future feature expansion.

EXAMPLE USAGE

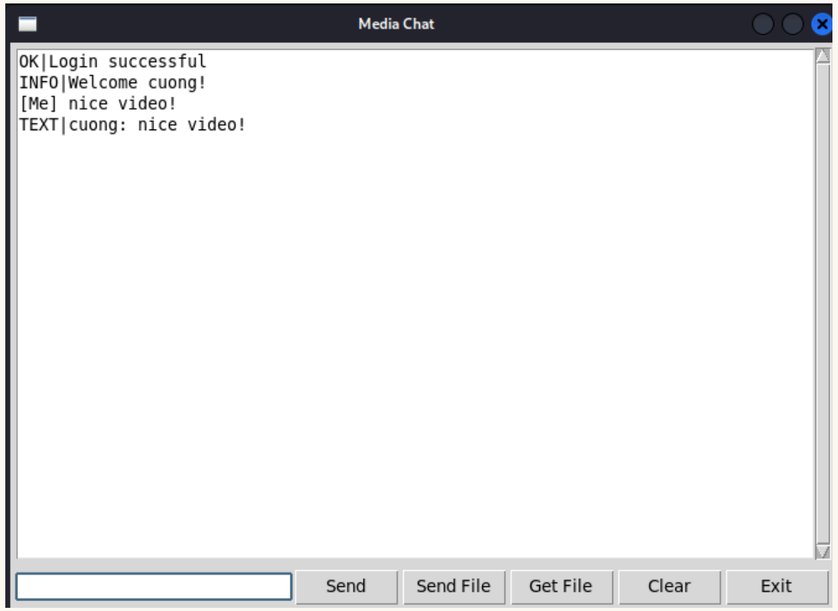




LOGIN

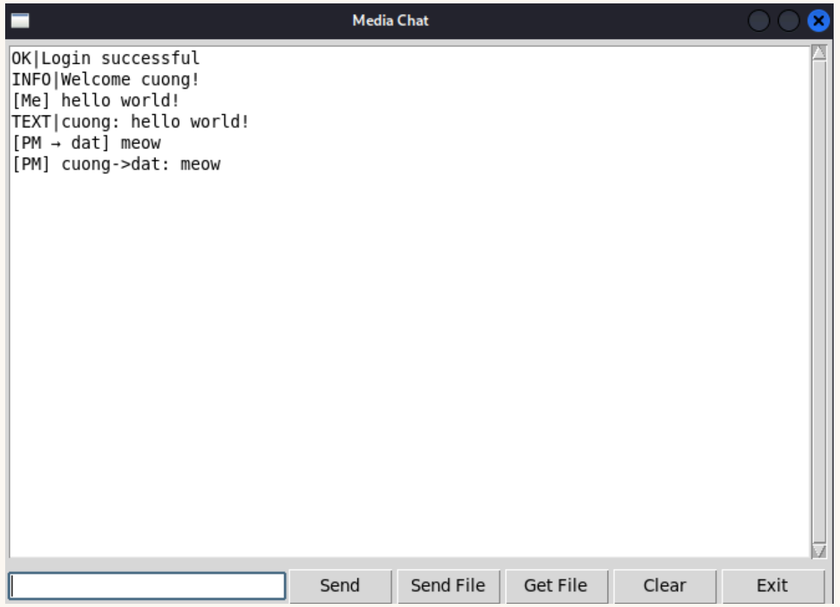
Enter credentials
→ "Login successful" confirmation.

EXAMPLE USAGE



SEND MESSAGE

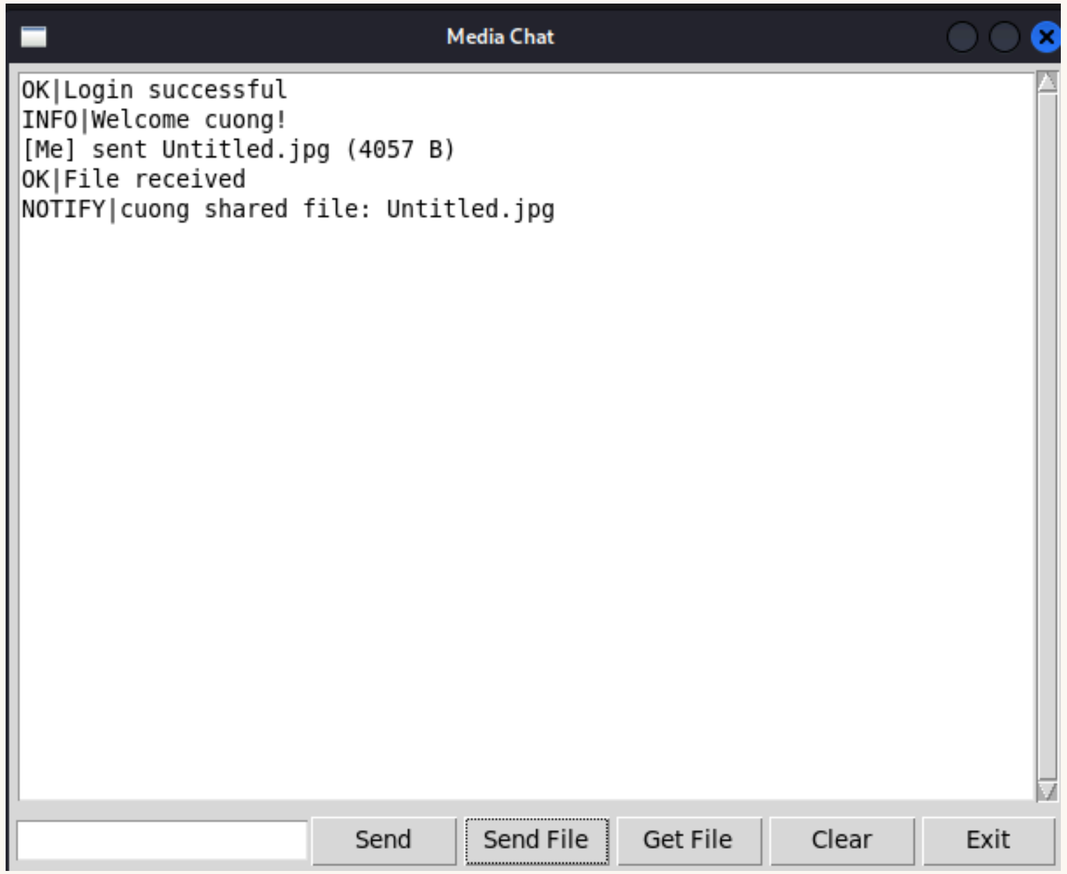
Type “Nice video!” into the message input field and press Enter or click Send.



PRIVATE MESSAGE

Use @username → only recipient sees it.

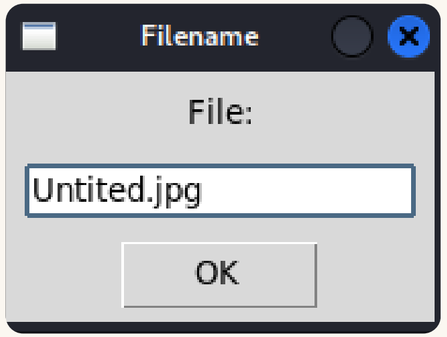
EXAMPLE USAGE

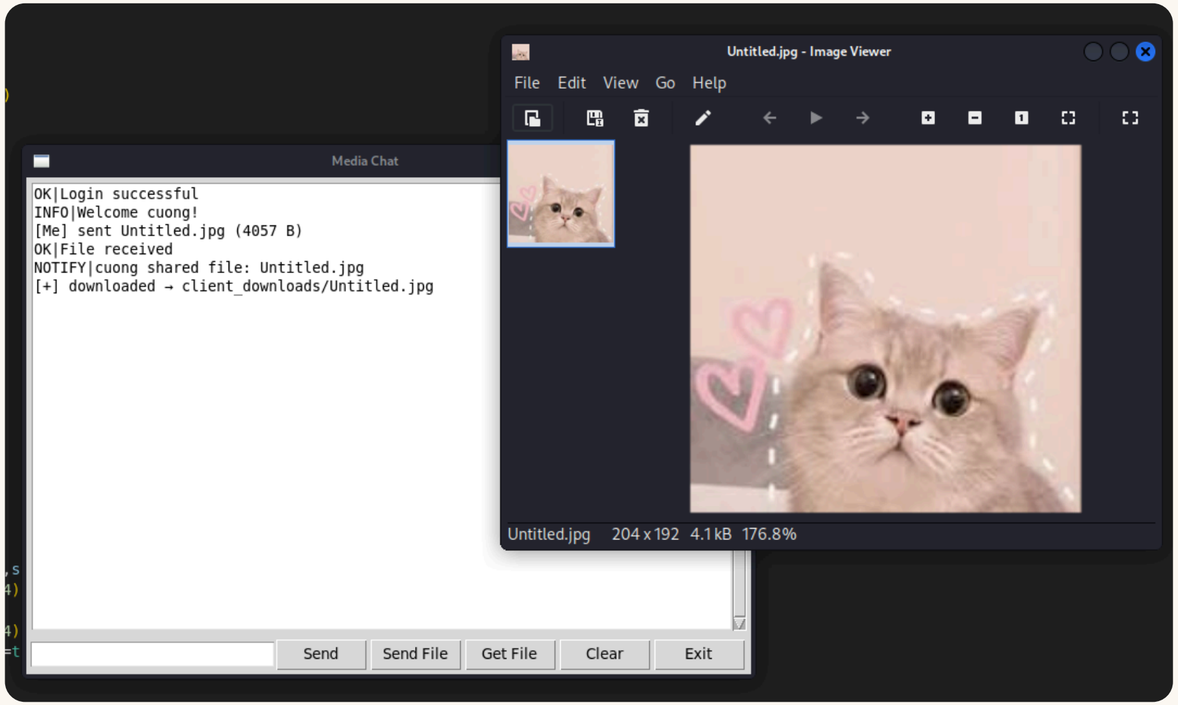


SHARE FILE

Click Send File, choose a file from the file dialog, and confirm.

EXAMPLE USAGE





DOWNLOAD FILE

Enter filename
→ file downloads and opens.

THANK YOU!