HO CHI MINH UNIVERSITY OF SCIENCE

FACULTY OF INFORMATION TECHNOLOGY

FTP CLIENT WITH VIRUS SCANNING VIA CLAMAVAGENTS

LAB PROJECT - (Course: Computer Networking)

TEAM: Just2chillGuys

MEMBERS:

Trương Tấn Dũng - 24127026

Nguyễn Kinh Quốc - 24127230

SUPERVISOR(S): Nguyễn Thanh Quân

Ho Chi Minh City, August 2025

Acknowledgements

This project could not have been completed without the guidance of our instructor(s), the support of our families, and the collaboration of all team members. We sincerely thank our supervisor(s) for their invaluable feedback throughout the project, and we appreciate each teammate for their dedication and contributions.

1. Group Introduction

1.1 Group details

Group name: Just2chillGuys

1.2 Members information

Name	Student ID
Trương Tấn Dũng	24127026
Nguyễn Kinh Quốc	24127230

1.3 Individual contributions

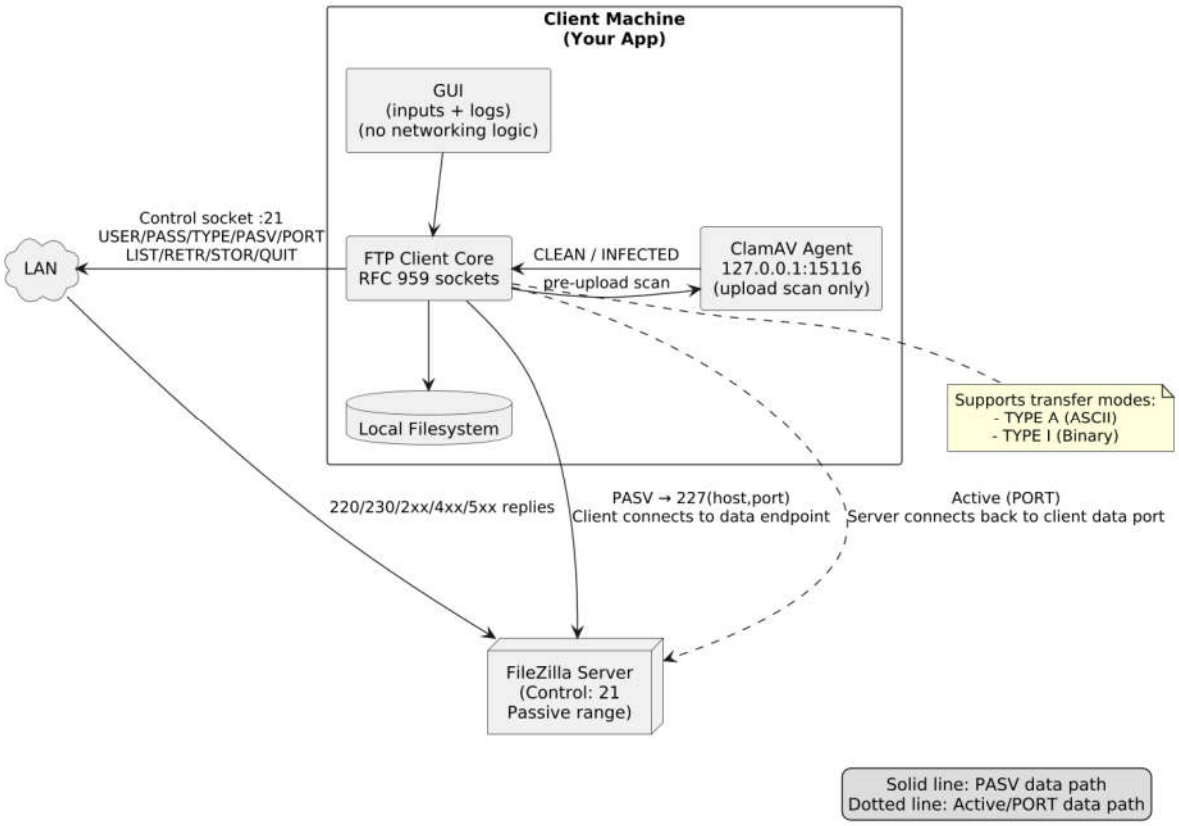
Name	Student ID	Team role	Description
Trương Tấn Dũng	24127026	Networking(Transfer & ClamAV) UI/UX Designer, Report Writer	Implemented the upload/download, integrated the ClamAV scanning agent into the transfer flow and designed the Tkinter/CustomTkinter UI with thread-safe progress/logging, error handling, and timeouts
Nguyễn Kinh Quốc	24127230	Storage & Session Engineer, Report Writer	Implemented local/remote file & directory operations with conflict/overwrite policy and error reporting, built session management and authored the README with setup, usage, and troubleshooting

2. System Design Overview

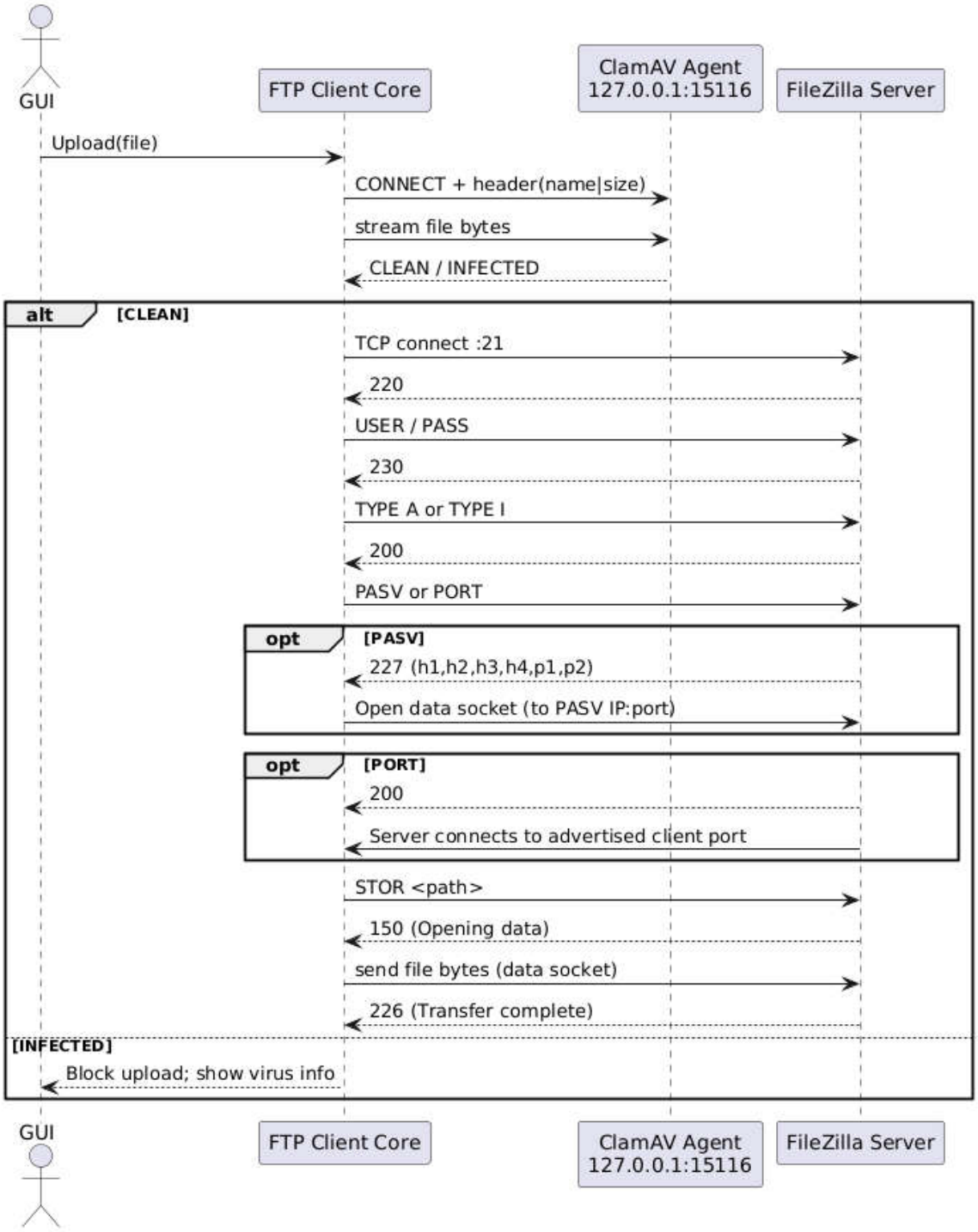
2.1 Project Objectives & Boundaries

- **Socket-level FTP client for LAN:** Implement an FTP client that speaks RFC 959 over raw TCP sockets against a LAN server (e.g., FileZilla Server).
- **Protocol correctness:** Keep a persistent **control** socket (TCP/21) and open per-operation **data** sockets. Implement core verbs: USER, PASS, PWD/CWD, TYPE, PASV/PORT, LIST, RETR, STOR, QUIT.
- **Both data connection modes:** Support **Passive (PASV)** *and* **Active (PORT)**. Default to PASV for simplicity; allow switching to Active (client advertises its IP/port and accepts the server's incoming data connection).
- **Both transfer modes:** Support **Binary (TYPE I)** as the safe default and **ASCII (TYPE A)** for text when CRLF normalization is needed.
- **Upload-only malware scanning (ClamAV agent):** Before any upload, send the file to a **local ClamAV agent** over TCP (127.0.0.1:15116) using a simple header <filename><SEPARATOR><filesize>, stream the bytes, and only proceed with STOR if the agent replies CLEAN. (Downloads are **not** scanned.)
- **Response handling & robustness:** Parse single/multi-line replies; surface actionable errors (530, 550, 425/426) and apply socket timeouts with limited retries on transient failures.
- **Optional security (RFC 4217):** Support **Explicit FTPS** — AUTH TLS on the control socket, PBSZ 0/PROT P for protected data channels when required.
- **Minimal GUI footprint:** The GUI collects inputs (host/port/credentials/paths), triggers the socket/scan flows, and displays logs/progress/scan results. All networking and scanning orchestration live outside the UI.

2.2 High-Level Architecture

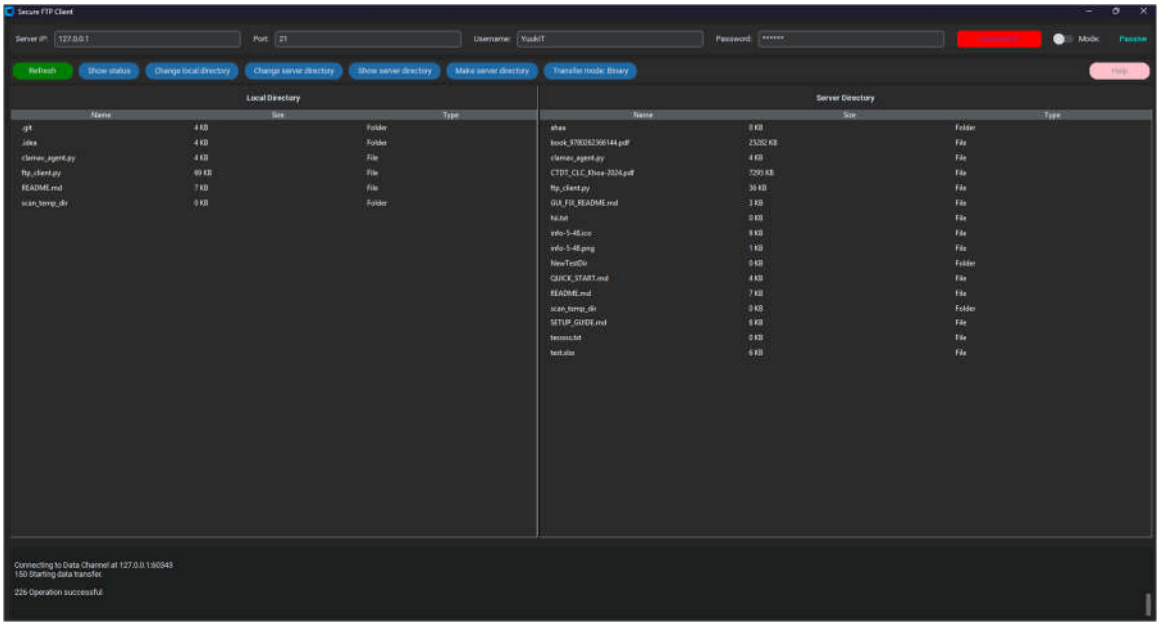


2.3 Upload Flow with ClamAV

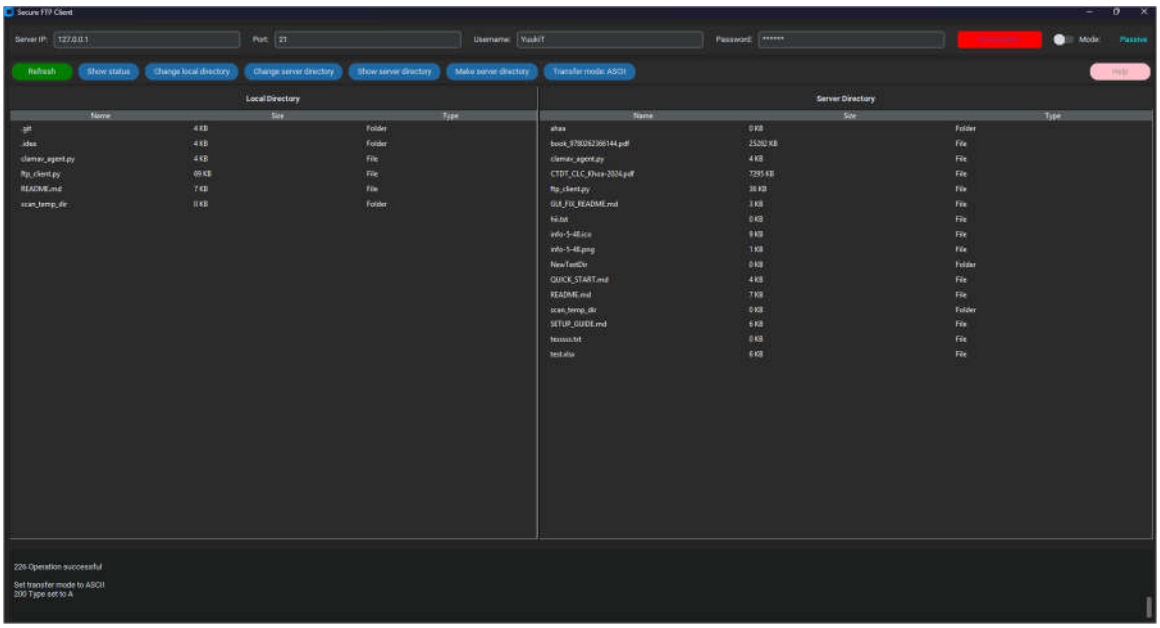


3. Screenshots of successful sessions

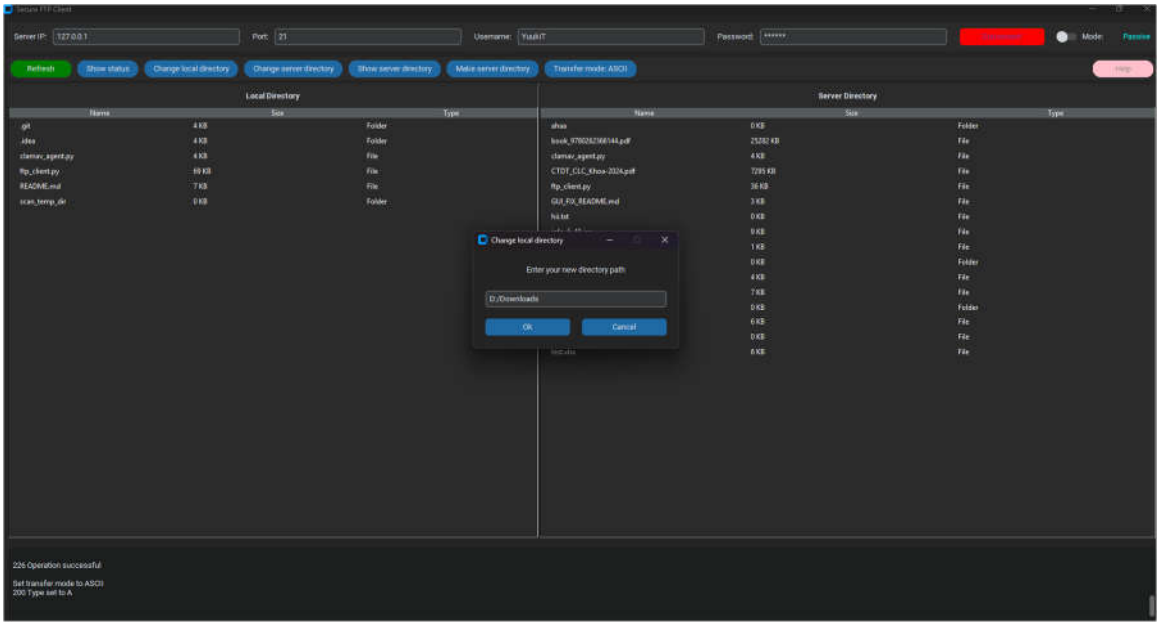
3.1 Connect & Login

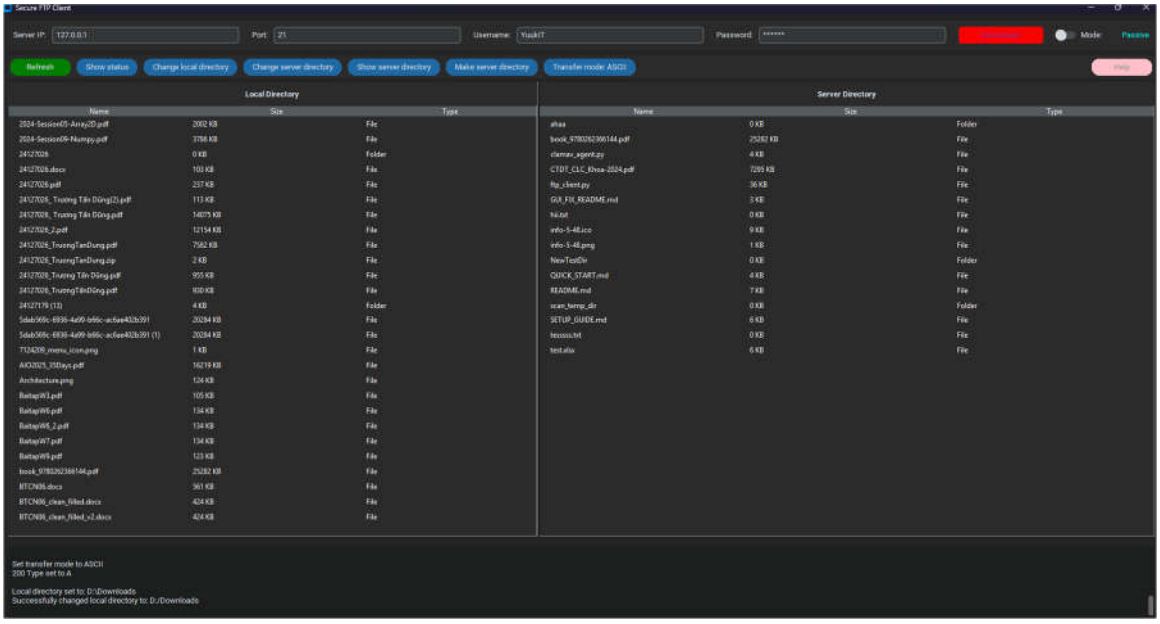


3.2 Set up transfer mode

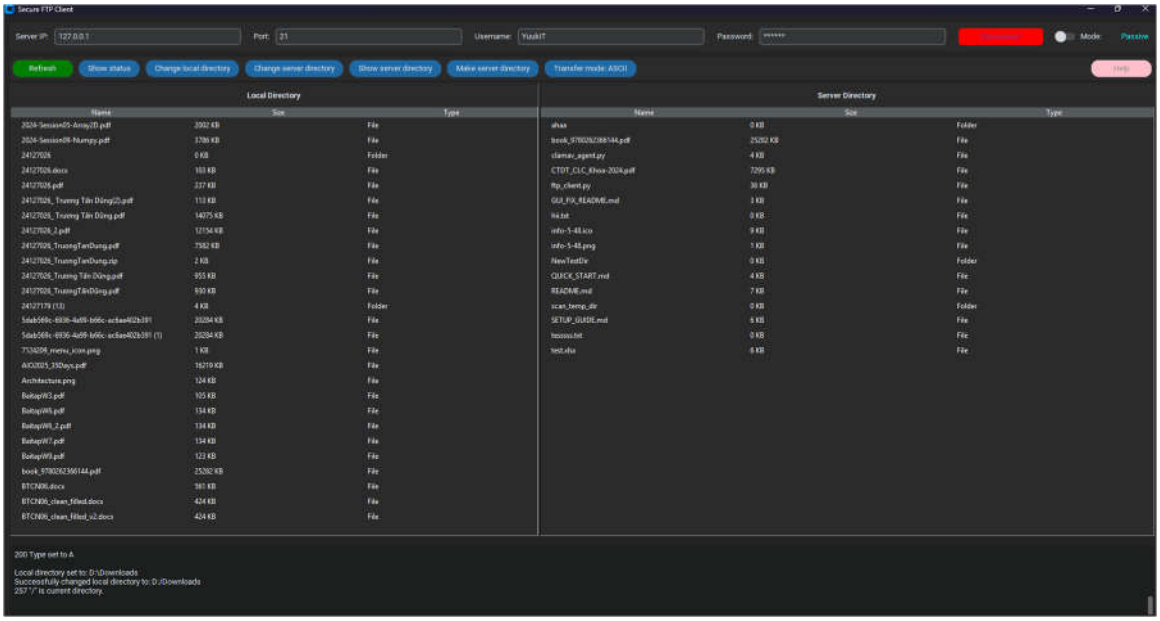


3.3 Change local/server directory

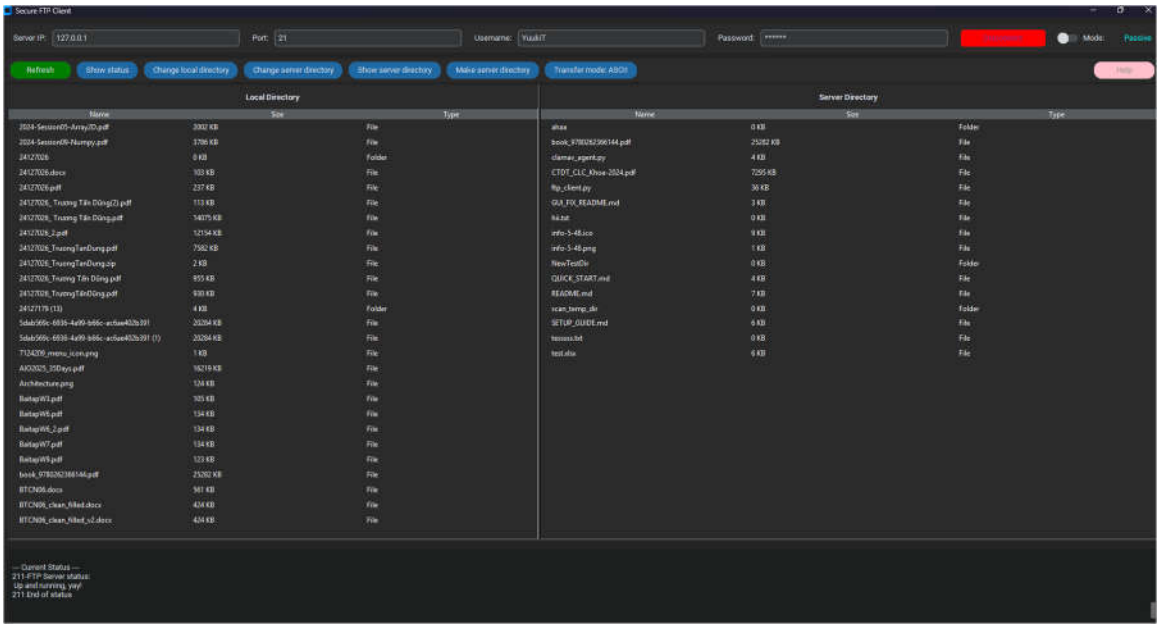




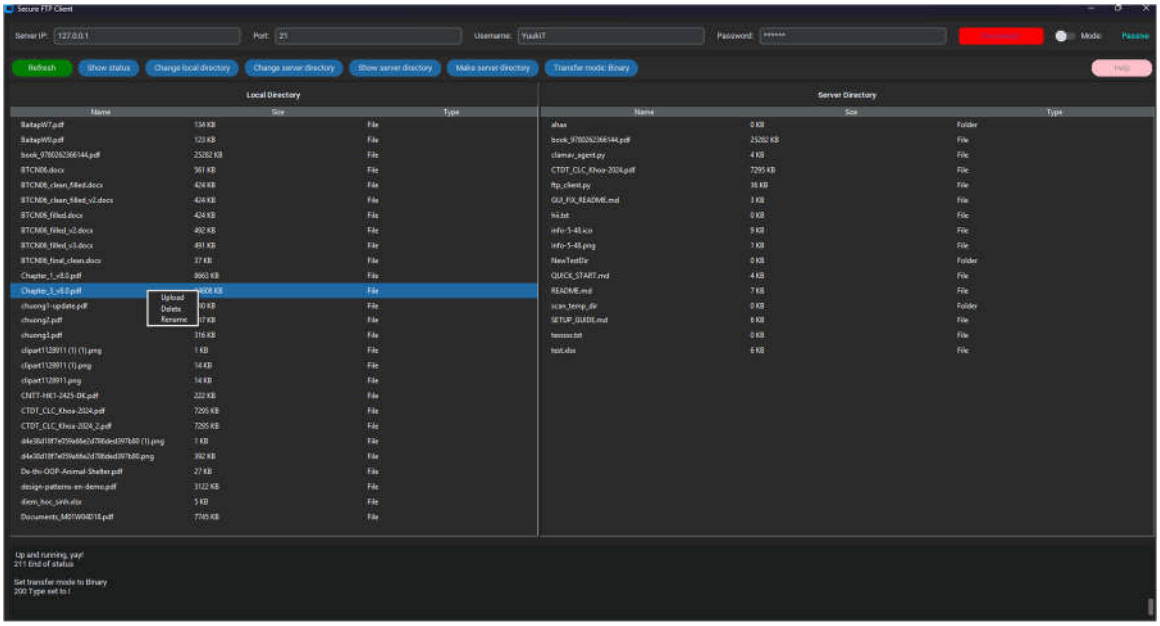
3.4 Show server directory

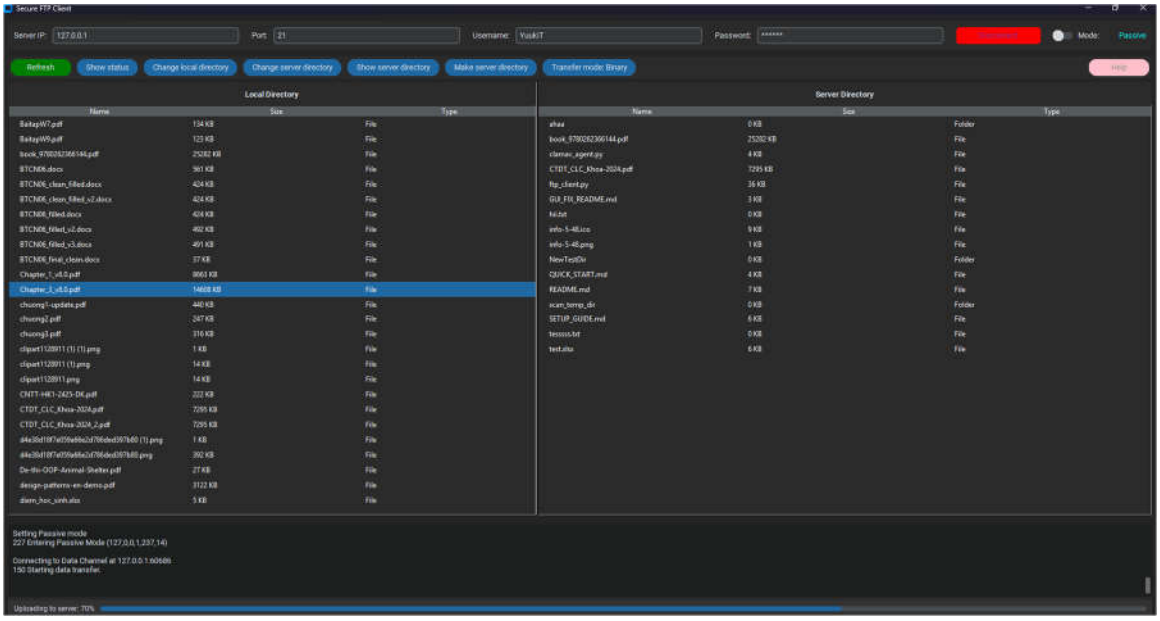


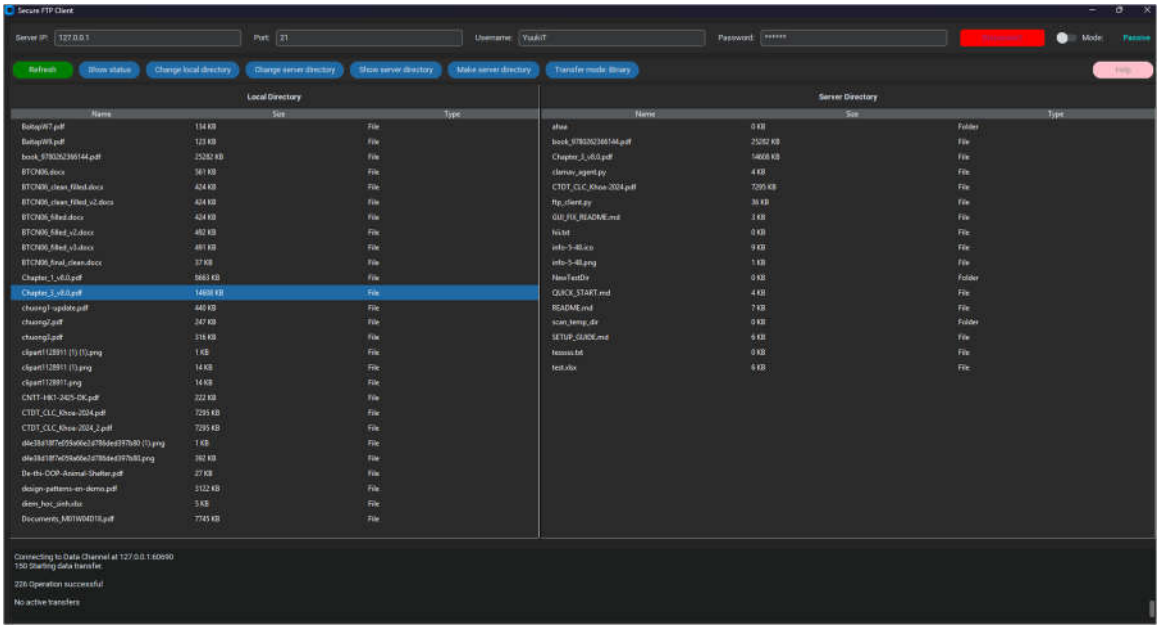
3.4 Show status:



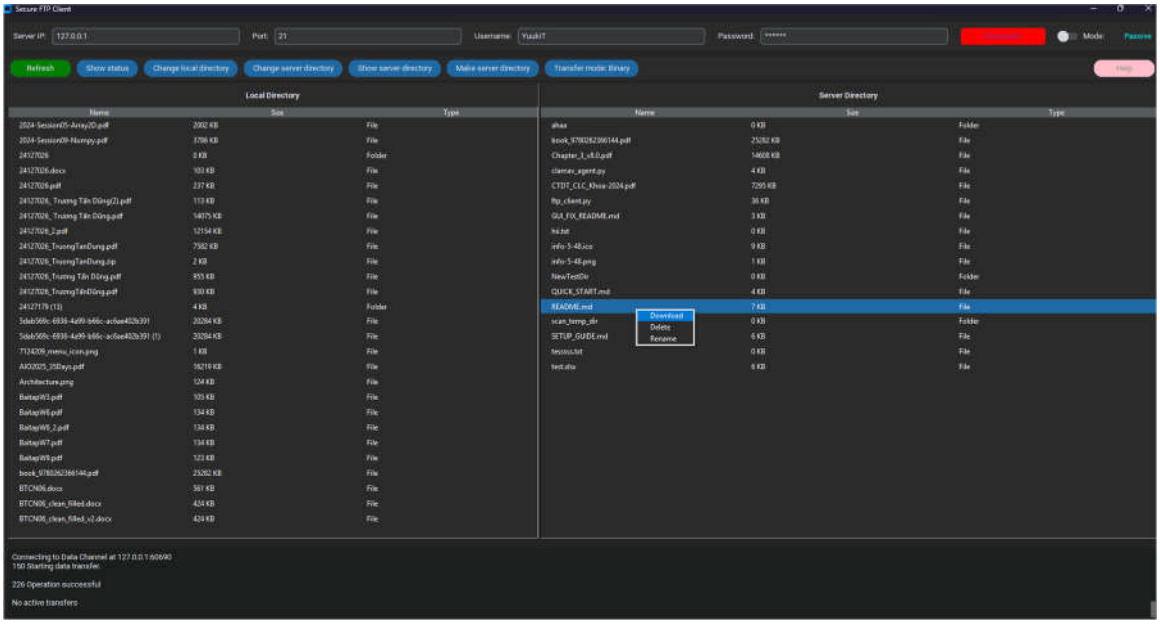
3.5 Upload

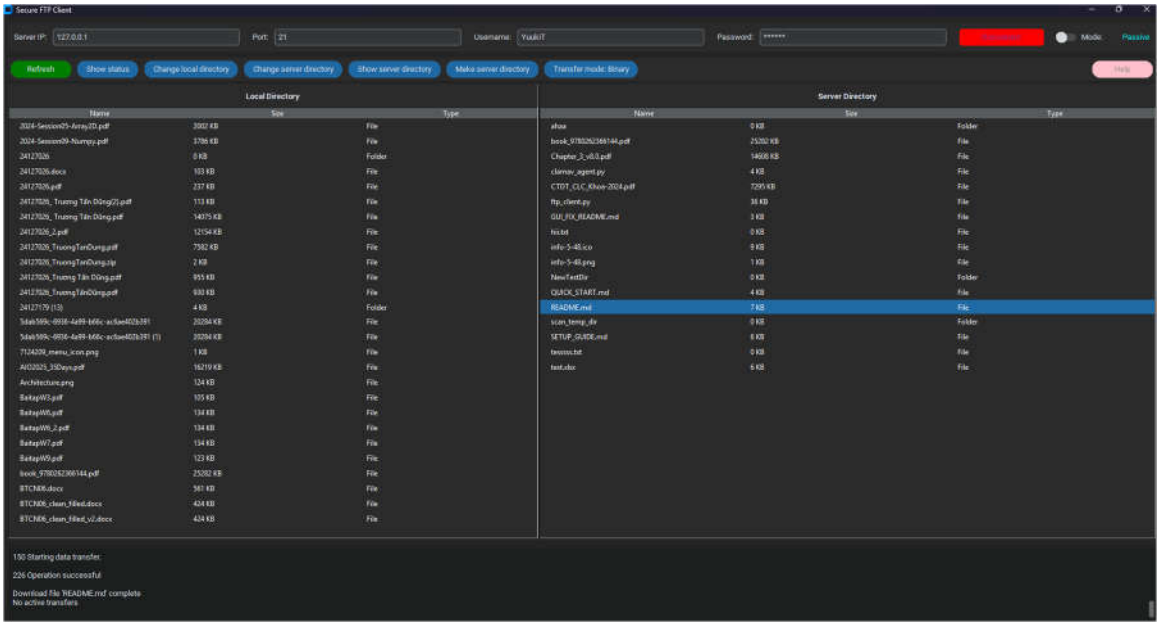

Secure FTP Client
Server IP: 127.0.0.1
Port: 21
Username: YuukiT
Password: ******
Mode: Passive
Refresh
Show status
Change local directory
Change server directory
Show server directory
Make server directory
Transfer mode: Binary
Help
Local Directory
Name
Size
Type
BaitapW7.pdf 134 KB File
BaitapW9.pdf 123 KB File
book_9780262366144.pdf 25282 KB File
BTCN06.docx 361 KB File
BTCN06_clean_filled.docx 424 KB File
BTCN06_clean_filled_v2.docx 424 KB File
BTCN06_filled.docx 424 KB File
BTCN06_filled_v2.docx 492 KB File
BTCN06_filled_v3.docx 491 KB File
BTCN06_final_clean.docx 37 KB File
Chapter_1_v8.0.pdf 8663 KB File
Chapter_3_v8.0.pdf File
chuong1-update.pdf File
chuong2.pdf File
chuong3.pdf 316 KB File
clipart1128911 (1) (1).png 1 KB File
clipart1128911 (1).png 14 KB File
clipart1128911.png 14 KB File
CNTT-HK1-2425-DK.pdf 222 KB File
CTDT_CLC_Khoa-2024.pdf 7295 KB File
CTDT_CLC_Khoa-2024_2.pdf 7295 KB File
d4e38d1897e059e86e2d786ded3978d0 (1).png 1 KB File
d4e38d1897e059e86e2d786ded3978d0.png 392 KB File
De-thi-OOP-Animal-Shelter.pdf 27 KB File
design-patterns-en-demo.pdf 3122 KB File
diem_hoc_sinh.xlsx 5 KB File
Documents_M01W04D1E.pdf 7765 KB File
Upload
Delete
Rename
Server Directory
Name
Size
Type
ahaa 0 KB Folder
book_9780262366144.pdf 25282 KB File
clamav_agent.py 4 KB File
CTDT_CLC_Khoa-2024.pdf 7295 KB File
ftp_client.py 36 KB File
GUI_FIX_README.md 3 KB File
hii.txt 0 KB File
info-5-48.ico 9 KB File
info-5-48.png 1 KB File
NewTestDir 0 KB Folder
QUICK_START.md 4 KB File
README.md 7 KB File
scan_temp_dir 0 KB Folder
SETUP_GUIDE.md 8 KB File
tesssss.txt 0 KB File
test.xlsx 6 KB File
Up and running, yay!
211 End of status
Set transfer mode to Binary
200 Type set to I


Secure FTP Client
Server IP: 127.0.0.1
Port: 21
Username: YuukiT
Password: ******
Mode: Passive
Refresh
Show status
Change local directory
Change server directory
Show server directory
Make server directory
Transfer mode: Binary
Help
Local Directory
Name
Size
Type
BaitapW7.pdf 134 KB File
BaitapW9.pdf 123 KB File
book_9780262366144.pdf 25282 KB File
BTCN06.docx 361 KB File
BTCN06_clean_filled.docx 424 KB File
BTCN06_clean_filled_v2.docx 424 KB File
BTCN06_filled.docx 424 KB File
BTCN06_filled_v2.docx 492 KB File
BTCN06_filled_v3.docx 491 KB File
BTCN06_final_clean.docx 37 KB File
Chapter_1_v8.0.pdf 8663 KB File
Chapter_3_v8.0.pdf 14608 KB File
chuong1-update.pdf 440 KB File
chuong2.pdf 247 KB File
chuong3.pdf 316 KB File
clipart1128911 (1) (1).png 1 KB File
clipart1128911 (1).png 14 KB File
clipart1128911.png 14 KB File
CNTT-HK1-2425-DK.pdf 222 KB File
CTDT_CLC_Khoa-2024.pdf 7295 KB File
CTDT_CLC_Khoa-2024_2.pdf 7295 KB File
d4e38d1897e059e86e2d786ded3978d0 (1).png 1 KB File
d4e38d1897e059e86e2d786ded3978d0.png 392 KB File
De-thi-OOP-Animal-Shelter.pdf 27 KB File
design-patterns-en-demo.pdf 3122 KB File
diem_hoc_sinh.xlsx 5 KB File
Server Directory
Name
Size
Type
ahaa 0 KB Folder
book_9780262366144.pdf 25282 KB File
clamav_agent.py 4 KB File
CTDT_CLC_Khoa-2024.pdf 7295 KB File
ftp_client.py 36 KB File
GUI_FIX_README.md 3 KB File
hii.txt 0 KB File
info-5-48.ico 9 KB File
info-5-48.png 1 KB File
NewTestDir 0 KB Folder
QUICK_START.md 4 KB File
README.md 7 KB File
scan_temp_dir 0 KB Folder
SETUP_GUIDE.md 8 KB File
tesssss.txt 0 KB File
test.xlsx 6 KB File
Setting Passive mode
227 Entering Passive Mode (127,0,0,1,237,14)
Connecting to Data Channel at 127.0.0.1:60685
150 Starting data transfer.
Uploading to server: 70%



3.6 Download





4. Summary of How Each Requirement Was Fulfilled

Requirement	How we implemented it
Connect & authenticate	Open a persistent TCP control socket to server:21; send USER/PASS; handle 220 → 331/230 login flow.
Directory listing	TYPE A or TYPE I → PASV (get 227) or PORT (get 200); open data socket; send LIST; expect 150 → 226.
Download (RETR)	Set transfer mode; negotiate data socket (PASV/PORT); RETR <path>; read to EOF; then 226.
Upload (STOR) with ClamAV (upload-only scan)	Pre-upload: send file to local ClamAV agent (127.0.0.1:15116); proceed only if CLEAN. Then negotiate data socket and STOR; expect 150 → 226.
Data connection modes	Support both: PASV (client connects to server's 227 IP:port) and Active/PORT (server connects back to client's advertised port).
Transfer modes	Support TYPE I (Binary) and TYPE A (ASCII); default to Binary for byte-exact files.
Error handling & robustness	Parse single/multi-line replies; timeouts on control/data sockets; retry selected transient failures; map 530/550/425/426 to clear messages.

Optional FTPS (if enabled)	Explicit TLS: AUTH TLS → wrap control; PBSZ 0, PROT P for protected data channels.
Minimal GUI	GUI only collects inputs and shows logs/progress/scan results; all networking lives in the socket layer.

5. Problems encountered

- Corrupted files (ASCII vs Binary)
  - **Symptom:** Images/archives broken; text lines altered.
  - **Root cause:** Using TYPE A for binary files (CRLF conversions) or the opposite.
  - **Fix:** Default to TYPE I (Binary); use TYPE A only for plain text that requires CRLF normalization.
  - **Evidence:** Byte size and optional hash match between source and destination.

- 550 File unavailable / wrong path
  - **Symptom:** LIST/RETR/STOR returns 550.Root cause: Using TYPE A for binary files (CRLF conversions) or the opposite.
  - **Root cause:** Wrong working directory or malformed/relative paths.
  - **Fix:** Use PWD/CWD to confirm location; normalize separators; prefer absolute paths for reliability.
  - **Evidence:** Same command succeeds after correcting path (`150/226` instead of `550`).

- FTPS handshake issues (Explicit TLS)
  - **Symptom:** After `AUTH TLS`, commands fail or data is unprotected.
  - **Root cause:** Missing `PBSZ 0` / `PROT P`, or certificate trust problems.
  - **Fix:** Sequence: `AUTH TLS` → wrap control socket → `PBSZ 0` → `PROT P`; configure trust for the server cert (self-signed in LAN if needed).
  - **Evidence:** Control returns `200` after `PBSZ/PROT`; transfers succeed over protected data channels.

- UI freezes & multi-transfer failures (GUI-driven)
  - **Symptom:** The window becomes unresponsive when starting `LIST/RETR/STOR`; with multi-select uploads/downloads, some items randomly fail (`425/426`, missing `226`, or `421 Timeout`).
  - **Root cause:** Network I/O executed on the **GUI event loop**, so the UI blocks. Kicking off several actions from the GUI causes **simultaneous use of a single control**

socket—replies interleave and commands race. (In addition, unsafe UI updates from background tasks can crash or deadlock.)

- **Fix:** Keep **all socket I/O off the UI thread**. Use **one background worker that owns the control connection** and processes a **FIFO job queue** (`LIST`, `RETR`, `STOR`, delete/rename), opening a fresh **data socket per job**. If you truly need parallel transfers, spawn **separate client instances**—each with its **own control socket**—instead of sharing one. The GUI should only enqueue jobs and receive progress via a thread-safe dispatcher (e.g., `after(...)`), never touching sockets directly.
- **Evidence:** With the queue/worker pattern, the UI stays responsive; control logs show clean, serialized sequences (`… → 150 → 226`); multi-file operations finish without random `425/426` errors.

6. Security Considerations

- Use Explicit FTPS for encryption where possible (server must be configured with a certificate).
- Restrict server accounts to the minimum necessary permissions and a defined home directory.
- Open only the required ports in the Windows Firewall (FTP 21 and passive range on the server).
- Validate file paths to prevent directory traversal in local operations.
- ClamAV scanning before upload/after download for high-risk environments.

7. References

- FileZilla Server documentation

- RFC 959 (FTP) and RFC 4217 (FTP over TLS)

- Python ftplib/ssl/Tkinter references