

Secure Chat Application – Project Report

1. Project Overview

This project implements a **secure, multi-client chat application** using Python. It provides **end-to-end encryption**, **user authentication**, and supports **sending text messages and files**.

The system is based on a **client-server architecture**, with:

- A **graphical interface** for clients (using `customtkinter`)
 - A **console-based interface** for the server
-

2. Key Features

Secure Communication

- **Key Exchange:** Uses **X25519** elliptic curve Diffie-Hellman protocol for secure key exchange.
- **Authenticated Encryption:** Messages and files are encrypted using **AES-256-GCM**.
- **Key Derivation:** Employs **HKDF** (HMAC-based Key Derivation Function) for AES key generation.

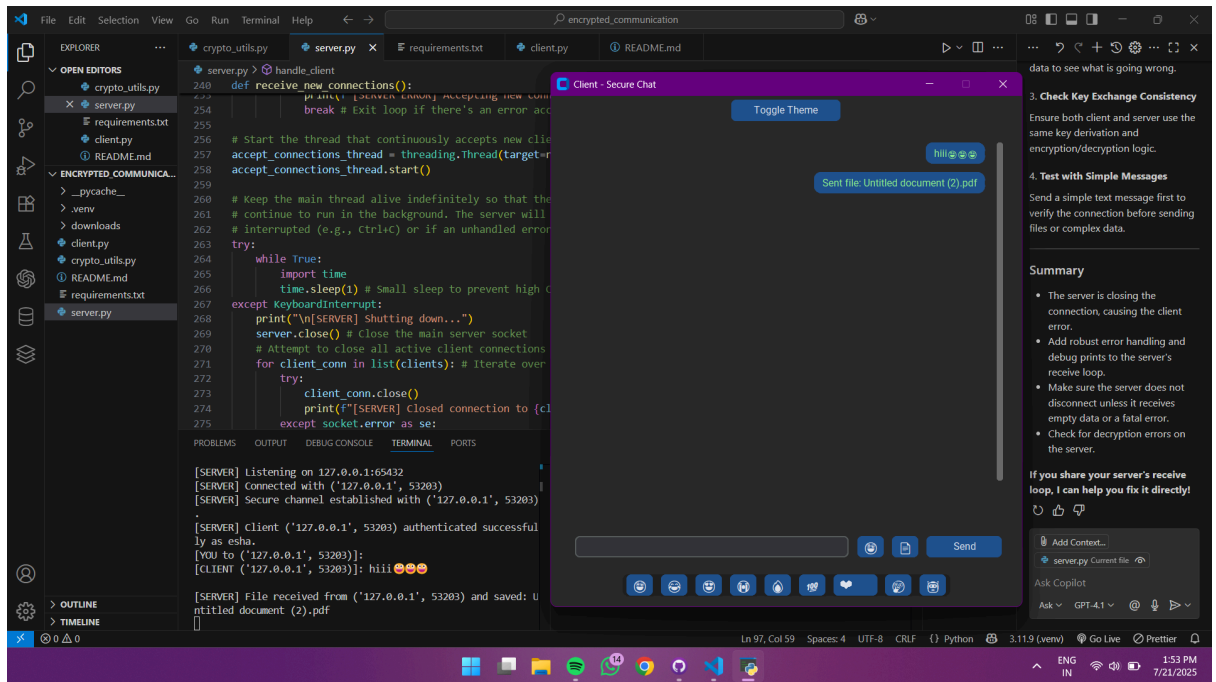
Multi-Client Support (Server)

- Server handles multiple clients concurrently using **threading**, allowing real-time communication with several users.

File Transfer

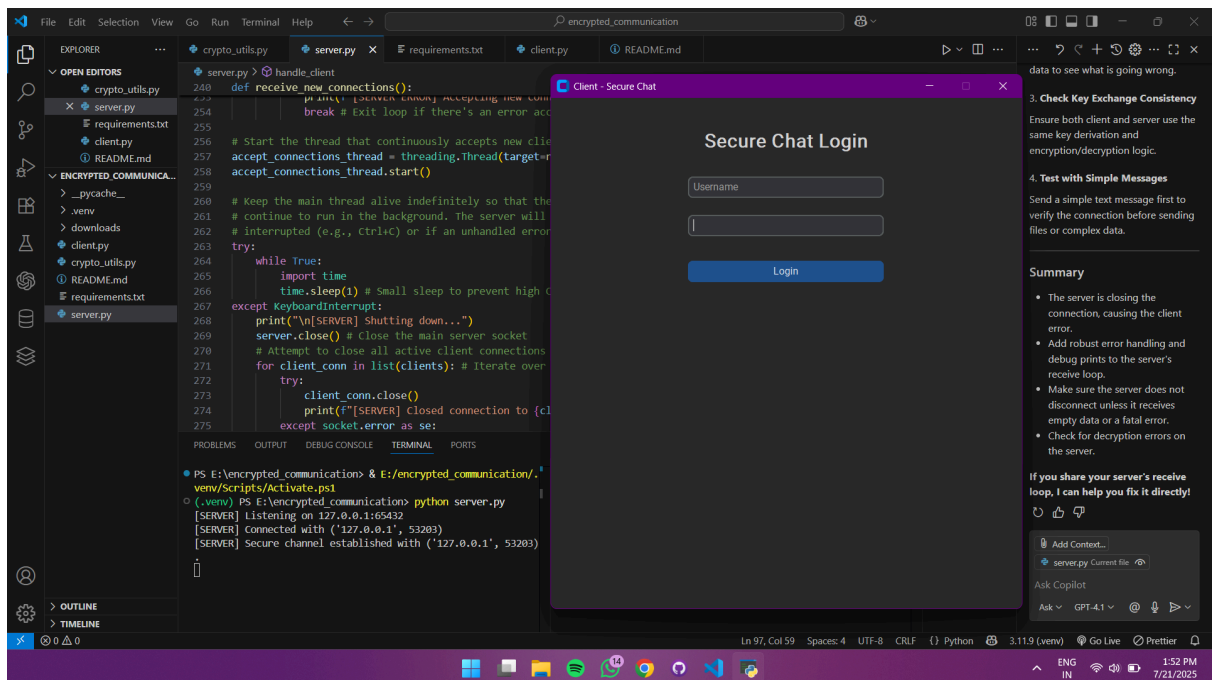
- Supports **encrypted file sharing**.

- Files are distinguished using a **b'FILE'** prefix and saved after decryption on the server.



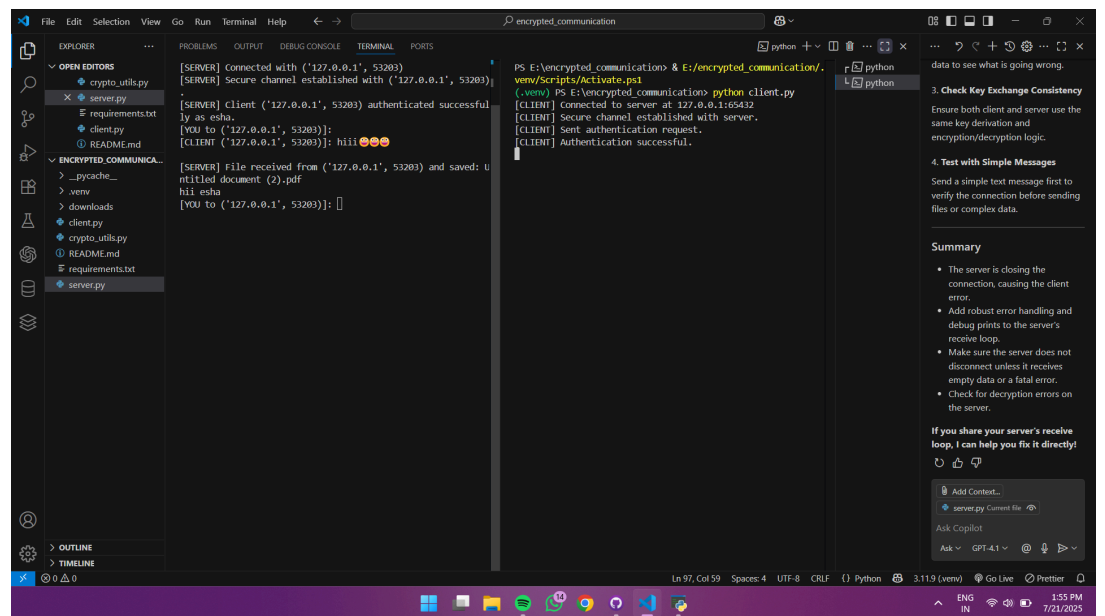
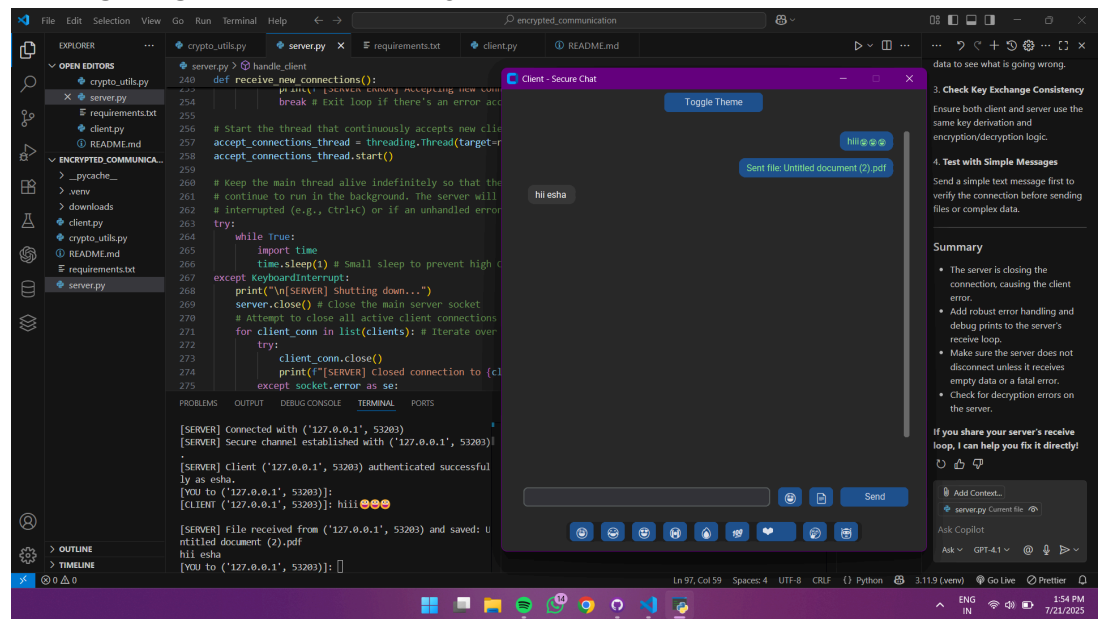
🔒 User Authentication

- Users must **login** with a **username** and **password** before chatting.
- Server validates credentials (currently hardcoded for demo purposes).



💬 Interactive GUI (Client)

- Built using **customtkinter**.
- Features:
 - **Dark/Light themes**
 - **Emoji support**
 - **Typing indicators** (like WhatsApp)
 - **Message alignment** (sent from right, received from left)



Robust Data Transfer

- Uses **length-prefixing** to ensure accurate message/file transmission.
 - Prevents message truncation or partial reads over the socket.
-

3. Architecture

Server (**server.py**)

- Listens for incoming connections
- Performs:
 - Secure key exchange
 - User authentication
 - Message decryption & broadcast
 - File saving (with decryption)
- Creates a thread per client for real-time handling

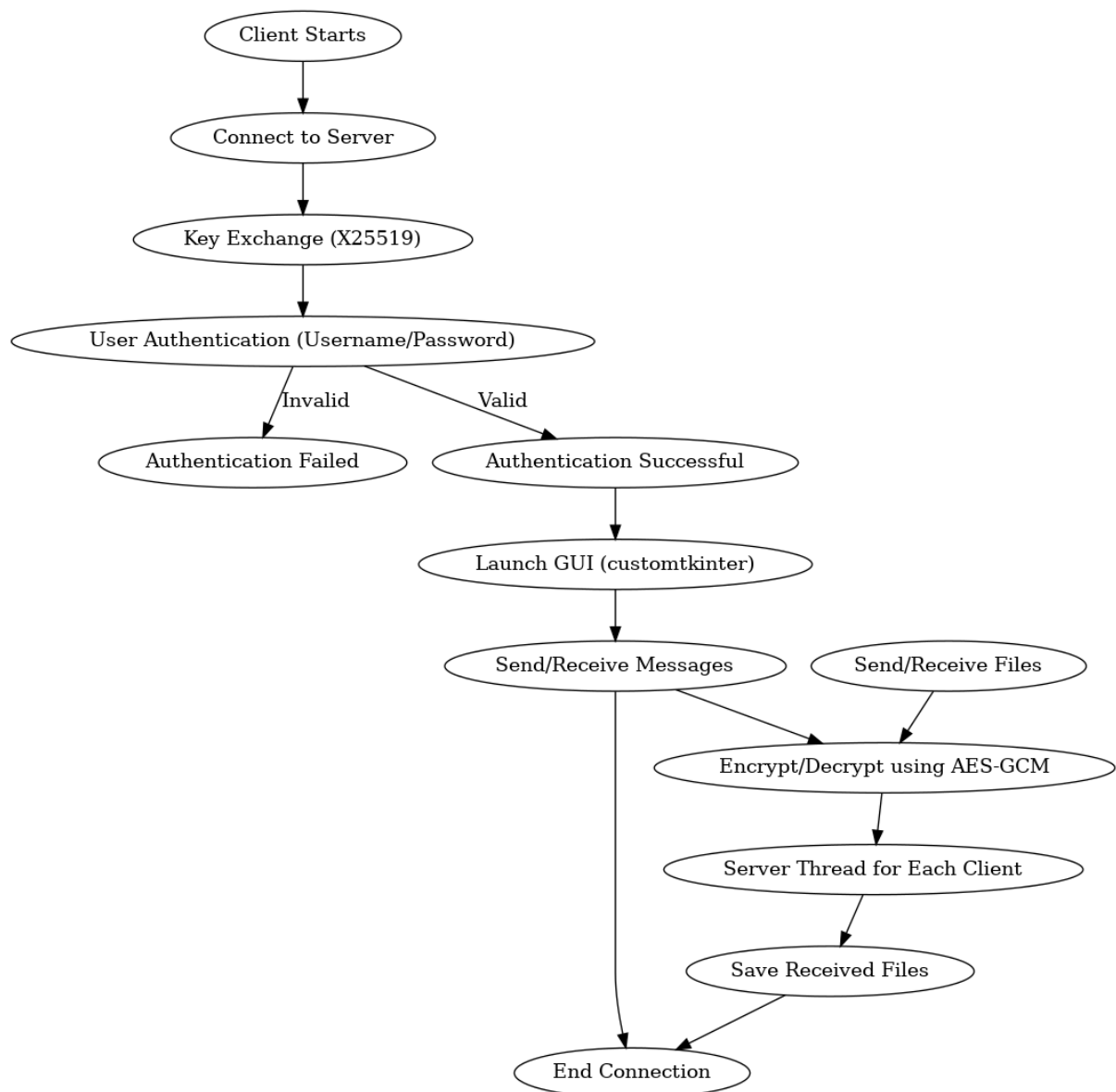
Client (**client.py**)

- Connects to server, performs key exchange & authentication
- Provides a GUI chat interface with:
 - Real-time messaging
 - File upload
 - Emoji picker
 - Typing indicator
 - Theme toggle

Crypto Utilities (**crypto_utils.py**)

- Manages:
 - Key generation
 - Shared secret derivation
 - AES encryption/decryption
 - Public key serialization/deserialization
-

4. Flowchart



5. Methodology

Foundational Setup

- Established TCP connection using `socket` module
- Built multi-threaded server for concurrency

Cryptographic Core

- Implemented `crypto_utils.py`

- Integrated **X25519**, **AES-GCM**, and **HKDF**

Reliable Data Transfer

- Added **4-byte length-prefixing** using `struct`
- Ensured complete message reads

GUI Development

- Used `customtkinter` to create modern GUI
- Added:
 - Message panels
 - Emoji picker
 - Typing status display
 - Theme toggle

Authentication Layer

- Added login screen to client
- Server checks credentials before allowing chat

File Transfer Implementation

- Enabled encrypted file sharing from client to server
- Used `filedialog` for file selection
- Server saves files post-decryption

Testing & Refinement

- Stress-tested encryption, communication, and file transfer
- Handled edge cases (incomplete reads, disconnects)

- Improved feedback and error messages

6. Technical Details

Category	Technology Used
Programming Language	Python 3.x
Networking	<code>socket</code> module (TCP/IP)
Concurrency	<code>threading</code> module
Cryptography	<code>cryptography</code> package
GUI Framework	<code>customtkinter</code>
Data Structuring	<code>struct</code> module (for message framing)
File Handling	<code>os</code> , <code>filedialog</code> from <code>tkinter</code>

7. Conclusion

This project demonstrates how **cryptography**, **networking**, and **GUI design** can be combined to build a **real-time secure communication system**. It provides a complete client-server model with a **focus on security, usability, and scalability**. Future improvements may include:

- Encrypted message storage
 - Group chat support
 - Voice/video calling
 - Database-backed user authentication
-