

Peer to Peer Communication App

(python)

**Project BY:**

**Abdul Muizz : 22f-3260**

**Muhammad Usman : 22f-3285**

**ABSTRACT**

Peer-to-Peer Encrypted Chat Application

This project implements a secure, multi-featured chat application using Python's socket programming and Tkinter GUI framework. The system enables users to create accounts, log in, and engage in both broadcast and private messaging with robust security features. Key functionalities include end-to-end message encryption, file sharing, emoji integration, and message management capabilities such as deletion. The client-server architecture supports multiple simultaneous users, with the server facilitating message routing and connection management. User authentication is handled through a local credential storage system, while communication leverages socket threading to enable real-time message exchange. The application demonstrates comprehensive networking techniques, graphical interface design, and practical security implementations in a desktop chat platform

**P2P Chat Application Technical Report**

### A Comprehensive Analysis of Features and Functionalities

## LAYOUT OF REPORT

1. Introduction
2. System Architecture
3. Core Features and Functionalities
4. Security Implementation
5. User Interface Design
6. File Transfer System
7. Technical Implementation Details
8. Future Enhancements
9. Conclusion

## 1. Introduction

The P2P Chat Application is a robust communication platform developed in Python, utilizing socket programming and threading for real-time message exchange. This report provides a detailed analysis of its features, architecture, and implementation.

## 

## 2. System Architecture

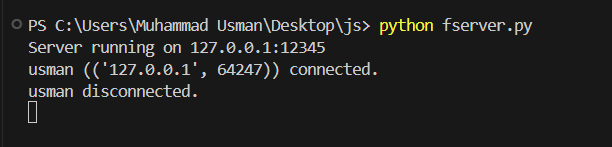
The application follows a client-server architecture with the following components:

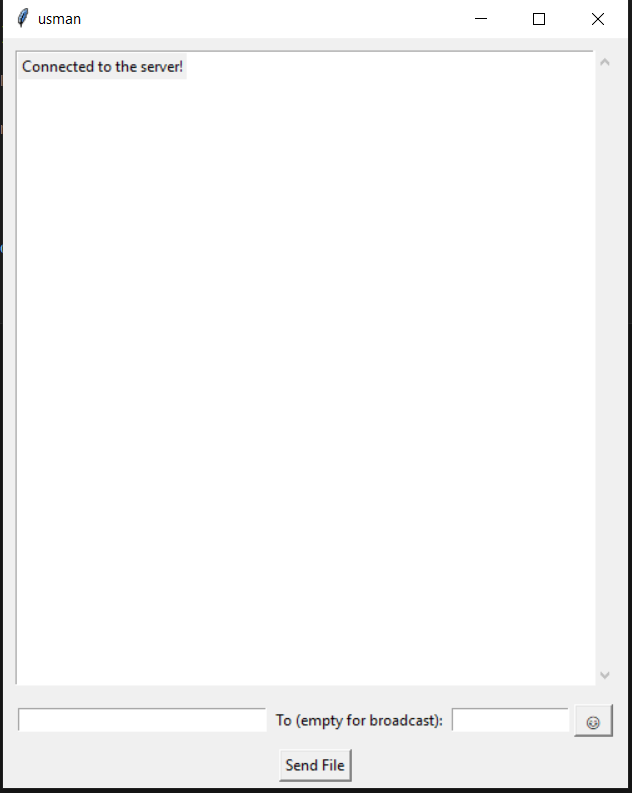
* Server Component (fserver.py)
* Client Component (fclient.py)
* GUI Interface (Tkinter-based)
* Private File Transfer System
* Group file transfer
* Authentication System
* Group messaging
* Private messaging
* Encryption and decryption method

### 2.1 Server Architecture

The server operates on localhost (127.0.0.1) and port 12345, managing:

* Client connections
* Message routing
* File transfer coordination
* User session management

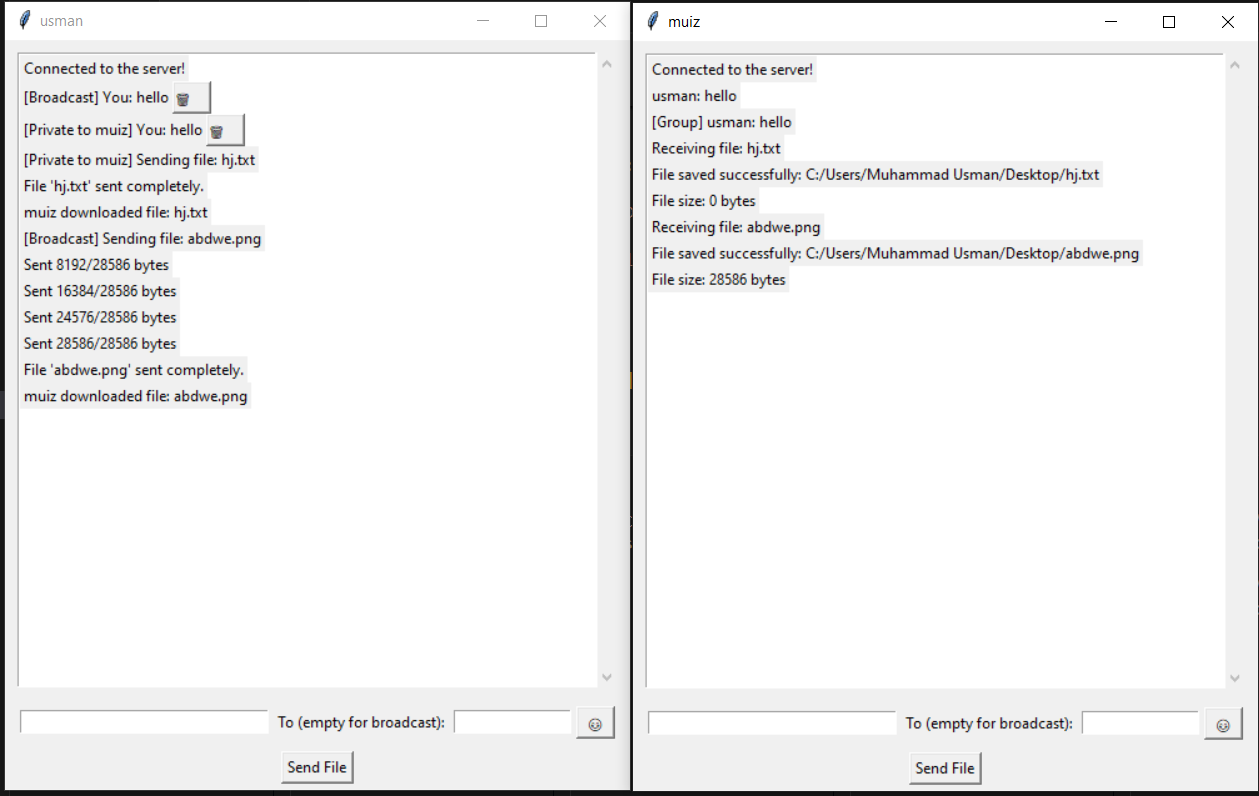




### 2.2 Client Architecture

The client component handles:

* User authentication
* Message sending/receiving
* File transfer operations
* GUI interactions



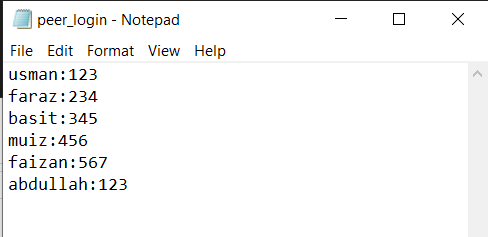
## 3. Core Features and Functionalities

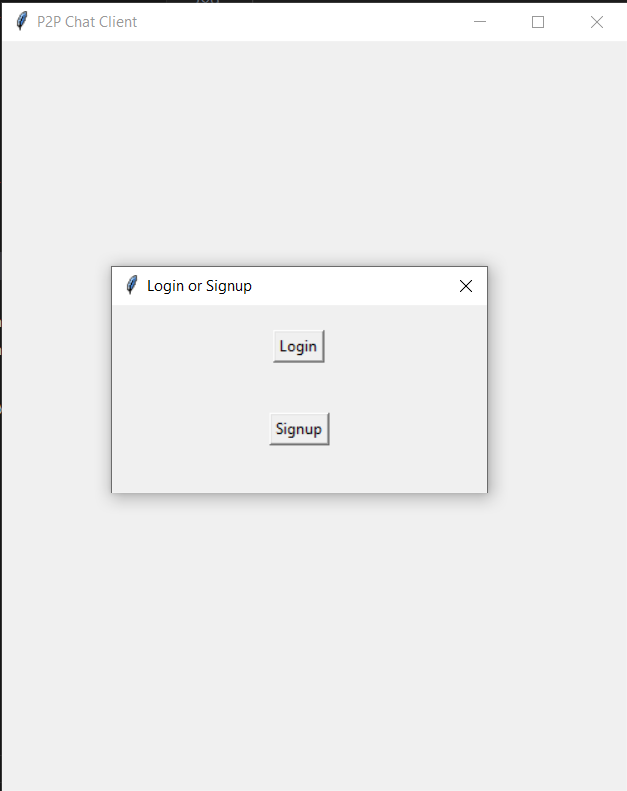
### 3.1 User Authentication System

* User Registration(SIGNUP)
  + Username creation
  + Password protection
  + Duplicate username prevention
  + Credential storage in peer\_login.txt
* Login System(LOGIN)
  + Username/password verification
  + Session management
  + Error handling for invalid credentials

WORKING OF LOGIN AND SIGN UP FUNCTIONING



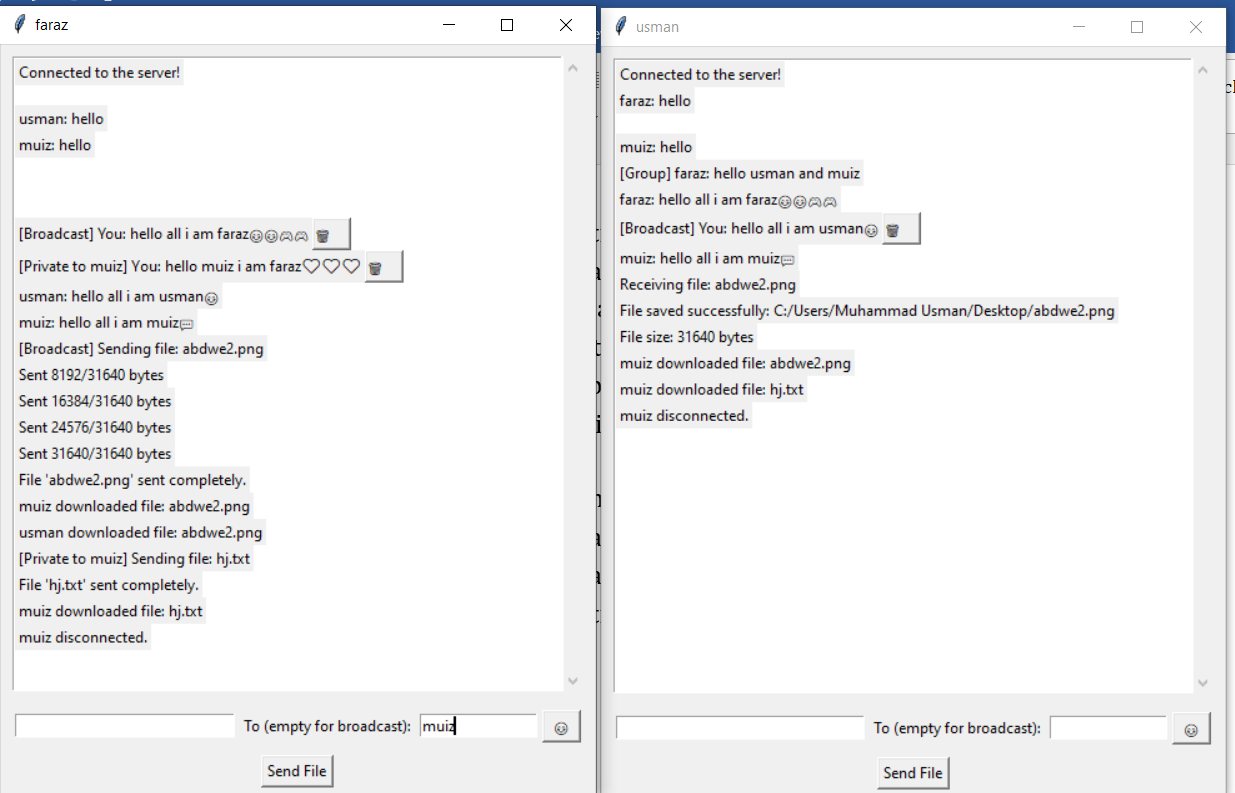




### 3.2 Messaging Features

1. Broadcast Messaging
   * Send messages to all connected users
   * Real-time message delivery
   * Message tracking with unique IDs
2. Private Messaging
   * Direct messages to specific users
   * Group messaging support
   * Recipient selection interface
3. Message Management
   * Message deletion capability
   * Message history display
   * Real-time message updates

Message sending and receiving video  

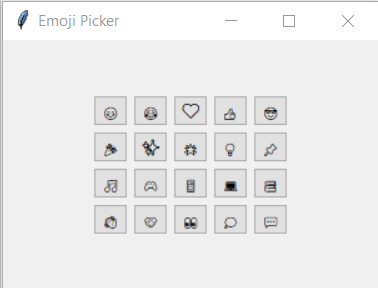



### 3.3 Emoji Support

* Built-in emoji picker
* Common emoji selection
* Quick emoji insertion
* Custom emoji grid layout

Emoji support video



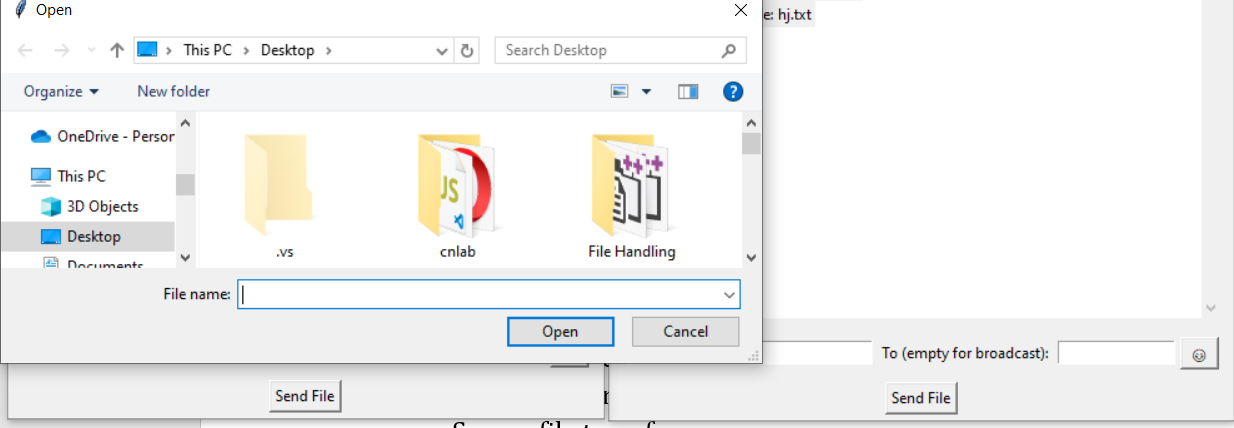
EMOJI  


### 3.4 File Transfer System

1. File Sending Capabilities
   * Support for all file types
   * Progress tracking
   * File size display
   * Chunk-based transfer
2. File Receiving Features
   * Save location selection
   * Download progress display
   * File integrity verification
   * Download completion notification

File sending and receiving video

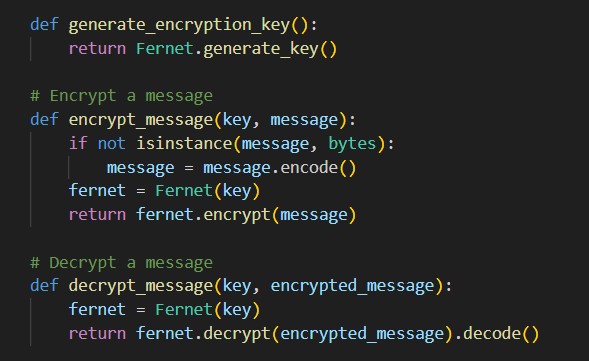




## 4. Security Implementation

### 4.1 Encryption System

* Fernet encryption implementation
* Secure key generation
* Message encryption/decryption
* Secure file transfer

**ENCRYPTION/DECRYPTION  
**

### 4.2 Authentication Security

* Password protection
* Secure credential storage
* Session observation
* Connection security

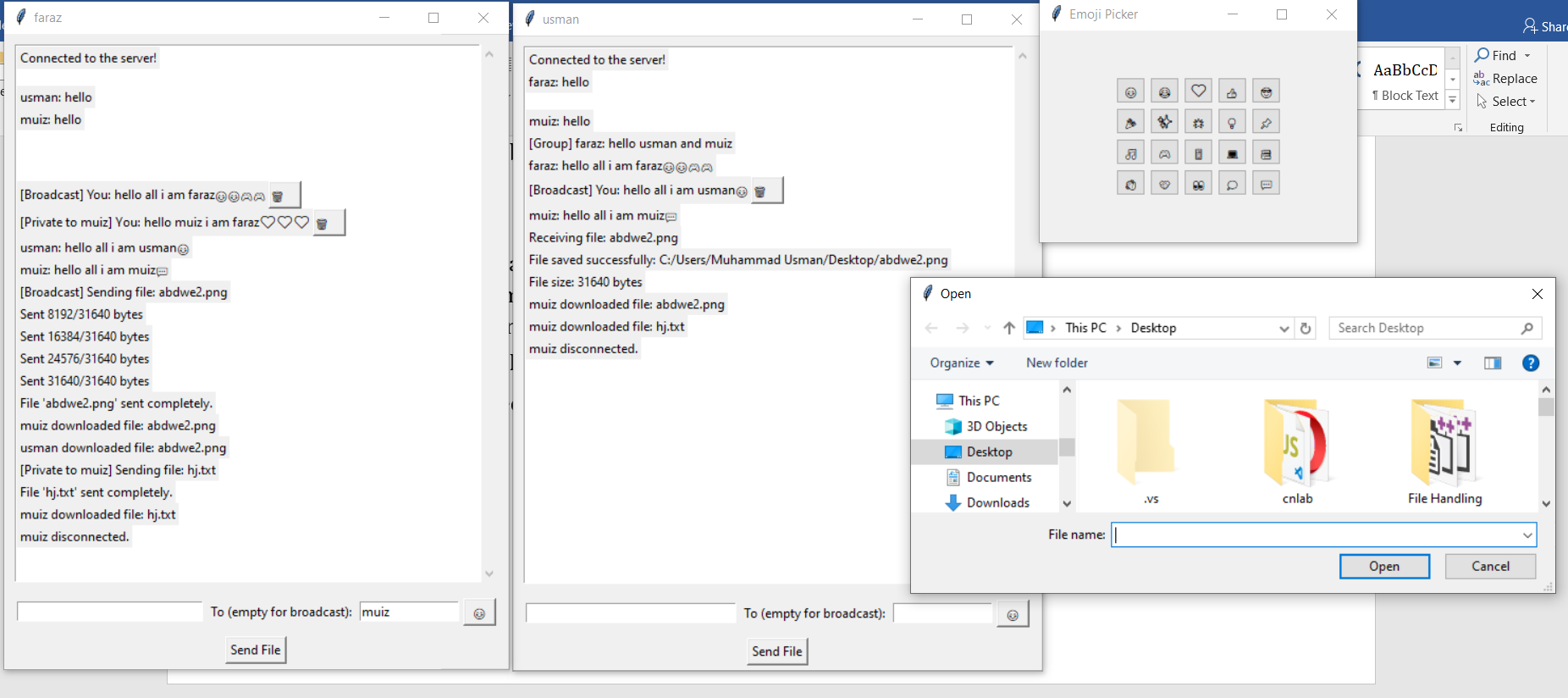
## 5. User Interface Design

### 5.1 Main Interface Components

* Chat display area
* Message input field
* Recipient selection
* File transfer buttons
* Emoji picker button

### 5.2 Visual Elements

* Scrollable chat history
* Message formatting
* User status indicators
* File transfer progress bars



## 6. File Transfer System

### 6.1 Technical Implementation

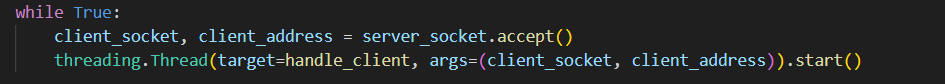
* Chunk-based transfer (8192 bytes)
* Binary data handling
* Progress tracking
* Error handling

### 6.2 User Features

* File selection dialog
* Save location selection
* Transfer progress display
* Completion notifications

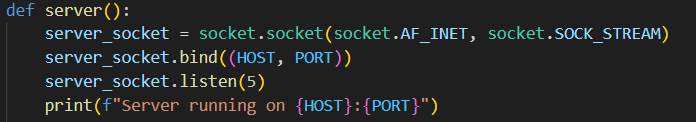
## 7. Technical Implementation Details

### 7.1 Threading Implementation

* Separate threads for:
  + Message receiving
  + File transfer
  + GUI updates
  + Server communications  
    

### 7.2 Socket Programming

* TCP/IP implementation
* Connection management
* Data packet handling
* Error recovery



### 7.3 Error Handling

* Connection loss recovery
* File transfer interruption handling
* Invalid input management
* Network error handling

## 8. Future Enhancements

Potential improvements for future versions:

1. End-to-end encryption

2. Voice and video chat capabilities

3. File sharing optimization

4. Advanced message formatting

5. User profile customization

6. Message search functionality

7. Offline message storage

8. Multi-server support

## 9. Conclusion

The P2P Chat Application demonstrates robust implementation of essential communication features while maintaining security and user-friendly interface. The combination of real-time messaging, file transfer capabilities, and emoji support makes it a versatile communication tool suitable for various use cases.

**P2P communication APP in Action**  
