Documentation for Chatroom Application

General Information

- **Python Version:** 3.11.7
- **Platform:** Multi-platform (Windows, macOS, Linux compatible)
- **Dependencies:**
  - Python standard libraries: `threading`, `socket`, `argparse`, `os`, `sys`, `time`
  - GUI libraries: `tkinter`, `customtkinter`, `PIL`

Client-Side Documentation

Overview

The client side of the chatroom application provides a graphical user interface for users to send and receive messages and files in real-time. It is responsible for handling user inputs, displaying incoming messages, and managing file transfers.

Key Components

Client Class

Main class that initializes the connection and the GUI.

- **start()**: Establishes the socket connection to the server, prompts user for a username, and launches the GUI setup.
- **send_message(textInput)**: Sends the user's message to the server.
- **request_online_users()**: Requests a list of currently online users from the server.
- **send_file()**: Opens a file dialog to select a file, then sends the selected file to the server.
- **setup_gui()**: Sets up the Tkinter GUI, including message list, entry fields, and buttons.

Send Class (threading.Thread)

Handles outgoing messages.

- **run()**: Continuously prompts the user for input and sends it to the server.

Receive Class (threading.Thread)

Handles incoming messages.

- **run()**: Listens for messages from the server and updates the GUI accordingly.
- **receive_file(filename, filesize)**: Receives file data from the server and writes it to the local file system.

User Interface

Uses `customtkinter` for a modern look and feel. Features include:

- Text entry for sending messages.
- Buttons for sending messages, sending files, and requesting online user lists.
- Real-time update of message history.

Server-Side Documentation

Overview

The server component manages all network connections, handles incoming data from clients, and distributes messages and files to other connected clients.

Key Components

Server Class (threading.Thread)

Manages client connections and broadcasts messages.

- **start_server()**: Initializes the server socket and listens for incoming connections.
- **sendAll(message, source)**: Broadcasts a message to all clients except the sender.
- **get_online_users()**: Compiles a list of all currently connected users.
- **remove_connection(client_socket)**: Removes a client from the list of active connections.
- **check_inactive_connections()**: Periodically checks for inactive connections and closes them.
- **broadcast_file(source_socket, filename, file_data)**: Sends a received file to all clients except the sender.

ServerSocket Class (threading.Thread)

Dedicated thread for handling communication with each client.

- **run()**: Manages all incoming messages or commands from a specific client and responds accordingly.
- **handle_file_transfer(initial_msg)**: Handles the reception and redistribution of files sent by clients.

Networking and Concurrency

Uses TCP sockets for reliable, ordered, and error-checked delivery of streams of bytes. Implements multithreading to handle multiple client connections simultaneously, ensuring responsive and scalable server performance.

Error Handling

Includes robust error handling to manage client disconnections, transmission errors, and exceptions, ensuring server stability.

Usage Instructions

1. **Starting the Server**: Run the server script on a machine, specify the host address and port. Ensure the server is accessible to all clients.
2. **Connecting as a Client**: Run the client script, connect to the server's IP address and port, and enter a username.
3. **Chat Interface**: Use the chat interface to send messages and files. Online users can be requested via a dedicated button.