```python
# 2024 Â© Idan Hazay client.pyw
# Import required libraries

from modules.logger import Logger  # Custom logging module
from modules import networking, receive, gui, dialogs  # Importing necessary modules

import socket, sys, traceback  # Standard libraries

from PyQt6 import QtWidgets  # PyQt6 for GUI handling


class Application:
    """
    Handles the initialization of the PyQt application, networking,
    and GUI setup for client-side operations.
    """

    def __init__(self):
        sys.excepthook = dialogs.global_exception_handler  # Set a global exception handler for unhandled exceptions
        self.qtapp = QtWidgets.QApplication(sys.argv)  # Initialize PyQt application

        self.network = networking.Network()  # Initialize networking module
        self.window = gui.MainWindow(self.qtapp, self.network)  # Initialize main GUI window
        self.start_app()  # Start the application loop

        sys.exit(self.qtapp.exec())  # Start the PyQt event loop and exit when it finishes

    def start_app(self):
        """
        Starts the application by displaying the main window,
        initiating the connection page, and setting up the receive thread.
        """
        self.window.show()
        self.window.not_connected_page(False)  # Show the "not connected" page initially

        self.receive_thread = receive.ReceiveThread(self.network)  # Initialize background thread for receiving data
        self.receive_thread.reply_received.connect(self.handle_reply)  # Connect received replies to handler

        self.window.receive_thread = self.receive_thread  # Attach the receive thread to the main window
        self.window.protocol.connect_server(loop=True)  # Attempt to connect to the server

    def handle_reply(self, reply):
        """
        Handles replies received from the server.
        Parses the response and handles errors or disconnects if necessary.
        """
        try:
            self.network.logtcp('recv', reply)  # Log received data

            to_show = self.window.protocol.protocol_parse_reply(reply)  # Parse the server's reply
            print(to_show)

            if to_show == "Invalid reply from server":
                print(reply)

            # If exit request is acknowledged, disconnect
            if to_show == "Server acknowledged the exit message":
                print('Successfully exited')
                self.network.sock.close()
                sys.exit()

        except socket.error as err:
            print(traceback.format_exc())
            return
        except Exception as err:
            print(traceback.format_exc())
            return

def main():
    """
    Main function to initialize and start the client application.
    Sets up secure connection and GUI for user interaction.
    """
    app = Application()  # Initialize the client application

if __name__ == "__main__":  # Run the main function if the script is executed directly
    sys.stdout = Logger()  # Redirect standard output to the custom logger
    main()
```