Requirements

### **1. Requirement Tính Năng (Functional Requirements)**

1. Giao diện dòng lệnh (Command Line Interface - CLI):
   1. Ứng dụng phải hoạt động qua giao diện dòng lệnh và cung cấp các lệnh như help, myip, myport, connect, list, terminate, send, và exit.
2. Hiển thị thông tin:
   1. **help**: Hiển thị hướng dẫn sử dụng các lệnh.
   2. **myip**: Hiển thị địa chỉ IP của máy đang chạy chương trình.
   3. **myport**: Hiển thị port mà chương trình đang lắng nghe kết nối.
3. Kết nối TCP giữa các peer:
   1. **connect** <**destination**> <**port**>: Tạo kết nối TCP đến một peer khác qua IP và port được chỉ định.
   2. Cần xử lý các trường hợp:
      1. IP không hợp lệ.
      2. Tự kết nối (self-connections).
      3. Kết nối trùng lặp.
4. Quản lý danh sách kết nối:
   1. **list**: Hiển thị danh sách các kết nối hiện tại, bao gồm cả kết nối được tạo bởi chương trình và kết nối từ các peer khác.
5. Quản lý kết nối:
   1. **terminate** <**connection** **id**>: Ngắt kết nối với một peer dựa trên ID từ danh sách list.
   2. Hiển thị lỗi nếu ID không hợp lệ hoặc kết nối không tồn tại.
6. Trao đổi tin nhắn:
   1. **send** <**connection** **id**> <**message**>: Gửi tin nhắn tới peer được chỉ định (ID từ danh sách list).
   2. Hiển thị thông báo khi tin nhắn được gửi thành công.
   3. Peer nhận tin nhắn cần hiển thị thông tin:
      1. Địa chỉ IP của người gửi.
      2. Port của người gửi.
      3. Nội dung tin nhắn.
7. Thoát khỏi ứng dụng:
   1. **exit**: Đóng tất cả các kết nối hiện tại và thoát chương trình.
   2. Các peer khác cần cập nhật danh sách kết nối để loại bỏ peer vừa thoát.

### **2. Requirement Phi Tính Năng (Non-Functional Requirements)**

1. Hiệu năng:
   1. Ứng dụng phải xử lý nhiều kết nối đồng thời (multi-threading hoặc asynchronous I/O).
   2. Đảm bảo không bị treo (hang) khi có nhiều kết nối hoặc khi có peer ngắt kết nối đột ngột.
2. Độ tin cậy:
   1. Ứng dụng phải xử lý tốt các lỗi như:
      1. Kết nối bị ngắt.
      2. Peer không phản hồi.
      3. Tin nhắn vượt quá độ dài tối đa (100 ký tự).
   2. Hiển thị thông báo lỗi rõ ràng và thân thiện với người dùng.
3. Không bị rò rỉ bộ nhớ (Memory Leak):
   1. Đảm bảo giải phóng tài nguyên (socket, bộ nhớ) khi kết nối bị ngắt hoặc khi chương trình thoát.
   2. Sử dụng các công cụ kiểm tra rò rỉ bộ nhớ (như Valgrind) nếu lập trình bằng C/C++.
4. Tuân thủ quy tắc lập trình (Coding Convention):
   1. Sử dụng các quy tắc đặt tên biến, hàm, và cấu trúc mã nguồn rõ ràng, dễ đọc.
   2. Tách biệt logic (logic xử lý) và giao diện (CLI) để dễ bảo trì.
   3. Thêm comment giải thích các đoạn mã quan trọng.
5. Mã nguồn sạch (Clean Code):
   1. Tránh các đoạn mã lặp lại (code duplication).
   2. Sử dụng các hàm/methods để chia nhỏ logic phức tạp.
   3. Đặt tên biến và hàm có ý nghĩa, dễ hiểu.
   4. Xử lý lỗi (error handling) đầy đủ, không để chương trình bị crash.