

Lab

3

BÁO CÁO BÀI THỰC HÀNH SỐ 3

Lập trình Sockets trong C#

Môn học: Lập trình mạng căn bản

Sinh viên thực hiện	Nguyễn Duy Hưng (24520602)
Số câu đã hoàn thành	6/6

Câu 1. Viết ứng dụng thực hiện gửi và nhận dữ liệu giữa hai bên sử dụng giao thức UDP (UDP Client và UDP Server).

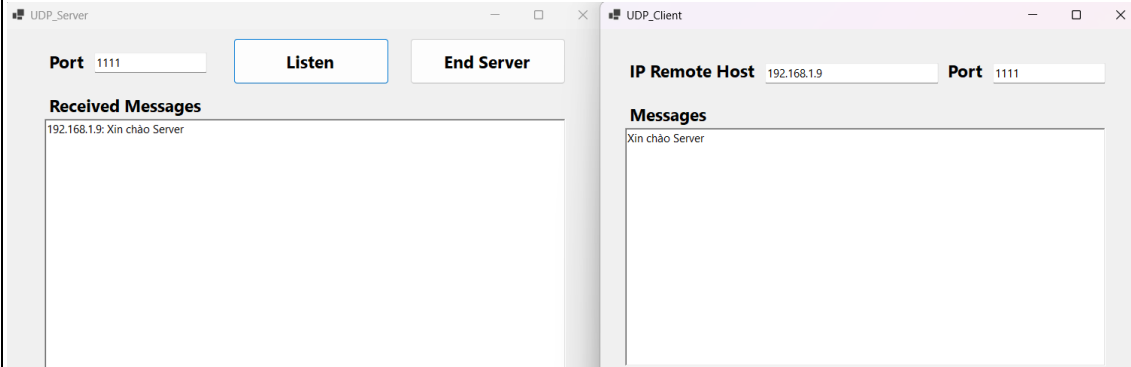
Trả lời:

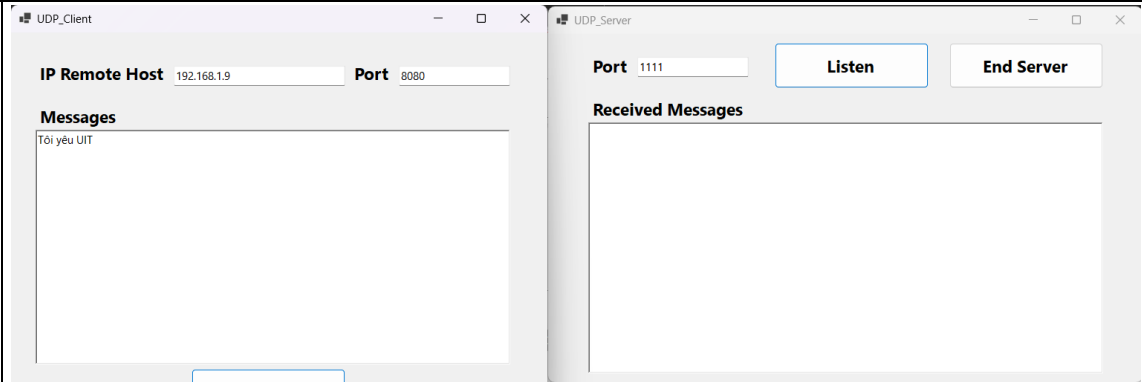
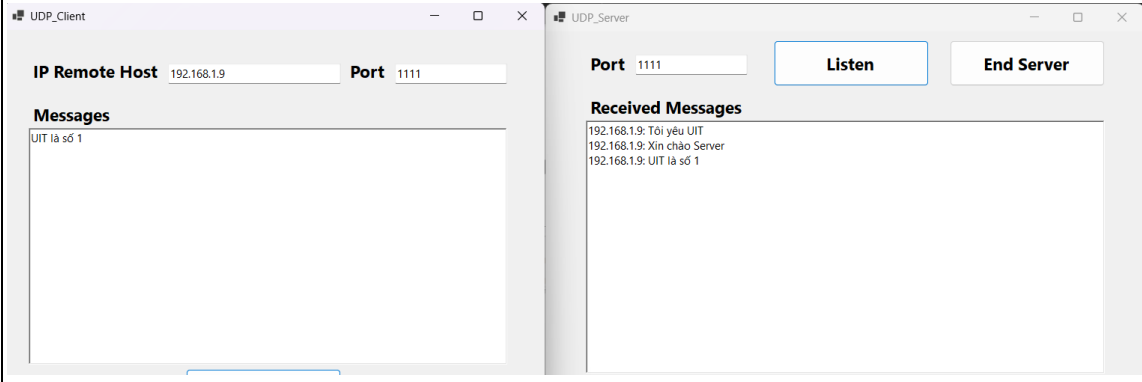
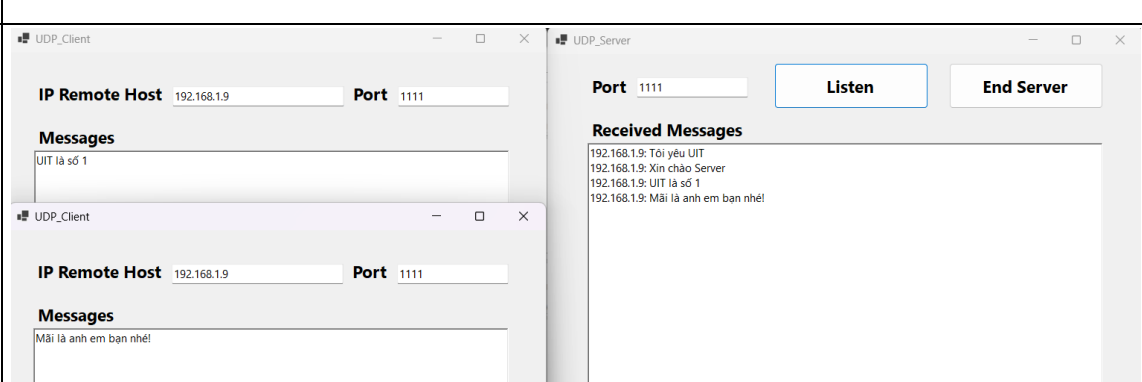
Ý tưởng chính: UDP Client nhập IP, port, thông điệp → gửi qua mạng → UDP Server lắng nghe tại port đó → nhận và hiển thị thông điệp lên màn hình. Cả hai dùng UdpClient, và Server sử dụng Thread để lắng nghe liên tục.

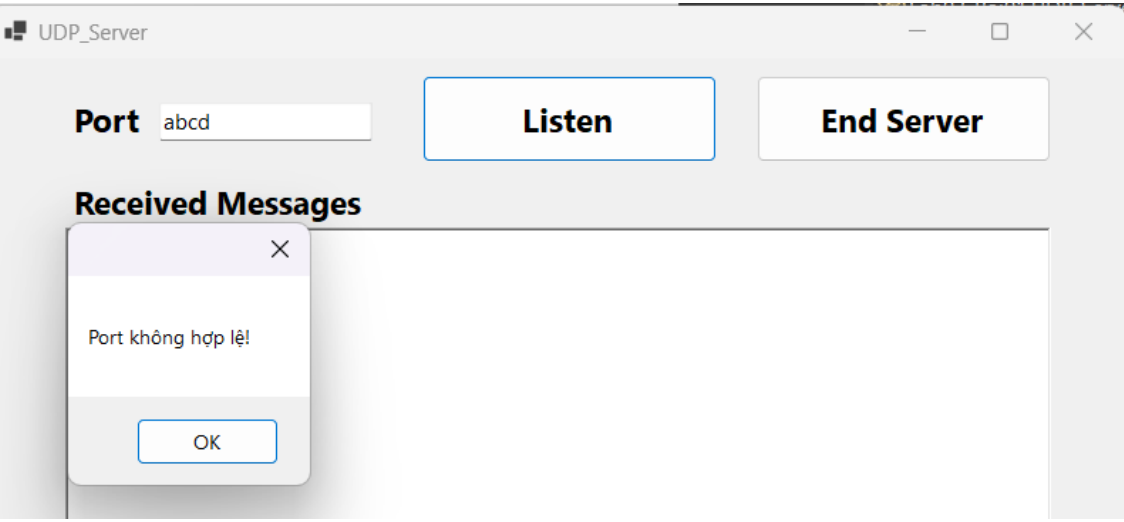
Các bước thực hiện chính:

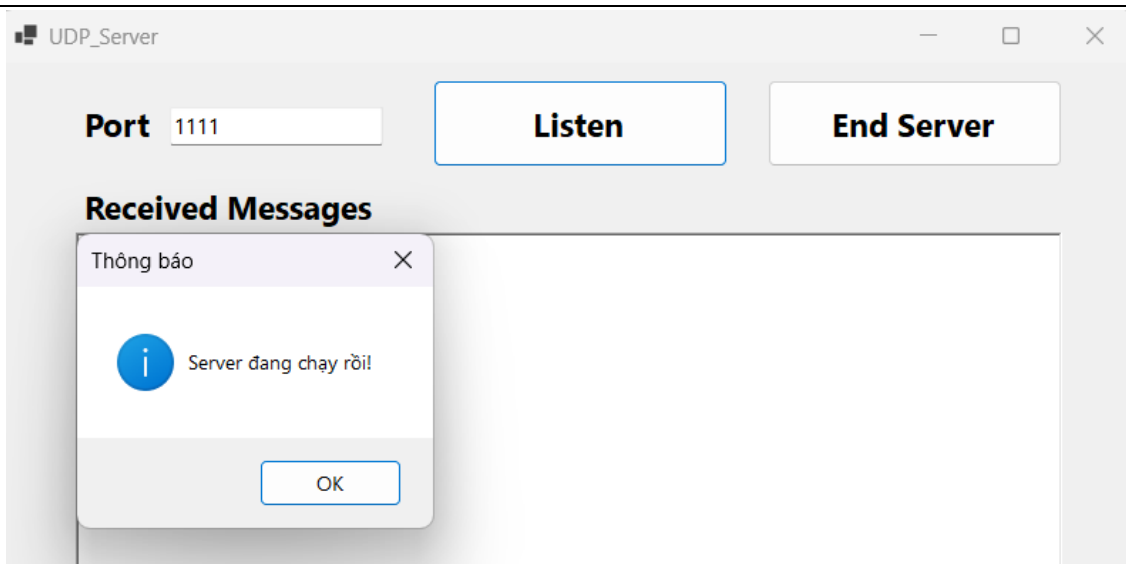
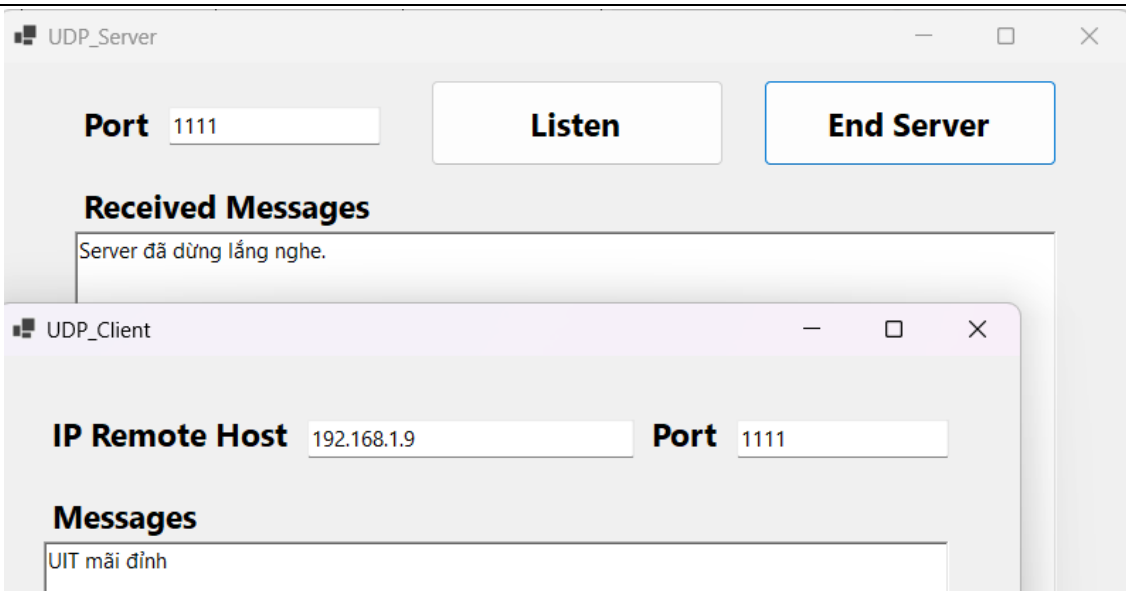
- 1. Nhận sự kiện: Người dùng nhấn nút Listen trên Server hoặc Send trên Client*
- 2. Client gửi dữ liệu: Hệ thống lấy IP, port và nội dung tin nhắn từ giao diện rồi gửi qua mạng bằng UdpClient*
- 3. Server nhận dữ liệu: Hệ thống lắng nghe tại cổng được nhập, nhận gói tin từ Client và giải mã nội dung*
- 4. Hiển thị kết quả: Server xuất nội dung nhận được (kèm địa chỉ IP) lên khung hiển thị, Client thông báo gửi thành công*

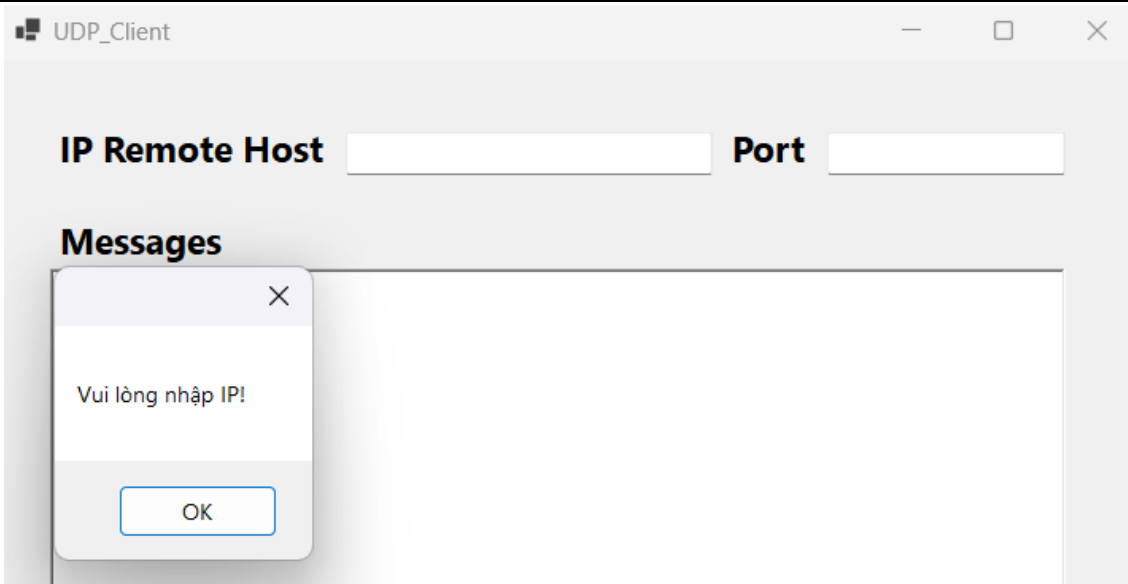
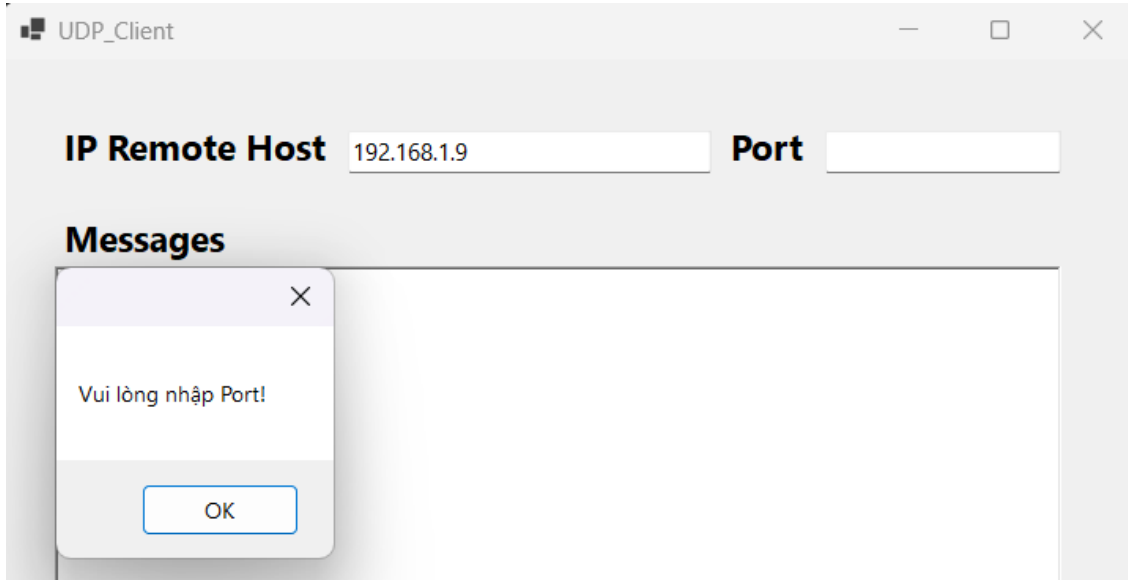
Test case:

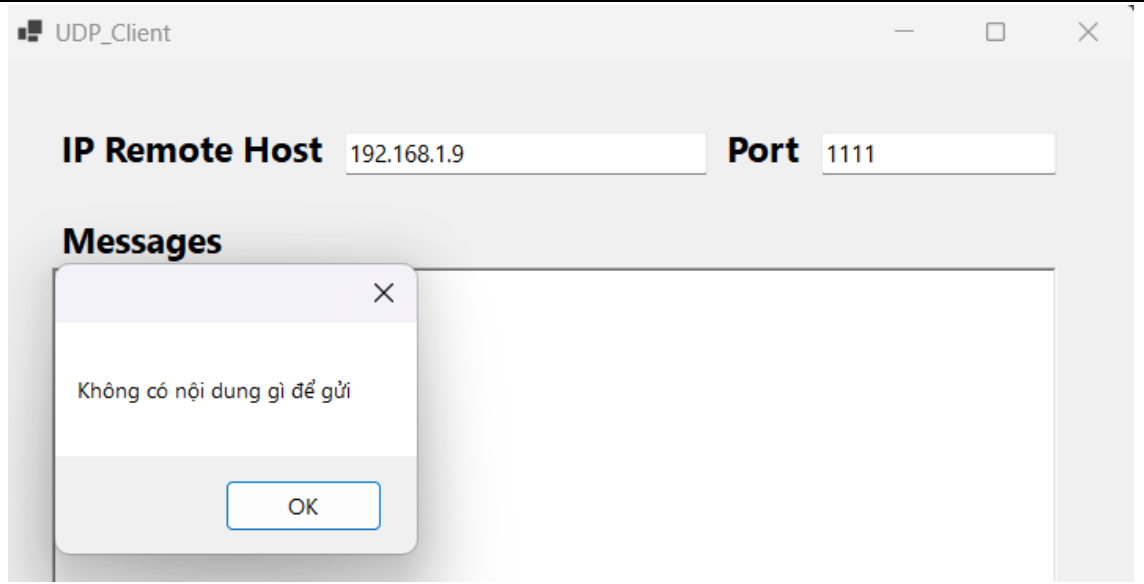
Mã TC	Mô tả	Dữ liệu đầu vào	Kết quả mong đợi	Hình ảnh kết quả
TC1.1	Gửi tin nhắn cơ bản	IP: 192.168.1.9 Server Port: 1111 Client Port: 1111 Message: "Xin chào Server"	Server nhận và hiển thị "192.168.1.9: Xin chào Server"	 <p>The screenshot shows two windows side-by-side. The left window, titled 'UDP_Server', has a 'Port' field set to '1111' and a 'Listen' button. Below it, a 'Received Messages' box displays '192.168.1.9: Xin chào Server'. The right window, titled 'UDP_Client', has an 'IP Remote Host' field set to '192.168.1.9' and a 'Port' field set to '1111'. Below it, a 'Messages' box displays 'Xin chào Server'.</p>

TC1.2	Gửi sai port	IP: 192.168.1.9 Server Port: 1111 Client Port: 8080 Message: “Tôi yêu UIT”	Server không nhận được tin	
TC1.3	Gửi nhiều lần liên tục	IP: 192.168.1.9 Server Port: 1111 Client Port: 1111 Message: “Tôi yêu UIT”, “Xin chào Server”, “UIT là số 1”	Server nhận đủ, hiển thị tuần tự	
TC1.4	Nhiều Client gửi cùng lúc	Client 1: IP: 192.168.1.9 Client Port: 1111 Message: “Tôi yêu UIT”, “Xin chào Server”, “UIT là số 1” Client 2:	Server nhận đủ, hiển thị tuần tự	

		IP: 192.168.1.9 Client Port: 1111 Message: “Mãi là anh em bạn nhé!”		
TC1.5	Port server không hợp lệ	Server Port: abcd	Hiện thị thông báo: “Port không hợp lệ”	

TC1.6	Nhấn Listen ở Server nhiều lần	Nhấn Listen lần thứ 2	Hiển thị thông báo: “Server đang chạy rồi!”	
TC1.7	Server đóng socket	Nhấn End Server	Hiển thị thông báo: “Server đã dừng lắng nghe” Client không thể gửi nội dung đến Server	

TC1.8	Client không nhập IP	IP rỗng	Hiển thị thông báo: “Vui lòng nhập IP!”	
TC1.9	Client không nhập Port	Port rỗng	Hiển thị thông báo: “Vui lòng nhập Port!”	

TC1.10	Client không nhập nội dung	Message rỗng	Hiển thị thông báo: “Không có nội dung gì để gửi”	
--------	----------------------------	--------------	---	--

Câu 2. Viết chương trình lắng nghe dữ liệu từ dịch vụ Telnet sử dụng kết nối TCP

Trả lời:

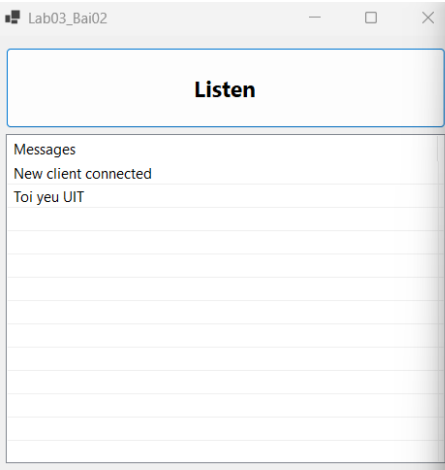
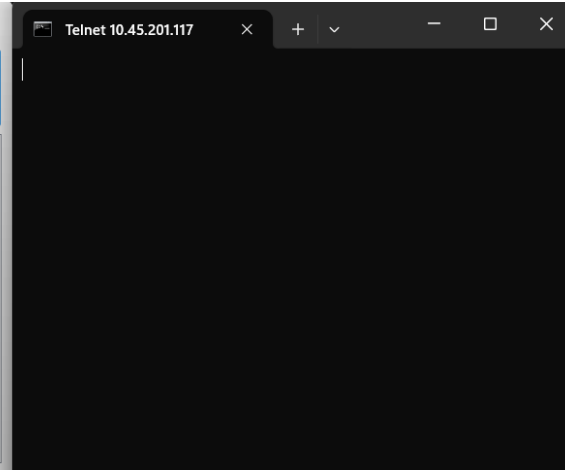
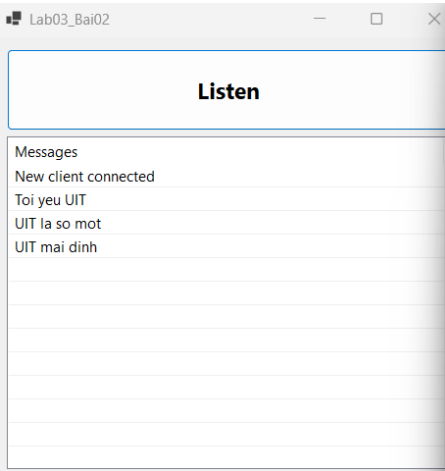

Ý tưởng chính: Chương trình sử dụng Socket TCP để lắng nghe (listen) kết nối đến từ dịch vụ Telnet (cổng 8080). Khi người dùng kết nối qua Telnet và gửi dữ liệu, chương trình nhận chuỗi ký tự và hiển thị lên giao diện

Các bước thực hiện chính:

1. Khởi tạo Socket TCP Server: Tạo socket với AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp. Gán IP và cổng (port 8080) cho socket
2. Lắng nghe kết nối: Gọi Listen() để chờ client (Telnet) kết nối. Khi có kết nối, chấp nhận bằng Accept()
3. Nhận dữ liệu: Tạo luồng (thread) để nhận dữ liệu từ client. Dùng Receive() để đọc chuỗi ký tự gửi từ Telnet
4. Hiển thị dữ liệu: Chuyển dữ liệu nhận được thành chuỗi (Encoding.ASCII). Ghi nội dung lên richtextbox trên giao diện

5. Đóng kết nối: Khi client ngắt Telnet, đóng socket và dừng lắng nghe

Test case:

Mã TC	Mô tả	Dữ liệu đầu vào	Kết quả mong đợi	Hình ảnh kết quả
TC2.1	Gửi 1 messages	Chuỗi: “Toi yeu UIT”	Phía Server nhận thông tin và xuất ra màn hình: “Toi yeu UIT”	 
TC2.2	Gửi nhiều messages	Chuỗi: “Toi yeu UIT”, “UIT la so mot”, “UIT mai dinh”	Phía Server nhận đầy đủ và hiển thị tuần tự	 

Câu 3. 1 Server – 1 Client Viết ứng dụng thực hiện gửi và nhận dữ liệu sử dụng giao thức TCP (TCP Client và TCP Listener)

Trả lời:

Ý tưởng chính: Chương trình mô phỏng việc truyền dữ liệu giữa 1 Server và 1 Client thông qua giao thức TCP.

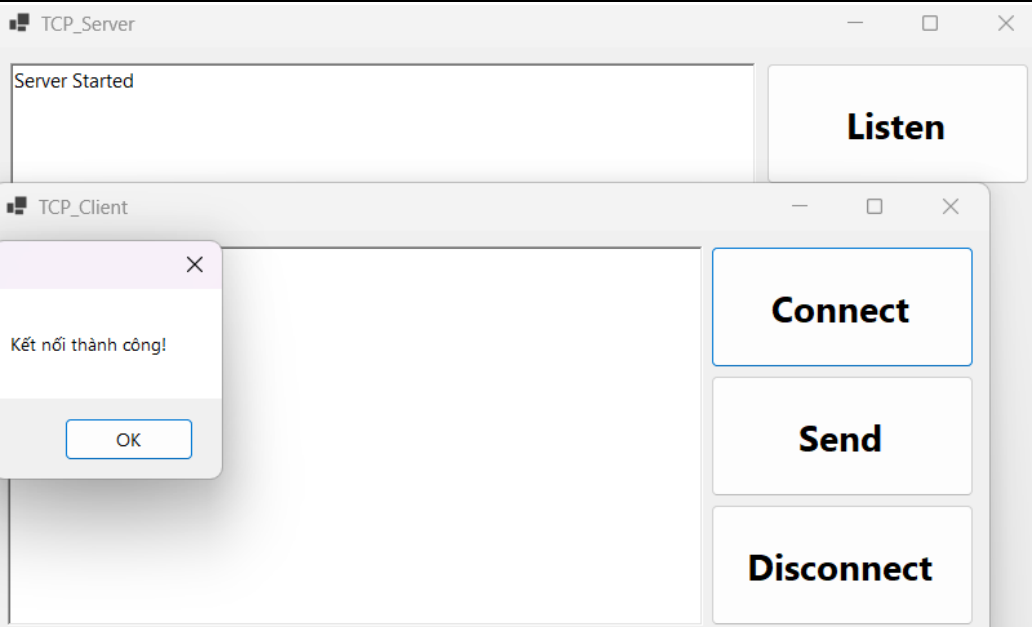
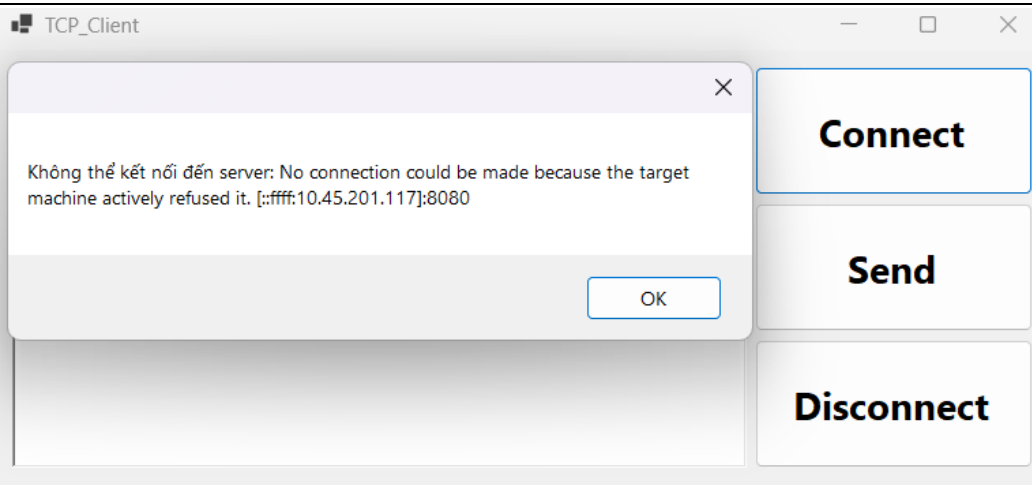
- *Server sử dụng lớp Socket để lắng nghe kết nối từ Client, nhận dữ liệu và hiển thị lên giao diện.*
- *Client sử dụng lớp TcpClient để kết nối đến Server qua địa chỉ IP và cổng (8080), sau đó gửi thông điệp mà người dùng nhập.*

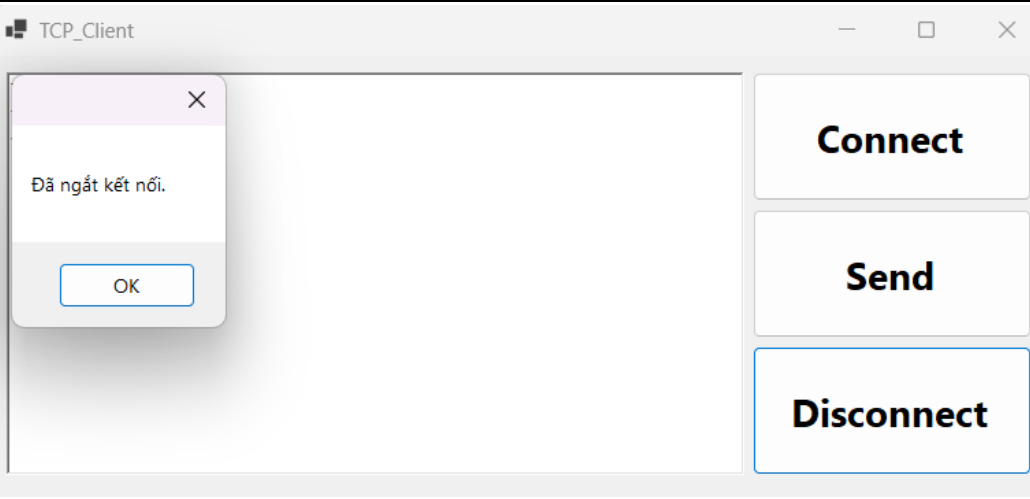
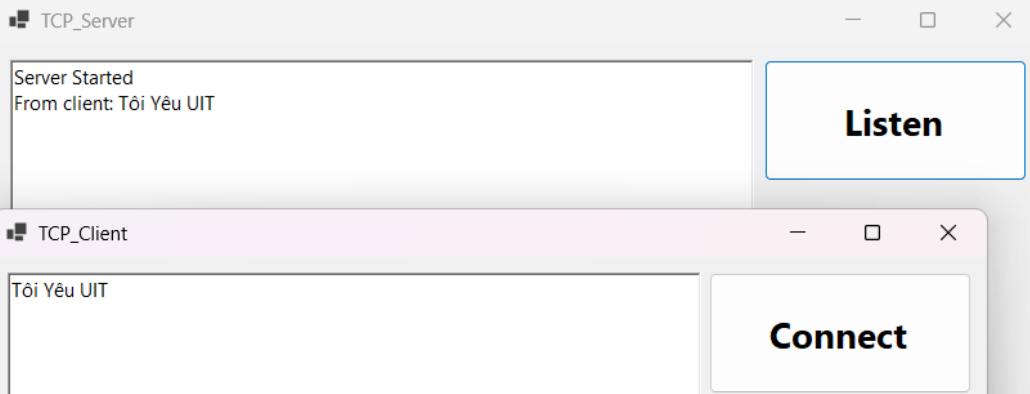
Các bước thực hiện chính:

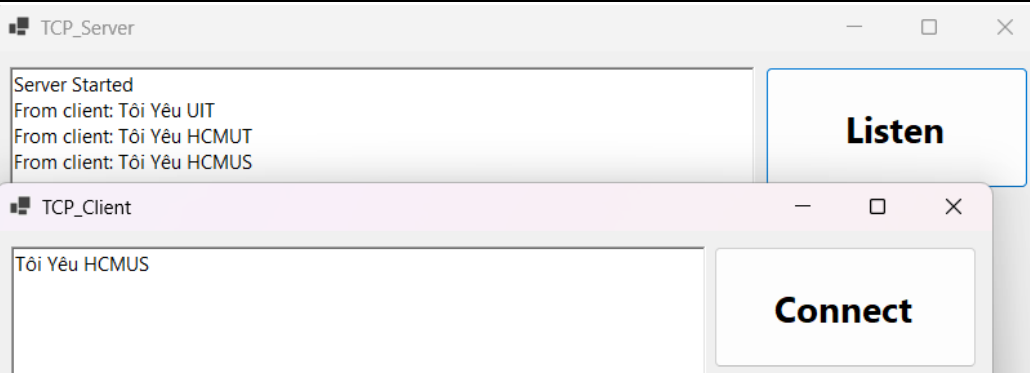
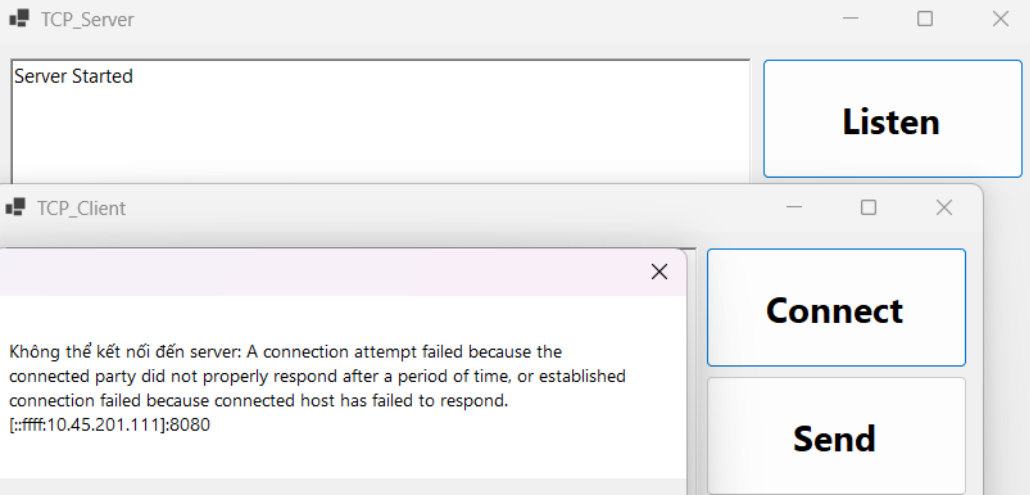
1. Khởi tạo Server: Tạo Socket Listener_Socket với kiểu Stream và giao thức Tcp. Gán địa chỉ IPAddress.Any và cổng 8080 để server chờ kết nối từ client. Gọi Bind() và Listen() để bắt đầu lắng nghe. Khi có kết nối, dùng Accept() để nhận Client_Socket.
2. Nhận dữ liệu từ Client: Tạo mảng Buffer để đọc dữ liệu từ client qua Receive(). Giải mã dữ liệu nhận được bằng Encoding.UTF8.GetString() và hiển thị lên rtb_messages của form. Vòng lặp chạy đến khi client ngắt kết nối.
3. Khởi tạo Client và kết nối Server: Client dùng TcpClient và kết nối tới IP của Server qua cổng 8080. Sau khi kết nối thành công, thông báo bằng MessageBox.
4. Gửi dữ liệu từ Client đến Server: Lấy nội dung từ rtb_messages.Text. Chuyển chuỗi thành byte (Encoding.UTF8.GetBytes()) rồi gửi qua NetworkStream.Write().
5. Ngắt kết nối: Khi người dùng nhấn “Disconnect”, client gửi thông báo “Client closed” rồi đóng NetworkStream và TcpClient.

Test case:

Mã TC	Mô tả	Dữ liệu đầu vào	Kết quả mong đợi	Hình ảnh kết quả
-------	-------	-----------------	------------------	------------------

TC3.1	Người dùng nhấn nút Connect	Client gửi yêu cầu kết nối tới Server	Hiển thị thông báo: “Kết nối thành công”	
TC3.2	Lỗi kết nối khi server chưa listen	Client gửi yêu cầu kết nối tới Server	Hiển thị thông báo: “Không thể kết nối đến server + ex.messages”	

TC3.3	Người dùng nhấn nút Disconnect	Client ngắt kết nối với Server	Hiển thị thông báo: “Đã ngắt kết nối”	
TC3.4	Gửi 1 message	Chuỗi: “Tôi Yêu UIT”	Xuất ra màn hình phía Server	

TC3.5	Gửi nhiều messages	Chuỗi: “Tôi Yêu UIT”, “Tôi Yêu HCMUT”, “Tôi Yêu HCMUS”	Xuất ra màn hình phía Server tuần tự	 <p>The screenshot shows two windows. The 'TCP_Server' window has a text area with the following content: 'Server Started', 'From client: Tôi Yêu UIT', 'From client: Tôi Yêu HCMUT', and 'From client: Tôi Yêu HCMUS'. It has a 'Listen' button. The 'TCP_Client' window has a text area with 'Tôi Yêu HCMUS' and a 'Connect' button.</p>
TC3.6	Lỗi khi kết nối sai IP Server hoặc Port	IP Server: 10.45.201.117 IP Client: 10.45.201.111	Hiển thị thông báo: “Không thể kết nối đến server + ex.messages”	 <p>The screenshot shows the same two windows as in TC3.5. The 'TCP_Server' window has 'Server Started' in its text area and a 'Listen' button. The 'TCP_Client' window has an error message in its text area: 'Không thể kết nối đến server: A connection attempt failed because the connected party did not properly respond after a period of time, or established connection failed because connected host has failed to respond. [::ffff:10.45.201.111]:8080'. It has 'Connect' and 'Send' buttons.</p>

Câu 4. 1 Server – Multi Client Quản lý phòng vé (phiên bản số 3)

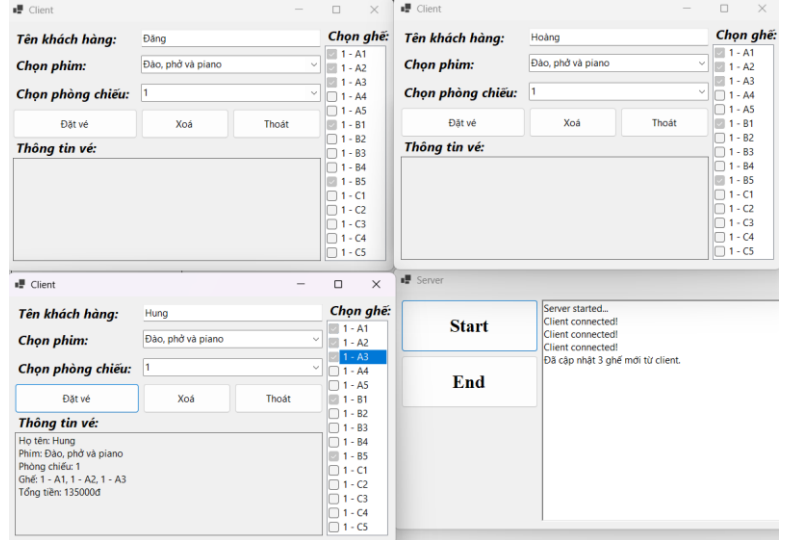
Trả lời:

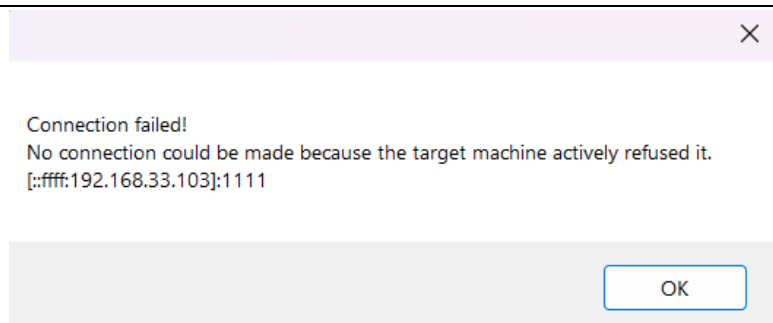
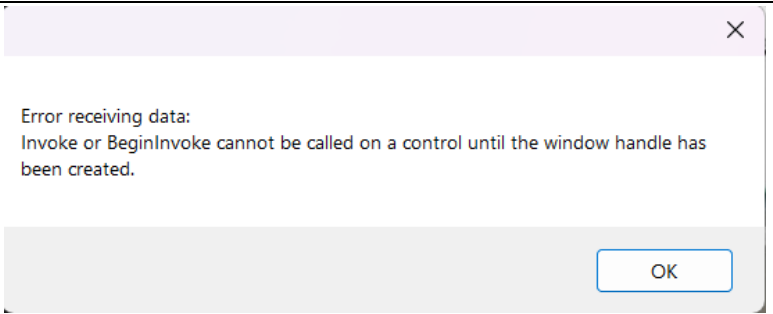
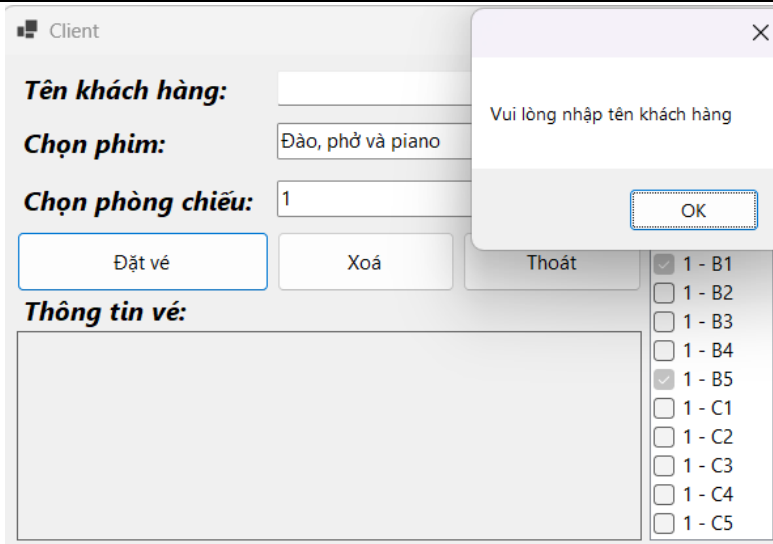
Ý tưởng chính: Khi server lắng nghe được client sẽ gửi dữ liệu từ file Input.json cho client xử lý, sau đó client đặt vé thì server sẽ nhận dữ liệu rồi thông gửi broadcast cho các client khác về các ghế đã được đặt. Logic xử lý đặt vé sẽ giống bài 1.

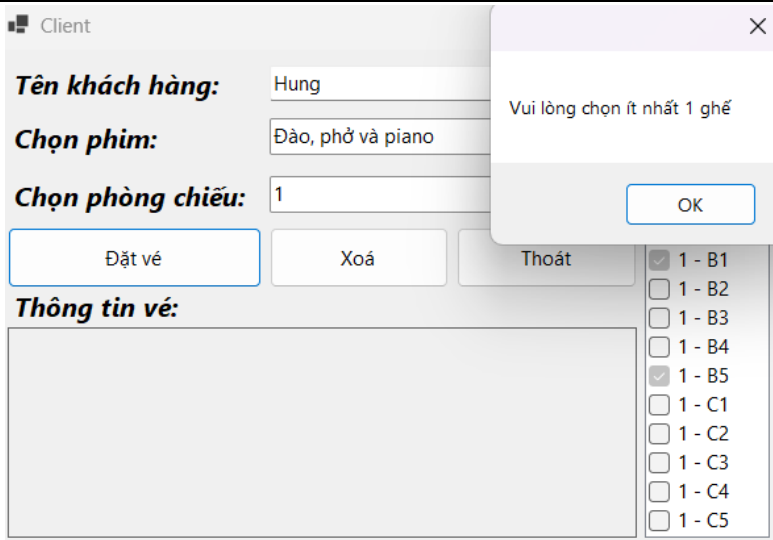
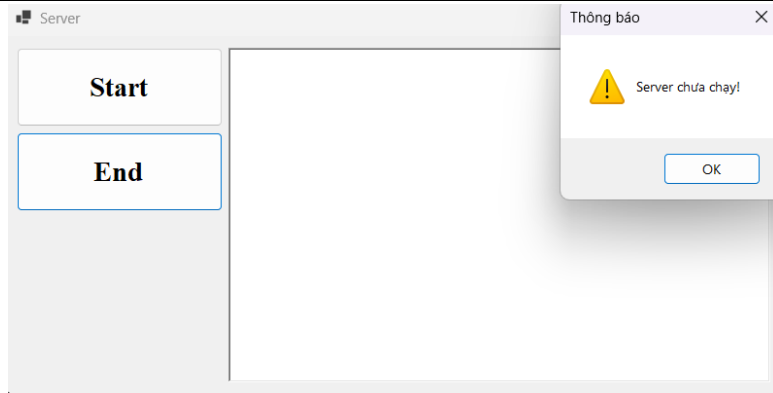
Các bước thực hiện chính:

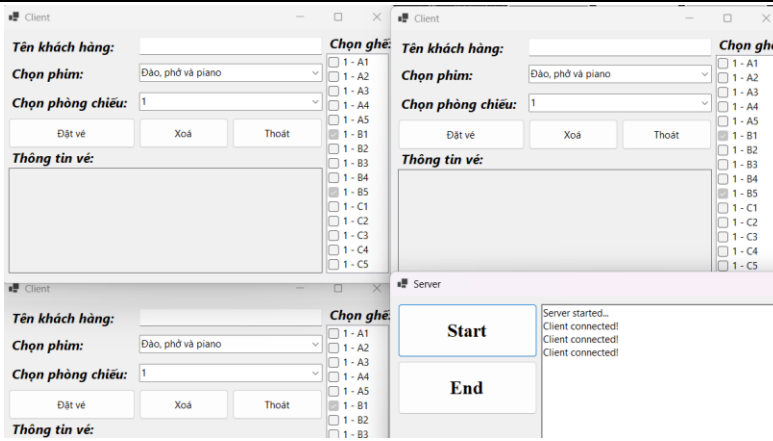
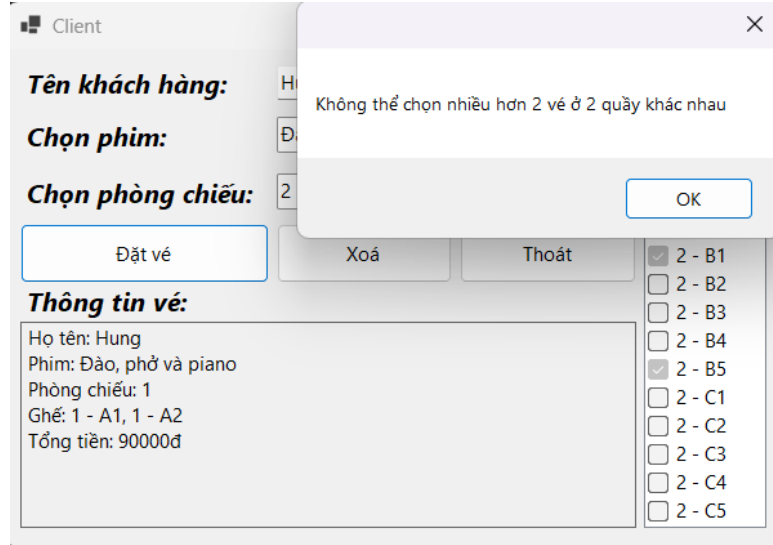
1. Khởi động server, bắt đầu lắng nghe các client
2. Client nhận dữ liệu từ server, client thực hiện chức năng đặt vé
3. Server nhận dữ liệu và broadcast cho các client khác về các chỗ đã được đặt
4. Tiếp tục cho đến khi hết vé

Test case:

Mã TC	Mô tả	Dữ liệu đầu vào	Kết quả mong đợi	Hình ảnh kết quả
TC4.1	Client đặt vé, client khác nhận được vị trí các ghế đã đặt	Phim: Đào, phố và piano Phòng chiếu: 1 Tên: Hung	Client khác nhận được vị trí ghế mà khách đầu tiên đặt	 <p>The screenshot displays the application's user interface. On the left, the 'Client' window for user 'Hung' is shown, with fields for 'Tên khách hàng' (Hung), 'Chọn phim' (Đào, phố và piano), 'Chọn phòng chiếu' (1), and 'Chọn ghế' (1-A3). Below these are buttons for 'Đặt vé', 'Xoá', and 'Thoát'. The 'Thông tin vé' section shows details for the selected seat. On the right, the 'Server' window is visible, showing a 'Start' button and a log area with messages: 'Server started...', 'Client connected!', 'Client connected!', and 'Đã cập nhật 3 ghế mới từ client.'.</p>

TC4.2	Server chưa mở nhưng đã mở client khiến client không kết nối được	Mở form client mà chưa khởi động client	Hiển thị thông báo: Connection failed! + ex.message	
TC4.3	Lỗi khi nhận data từ server	Client không nhận được dữ liệu từ server	Hiển thị thông báo: Error receiving data: + ex.message	
TC4.4	Không nhập tên khi đặt vé	Tên khách hàng để trống	Hiển thị thông báo: Vui lòng nhập tên khách hàng	

TC4.5	Người dùng không chọn ghế	Chưa ghế nào được chọn	Hiện thị thông báo: Vui lòng chọn ít nhất 1 ghế	
TC4.6	End server khi server chưa chạy	Người dùng bấm nút End	Hiện thị thông báo: Server chưa chạy!	

TC4.7	Server kết nối với nhiều client cùng lúc	Mở 3 form client kết nối với server	Server nhận được kết nối với cả 3 client hiển thị ra 3 thông báo: Client connected!	
TC4.8	Người dùng chọn nhiều hơn 2 vé ở 2 quầy	Chọn 2 vé ở phòng 1, 1 vé ở phòng 2	Hiển thị thông báo: Không thể chọn nhiều hơn 2 vé ở 2 quầy khác nhau	

Câu 5. 1 Server – Multi Client Hôm nay ăn gì? (phiên bản số 3)

Trả lời:

Ý tưởng chính: Xây dựng một ứng dụng **Client – Server** cho phép nhiều người dùng cùng tham gia chọn món ăn.

Mỗi người có thể:

- **Thêm món ăn** (kèm tên và hình ảnh).
- **Xem danh sách tất cả món ăn.**
- **Chọn ngẫu nhiên món ăn toàn cục.**
- **Chọn ngẫu nhiên món ăn riêng của chính mình.**

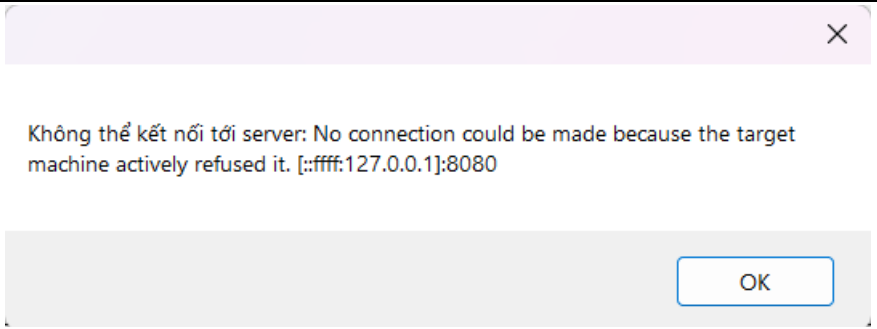
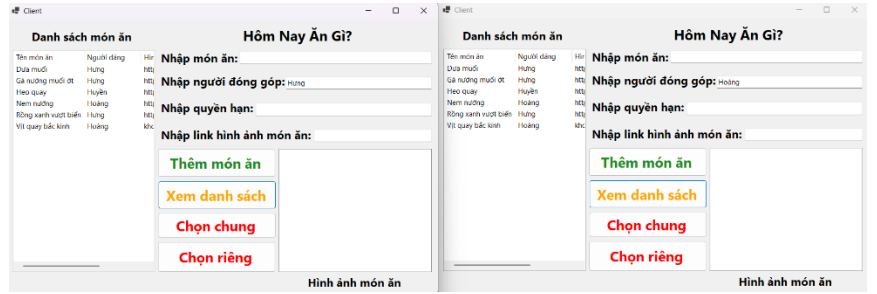
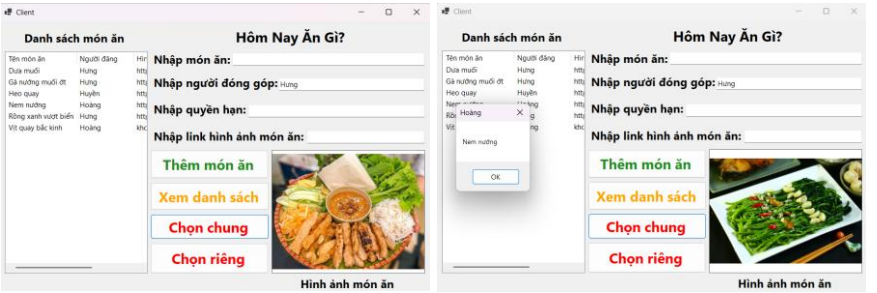
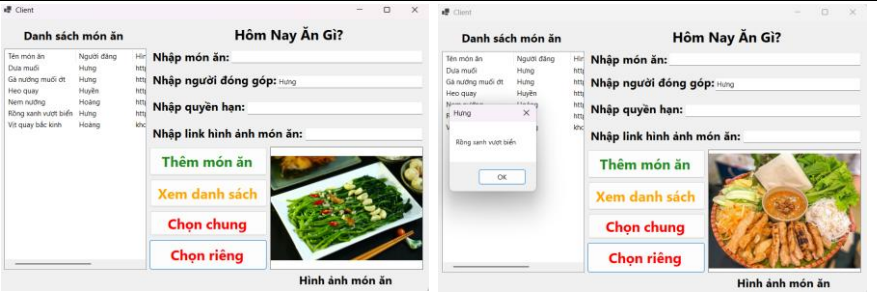
Server quản lý dữ liệu chung bằng **SQLite**, còn các Client giao tiếp với Server qua **TCP socket**.

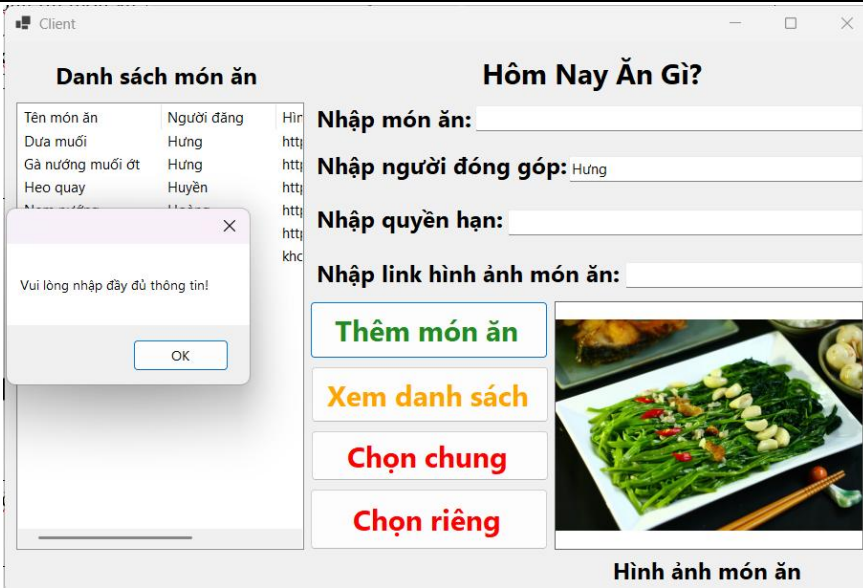
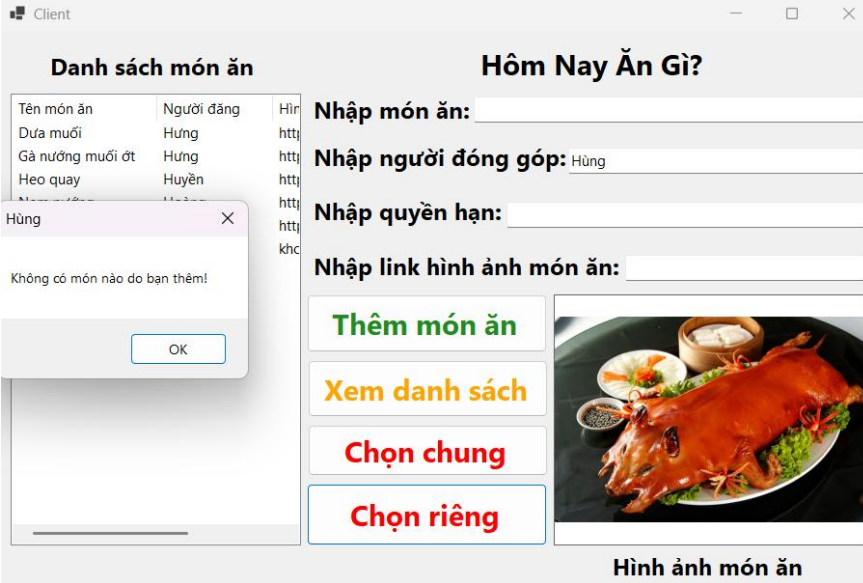
Các bước thực hiện chính:

1. Người dùng mở form và bắt đầu kết nối với server
2. Người dùng thêm món, server sẽ lưu vào DB, broadcast món ăn vừa mới được thêm vào
3. Người dùng xem món, server sẽ truy vấn toàn bộ DB và xuất ra các món ăn đã được các người dùng thêm vào
4. Người chọn món chung, server sẽ truy vấn toàn bộ DB và xuất ra 1 món ăn ngẫu nhiên cho người dùng
5. Người dùng chọn riêng, server sẽ truy vấn toàn bộ DB và tìm ra các món ăn của người dùng đó. Sau đó xuất ra 1 món ăn ngẫu nhiên

Test case:

Mã TC	Mô tả	Dữ liệu đầu vào	Kết quả mong đợi	Hình ảnh kết quả
-------	-------	-----------------	------------------	------------------

TC5.1	Server chưa mở nhưng đã mở client khiến client không kết nối được	Mở form client mà chưa khởi động client	Hiện thị thông báo: Connection failed! + ex.message	
TC5.2	Khi 1 người thêm món ăn các client khác cũng nhận được món ăn vừa mới thêm vào	Client tên Hoàng thêm món Vịt quay bắc kinh,	Các client còn lại sẽ nhận được món ăn mới thêm, ở đây là client Hưng cũng có món đó trong danh sách	
TC5.3	Người dùng chọn món chung	Client tên Hưng chọn ngẫu nhiên một món ăn trong danh sách chung	Hiện thị thông tin món ăn bằng message box sau khi nhấn ok sẽ xuất ra hình ảnh món ăn	
TC5.4	Người dùng chọn món riêng	Client tên Hưng chọn ngẫu nhiên một món ăn trong danh sách riêng	Hiện thị thông tin món ăn bằng message box sau khi nhấn ok sẽ xuất ra hình ảnh món ăn của Hưng thêm vào	

TC5.5	Người dùng không nhập đủ thông tin khi thêm món	Một số trường cần nhập bị thiếu	Hiện thị thông báo: “Vui lòng nhập đủ thông tin”	
TC5.6	Client chọn riêng nhưng chưa có món ăn nào	Người dùng mới chưa có món ăn	Hiện thị thông báo: “Không có món ăn nào do bạn thêm”	

Câu 6. 1 Server – Multi Client Viết chương trình Chat Room

Trả lời:

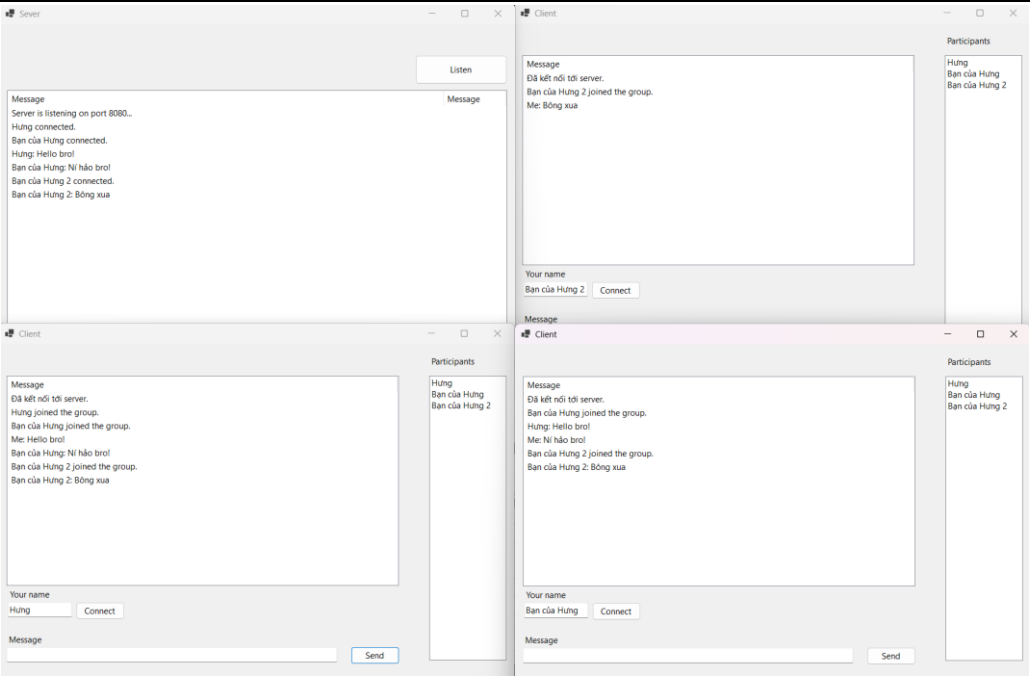
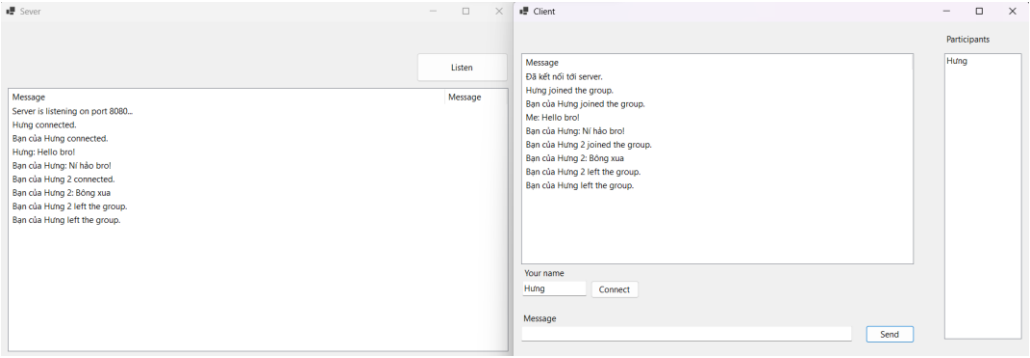
Ý tưởng chính: Tạo một TCP server để quản lý tin nhắn của người dùng. Khi chat chung thì tin nhắn sẽ được gửi đến server rồi broadcast đến các client khác, còn nếu chat riêng thì tin nhắn chỉ được gửi đến người được chọn.

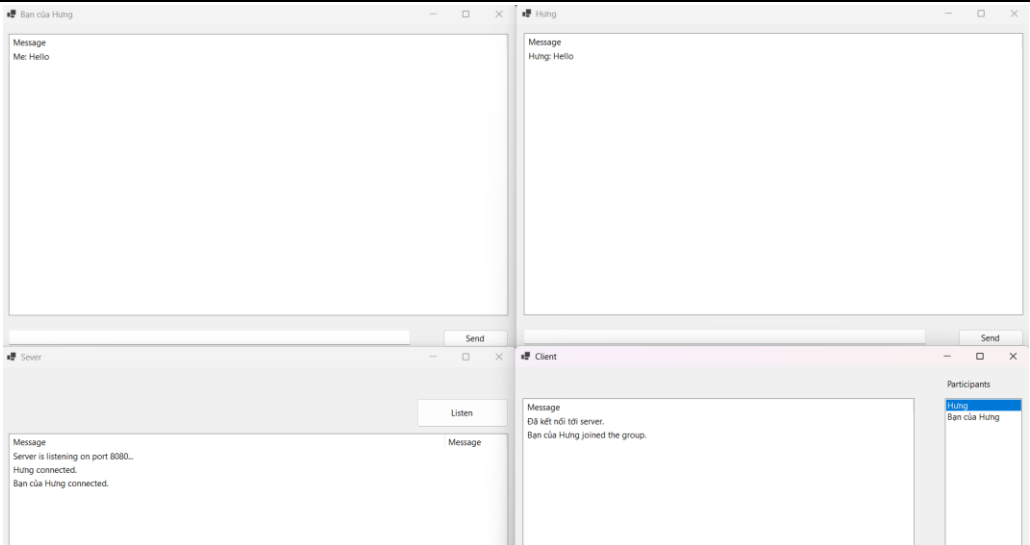
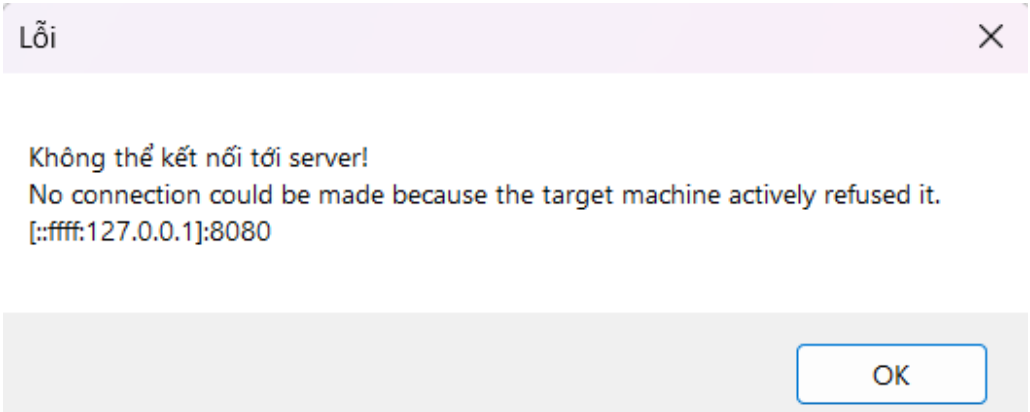
Các bước thực hiện chính:

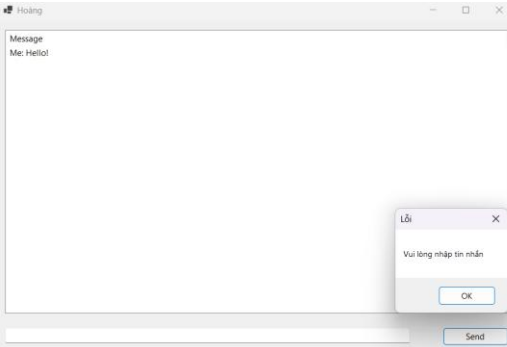
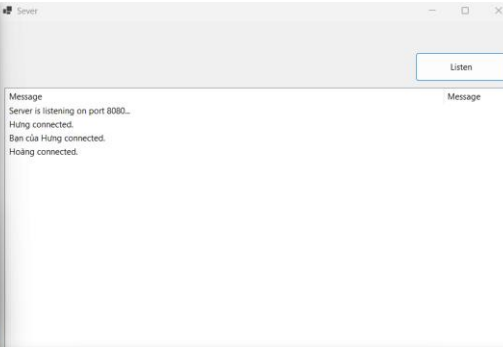
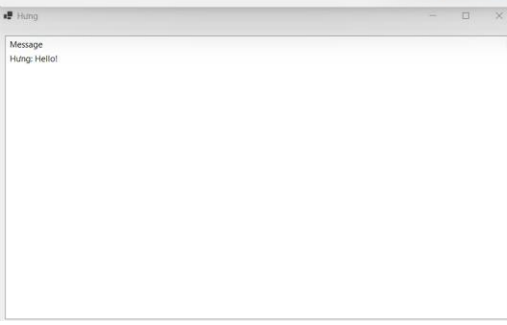
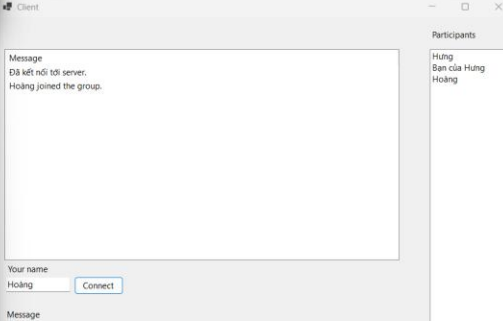
1. Người dùng kết nối tới Server bằng nút connect
2. Người dùng bấm nút send sẽ gửi tin nhắn đến server và server sẽ broadcast đến các client khác
3. Người dùng chọn người để chat riêng bằng cách click vào tên client đó, sau đó sẽ có form mở riêng để chat
4. Người dùng nhấn nút gửi và tin nhắn sẽ được gửi đến cho người đã được chọn.

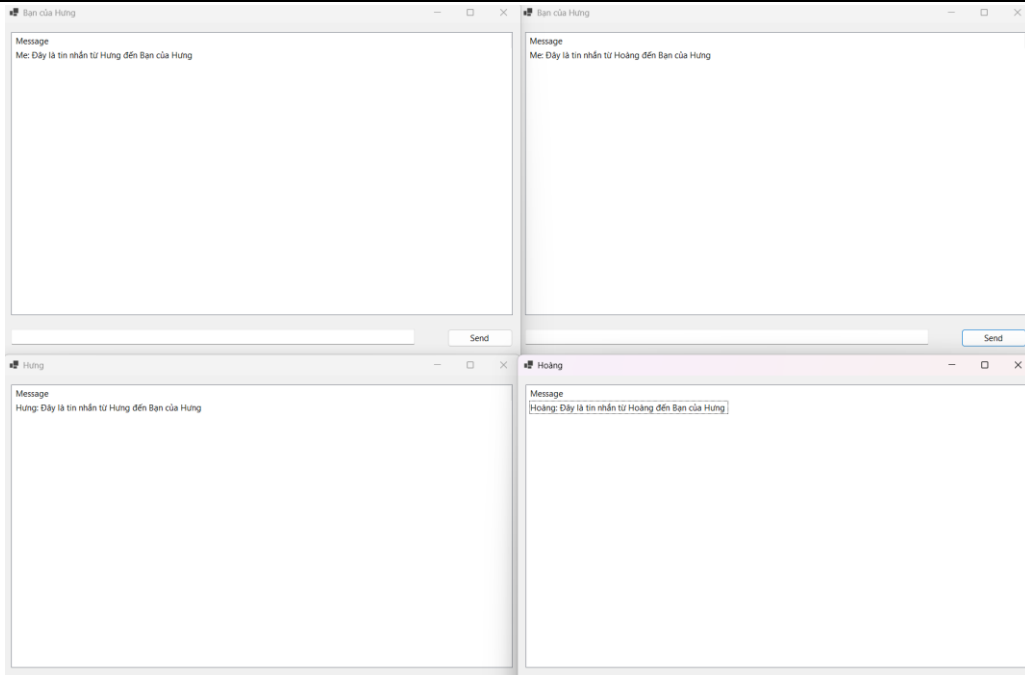
Test case:

Mã TC	Mô tả	Dữ liệu đầu vào	Kết quả mong đợi	Hình ảnh kết quả
-------	-------	-----------------	------------------	------------------

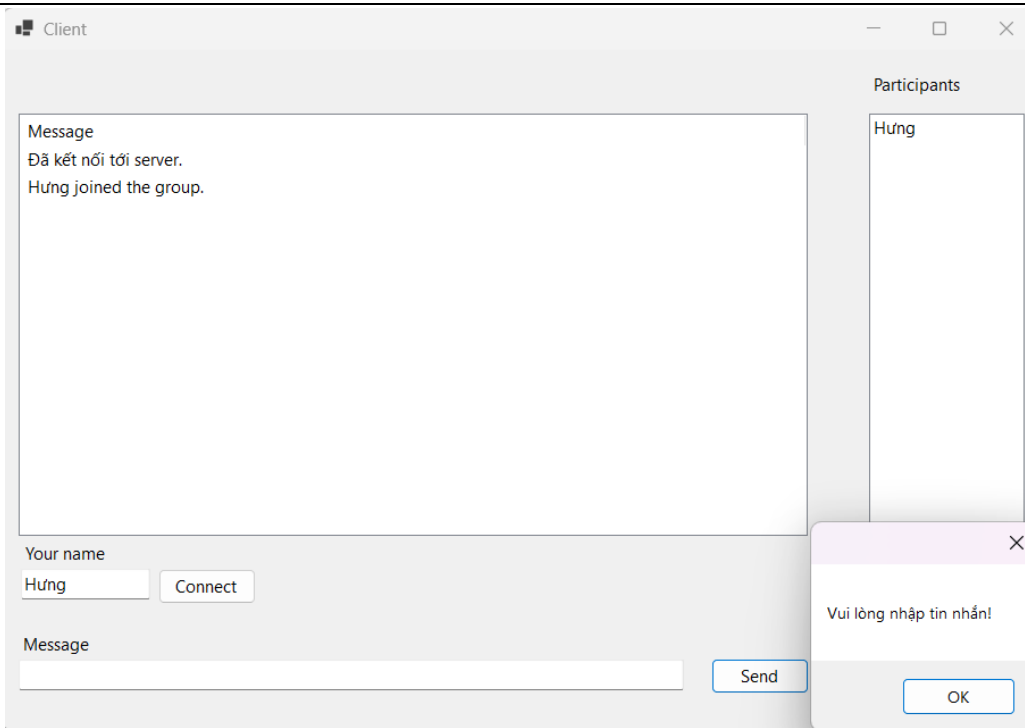
TC6.1	Chat chung với nhiều client	3 Client lần lượt gửi các tin nhắn	Server hiển thị được tin nhắn và broadcast được đến với các client khác	
TC6.2	Client ngắt kết nối với server	Đóng các form client	Server hiển thị ra thông báo các client đã ngắt kết nối (left the group) và broadcast thông tin đó đến các client đang hoạt động	

TC6.3	Chat riêng giữa 2 client	Người dùng chọn client muốn chat riêng sau đó gửi tin nhắn	Server nhận tin nhắn rồi gửi cho người được chọn nhưng không hiển thị tin nhắn để bảo mật. Người được chọn nhận được tin nhắn từ người gửi	
TC6.4	Lỗi khi server chưa listen nhưng client đã kết nối	Client dùng nút connect tới server	Hiện thị thông báo: “Không thể kết nối tới server! + ex.message”	

TC6.5	Khi chat riêng nhưng không nhập tin nhắn	Client để trống tin nhắn	Hiển thị thông báo: “Vui lòng nhập tin nhắn”	   
-------	--	--------------------------	--	---

TC6.6	Nhiều người dùng cùng chat riêng với 1 người	Client Hung và Client Hoàng cùng nhắn riêng với Client Bạn của Hung	Client Bạn của Hung sẽ nhận được tin nhắn riêng từ mỗi người dùng	 <p>The screenshot displays four overlapping chat window mockups. The top-left window is titled 'Bạn của Hung' and shows a message from 'Me' (the user) to 'Bạn của Hung'. The top-right window is also titled 'Bạn của Hung' but shows a message from 'Me' to 'Hoàng'. The bottom-left window is titled 'Hung' and shows a message from 'Hung' to 'Bạn của Hung'. The bottom-right window is titled 'Hoàng' and shows a message from 'Hoàng' to 'Bạn của Hung'. Each window has a title bar with standard minimize, maximize, and close buttons, and a 'Send' button at the bottom right of the message area.</p>
-------	--	---	---	---

TC6.7	Lỗi khi tắt form server sau đó mở lại rồi bắt đầu listen	Tắt form server sau đó mở lại từ form broadcast	Hiển thị thông báo lỗi do server đang chạy và chưa được tắt hẳn ở lần trước	<p>The screenshot shows a window titled "Sever" (sic). It contains a "Listen" button. Below the button is a message box with the text: "Server Error: Only one usage of each socket address (protocol/network address/port) is normally permitted."</p>
-------	--	---	---	---

TC6.8	Khi chat chung nhưng không nhập tin nhắn	Client để trống tin nhắn	<p>Hiển thị thông báo: “Vui lòng nhập tin nhắn!”</p> 
-------	--	--------------------------	--