Assginemt 4

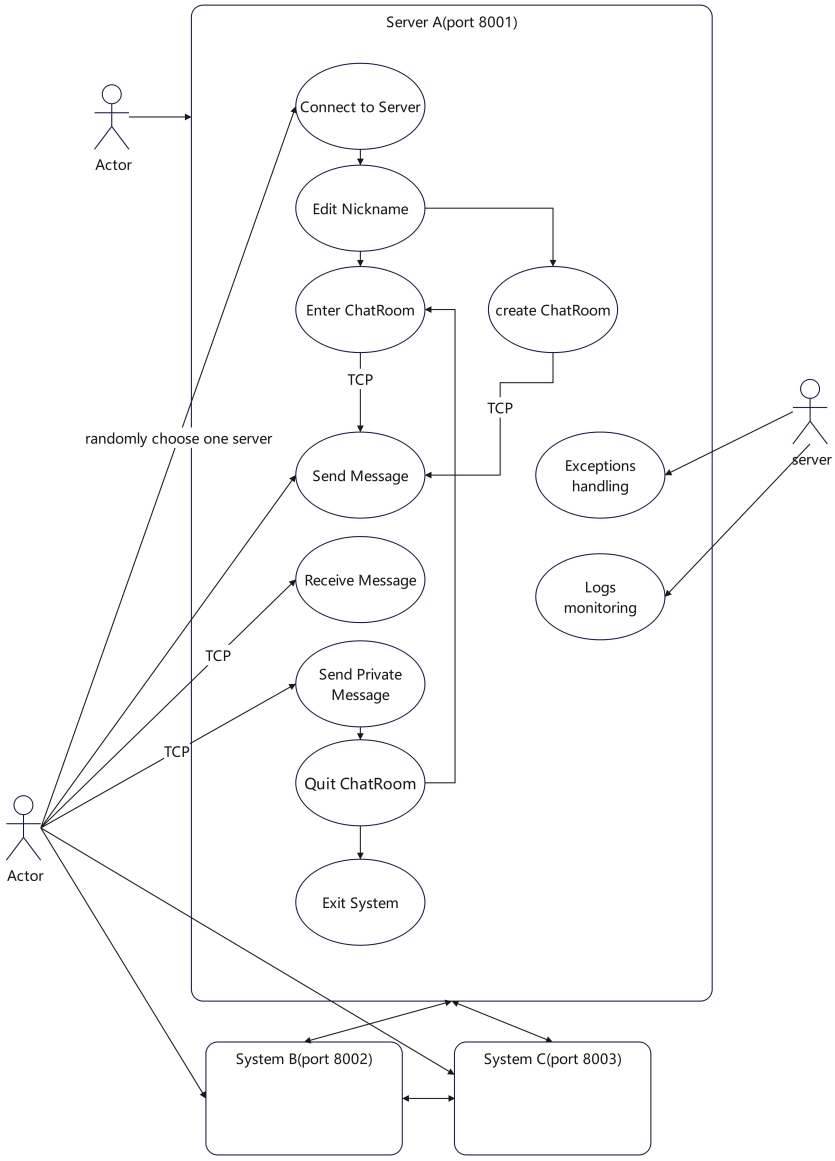
Zijing Li

000213415

Zijing.Li@student.lut.fi

UML Diagram

Use Case:



Function Analysis:

This system is a distributed system that supports multi-user communication. There are three server instances listening to ports 8001,8002,8003 respectively, which reflects the scalability and higher availability of a distributed system. Users connect to the servers using TCP and have permission to perform the following functions:

1. Connect to the server via IP address
2. Edit nicknames
3. Select OR create a chat room
4. Enter the chat room, chat with that member in a group, send and receive messages.
5. Chat privately with members in the chat room.
6. Exit the chat room and re-select to join.
7. Disconnect from the server.

The server side has the following functions:

1. Deliver messages between clients.
2. Supports multi-threaded client connections.
3. Performs exception handling.
4. Understand client operation.

Distributed features:

1. Subject to physical constraints, use different ports to emulate servers in different physical locations and can be easily scaled. Users randomly connect to available ports, which improves the load capacity of the system and ensures load balancing on each port to a certain extent.
2. Clients listening on different ports can still communicate and share information across ports.
3. The system provides high concurrency, users can use the functions at the same time without being affected.
4. The system can track logs to understand the client's communication, and will not cause the entire system to crash due to anomalies, and can give the appropriate error messages.

Transparency:

1. The client enjoys transparency of access and transparency of location distribution. The client does not need to know how the server of the chat room is deployed and developed, or in what physical location the local functionality is distributed; the client only needs to interact with the functionality at the user interface to use the system. Users are randomly assigned to different ports without being affected, hiding the distributed nature.
2. When a client has an exception, such as a user forcibly terminating a connection to the server or not following the rules for communication, the error will not affect the normal operation of the server or other clients.

3. Chat rooms, nicknames and other information can be created on the client side without affecting the normal use of other users.

Scalability:

1. The system can easily expand more available ports to support the use of more clients, strengthen the load capacity of the server, and prevent overloading of individual ports.
2. Chat rooms can be added and created at will to support concurrent communication across multiple chat rooms.

Exception Handling:

1. I used a lot of try-except on the server side, which can solve most of the client-side exceptions, and print out the corresponding reasons in the terminal.
2. Any exception will not affect the normal operation of the entire server, even if the client is forced to terminate the connection, but also to ensure that it does not affect the use of other users, and release the resources of the port.

In addition, users entering or leaving the chat room can receive information about their entry and exit, reflecting the consistency of the distributed system, and real-time monitoring of the system process.