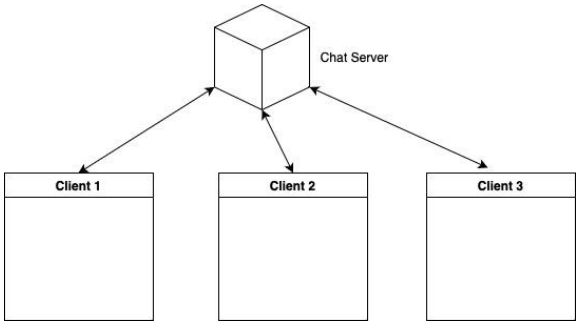
**Distributed Computing (COMP3008)**
**Assignment 1 (Part A) – 32 Marks**
**Build a basic chat application using .NET**

**Objective:** The objective of this assignment is to guide students through the process of building a chat application using .NET Windows Communication Foundation (WCF). By completing this assignment, students will learn to create a server that enables multiple clients to communicate in real-time through a Windows Presentation Foundation (WPF) client application.

This is a group assignment (a group of two or three members). The maximum mark is placed right after each requirement. In the marking criteria, there are six levels of implementation, i.e., zero attempts (0% -10% complete), novice (efforts shown – around 10%-30% are completed), promising (around 30%-50% are completed), intermediate (about 50%-70% of the requirements are achieved), competent (about 70%-90% of the requirements are achieved, and proficient (90% or above level are achieved).

You must implement at least two projects in the solution: a) Chat Server and b) Chat Client.

The basic interaction/system design is as follows. Note that we can have more than three clients. Theoretically, the number of clients is infinite; however, fortunately, you don't have to test the scalability and load balancing (😊). However, test the system with client numbers ranging from 1 to 5.




Since you are developing it in a single machine, give a static ip: port to the server which is known to the clients. This is not going to be a peer-to-peer system (for simplicity).

**A. The functional specification of the chat server [11 Marks] and marks breakdown:**

1. **User Management:** The chat server manages unique client log-ins. Each client will try to join only with a username, and the server checks if this name is unique, i.e., no two clients with the same username should not be present in the system at the same time [2 Marks]
2. **Chat Room Management:** The server handles the creation, joining, and leaving of chat rooms. Each chat room can have multiple participants. [2 Marks]
3. **Message Distribution:** When a user sends a message, the server ensures that the message is distributed to all participants within the same chat room. [2 Marks]

4. **Private Messaging:** The server supports private messaging, allowing users to send messages to specific individuals within the same chat room. [2 Marks]
5. **File Sharing:** The server facilitates the sharing of files between users within a chat room. It manages the transfer of files and ensures that only participants in the chat room can access shared files. For simplicity, we only handle image files and text files [3 Marks]

**B. The functional specification of the Chat Client in WPF (Graphical User Interface - GUI) [16 Marks] and marks breakdown:**

1. **User Login:** The chat client prompts users to log in using a unique name. After successful authentication, the user gains access to the chat rooms. [2 Marks]
2. **Chat Room Selection:** The client displays a list of available chat rooms. Users can choose a chat room to join based on their interests or affiliations. [2 Marks]
3. **Chat Room Creation:** The client can also create a chat room with a unique name. The server will notify if the chat room is created. [2 Marks]
4. **Chat room messaging:** Users can send and receive messages within the chat room. Messages are displayed in the chat area as they are received. [2 Marks]
5. **Private Messaging:** Users can initiate private conversations with other users by selecting their names and sending private messages. Users will also be able to receive private messages. [2 Marks]
6. **File Sharing:** The client allows users to select files (images, documents, etc.) to be shared within the chat room. Shared files are displayed as links that other participants can access. [2]
7. **Logout:** Users can log out of the chat client when they're done using it. [1]

If you are uncomfortable with threading, you can only work with sequential operations. For example, to receive a private message, you can initiate the remote call by pressing a relevant button, and then the server sends the message to the client. It is not real-time, and this is called ‘pull’ strategy, i.e., client will always pull information from the server.

For GUI updates, use **refresh buttons (more than one as required)** to update to the current information. For example, if clients join/leave a chat room, other clients can get the updated client list by refreshing their GUI. [3 Marks]

Note that if you are uncomfortable with a GUI-based client, you can also develop a console-based client where the instructions are given in commands. However, **the maximum point available for console-based clients is 7**.

**C. Additional requirements [5 Marks]:**

We will make the ‘pull’ requests more advanced by placing them in a different thread to listen to any messages from the servers, i.e., getting chat room messages, private messages and files shared by other clients. It makes it more real-time. The thread will periodically send requests to the server, and if it receives the information, it updates the GUI.

a. Apply separate threads to update the current list of chat rooms and available users [Marks 2]

b. Apply separate threads to receive chat room messages and private messages [ Marks 3]

**Submission due date: 24 September 2023 23:59 AWST.**

**Submission Guidelines:**
Submit your assignment electronically, via Blackboard, before the deadline. **If you submit in a group, all the members must submit the same files.**
To submit, do the following:

1. Fill out and sign a declaration of originality. A photo, scan or electronically-filled out form is fine. Whatever you do, ensure the form is complete and readable!
Place it (as a .pdf, .jpg or .png) inside your project directory.
2. Zip your entire project directory. Leave nothing out.
3. Submit your zip/tar.gz file to the assignment area on Blackboard.
4. Re-download, open, and run your submitted work to ensure it has been submitted correctly.

You are responsible for ensuring that your submission is correct and not corrupted. You may make multiple submissions, but only your newest submission will be marked. The late submission policy (see the Unit Outline) will be strictly enforced. Please note:

- DO NOT use WinRar.
- DO NOT have nested zip/tar.gz files. One is enough!
- DO NOT try to email your submission as an attachment. Curtin's email filters are configured to discard emails with potentially executable attachments silently. In an emergency, if you cannot upload your work to Blackboard, please instead upload it to Google Drive, a private GitHub repository, or another online service that preserves immutable timestamps and is not publicly accessible.

**Marking Demonstration:** You must demonstrate and discuss your application with a marker in a one-to-one online/in-person session. Most of the marks for your assignment will be derived from this demonstration. The demonstrator will ask you to rebuild and run your application (provided by the demonstrator) and demonstrate its major features. They may ask you about any aspect of your submission. **You will be asked about your contribution percentages during the demo. Your mark from the group submission will be adjusted based on your contribution percentage. You must specify your contribution in the individual report (check part B of the assignment).**

The demonstration schedule and policy will be published later (Check blackboard announcements). We may also cancel the demonstration-based marking due to unavoidable circumstances. In that case, it will be a full inspection-based marking.

**Academic Integrity:** This is an assessable task. If you use someone else's work or obtain someone else's assistance to help complete part of the assignment that is intended for you to complete yourself, you will have compromised the assessment. You will not receive marks for any parts of your submission that are not your original work.

Further, if you do not reference any external sources, you are committing plagiarism and collusion, and penalties for Academic Misconduct may apply. Please see Curtin's Academic Integrity website for information on academic misconduct (which includes plagiarism and collusion).

The unit coordinator may require you to provide an oral justification of or to answer questions about any piece of written work submitted in this unit. Your response(s) may be referred to as evidence in an Academic Misconduct inquiry.

**End of Assignment**