**COSC 4333: Chat Service**

**Group Project**

**Due 10:00am. Dec. 2, 2024 (Monday)**

**By: Wesley Hargrave,**

**Thien Ho,**

**Andrew Hoffman**

### Introduction

This project involves the design and implementation of a multi-threaded chat room service in C programming, which features a central chat server capable of managing multiple chat rooms and clients concurrently, complemented by lightweight client applications for seamless communication. In this report, we will give a comprehensive look into its functionality and demonstrate how the server and clients work together to communicate alongside discussing challenges that we encountered during development on the project.

### Functionality and Implementation

The project is built around a client-server model, with components such as the chat server and the chat client that interact over a socket-based network, with the server acting as the central hub for managing multiple chatrooms and client interactions. The server also listens for JOIN and LEAVE requests on a predefined port, dynamically creating threads for each chat room, as it is designed to ensure efficient message processing and distribution within individual chatrooms. On the client’s side, the application establishes a connection with the server and offers a user-friendly interface for sending and receiving messages, with communication between clients and the server being handled through socket-based data exchange, enabling real-time interactions.

Implementation involves two programs: ChatroomServer.c and ChatroomClient.c. The server program initializes a listening socket on a predefined port and handles JOIN requests by creating or assigning threads for chatrooms using pthread\_create(), where each thread manages message broadcasting, forwarding messages between clients in the same room. When a client sends a LEAVE request, the server removes the client, terminating the room thread and freeing up resources if no users remain.

As for the client, they connect to the server with a JOIN request for a specific chat room. Once connected, it continuously sends user messages and displays incoming messages in real time, and when they are done, they can exit by sending a LEAVE request, which closes the connection and allows the server to act.

### Challenges and Solutions

Some of the challenges encountered during this project required good solutions to ensure a more efficient chat room service. One primary challenge was handling multiple client connections at once, which could be addressed using multi-threading, where each chat room was managed by a dedicated thread. Another concern was ensuring message broadcasting was correct, which required careful synchronization to avoid race conditions. There was also a need to ensure that the input from each client did not cause a buffer overload when transmitted to the server and concurrently to other clients.

### Conclusion

This multi-threaded chatroom service was designed and implemented using socket programming and multi-threading in C, combining concepts like networking and concurrency into a useful application. Despite difficult challenges such as managing concurrency and code testing, this project gave insight into how distributed systems may work in real life and help give experience into tackling more complex endeavors in the future.