CS5700 Final Project
Akshay Rao

This is a report on my final project for CS5700 - Fundamentals of Networking.

Project : Instant Messaging Chat Application

Repo link: https://github.com/akshayrao96/chat-program

Introduction:

This report is an analysis of an instant messaging chat application project. I developed this program to better understand and implement client-server socket connection using the Python programming language. The project comprises 3 parts: server.py, client.py and chat-client.py. This program showcases an application of TCP/IP networking concepts, whereby TCP connection is used to ensure secure and safe communication between servers and clients.

Application, Significance and Purpose:

Instant messaging has evolved at a drastic rate, such that I have never questioned all the intricacies happening behind the scenes. I can open up a group chat, send a message, and expect everyone to see and receive the message.

When developing this program, I realized how much complexity and maintenance has to happen for a variety of processes to synchronize and work together. Instant messaging communication has become the forefront of all communication methods, and with information pervasively spreading at lightning fast rates, it's imperative that messages are sent and received quickly.

I developed this program to be a functional chat room, whereby a server is hosted, and multiple clients are able to join a chatroom and talk to one another. This project lays the groundwork to better understand how data is transmitted over networks.

CS5700 Final Project
Akshay Rao

Features and Design:

I will go over the 3 files that are in my repository for the project.

My server.py file is the server for this instant messaging chat application. The server is configured to listen on port 9000 (an arbitrary free port), handling incoming client connections. Aspects of socket configuration, binding and listening are implemented. It keeps track of all clients that have connected, and has a broadcast function to send a message to all clients connected.

My client.py file is a testing client for this program. It contains the logic for a client, establishes connection with the server and communicates with it.

My chat-client.py file is a sophisticated version of the client.py file - it contains all the logic but also adds a graphical user interface for users to use, just like a real-world chat application.

Implementation, Methodology and Testing:

The implementation and testing were done in several parts. Initially, the server's capability to run without errors on port 9000 was the first testing phase. Ensuring the server's stability was crucial before introducing client connections.

Subsequently, client.py was written, to mainly facilitate the logic between the client and the server. As seen from the source code, it establishes connection to the server. Logic of communication back and forth was implemented, such as the server prompting the client for the name, the information passed as bytes back to the server to store in a client list. When the client was sent the message, functionality of the server to receive the message, and broadcast it to all other connected clients was tested.

To build a more friendly user interface, chat-client.py was programmed. THe next steps were to implement the interface design, make sure the client and server were both multithreaded such

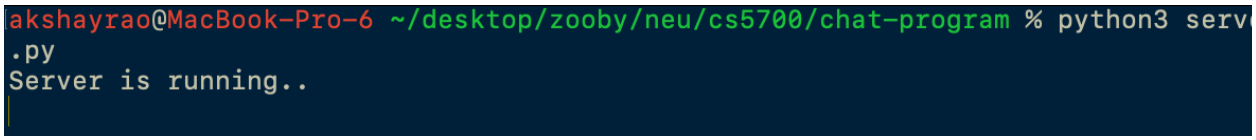
that clients could send messages, and servers could handle received messages in a modular fashion.

Graceful exiting from the client was implemented, so that the program can handle multiple clients that are simultaneously joining and leaving. Functionality of the TCP communication was imperative and the aim of the program, so additional features were not added, due to potential complexities and bugs.

Testing was performed in a localhost environment.

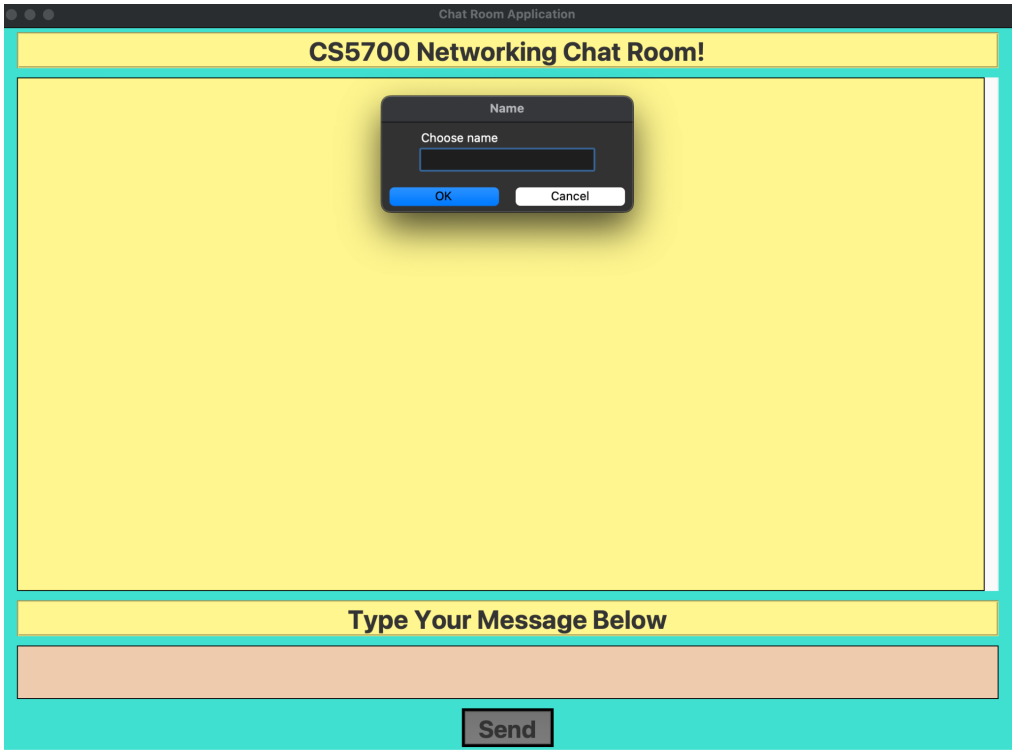
Program Flow:

Server starts running and is listening

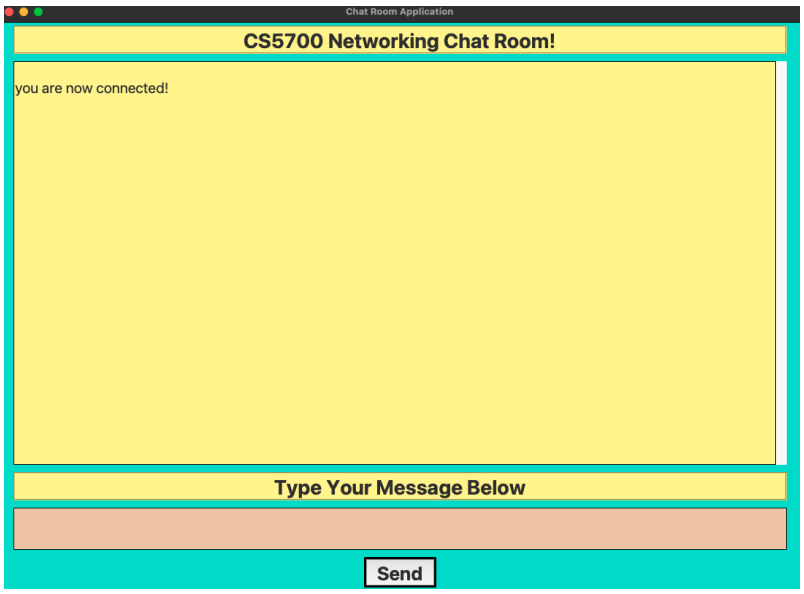


Client runs, and connects to the server. Pop up of client shows up, prompting name:

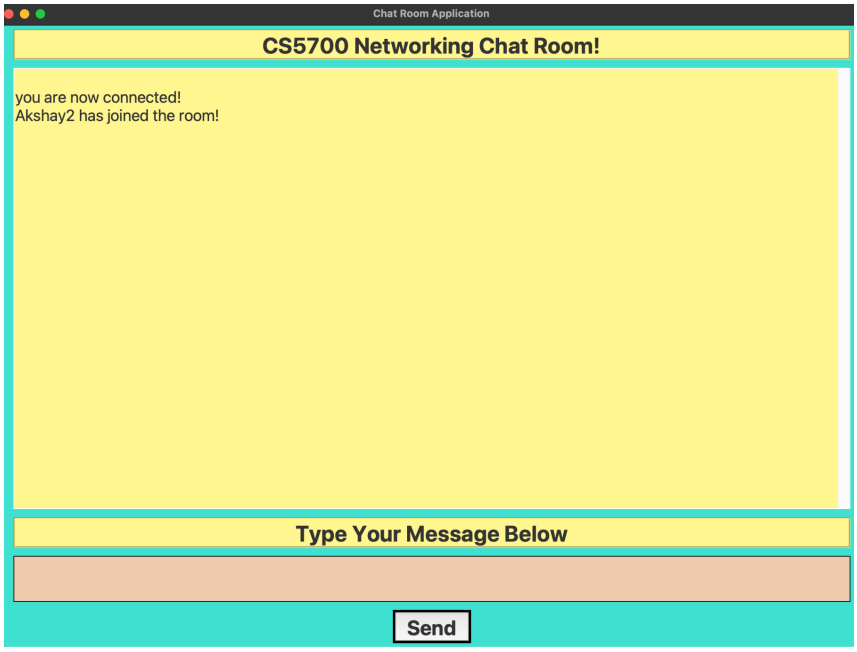
CS5700 Final Project
Akshay Rao



Inputting name Akshay into textbox:

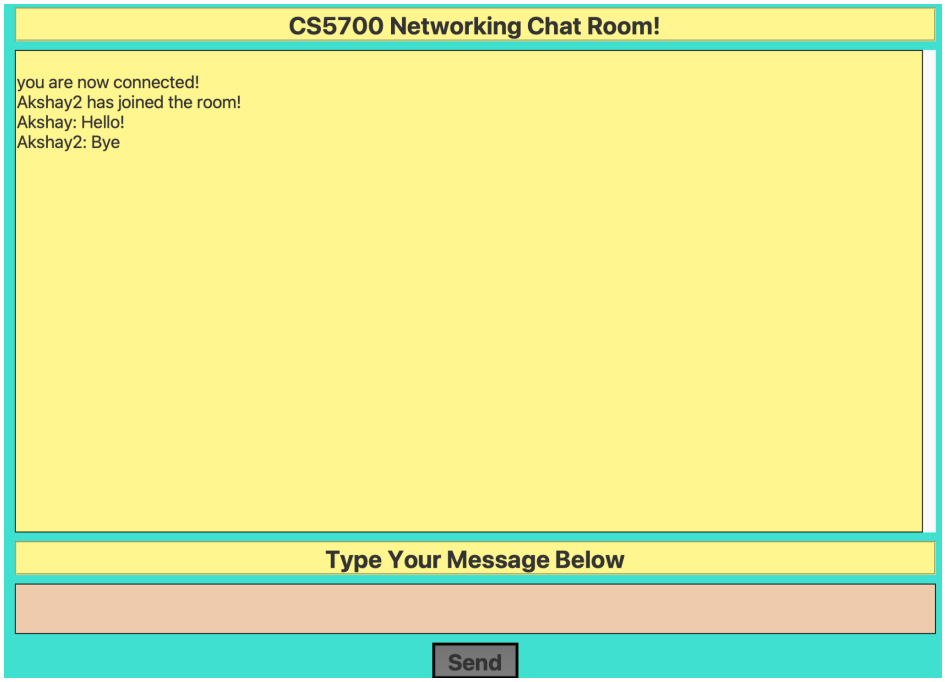


When another client connects, I input the name Akshay2 on there. This is now the UI for first client, Akshay:

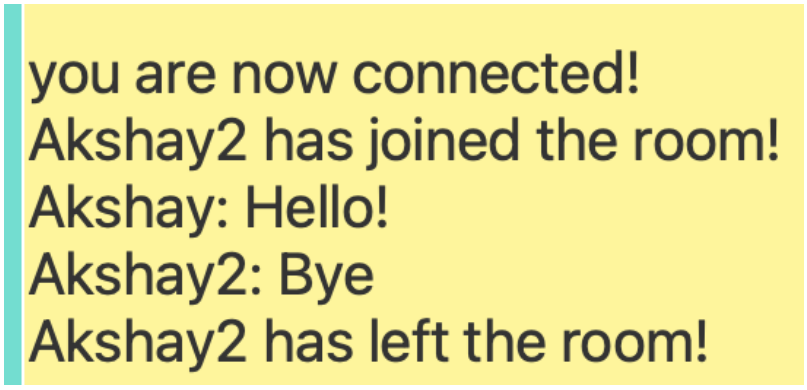


Now both clients have their own UI, and can chat to each other! Suppose Akshay sends hello, and Akshay2 sends bye, this is now the chatbox.

CS5700 Final Project
Akshay Rao



Graceful quitting is added, such that if a client types “quit” or “q”, or closes the box, they leave the room. All other clients are notified. Akshay2 leaves the server:



Improvements, Future Work:

CS5700 Final Project
Akshay Rao

This chat application parallels features found in established messaging platforms. It serves as a rudimentary model, primarily focused on a functional chat program and networking communications. Lots of features could be added, but there are tradeoffs with scalability and performance overheads. Improvements I considered were adding a server admin (broadcasting messages, kicking users), message history, and user authentication (username and password). However, adding features is volatile to the system, and potentially involves breaking the system if errors occur. Future works would have to be tested robustly in different branches, whereby changes are integrated one by one.

Conclusion:

In conclusion, this chat application provides a rudimentary model for real-world instant messaging platforms. It was a good learning tool that highlighted efficient handling and processing of real-time communication systems

Reflection:

I thoroughly enjoyed developing this application, which was challenging. Being unfamiliar with Python, this project provided valuable exposure to networking fundamentals and program flow control - as opposed to using other unfamiliar languages where I'd have to spend more time on intricate syntax and development tools.

Despite the codebase looking small, getting the intricacies of ensuring network communication among server and clients was extremely important, and involved meticulous analysis for fixing different bugs. Just a single bug could hinder the working program, so I learned a lot about networking fundamentals and tracing the routes of how messages are being communicated between server and clients.

It underscored the importance of iterative testing and development, with starting from a basic client server interaction to one that involved a multi-threaded, GUI based application.

Sources:

CS5700 Final Project
Akshay Rao

Here are the list of sources I used to help me build this application:

- CS5700 lectures
- Python Socket Programming resources: https://realpython.com/python-sockets/
- https://docs.python.org/3/howto/sockets.html
- Tkinter documentation : https://docs.python.org/3/library/tk.html
- https://docs.python.org/3/library/tkinter.html