Network Program Design

Crab Chat

# Project Members

* 1. Evan Binkley, in charge of server and project manager
  2. Austin Swartley, in charge of server
  3. Peter Schaefer, in charge of client and library

# Communication Plan

* 1. Snapchat, Discord, and GitHub will be used for general communicating and collaboration.
  2. Zoom and Discord will be the main method of conducting team meetings and focused collaboration efforts.

# Task Management

## Program Design (Theorizing)

* + 1. JSON packet formatting and organization idea (Create format)
    2. Create possible schedules and due dates.
    3. Have consistent programming format practices for readability.
    4. Create a sequence diagram.

## Application Design (Writing)

### Client (Peter Schaefer)

* + - 1. Basic nickname sending
      2. JSON packet management and sending
      3. Server-to-Client push receiving and printing

### Server (Austin Swartley & Evan Binkley)

* + - 1. Basic server user setup (Executable startup process)
      2. Thread management for user connections
      3. Stream handling per thread, and JSON packet handling
      4. Server-to-Client message pushing and updating
      5. Signal Handling and Graceful closing

### Library (Peter Schaefer)

* + - 1. Library (lib.rs) will be updated as per sufficient needs (When applicable)

# Programming Languages

* 1. The Client-side program for our project will be written and running in Rust.
  2. The Server-side program for our project will be written and running in Rust.
  3. The Library for our project will be written in rust, and shared between both programs for what is needed.

# Project Scope

* 1. Client-side message and nickname sending
  2. Server-side message receiving
  3. Server-side message management and message pushing / updating
  4. User inputted program termination

# User Interaction

## Client-side

* + 1. User inputs username / nickname
    2. User sends messages to terminal for processing
    3. Client updates board of messages as server pushes and receives them

## Server-side

* + 1. Server awaits port number for socket.
    2. Server processes information and pushes packet based data to clients based on receiving.
    3. Users can input termination signal / character sequence for graceful close.

# Shared Functionalities

* 1. JSON packet formatting
  2. Encoding and decoding packets sent and received

# Application Protocol

* 1. Specifically design JSON packets. Each packet contains meta-data about the type of packet, other possible meta-data including date and time, and the actual message sent.
  2. Client will send a message and then wait until another receiving a packet from the server.
  3. Server will receive a message and then push it to all connected client.

# Sequence Diagram

* 1. The sequence diagram is attached to this submission as a separate file.

# Test Plan

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case Number** | **Brief Description** | **Expected Results** | **Tester Name** | **Date Tested** | **Actual Results** | **Success or Fail** | **Correction** |
| 1 | Nickname taken | Error message stating nickname is taken and then ask for a new name. |  |  |  |  |  |
| 2 | Send a message | Push a message from the server to all connected clients |  |  |  |  |  |
| 3 | Close a client | User inputs the close command and then their client session will be closed and cleaned up on server side |  |  |  |  |  |
| 4 | Long message | Cuts the message to the character limit and informs the user on where the message went over the limit. Sends the most characters that were inside the limit to all clients. |  |  |  |  |  |
| 5 | Empty Message | If a user sends an empty message, then the message should still send to the server and all the other users. |  |  |  |  |  |
| 6 | Send message with non-standard characters | If a user sends a non-standard character, i.e., an emoji, the message should still go through to the server and other users. |  |  |  |  |  |
| 7 | Large amount of users all inputting messages. | If there are many users all sending messages at once, we would like to see no felt drop in performance. |  |  |  |  |  |