CAB403 – Assignment: Process Management and Distributed Computing

TEAM:

Kevin Upton – n9214305

CONTRIBUTION:

All team members contributed equally.

COMPLETION:

Completed task 3 – Seems to be no problem.

Import Structures:

Control	The main control class responsible for holding all information. Setup in the main.
Instance	On the server side, it is the representative for each thread instance. Contains the thread information. Used in the thread pool. An x amount of instances are created on server start. Each instance is created then initially put to sleep, waiting to be assigned a connection. Once the connection has been lost, the instance resets and goes back to sleep.
List	A custom structure used to store an array of any datatype. Handy in storying dynamic arrays that change size. Allowing adding and popping data.
LeaderWorker	The worker responsible for changing user scores, then reorganising their location in the leaderboard. Rather than sorting on retrieval, the leaderboard is reorganised every time someone's score changes. (Separate thread).
Leaderboard	Retrieves a specific amount of users from their organised array, starting from a specific location. (users are organised in better to worse in the array). Contains the data in order at the time of creation.
Event	Responsible for handling incoming packet information. Each packet has a specific event associated to it and the event runs the packet accordingly.
Connection	Responsible for handling the connection between the client and the server. Creating, deleting, sending and receiving of the sockets.
User	Contains the user instance of the game. Only one is created on new user login. After the same user logs in, it reloads the same instance of the User. Contains the name, amount of wins, and the amount played.
Game	Contains information about the current game. If it is on the server side it contains the word, other on the client it just contains the word size and the whereabouts of the known letters. Also keeps track of the number of guesses.

Synchronisation Primitives:

The protection lies in that the client does not contain any of the actual words. The only thing that the client knows is the size of the word_a and word_b, and the whereabouts of the known characters in these words. All of this information is stored in the Game structure.

How it works:

User sends (char) to server.

Server sends back:

- Guesses Remaining
- Word_a with the visible characters and an '_' where there are unknown characters.
- Word_b with the visible characters and an '_' where there are unknown characters.
- Word_a size
- Word_b size
- Guessed characters.

So each time the server responds, it updates the client model, so there is no opportunity for misinterpretation on the client side. Keeping the client in sync with the server at all times.

Victory and loss is calculated also on the server, so there is no opportunity to cheat the user's score.
