PROG2121 – WINDOWS PROGRAMMING

A05 – TCP/IP

OVERVIEW

TCP/IP is the most common inter-process communications protocol. It is an industry standard that allows devices on many platforms to communicate with one another.

This assignment requires you to write a multi-user application supporting a word guessing game.

OBJECTIVES

- Implement simple threaded solutions in Windows applications.
- Compare several methods of inter-process communications between Windows applications.

ACADEMIC INTEGRITY AND LATE PENALTIES

- Please refer to the SET Policies document regarding Academic Integrity Information
- Please refer to the SET Policies document regarding Late Policy

EVALUATION

- Please see the Rubric provided in eConestoga for this assignment

REQUIREMENTS

Submission in groups of 2

You will build a client-server application that will support multiple users at the same time. The application is a simple word guessing game where the application gives you a string of 80 characters and you are to find as many words as you can in that string. The words are only valid if the sequence of characters is in the same direction – either forwards or backwards. For example, in the sequence of the following characters:

thisawhenaeb

you will find: this a saw was when hen an be bean

Client Requirements

1. The application must be written as a WPF application in C#.
2. You must have a way for the user to enter an IP address, Port number, a name that represents the user and the time limit for the game. This data is used to connect to the server. If there is a problem connecting to the server, the user must be informed so the user can try again.
3. If connected successfully, you will be notified by the server sending you a string of 80 characters along with the number of words that are in the 80 characters.
4. You will repeatedly guess words until either you guess them all or the timer runs out.
5. At each guess, the server will tell you if your guess is correct and what the current number of correct words is. It is important to note that the *server* keeps track of the number of correct words.
6. Once the game is completed, the user is asked by the server if they want to play again. If so, it starts another game with a new list of characters. If not, then the server ends the game session.
7. At any point during the game, the user should have a way to end the game session. If in the middle of a game, the server must confirm if the user really wants to end the game.
8. As you make progress through this assignment, you must make regular commits to the GitHub repository for this assignment.
9. You will demonstrate the game in the labs after the assignment has been submitted.

Server Requirements

1. The server is a console application written in C#.
2. The server is a multi-threaded, multi-user application that allows several clients to connect and play the game at the same time.
3. When a client establishes a new connection, the server starts a new guessing game. The server chooses a text file randomly from a configurable directory. The text file must have an 80-character string from which to guess words, a number of words that exist in the character string on the next line, and the words that can be found in the string listed one word per line. The only information sent to the user is the 80-character string and the number of words to be found.
4. At each guess by the user, the server determines if the word is in the list and notifies the user, along with a new number of words still to find.
5. If the user wins or the time limit is exceeded, the server asks if the user wants to play again. If so, a new game is started.
6. If the user chooses to end the session before the game ends, the server asks for confirmation. If confirmed, the game session is ended.

7. The server must be able to be shut down gracefully. This means that all clients running a game at the time of shutdown must be notified so the user can be told about the shutdown.
8. As you make progress through this assignment, you must make regular commits to the GitHub repository for this assignment.
9. You will demonstrate the game in the labs after the assignment has been submitted.

Notes

- In the client, please consider the user interface design. Make the application easily usable.
- The application will be tested by running 1 server, and multiple clients at the same time.
- Communication between the client and server should be "disconnected". Like a web server, the client makes a single request, and the server responds. After that, the TCP/IP connection ends.
- The state of the game needs to be maintained at the server. Each time the client makes a request, the previous state is recalled to decide on the next state. This state information should be stored in an appropriate data structure.
- To facilitate the communication, you should establish a message protocol between the client and the server.
- The server class should be designed in such a way that it does not have any console input or output. You may use the Main() or another class as the UI class.
- You may share the text files with the data for each game.
- *This is a fairly involved assignment. DO NOT leave it to the last minute. If you carefully consider the design before you start coding, it will be easier to implement.*

FILE NAMING REQUIREMENTS

- You will be creating two projects. You may have two separate solutions or just one with both projects in it.
- Although there are no specific naming requirements, please name the projects such that it is easily determined if they are the client or the server.

SUBMISSION REQUIREMENTS

- The full solution(s) must be placed in one zip folder.