1.1 Machine Project Specifications – File Exchange System

Hello Class,

The final output of this course will be a File Exchange System, enabling clients to store, share, and fetch files from a single server using either TCP or UDP protocol. The File Exchange System will consist of a server application and a client application. Additionally, the project may be implemented using Java, Python, or C programming languages. Groups of up to two (2) students are allowed.

Client Application Specifications:

1. The client application will function as the User Interface of a user when using the File Exchange System.
2. An input field should be included to allow the following input commands:

Description	**InputSyntax**	**Sample Input Script**
Connect to the server application	`/join <server_ip_add> <port>`	`/join 192.168.1.1 12345`
Disconnect to the server application	`/leave`	`/leave`
Register a unique handle or alias	`/register <handle>`	`/register User1`
Send file to server	`/store <filename>`	`/store Hello.txt`
Request directory file list from a server	`/dir`	`/dir`
Fetch a file from a server	`/get <filename>`	`/get Hello.txt`
Request command help to output all Input Syntax commands for references	`/?`	`/?`

3. An output area should also be included to display server status from other users as well as system messages from the interaction of the client and the server application.

Description	**Sample Output Script**
Message upon successful connection to the server	`Connection to the File Exchange Server is successful!`
Message upon successful disconnection to the server	`Connection closed. Thank you!`
Message upon successful registration of a handle or alias	`Welcome User1!`
Message upon successful sending a file to server with timestamp (i.e. User1 is storing file to server)	`User1<2023-11-06 16:48:05>: Uploaded Hello.txt`

Message upon successful receipt of the directory list from the server.	`Server Directory`
	`Hello.txt`
	`IMG001.bmp`

Message upon successful receipt of the requested file.	`File received from Server: Hello.txt`

4. Error messages should also be displayed.

Description	**Sample Output Script**
Message upon unsuccessful connection to the server due to the server not running or incorrect IP and Port combination	`Error: Connection to the Server has failed! Please check IP Address and Port Number.`
Message upon unsuccessful disconnection to the server due to not currently being connected	`Error: Disconnection failed. Please connect to the server first.`
Message upon unsuccessful registration of a handle or alias due to registered "handle" or alias already exists	`Error: Registration failed. Handle or alias already exists.`
Message upon unsuccessful sending of a file that does not exist in the client directory.	`Error: File not found.`
Message upon unsuccessful fetching of a file that does not exist in the server directory.	`Error: File not found in the server.`
Message due to command syntax	`Error: Command not found.`
Message due to incorrect or invalid parameters	`Error: Command parameters do not match or is not allowed.`

5. Commands from the user, following the identified input commands and parameters, should be able to be received.

Server Application Specifications:

1. The server application will function as the service or program to which client applications will connect, facilitating interaction among clients in the File Exchange Application.

Note that additional features and functionalities may be considered for bonus points only after all requirements have been successfully fulfilled. Additional features include a Graphical User Interface and support for group messaging. The bonus for additional features may not exceed 20% of the total score for the machine project.

The rubrics for grading, as well as the demo kit, will be provided (refer to the Demo Kit file). The rubrics will include the breakdown of points for each identified functionality and will also outline the scope for possible bonus points. Your projects will be deployed, tested, and evaluated by your instructor using the submitted server and client application. However, your instructor may also request a project demo. To expedite the scoring process, you will be required to upload a short video (screen recorded) demonstrating all the functions specified in the demo kit (upload to your Google or One Drive DLSU account). The video should not exceed 2 minutes. Please ensure the file sharing access option is enabled and share the link of the file in the submission comment section.

The due date for the machine project is August 1, 2024, at 11:59 PM. However, further instruction should be coordinated with your instructor.

Failure to submit or incomplete submission (where the program is not functioning) will result in a grade of 0.0 for this assessment.

Upon completion of the machine problem, all source code and files used in the project should be archived (.zip). If any special libraries are used for the applications, include a list of those libraries and their respective versions for reference during testing.

Have fun!

To aid you in understanding how to implement a basic client-server chat application, you may use the following tutorials and examples for reference:

- Socket Programming Slides, Socket Programming Tasks
- https://realpython.com/python-sockets/
- https://www.studytonight.com/network-programming-in-python/working-with-tcp- sockets
- https://www.studytonight.com/network-programming-in-python/blocking-and- nonblocking-socket-io
- https://www.studytonight.com/network-programming-in-python/working-with-udp- sockets
- https://tutorialedge.net/python/udp-client-server-python/
- https://github.com/ratanak1010/Java-UDP-Chat
- https://www.geeksforgeeks.org/working-udp-datagramsockets-java/