1.1 Machine Project Specifications – File Exchange System

Hello Class, the final output of this course would be a **File Exchange System**, allowing clients to be able to store, share and fetch files from a single server TCP or UDP protocol. The **File Exchange System** would need to be comprised of a server application and a client application. You may also choose to use either C, Java, or Python programming languages for this project. Group up to three (3) students.

Client Application Specifications:

1. The client application would function as the User Interface of a user when using the File Exchange System.
2. The client application should contain an input field to allow the following input commands:

Description	**Input Syntax**	**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. The client application should also contain an output area 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. The client application should be also be able to display error messages

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. The client application should be able to receive commands from the user following the identified input commands and parameters.

Server Application Specifications:

1. The server application would function as the service or program where client applications would connect to, in order to interact with other clients in the File Exchange Application

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

The rubrics for grading as well as the demo kit would be provided (see Demo Kit file). The rubrics would include the breakdown of points for each identified functionality, as well as would include the scope for possible bonus points. Your instructor would deploy, test, and evaluate your projects using the submitted server and client application, however a project demo may also be requested by your instructor. To facilitate the scoring faster, you will need to upload a short video (screen recorded) demonstrating all the functions specified in the demo kit (upload in your Google or One Drive DLSU account). Video should not exceed 2 minutes. Share the link of the file in your submission comment section. Please check the sharing access option of the file.

The due date of the machine project would be on **December 5, 2023 11:59PM**, with the last day of accepting late submissions by **December 8, 2023 11:59PM**, resulting a deduction of 10% per day from the due date.

No submission or incomplete submission (program is not working) would result in a grade of 0.0 for this assessment.

After completing the machine problem, archive (.zip) all the source code and files you used in your project. If you are using any special libraries for your applications, include a list of those libraries and their respective versions for reference during the 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/