Test Report for File Management of Client-Server Application

Assignment-3 is associated with the File Management of client-server application which contains asserts as well as the Unittest module for both the server and client. The entire code for client-server has been equally divided into 6 python files which consist of “*fileopn.py*”, “*fileclient.py*”, “*fileclient1.py*”, “*fileserver.py*”, “fileserver1.py”, and “test.py”.

In the above-mentioned programming files, the “*fileopn.py*” comprises the methods for creating, reading, writing, displaying a list of folders under a current working directory, and moves the current working directory for the current user to the specified folder. The “*test.py*” file includes five different test cases of different users, which has been running successfully thus indicating that every user has their own unique memory space available to execute their instructions for file management.

On selection of the first instruction in a file operation, it helps to create a folder such that the instruction written by the user is tested and when it is executed successfully then it states that the test was passed. Thereafter command is tested and executed in a file operation for writing into a file that might be present in the current working directory or a folder, this functionality of the writing module is also qualified. Subsequently, the third instruction is to read from a given file name, this module was also verified and approved. After that, the fourth instruction is used to display a list of folders present in the current working directory, this functionality was also assessed and passed. At last, the fifth instruction is utilized to change the path of the given folder, this module is also analyzed and verified too.

The Unittest module was imported in the “*test.py*” file. For testing the file operations, there is a Test_FileOpt class in the “*test.py*” file which was created to analyze the “*fileopn.py*”.

These Unittest were chosen for coding because it should be done in a distinct file by including the objective file, where it needs to be tested. The features present in the Unittest module helps to test server and client projects where doctest does not provide like that.

The basic file operations are present in the “*fileopn.py*”.This file provides five different functionalities where each of these functions was assessed and passed with the help of the Unittest module.

Unittesting is done by calling every function and their respective parameters are filled. The result value of these called functions is in turn stored into a variable. Via asserts we can assess both the returned and expected value.

The “*test.py*” file was implemented with five distinct types of test cases of five different users with the respective folder and file names of their choice. As the test cases mentioned in the “*test.py*” are created for a simple understanding of file management of the client-server application. But when this project is scaled up in real-time scenarios then I believe that these mentioned five cases are not sufficient and can be analyzed with multiple amounts of differences in inputs.

Areas like login and register functions present in both the client and server code were not tested. The main reason behind this is when I run the server code the testing could not be able to complete on its whole as the server and client runs simultaneously. For further details of the login and register files, please examine my code files of “*fileserver.py*”, “*fileserver1.py*”, “*fileclient.py*” and “*fileclient1.py*”.