Applied Programming in Python

Assignment Lab 3

A simplified client – server solution for file management

Group members:

1 – Mohammad Hossein Abdsharifi (19890907 - 5791)

2 – Ripan Kumar Dhar (19880727 - 8398)

Introduction

This simple client-server implemented by python is designed to make a connection under the localhost (127.0.0.1) and port 8080 to model a program or an application to handle requests coming from users (client). The server is waiting to respond request incoming from the client as soon as received. This is a simple model for request-respond on the internet which we widely used every day such as, websites, chat applications, and email service. In the following, we will discuss about the client-server modules provided for this assignment. Basically, four main modules are conducted to handle all parts of the tasks as well as possible.

In addition, we will explain how the work is divided between group members to cope with different challenges during the coding. Due to a new situation in the world, such as outbreaking the COVID-19, we did all the discussion remotely, therefore, it takes a bit longer than usual to find an algorithm for completing the task. However, it was a nice experience for both members to solve a challenge remotely. A good platform "*GitHub version control*" was used to have full control of the code and monitoring all the steps by group members.

Client - Server

For doing a simple client-server in python, we start by doing a simple socket programming provided by the course materials and tried to establishing connections under the localhost address. After that, we tried to learn and start working with 'os' libraries and sending requests and receiving responses to understand how we can work with files, folders, and directories. Working with files and folders in python, required to import some libraries and learning many new things from the lectures and the internet. In the following, we will describe how the client-server is looks:

1 – Making a connection between server and client as soon as we run the server module without inserting any commands.

2 – Make a specific folder in the current working directory and check whether the folder is created or not (if not created, showing proper message).

3 – Making a file for writing in the first time and check whether it is able to write in the .txt file.

4 – Making a file for reading for the first time and check whether it is able to read from the .txt file.

5 – Making a new folder with the given name in any working directory.

6 – Change the working directory to any directory.

7 – Registering a new username and password. For each step we mentioned, server is able to handle it or provide proper message in case the user gives the wrong format for the input commands. In

all steps, we tried to find different errors occurred during our trials in order to handle all possible errors.

The table below, we explained about all possible commands that this client-server can handle:

Service Commands	**Description**	**Options**
register <username> <password> <privileges>	In the first step, user should create a username with specific privileges 'admin or user'. Each privilege has its own accessibility.	-
login <username> <password>	The second step is login by the registered username and registered password. If the username and password is not match, a proper message will show to user.	-
change_folder <name>	Move the current working directory *for the current user* to the specified folder residing in the current folder. If the given name is not point to the correct folder's name, a proper message will show.	To walk back to the previous folder, instead of <name>, two dots (..) can return the user to previous folder.
list	The 'list' command is showing all the files and folders in the current working directory with some information such as Size, Time and Date of creation.	-
write_file <name> <input>	Write the data in created text file with the mentioned format. If the pointed folder is not created yet, it has to be created.	If the <input> is empty, the created file will be removed.
read_file <name>	Read the first hundred characters from the given name folder in the current working directory. If the given name is not in the current working directory, a proper message will show.	A service request without a <name> variable should close the currently opened file from reading.
create_folder <name>	Create a new folder with the given name in the current working directory. If the folder with the given name is already exist, a proper message will show.	-
delete <username> <password>	Deleting the registered username with its password. Only privilege with admin level is permitted to delete username.	-

commands	Print information about all available commands, including expected input and what alternative usage they have.	If this command is followed by ‘issued’, then it will show all the commands user is used. If this command is followed by ‘clear’, then all the commands issued will clear from client.
quit	The quit command will logout the user and close the connection to the server.	-

Challenges and difficulties:

The module server is conducted under the one class called ‘my_server’. Module client is also implemented under a class named ‘my_client’ in order to able to make inheritance.

During the programming, we faced various challenges that we had to cope with. Each command has its own challenges, for example, in the ‘write_file’, the user can write in the file, but for the second time, it did not go to the second line.

In the command ‘read_file’, we had some problem with reading the first one hundred characters for each time we inserted the commands.

Initially, after registering a new username and its password, the user was able to login with any password! In this case, we searched a lot to handle our problem and work with dictionaries in python.

A new problem we faced after completing the most part of the program was removing the root folder after using quit commands. For finding this bug in our program, we did debug many times and finally, we found an extra remove operation in our program.

Error Handling:

In order to handle different errors that occurred in the programming, we handled many errors as much as possible to avoid any unexpected errors. Here we listed different exceptions we used in our program:

1 – OS Error

2 – File Exists Error

3 – Fil Not Found Error

4 – Key Error

5 – Attribute Error

5 – Index Error

6 – Index Error

By the mentioned exceptions, we handled 95 percent of error happened during the test. In case an error occurring under these exceptions, a proper message between two stars **‘* proper message *’** will show. In addition, the pylint value is checked to make sure the code is implemented with a high level of coding standards.

Basic requirements for running the program:

The minimum requirement to run the program is having python 3+. The program is able to run in any kind of editor (VS-Code, PyCharm and etc.)

Task Division:

As a team working, both members tried to divide all tasks fairly as much as possible. Since all jobs are conducted remotely, the TeamViewer platform is used to share computers between group members. Almost parallelly with coding, the report was written mostly by Ripan Kumar Dhar. Since one of the computer machines was used Windows OS, most coding part is done by one computer, but the idea and algorithm came up with many discussions and searching on the internet.