Project 1 – Report

I have neither given nor received unauthorized assistance on this work. I will not post the project description and the solution online.

Sign: 1. Mohit Shailesh Kulkarni (1002031021) Date: 02/07/2023

2. Shaunak Gajanan Amble (1002090283) Date: 02/07/2023

Overall Implementation:

In this project we have used Python ver. 3.11.4 as the programming language. We found out that there exists a python library called xml-rpc and have used the same for handling requests between client and the server. In the background, the library uses XML to serialize data between the client and server. We have developed server-side functions in accordance with the task's requirements, and we subsequently run these functions on the client side. Clients can interact with servers using http to rpc calls. In order to access server-side functions, the client must construct a proxy. Clients use this port to make RPC calls after the proxy binds the IP address to the port. This is how the client and server speak to one another.

PART-1

How we Implemented:

- There are two folders and two python files for the client and server.
- It is a menu driven program in which the upload option can be used to upload the file to the server which is present in the client.
- The rename option will rename the file present in the client and the server.
- The delete option will delete the file present in the client and the server.
- Some basic error handling is done, and the success 200 status code can be seen in the server.

What we learned:

- Learned how to use the python xml-rpc library.
- It is complex to implement socket programming compared to the use of xml-rpc library.
- Learned about file handling and synchronization between client and server.

Issues faced:

We were absolutely not familiar with the xml-rpc library and had no idea how to use it. We had to go through the documentation and watch some videos to learn about it. We had some fair amount of idea about programs which upload and download files from a server but had never implemented it so, we learned more about it when we implemented it hands-on.

Screenshots of the results:

Upload

```
mohitkulkarni@FVFH415MQ6LX part1 % python3 client_part_1.py
1 -- Upload File
2 -- Download File
3 -- Rename File
4 -- Delete File
5 -- Exit
Select one of the options given below:1
Please enter the name of the file that you want to upload:start.txt
Uploading the file mentioned above to the server
File is Uploaded succesfully
1 -- Upload File
2 -- Download File
3 -- Rename File
4 -- Delete File
5 -- Exit
```

Download

```
mohitkulkarni@FVFH415MQ6LX part1 % python3 client_part_1.py
1 -- Upload File
2 -- Download File
3 -- Rename File
4 -- Delete File
5 -- Exit
Select one of the options given below:2
Please enter the name of the file that you want to download:start.txt
Finished downloading file
```

Rename

```
mohitkulkarni@FVFH415MQ6LX part1 % python3 client_part_1.py
1 -- Upload File
2 -- Download File
3 -- Rename File
4 -- Delete File
5 -- Exit
Select one of the options given below:3
Enter the name of the file you want to renamestart.txt
Enter new name for the fileretest.txt
Successfully renamed in Client
Successfully renamed in server
```

Delete

```
1 -- Upload File
2 -- Download File
3 -- Rename File
4 -- Delete File
5 -- Exit
Select one of the options given below:retest.txt
 Selected option is invalid. Please try again
Enter the name of the file you want to delete on Client and Serverretest.txt
Deleted file retest.txt from client
Deleted retest.txt from server
```

Server Messages

```
PS C:\Users\AmbleShaunakGajanan\Desktop\proj1final\Proj_1\Proj_1\part1> py server_part_1.py
Server has started
127.0.0.1 - - [02/Jul/2023 14:05:26] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [02/Jul/2023 14:05:51] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [02/Jul/2023 14:06:27] "POST /RPC2 HTTP/1.1" 200 -
```

PART-2

How we Implemented:

- The logic that we used in this part is that in the beginning we initialized the current time and each file in the client folder for consistency checking.
- For synchronized checking we made the thread sleep for 20secs.
- Then the function will run after every 20secs and check for the following things :-
    - All the files in the client folder.
    - The modified time of the files. The current time which we initialized in the beginning will be used here, if the modified time is greater than the current time the file is modified.
    - If the file is modified, then upload it to the server.
    - Then delete the file from the server.

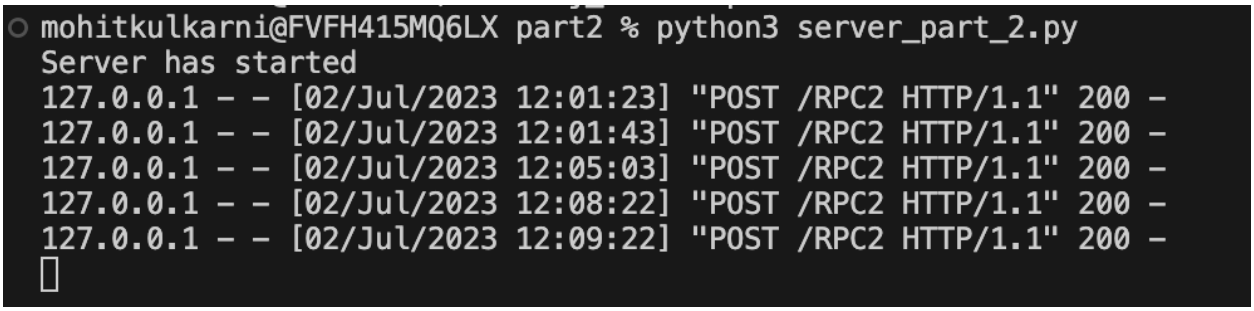
What we learned:

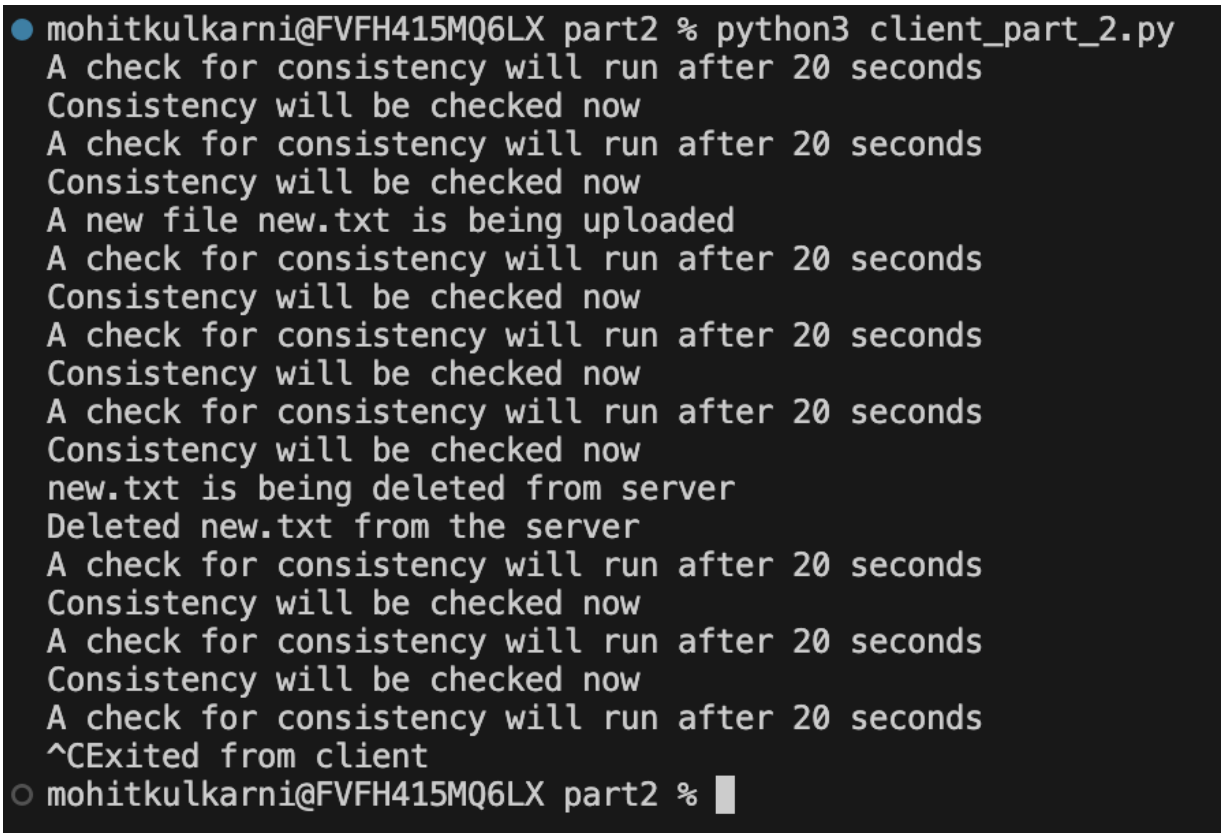
- How to synchronize the client and the server.
- How the live updates work and how the changes made to the client side will affect and make changes to the server side.

Issues faced:

The synchronization of the client and the server side was the toughest part, the changes made in the client side were not getting reflected in the server side. The issue was being caused by the comparison of time.

Screenshots of the results:





PART-3

How we Implemented:

- The client makes RPC calls to the server using the 'xmlrpc.client.ServerProxy' object.
- The server registers the RPC functions and executes the requested operations when called by the client.
- The server maintains a 'THREAD_CLIENT' dictionary to store the results of asynchronous operations.
- The client uses threads to handle asynchronous operations and stores the thread objects in the 'thrs' list for tracking purposes.

What we learned:

- How to create a multi-threaded client for handling concurrent tasks.
- Understanding of synchronous and asynchronous RPC calls.
- Synchronization techniques for thread safety.

Issues Faced:

- Ensuring proper communication and error handling between the client and server.
- Managing thread synchronization and ensuring the correct handling of thread results.

Screenshots of the results:

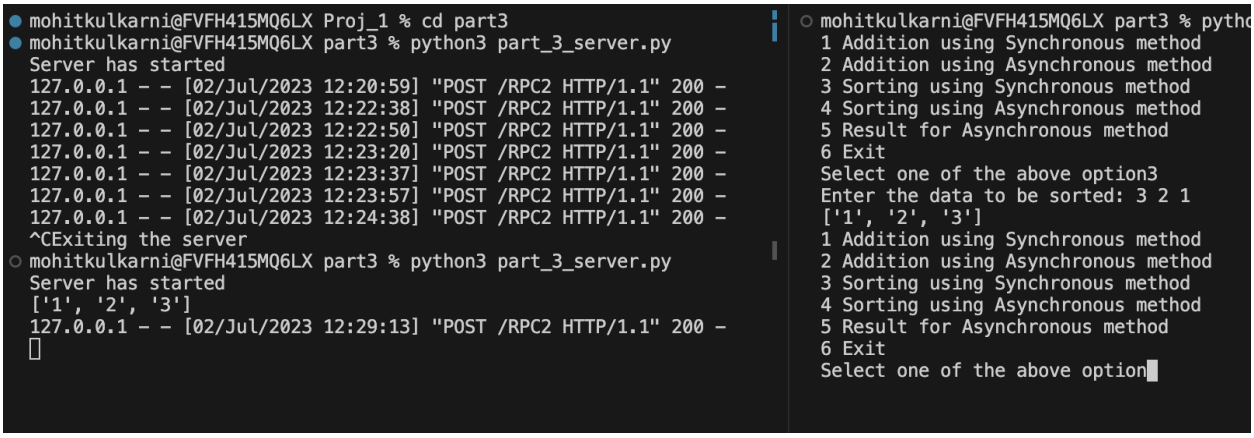
Synchronous Addition

```
1 Addition using Synchronous method
2 Addition using Asynchronous method
3 Sorting using Synchronous method
4 Sorting using Asynchronous method
5 Result for Asynchronous method
6 Exit
Select one of the above option1
Enter the first number: 30
Enter the second number: 40
70
```

Asynchronous Addition

```
1 Addition using Synchronous method
2 Addition using Asynchronous method
3 Sorting using Synchronous method
4 Sorting using Asynchronous method
5 Result for Asynchronous method
6 Exit
Select one of the above option2
Enter the first number30
Enter the second number40
ID for Asynchronous add: Add-25
1 Addition using Synchronous method
2 Addition using Asynchronous method
3 Sorting using Synchronous method
4 Sorting using Asynchronous method
5 Result for Asynchronous method
6 Exit
Select one of the above option5
Please enter the ID:Add-25
70
```

Synchronous Sorting



Asynchronous Sorting