PROJECT 1 REPORT

Preetham Karanth Kota – 1002076418

pxk6418@mavs.uta.edu

I have neither given nor received unauthorized assistance on this work

Signed: Preetham Karanth Kota Date: 09/23/2022

1 Technologies Used:

- Designed and developed the application for the said requirement using Python
- Used XMLRPC library provided by python for remote procedure call between client and server
- Implemented the said application in Linux environment. Any Linux machine would suffice to run this application
- Used Multithreading concept

2 Things Learnt:

- Got good know how about server client model
- Got to know basics of RPC and its application and usage
- Hands on python programming language and multi-threading

3 Project - Part 1:

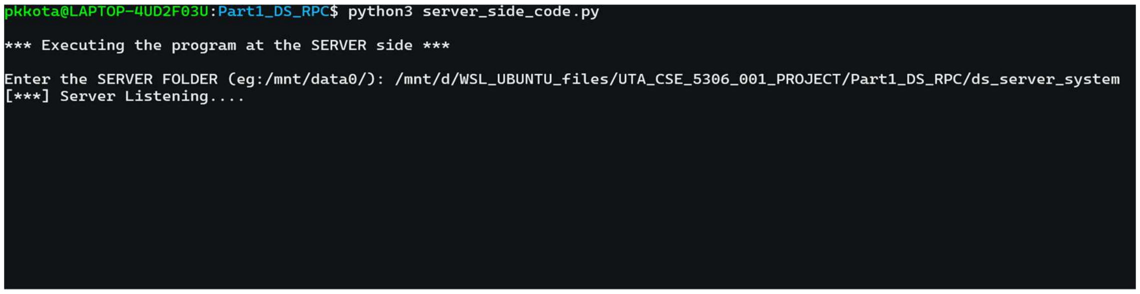
Implemented a multi-threaded file server that supports UPLOAD, DOWNLOAD, DELETE, and RENAME file operations. Use different folders to hold files downloaded to the client or uploaded to the server.

3.1 Implementation:

- Used Linux i.e. ubuntu to develop and design this particular software.
- Option is given to user i.e the user is supposed to enter 1,2,3,4 and 5 to upload, download, delete, rename and exit the program respectively at the client side.
- Before giving the said options to the user, the user is asked to enter the folder where the files are to be stored or picked to perform this operation both at client and server end.
- After entering the folder location, the user is given the option to Upload the file on to the server by entering the file name at client end. Download the file from the server by entering the file name. Delete the file from the server by entering the file name. Rename the file by entering the original file name and the name that is supposed to be the new name of the file.
- The python library named xmlrpc is used at client and server end to communicate via RPC.
- Multi-threading is used at server end where separate thread handles individual file operations.
- Open any Linux termina and run the command "python3 helper_thread_server_side.py" to get the server running
- Open any Linux terminal and run the command "python3 helper_thread_server_side.py" to get the client running

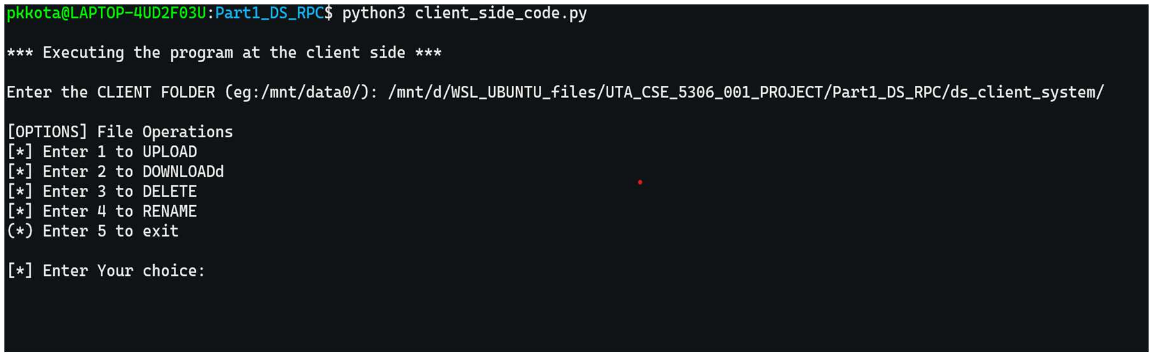
3.1.1 Upload :

- Enter the folder for client and server end. i.e folder from where the files is picked at the client end to send it to the server and the folder where the said file is stored at the server



```
pkkota@LAPTOP-4UD2F03U:Part1_DS_RPC$ python3 server_side_code.py

*** Executing the program at the SERVER side ***

Enter the SERVER FOLDER (eg:/mnt/data0/): /mnt/d/WSL_UBUNTU_files/UTA_CSE_5306_001_PROJECT/Part1_DS_RPC/ds_server_system
[***] Server Listening....
```

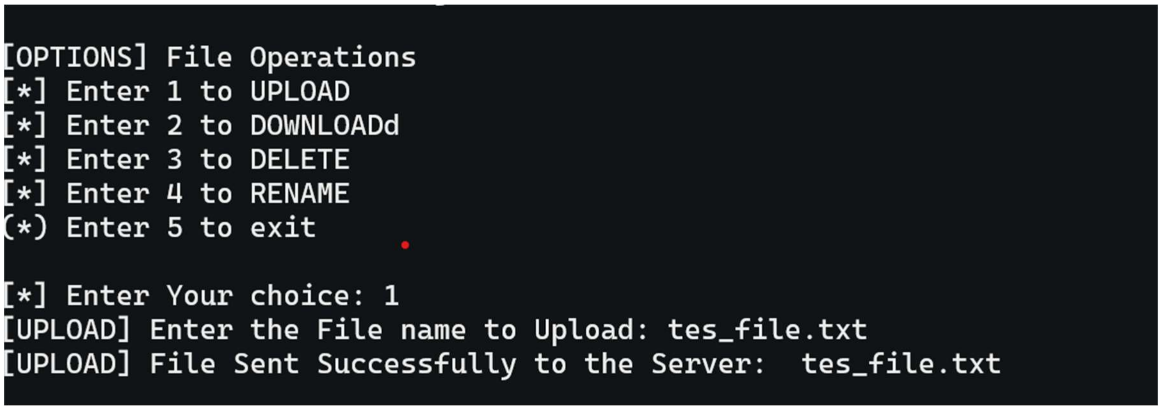
Fig 1: Entering file name at server side



```
pkkota@LAPTOP-4UD2F03U:Part1_DS_RPC$ python3 client_side_code.py

*** Executing the program at the client side ***

Enter the CLIENT FOLDER (eg:/mnt/data0/): /mnt/d/WSL_UBUNTU_files/UTA_CSE_5306_001_PROJECT/Part1_DS_RPC/ds_client_system/

[OPTIONS] File Operations
[*] Enter 1 to UPLOAD
[*] Enter 2 to DOWNLOADd
[*] Enter 3 to DELETE
[*] Enter 4 to RENAME
(*) Enter 5 to exit

[*] Enter Your choice:
```

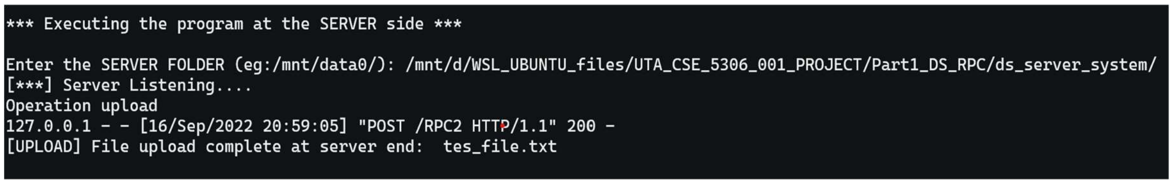
Fig2: Entering the file name at client end

- To upload enter option 1 and the file name present in the mentioned client side folder.The file will be uploaded to or sent to the server end.



```
[OPTIONS] File Operations
[*] Enter 1 to UPLOAD
[*] Enter 2 to DOWNLOADd
[*] Enter 3 to DELETE
[*] Enter 4 to RENAME
(*) Enter 5 to exit

[*] Enter Your choice: 1
[UPLOAD] Enter the File name to Upload: tes_file.txt
[UPLOAD] File Sent Successfully to the Server:  tes_file.txt
```

Fig 3: The file is sent to the server from the client side.

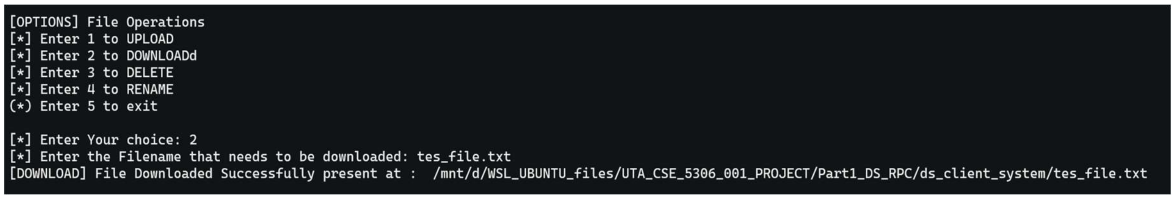


```
*** Executing the program at the SERVER side ***

Enter the SERVER FOLDER (eg:/mnt/data0/): /mnt/d/WSL_UBUNTU_files/UTA_CSE_5306_001_PROJECT/Part1_DS_RPC/ds_server_system/
[***] Server Listening....
Operation upload
127.0.0.1 - - [16/Sep/2022 20:59:05] "POST /RPC2 HTTP/1.1" 200 -
[UPLOAD] File upload complete at server end:  tes_file.txt
```

Fig 4: The file is uploaded or received at the server end from the client

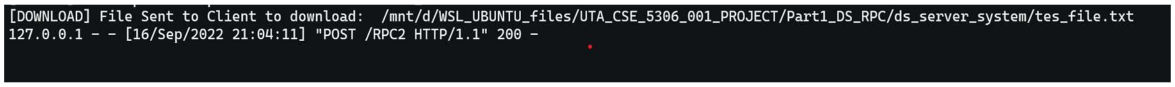
3.1.2 Download:

- Enter the option 2 at client end. And the mention the file that needs to be downloaded from the server.
- Once entered the file will be sent from server to the client



```
[OPTIONS] File Operations
[*] Enter 1 to UPLOAD
[*] Enter 2 to DOWNLOADd
[*] Enter 3 to DELETE
[*] Enter 4 to RENAME
(*) Enter 5 to exit

[*] Enter Your choice: 2
[*] Enter the Filename that needs to be downloaded: tes_file.txt
[DOWNLOAD] File Downloaded Successfully present at :  /mnt/d/WSL_UBUNTU_files/UTA_CSE_5306_001_PROJECT/Part1_DS_RPC/ds_client_system/tes_file.txt
```

Fig 5: File name entered at client side is downloaded

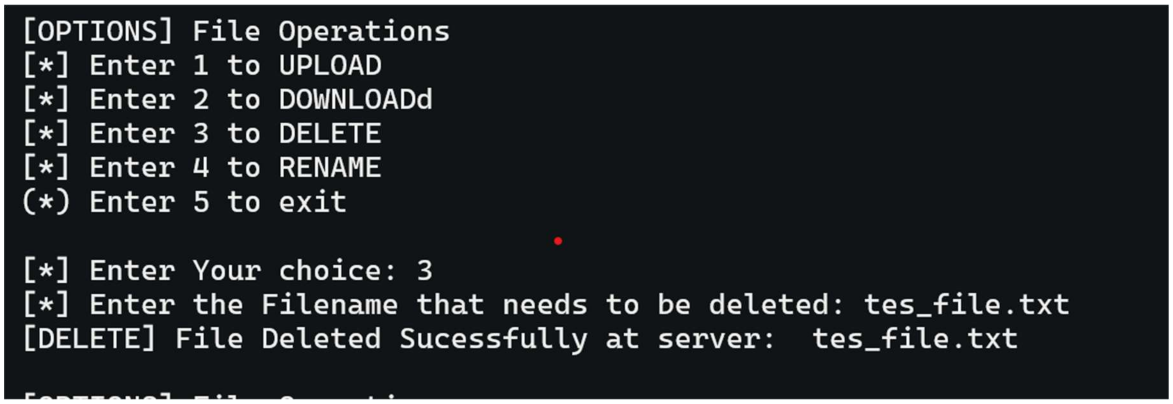


```
[DOWNLOAD] File Sent to Client to download:  /mnt/d/WSL_UBUNTU_files/UTA_CSE_5306_001_PROJECT/Part1_DS_RPC/ds_server_system/tes_file.txt
127.0.0.1 - - [16/Sep/2022 21:04:11] "POST /RPC2 HTTP/1.1" 200 -
```

Fig 6: File is sent from the server to client

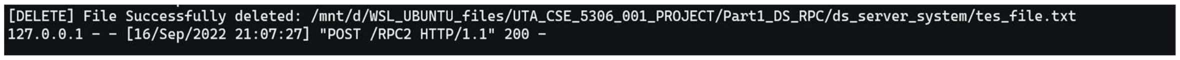
3.1.3 Delete:

- Enter the option 3 at the client end and mention the file name that needs to be deleted. The same would be deleted both at server and client end if the file is present.



```
[OPTIONS] File Operations
[*] Enter 1 to UPLOAD
[*] Enter 2 to DOWNLOADd
[*] Enter 3 to DELETE
[*] Enter 4 to RENAME
(*) Enter 5 to exit

[*] Enter Your choice: 3
[*] Enter the Filename that needs to be deleted: tes_file.txt
[DELETE] File Deleted Sucessfully at server:  tes_file.txt
```

Fig 7: After entering the option 3 and file name the file is deleted at the client.

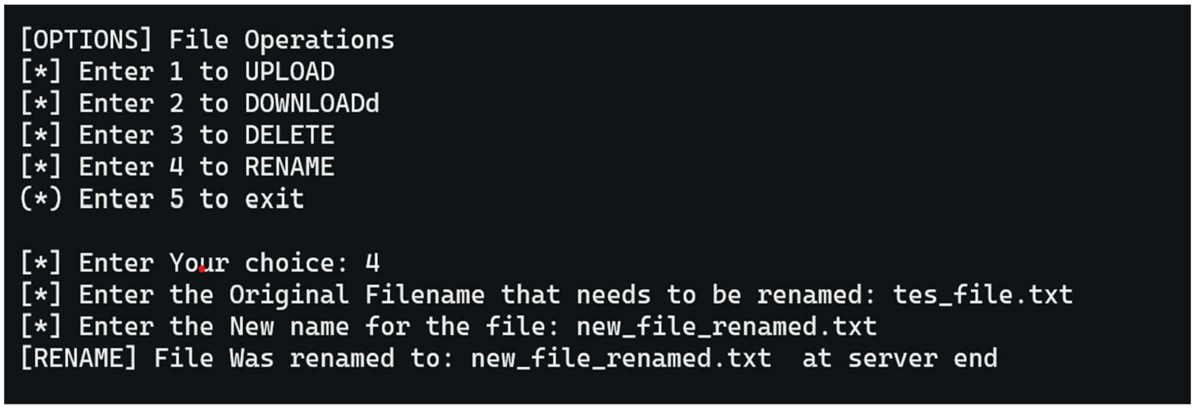


```
[DELETE] File Successfully deleted: /mnt/d/WSL_UBUNTU_files/UTA_CSE_5306_001_PROJECT/Part1_DS_RPC/ds_server_system/tes_file.txt
127.0.0.1 - - [16/Sep/2022 21:07:27] "POST /RPC2 HTTP/1.1" 200 -
```

Fig 8: The file name entered at the client end is deleted

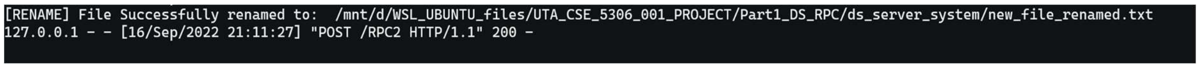
3.1.4 Rename:

- Enter the Option 4 at client end and enter the original file name and the new name. The said file would be renamed to the new name provided both at server and client end.



```
[OPTIONS] File Operations
[*] Enter 1 to UPLOAD
[*] Enter 2 to DOWNLOADd
[*] Enter 3 to DELETE
[*] Enter 4 to RENAME
(*) Enter 5 to exit

[*] Enter Your choice: 4
[*] Enter the Original Filename that needs to be renamed: tes_file.txt
[*] Enter the New name for the file: new_file_renamed.txt
[RENAME] File Was renamed to: new_file_renamed.txt  at server end
```

Fig 9: New and old file name is entered as prompted at client end. The file is renamed as a result.



```
[RENAME] File Successfully renamed to:  /mnt/d/WSL_UBUNTU_files/UTA_CSE_5306_001_PROJECT/Part1_DS_RPC/ds_server_system/new_file_renamed.txt
127.0.0.1 - - [16/Sep/2022 21:11:27] "POST /RPC2 HTTP/1.1" 200 -
```

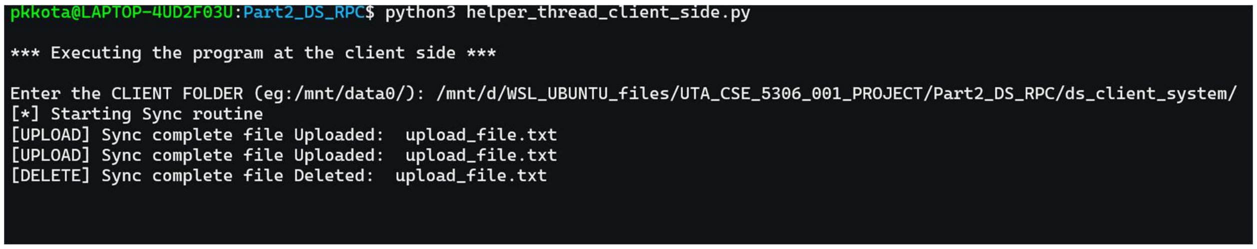
Fig 10: The Said file name is renamed

4 Project Part 2:

A python Application to emulate a synced folder system using RPC (such as Dropbox on Microsoft Windows.

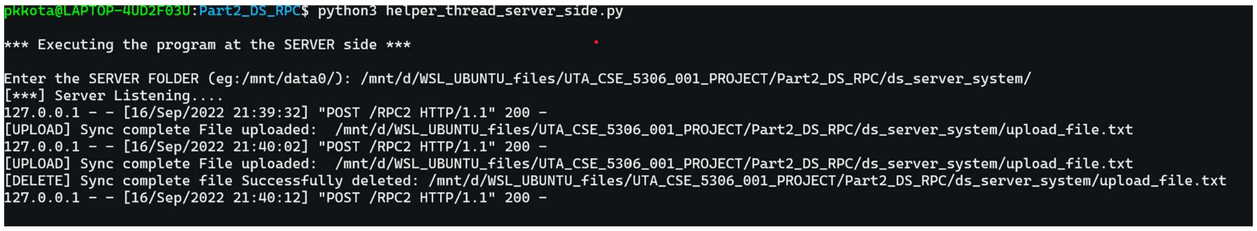
4.1 Implementation:

- Used Linux i.e. ubuntu to develop and design this particular software.
- The user is asked to enter the folder for client and server side which are to be synched.
- A timer thread is spawned every 10 seconds at the client end.
- It checks whether a new file is added , deleted or modified based on the md5hash and the set of files name stored every 10 seconds based on the md5hash stored in a dictionary of the old files.
- After comparison of this md5hash the file is uploaded to server if there is any change or any update in the existing file.If a new file is added the file is uploaded to server. If a file is deleted the same is updated to the server.
- If a file is renamed at client end that operation too is achieved by deleting the old file and uploading the new file to the server.
- Open any Linux termina and run the command "python3 helper_thread_server_side.py" to get the server running.
- Open any Linux terminal and run the command "python3 helper_thread_server_side.py" to get the client running.



```
pkkota@LAPTOP-4UD2F03U:Part2_DS_RPC$ python3 helper_thread_client_side.py

*** Executing the program at the client side ***

Enter the CLIENT FOLDER (eg:/mnt/data0/): /mnt/d/WSL_UBUNTU_files/UTA_CSE_5306_001_PROJECT/Part2_DS_RPC/ds_client_system/
[*] Starting Sync routine
[UPLOAD] Sync complete file Uploaded:  upload_file.txt
[UPLOAD] Sync complete file Uploaded:  upload_file.txt
[DELETE] Sync complete file Deleted:  upload_file.txt
```

Fig 11: File sync i.e upload delete being carried out at client end if there is any change



```
pkkota@LAPTOP-4UD2F03U:Part2_DS_RPC$ python3 helper_thread_server_side.py

*** Executing the program at the SERVER side ***

Enter the SERVER FOLDER (eg:/mnt/data0/): /mnt/d/WSL_UBUNTU_files/UTA_CSE_5306_001_PROJECT/Part2_DS_RPC/ds_server_system/
[***] Server Listening....
127.0.0.1 - - [16/Sep/2022 21:39:32] "POST /RPC2 HTTP/1.1" 200 -
[UPLOAD] Sync complete File uploaded:  /mnt/d/WSL_UBUNTU_files/UTA_CSE_5306_001_PROJECT/Part2_DS_RPC/ds_server_system/upload_file.txt
127.0.0.1 - - [16/Sep/2022 21:40:02] "POST /RPC2 HTTP/1.1" 200 -
[UPLOAD] Sync complete File uploaded:  /mnt/d/WSL_UBUNTU_files/UTA_CSE_5306_001_PROJECT/Part2_DS_RPC/ds_server_system/upload_file.txt
[DELETE] Sync complete file Successfully deleted: /mnt/d/WSL_UBUNTU_files/UTA_CSE_5306_001_PROJECT/Part2_DS_RPC/ds_server_system/upload_file.txt
127.0.0.1 - - [16/Sep/2022 21:40:12] "POST /RPC2 HTTP/1.1" 200 -
```

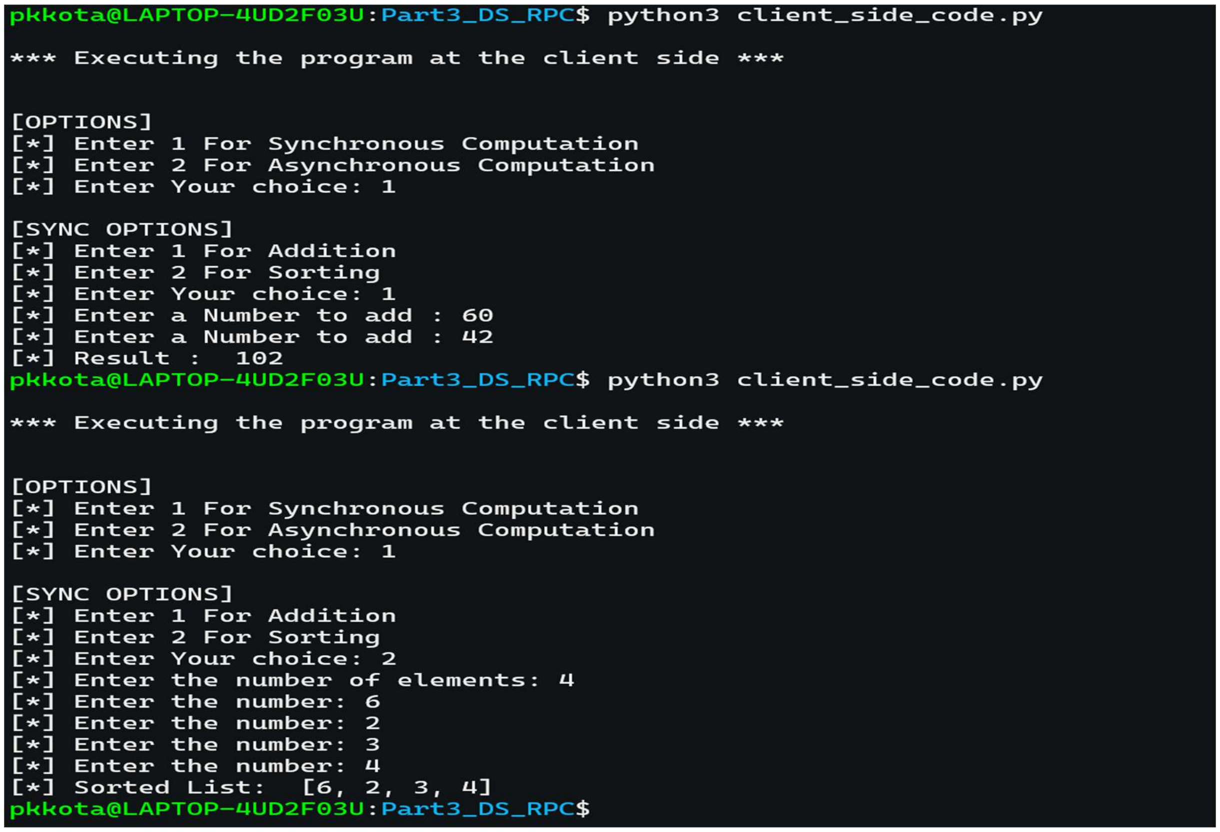
Fig 12: File sync at the server end received from the client

5 Project - Part 3:

A python application which uses RPC to compute add and sort functionality synchronous and asynchronously

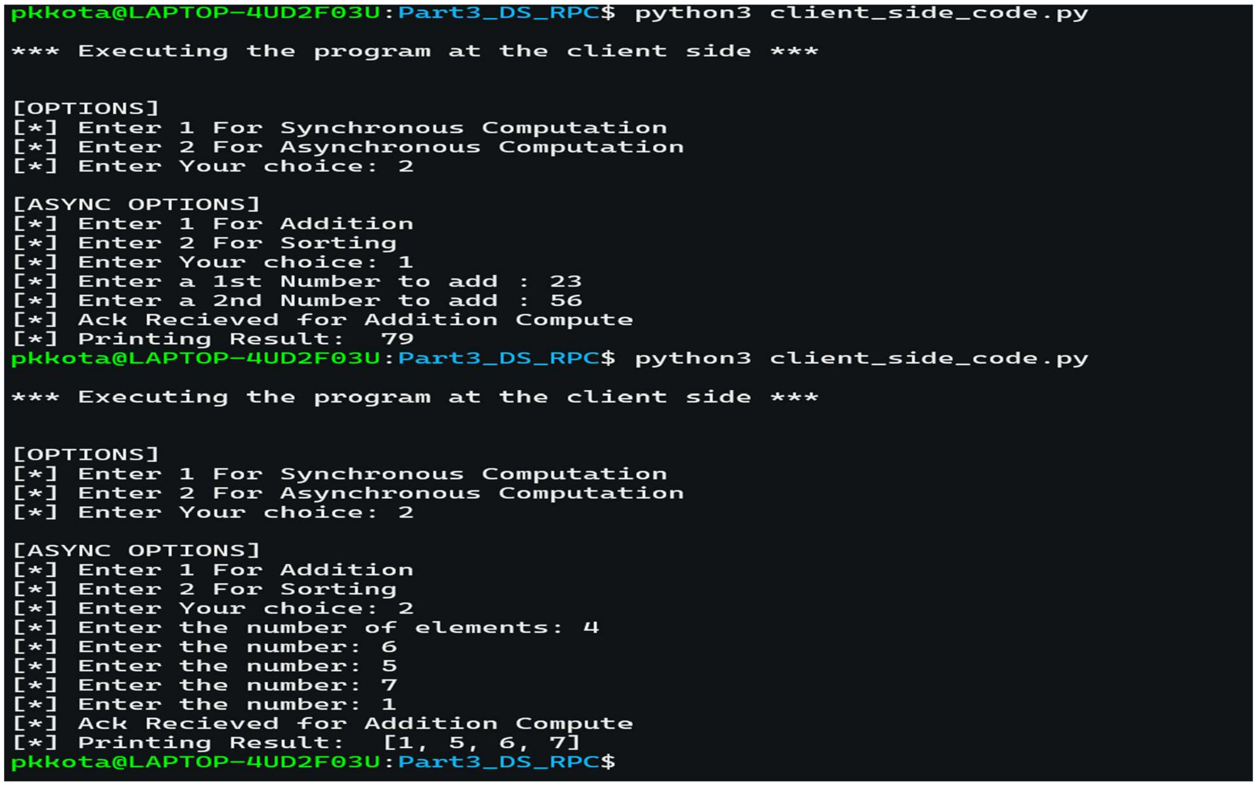
5.1 Implementation:

- Used a simple xmlrpc call from client to server to perform asynchronous add and sort
- The user is asked to enter option 1 or 2 for synchronous and asynchronous compute respectively.
- When chosen for synchronous/asynchronous, the user is then asked to enter option 1 or 2 for addition and sorting respectively
- When chosen addition the user is asked to enter the numbers to add and the result is printed on the prompt
- When sorting is chosen the user is asked to enter the length of the array and the elements. A sorted list is expected at the end of the compute.
- Incase of the synchronous compute be it addition or sorting the client waits for the reply i.e the results from the server
- Meanwhile for asynchronous compute the client doesn't wait for the results from the server instead the server returns none as ack and at the end the result is queried from the client end.
- Regular xmlrpc is used for synchronous communication, multicall functionality from xmlrpc library is used to achieve asynchronous communication.



```
pkkota@LAPTOP-4UD2F03U:Part3_DS_RPC$ python3 client_side_code.py

*** Executing the program at the client side ***


[OPTIONS]
[*] Enter 1 For Synchronous Computation
[*] Enter 2 For Asynchronous Computation
[*] Enter Your choice: 1

[SYNC OPTIONS]
[*] Enter 1 For Addition
[*] Enter 2 For Sorting
[*] Enter Your choice: 1
[*] Enter a Number to add : 60
[*] Enter a Number to add : 42
[*] Result :  102
pkkota@LAPTOP-4UD2F03U:Part3_DS_RPC$ python3 client_side_code.py

*** Executing the program at the client side ***


[OPTIONS]
[*] Enter 1 For Synchronous Computation
[*] Enter 2 For Asynchronous Computation
[*] Enter Your choice: 1

[SYNC OPTIONS]
[*] Enter 1 For Addition
[*] Enter 2 For Sorting
[*] Enter Your choice: 2
[*] Enter the number of elements: 4
[*] Enter the number: 6
[*] Enter the number: 2
[*] Enter the number: 3
[*] Enter the number: 4
[*] Sorted List:  [6, 2, 3, 4]
pkkota@LAPTOP-4UD2F03U:Part3_DS_RPC$
```

Fig 13 : Synchronous add and sort



```
pkkota@LAPTOP-4UD2F03U:Part3_DS_RPC$ python3 client_side_code.py

*** Executing the program at the client side ***


[OPTIONS]
[*] Enter 1 For Synchronous Computation
[*] Enter 2 For Asynchronous Computation
[*] Enter Your choice: 2

[ASYNC OPTIONS]
[*] Enter 1 For Addition
[*] Enter 2 For Sorting
[*] Enter Your choice: 1
[*] Enter a 1st Number to add : 23
[*] Enter a 2nd Number to add : 56
[*] Ack Recieved for Addition Compute
[*] Printing Result:  79
pkkota@LAPTOP-4UD2F03U:Part3_DS_RPC$ python3 client_side_code.py

*** Executing the program at the client side ***


[OPTIONS]
[*] Enter 1 For Synchronous Computation
[*] Enter 2 For Asynchronous Computation
[*] Enter Your choice: 2

[ASYNC OPTIONS]
[*] Enter 1 For Addition
[*] Enter 2 For Sorting
[*] Enter Your choice: 2
[*] Enter the number of elements: 4
[*] Enter the number: 6
[*] Enter the number: 5
[*] Enter the number: 7
[*] Enter the number: 1
[*] Ack Recieved for Addition Compute
[*] Printing Result:  [1, 5, 6, 7]
pkkota@LAPTOP-4UD2F03U:Part3_DS_RPC$
```

Fig 14: Asynchronous sort and add.