Report Project 1

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

Sign: 1. Vaibhavi Purushottam Arjunwadkar Date: 8 Feb 2023

2. Mrunmai Nitin Magar Date: 8 Feb 2023

Group Member Information:

Member 1: Vaibhavi Purushottam Arjunwadkar (1001826818)

Member 2: Mrunmai Nitin Magar (1002092125)

Implementation:

We have used python's xml-rpc standard library file for handling requests between client and server. The module employs XML in the background. for data serialization between the server and client. We have created two separate files for each of the three tasks, one for the client-side coding and other for the server-side coding. According to requirement in the task, we have created functions in the server and then we execute these functions from the client side. Client can execute by http to rpc calls to server. Client needs to create proxy to access server-side functions. Proxy binds IP address to the port and then this port is used by client to make rpc call. This is how client communicates with server.

Part 1:

How we have Implemented:

- Created two folders one for client and one for server
- Upload option can be used to create the file in the server which we have created in the client
- Rename option can be used to rename the specific file at client as well as at the server side.
- Delete option can be used to delete file from both client as well as from the server side.
- On server side it shows POST and 200 as a status code with IP address and date as I have kept the logging value as a True at the server-side coding.

What we have learned:

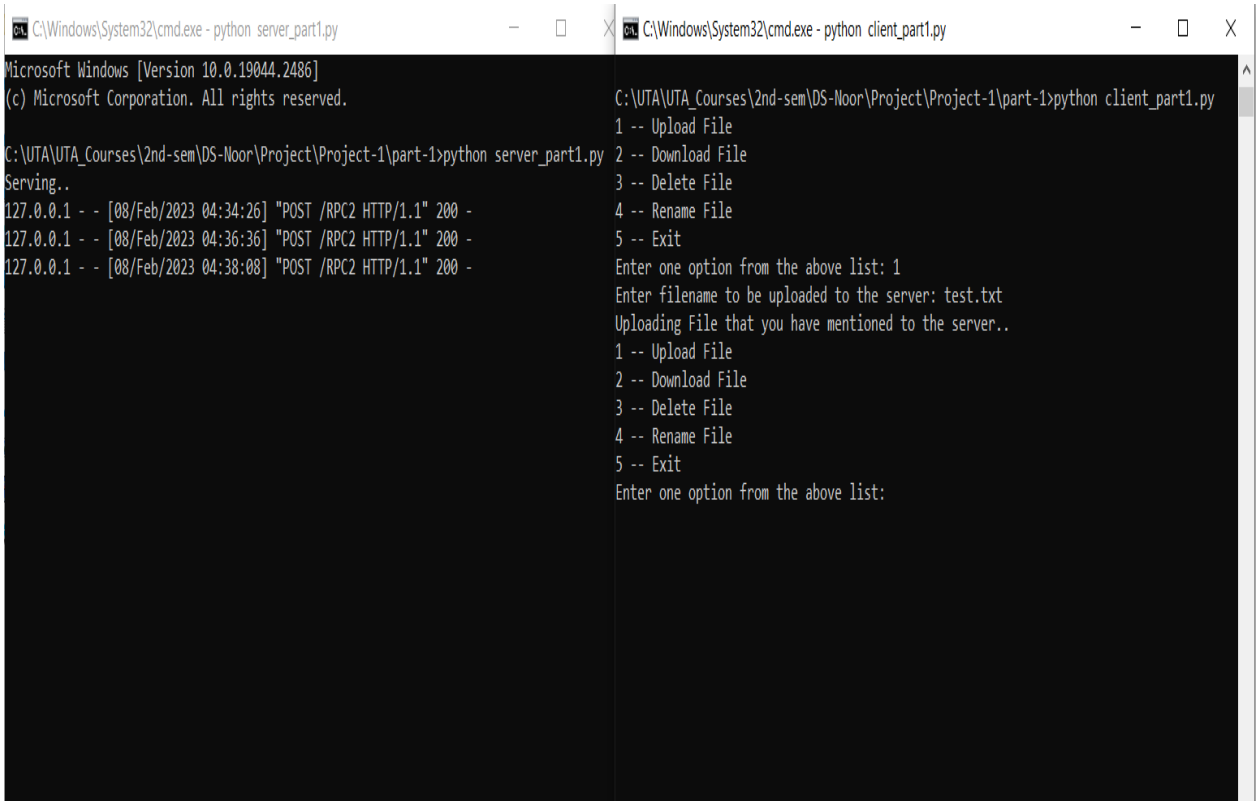
- How the xml-rpc library works in python
- How to follow pythonic way of coding to get a clean and understandable code.
- The benefit of using xml-rpc module which saves lot of time and energy instead of using traditional way of socket programming.

Issues Faced:

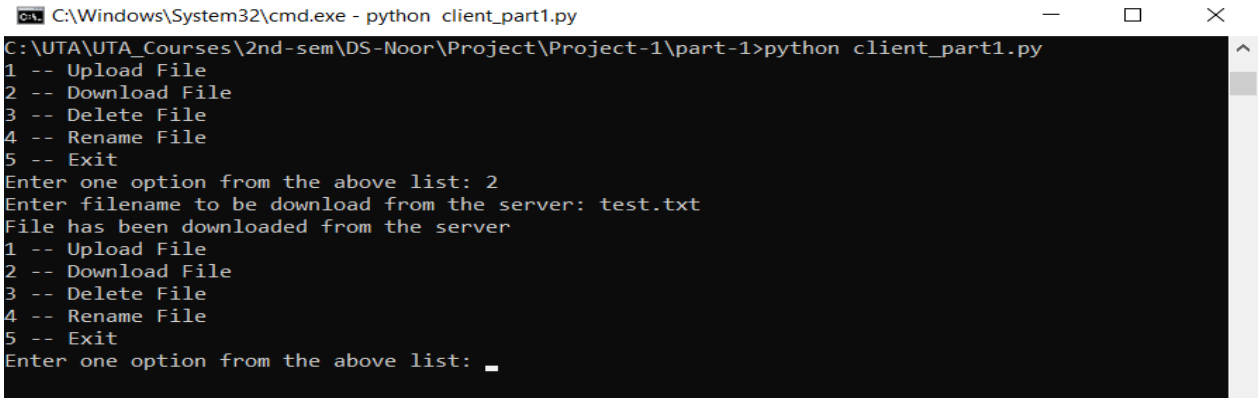
We both are new to python language and was not aware of how to use or implement xml-rpc. We have faced lot of issue like what kind of functions are supported by xml-rpc and when to use which function etc.

Screenshots for each operation:

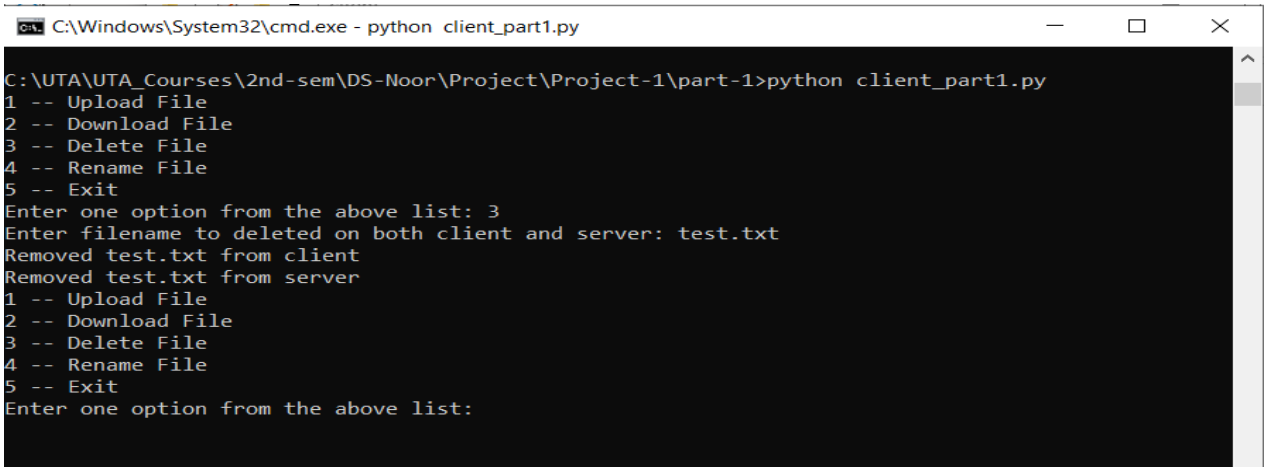
Upload



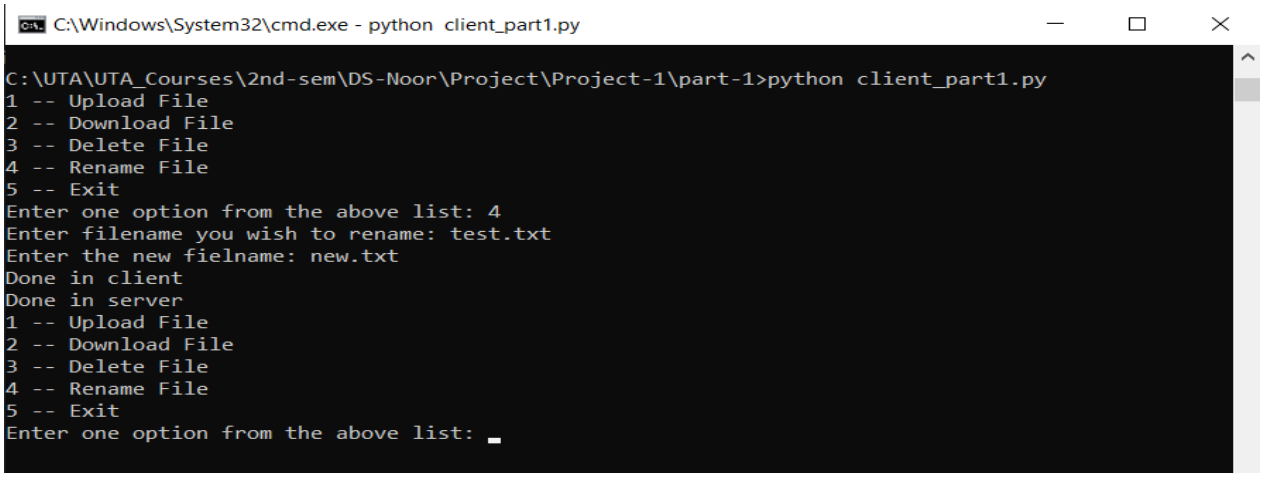
Download



Delete



Rename



Part 2:

How we have Implemented:

- We have initialized current time and all the files in client folder before running the loop to check consistency.
- We have made the process/thread sleeps for 15 sec
- Scan function runs after every 15 secs and will check the below things:
  - List of files in client folder
  - Check the modified time of each file.
    (If the modified time > current time = file is modified)
  - Upload file to server
  - Delete the file in server

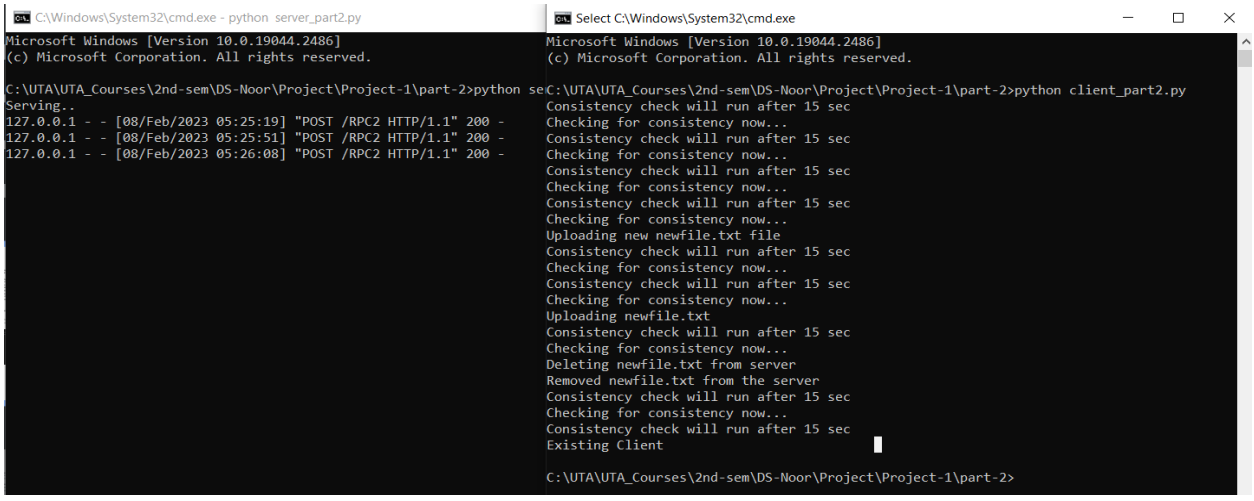
What we have learned:

- How and where to implement sleep function.
- How to reflect changes from client side to server side.

Issues Faced:

- Was unable to implement scan for consistency after every 15 seconds.
- Faced difficulty while comparing the current time and the modified time of the file while deletion.

Screenshot:



Part 3:

How we have Implemented:

- Created two python files naming server_part3 and client_part3.
- To execute synchronous operations, we have used basic xml-rpc module function that triggers function from client to rpc call to server function
- To execute asynchronous operations, we have used threading module to start thread, save the entry in list.
- When addition is chosen, the user is required to enter the numbers to be added, and the result is displayed on the prompt.
- When sorting is selected, the user is asked to input the array's elements. At the end of the computation, a sorted list is expected.
- Result is saved on server side with id and result, and for displaying the result select from the menu given on client side by inputting given thread id.

What we have learned:

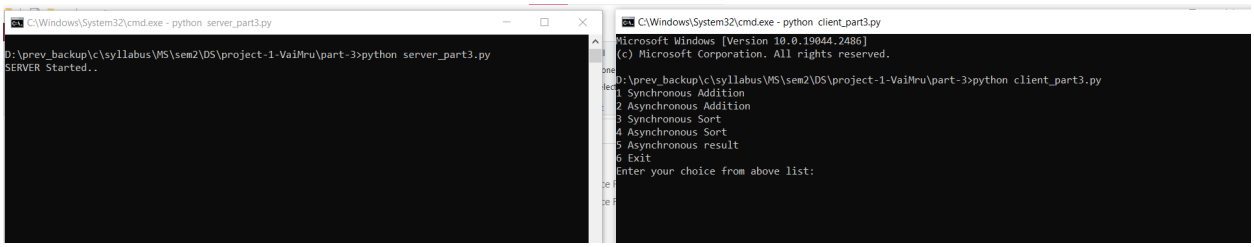
- we learnt to save result in server side instead of client side which is comparatively easy to implement.
- Learned about main thread and one of the limitation of python that is one thread policy.

Issues Faced:

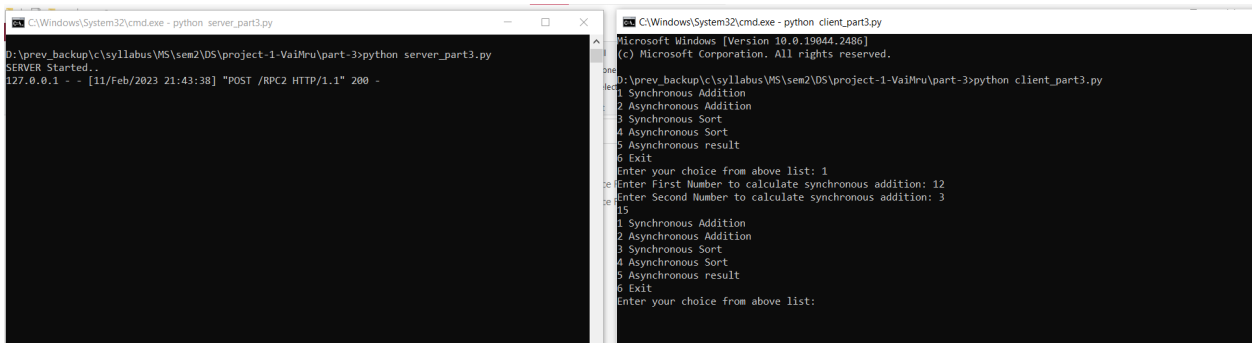
- While writing and executing code for synchronous addition operation, we were having the output as a concatenated string of numbers and not the result of addition operation. We got rid of that error by correcting the logic for that block of code.

\Screenshots for each operation:

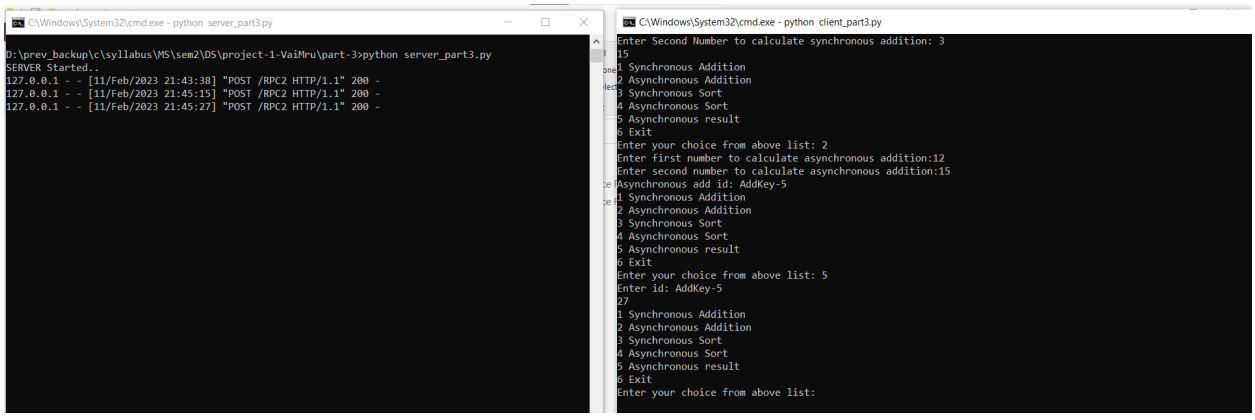
1. **Displaying menu**



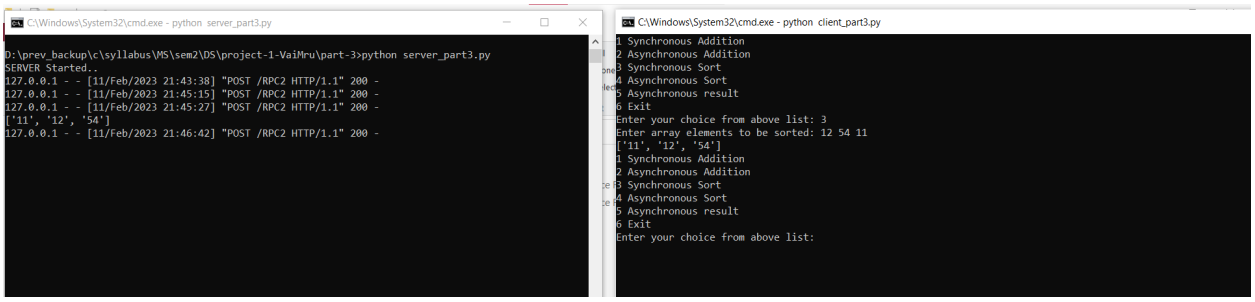
2. **Synchronous operation (Addition)**



3. **Asynchronous operation (Addition)**



4. Synchronous operation (Sort)



5. Asynchronous operation (Sort)