**Project 1**

Team Members

**1. Harsh Shah - 1002057387**

**2. Vashishth Gajjar - 1002160256**

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

Sign: Vashishth Gajjar Harsh Shah Date: 09/15/2024

Implementation

We divided project implementation with 4 main phases as follows:

1. **Planning** –

* Defining project’s requirements – This project entails developing a computation server for synchronous and asynchronous remote procedure calls (RPC) supporting add and sort operations, as well as a multi-threaded file server, supporting file operations and a synchronized storage service like Dropbox for automatic file synchronization.
* Selecting technologies – It was difficult to choose between the Java and Python programming languages. We chose Python owing to all the available libraries, such as "*xmlrpc*" for creating RPC and threading for developing asynchronous RPC or multithreading.
* Purpose – We analysed what exactly each function should perform.
* Created test files with dummy data.

1. **Coding** –

* Prior to beginning the actual coding, we explored the libraries, built-in functions, and the way they were implemented.
* We started with server implementation for a multi-threaded file server (Part 1), which includes the subsequent functions: upload, download, delete, and rename. Later, we completed the client implementation, which communicates with the server to conduct the previously described file actions.
* Implemented Part 3 for computational operations like addition and sorting, using synchronous and asynchronous RPCs.

1. **Testing** -

* Evaluated the server and client files with all the edge test cases to ensure all the operations are correctly executed. Evaluated whether RPC works as per expectations.
* Analysed if all the requirements gathered during planning phase are satisfied.
* Managed errors such as “file not found”, “server unavailable”.

1. **Documentation** –

* Developed a README file for the whole project, which includes instructions on how to compile and run the program.
* The final stage was to write this report. We developed a concise report that covers the project's general flow.

Learnings

1. **RPC** - Understood the concept of RPC for synchronous and asynchronous, where in synchronous RPC blocks the client until the server completes the request and returns response; whereas asynchronous RPCs allow the client to continue executing other tasks while waiting for a response from the server. We explored various libraries such as “*xmlrpc”* & *“asyncio”*.
2. **Multithreading** - This capacity of simultaneous execution is particularly useful when clients need continuous service without interruptions from time-consuming tasks.
3. **Asynchronous** - Asynchronous programming is required for managing non-blocking operations and enhancing application responsiveness. It includes implementing asynchronous RPCs, which allow clients to send requests and perform other tasks while awaiting a response from the server.

Issues Encountered

1. **Multithreading** - Managing several threads for each request caused implementation difficulties as well as certain undefined errors
2. **Incorrect outputs** - These errors resulted in unwanted output
3. **Time-consuming** - Asynchronous implementation in the computation server was a time-consuming task. It was easy to comprehend in theory, but it took longer to implement
4. We brainstormed all potential test scenarios and implemented those that were not satisfied.

Team members contribution

All the implementation, report and documentation were done by both of us, but the major role is as follows:

1. Harsh Shah - *File Server*
2. Vashishth Gajjar - *Computation Server*