I have neither given nor received unauthorized assistance on this work.

Signed: Hardik Jain (ID: 1001954448) Date: 10/07/2021

Perin Shah (ID: 1001867540)

**README**

**Project Content:**

Assignment 1 Folder contains files:

1. client.py
2. server.py
3. client (Folder)
4. server (Folder)
5. Readme.docx

Assignment 2 Folder contains files:

1. client1.py
2. client2.py
3. server.py
4. Readme.docx
5. Client1 (Folder)
6. Client2 (Folder)
7. Server (Folder)

Assignment 3 Folder contains files:

1. server.py
2. client.py
3. Readme.docx

Assignment 4 Folder contains files:

1. server.py
2. client.py
3. Readme.docx

**Instructions to compile the programs:**

* 1. Unzip the files into a folder. There will be 4 sub-folders in it.
  2. For further details, see the Readme File in each directory.

**Libraries used:**

1. Socket
2. OS
3. Threading
4. Logging
5. Pickle
6. Json
7. Numpy
8. Time

**Command to Install libraries**

1. Pip3 install {library\_name}

**Description:**

***Assignment1:***

Implement a basic single-threaded file server that supports the four basic operations UPLOAD, DOWNLOAD, DELETE, and RENAME. It was implemented using message-oriented communication protocol. The client and the server reside on the same machine but communicate with each other using different ports. Different folders are used to hold server files and client files to perform the above-mentioned operations.

***Note: Read the Readme File in Assignment 1 Folder for the instruction to run the program.***

***Assignment 2:***

Implemented a multi-threaded file server that supports the four basic operations: UPLOAD, DOWNLOAD, DELETE, and RENAME. It was implemented using message-oriented communication protocol. Multiple clients and server reside on same machine but communicate with each other using ports. There are three folders: “Client1”, “Client2” and “Server” to store files on which above operations are performed.

***Note: Read the Readme File in Assignment 2 Folder for the instruction to run the program.***

***Assignment 3:***

Designed a computational program which is based on Synchronous Remote Procedure Call (RPC). Parameters are taken from the user and sent to the server where all computation takes place. While the server performs computation, client waits for the result. After completing the operation server sends the result back to client. Server performs the following operations:

* Addition of 2 numbers
* Calculate the value of PI
* Sort an ARRAY
* Matrix multiplication.

***Note: Read the Readme File in Assignment 3 Folder for the instruction to run the program.***

***Assignment 4:***

Designed a computational program which is based on asynchronous and deferred synchronous Remote Procedure Call (RPC). Parameters are taken from the user and sent to the server where all computation takes place. Meanwhile the client keeps on performing next operations and once server has completed the operation, it sends the output to client. Server performs the following operations

* Addition of 2 numbers
* Calculate the value of PI
* Sort an ARRAY
* Matrix multiplication.

***Note: Read the Readme File in Assignment 4 Folder for the instruction to run the program.***

**Concepts Learned:**

Learned how the message-oriented protocol uses sockets to send and receive messages on the server. We also learned the difference between single threaded and multithreaded systems and how to build them. Also learnt when to use Remote Procedure Call (RPC) and when to use Message Oriented Protocol (MOP), as well as the differences between RPC and MOP. Implementing applications utilizing multiple messages passing protocols provided insight into how to convey parameters or messages to the Server and back to the Client in various ways.

**Issues Occurred:**

* Changing a single-threaded server to a multi-threaded server that receives requests from all clients.
* Implementing RPC without using libraries.
* Controlling multiple threads while implementing asynchronous and deferred synchronous Remote Procedure Call.

**Why Locking Is Important?**

When one client uses a file to conduct an action and other clients use the same file at the same time in a Multithreaded Server, the result of the computing operation performed on that data will be different. To overcome this problem, the resource on which one thread is now working should be locked. As a result, the other thread will have to wait until the thread that is accessing the resource leaves it after performing the operation on the data. This would result in a Multithreaded Server that is safe and maintains consistency.