**Distributed Systems**

CSE 5306-002 PROJECT REPORT

**FILE CLIENT-SERVER COMMUNICATION IN MULTI-THREAD**

**Submitted by**

Shubash Muniyappa (1001915563)

Sai Rohit Kalyan Gandham (1002070724)

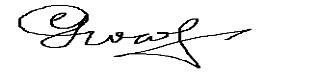
Instructor: Jia Rao

**Academic Integrity**

I have neither given nor received unauthorized assistance on this work

Signed:A picture containing diagram

Description automatically generated Date: 10/04/2022

Signed  Date: 10/04/2022

**Introduction**

The proposed project focuses on leveraging REMOTE PROCEDURE CALL (RPC) communications to establish "multi-threaded file server connectivity." This is one of the few projects that has the ability to CREATE, RENAME, UPLOAD, DOWNLOAD, and DELETE a file on both the client and server sides.

Remote Procedural Call is a communication protocol that might be helpful for sending a client-to-server service request. RPC employs a client-server architecture. When a user attempts to carry out any of the CRUD operations in this project, the server side will synchronously update.

A class that unites the client and server code can automatically handle and make the file transfer between the client and server transparent to users. We can also observe the compute server in this project supporting synchronous RPC add(i,j) and sort(array A) actions. In asynchronous RPC, the client places the RPC call, waits for the server to acknowledge it, and then moves forward after it does. On the other side, the server returns the call's outcome to the client after finishing the calculation.

**Implementation**

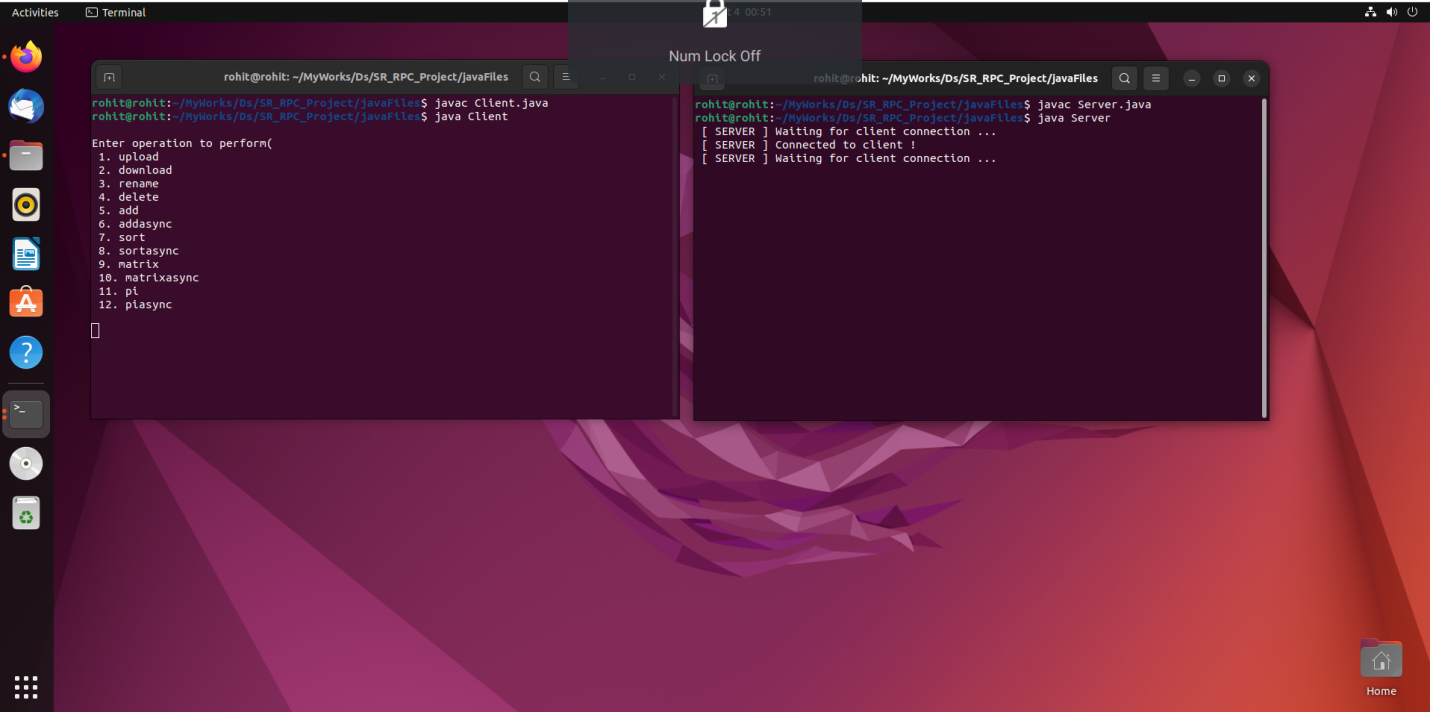
* We wrote the entire project in the JAVA programming language. There are three program files client code, server code and client handler code in this project.
* We are using localhost port \*\*\*\* on the web server. The server implemented in such a way that it is always running and accessible by multiple clients at the same time for various operations.
* Client code is initiated and connected to server address using server port. Concept of multiple threads is used for a single client to perform multiple operations.
* Client handler class has been created where various operations are defined to handle file such as fileDownload(), fileUpload(), fileDelete(), fileRename().
* We initially implemented the code in InteliJ (IDE) and made necessary changes for the code to work in Ubantu environment.

**Challenges**

* Dealing with asynchronous and synchronous server connection was the biggest problem, but we were able to get through it by utilizing the async and await attributes.
* Secondly, we faced major problem in implementing matrix multiplication, server reads the input as a string. Hence, we converted the strings to integers by splitting with **“:”** and **“,”**.

**Observations**

* Once both server and client codes are complied, client establishes the connection with the server.



* We can initiate multiple clients to simultaneously request for multiple Operations.

Text

Description automatically generated

* All of the material contained in the chosen file will be seen when you select the download option. If "download" is present, we verify the client code parameter, and the client code will call the download method.
* A new file will be created in the project's existing folder when the upload feature is used. The client code is checked for the parameter, and if it contains the word "upload," the upload method is invoked.
* The filename that was already present will be changed to the one you specify when utilizing the rename capability. If the word "rename" appears in the parameter in the client code, the rename method is called.
* The existing file that is present in the project's folder will be deleted when you use the delete option. The client code is examined, and if the parameter is "delete," the delete procedure is invoked.
* The server processes requests sent by clients and sends an acknowledgement back to the client in synchronous RPC.
* Every time a client submits a request to the server, regardless of whether the request is being processed or not, the server sends an acknowledgment which is referred as asynchronous RPC.
* We couldn’t observe any difference in addition operation in synchronous and asynchronous communication. But it is thinly found in matrix operation.

Text

Description automatically generated

**Contributions**

* Server-Client communications including basic file operations like Upload, Download, Delete and rename is implemented by Shubash.
* Synchronous and asynchronous RPC is established and Add, Sort, matrix Multiplication, value of Pi is implemented by Rohith.
* Readme file and report was of a mutual contribution.