Executive Summary

1. Assignment Overview -

The purpose of this assignment is to enhance our understanding of distributed systems by implementing Remote Procedure Calls (RPC) and multi-threading in a key-value store project. This assignment, which builds on Project #1, calls for switching client-server communication from socket-based interactions to RPC in order to standardize and streamline remote calls and enable more seamless communication. Implementing RPC, possibly through Java RMI introduces us to methods of efficiently invoking functions on remote servers. The project also requires that we turn our server into a multi-threaded system that can process several client requests for PUT, GET, and DELETE operations at the same time. To avoid data inconsistency from concurrent operations, this calls for the use of thread pools or implicit concurrency management techniques like concurrent hashmap while guaranteeing mutual exclusion.

2. Technical Expression -

Completing this assignment deepened my understanding of remote client-server interactions and concurrency management in distributed systems. Transitioning from socket-based communication to Remote Procedure Calls (RPC) introduced challenges that pushed me to explore Java RMI. Moving the client-server communication to RPC not only streamlined the communication process but also offered a new perspective on creating more scalable and modular networked applications.

Implementing a multi-threaded server required me to think critically about synchronization and mutual exclusion to prevent race conditions during concurrent PUT, GET, and DELETE operations. The experience highlighted the importance of balancing concurrency and resource management, as efficient threading was necessary to avoid bottlenecks without compromising data accuracy. Overall, this assignment gave me a solid foundation in managing concurrent operations and handling remote method invocation, both critical for modern distributed systems. These lessons will guide me in future projects where scalable, responsive and robust server-client architectures are essential.