**Name: Aditya Deshpande**

**Assignment Overview:**

The project assignment is designed to implement an easy key-value store client-server application. The communication between client and server would be through network protocols like TCP and UDP. The client takes input from the user through CLI to send GET, PUT, DELETE requests to the server. The server is designed to process these requests and store key value pairs in HashMap and then retrieve or delete values based on the user requests. The client should be robust and should handle errors efficiently such as server timeouts or malformed packets. It should continue sending valid requests even if previous requests fail. Both client and server should have a logging mechanism to log each activity taking place with precision in terms of milliseconds. The server should handle malformed requests effectively without crashing and log these requests to its log file. The overall system should support two types of protocols: TCP which is a connection-oriented protocol which guarantees delivery of packet and UDP which is connectionless protocol i.e. faster but does not guarantee delivery of datagram packets. The project involves pre-populating keys through client-side so that server can handle 5 PUT, 5 GET and 5 delete requests immediately. The server will run infinitely, waiting for requests from client. The client application is responsible for making a connection with the server and sending the request. The project helps in practicing concepts related to Socket programming, network protocols and communication practically. It also focuses on the importance of error handling mechanisms, logging feature so that client and server continue to be operational in case of some network failure or incorrect input from the user. The overall goal of this project is to create a reliable, scalable distributed key-value store server following best practices.

**Technical Impression:**

Implementing this project helped me in learning to program TCP, UDP protocols in JAVA. The process of implementing communication channel between a client and a server using TCP and UDP protocols helped me understand their differences. As UDP is connectionless protocol, it has to be secured and checks should be in place through our code to maintain its reliability. This made me understand use of error-handling mechanisms so that client-server does not crash even in case of invalid parameters or results. Socket programming is important for enabling direct channel of communication between the client and the server using TCP or UDP allowing them to transfer data efficiently over the network. The logging feature in which both the client and server interactions are recorded with timestamps with precision of milliseconds made me realize how important it is to track down the bugs and debug them in real world and how this logging makes debugging easy. The project also helped me in understanding the importance of ports, how to handle communication through a port. I also liked the deployment of my project on docker. Docker is important for this project as it helps in creating consistent environments for both client and server applications making it easier to run and manage them across different systems. By containerizing the application using Docker, compatibility issues are reduced during deployment and execution. Initially I faced challenge in implementing docker but the Readme file provided steps which helped me in implementing it.

I think this application has several use cases. For example, maintaining a shopping cart on Amazon where it is important to map items to user ID of logged in user. Along with this, I think this application has use case in caching mechanisms. Frequently accessed data is cached in the map with particular id and the performance of web application is improved. Overall, this project helped me in developing a practical foundation in development of distributed system, it would be interesting to now work on project 2 which would involve RPC and multithreading.