**Rohit Kumar Sadhukhan**

**BCSE III A1**

**Roll : 001910501023**

**Internet Tech.**

**Assignment 1**

**Problem Statement:**

Implement a key-value store using socket programming. The server implements the key-value store and clients make use of it. The server must accept clients’ connections and serve their requests for ‘get’ and ‘put’ key value pairs. All key-value pairs should be stored by the server only in memory. Keys and values are strings.

The client accepts a variable no of command line arguments where the first argument is the server

hostname followed by port no. It should be followed by any sequence of “get <key>” and/or “put <key> <value>”.

./client 192.168.124.5 5555 put city Kolkata put country India get country get city get Institute

India

Kolkata

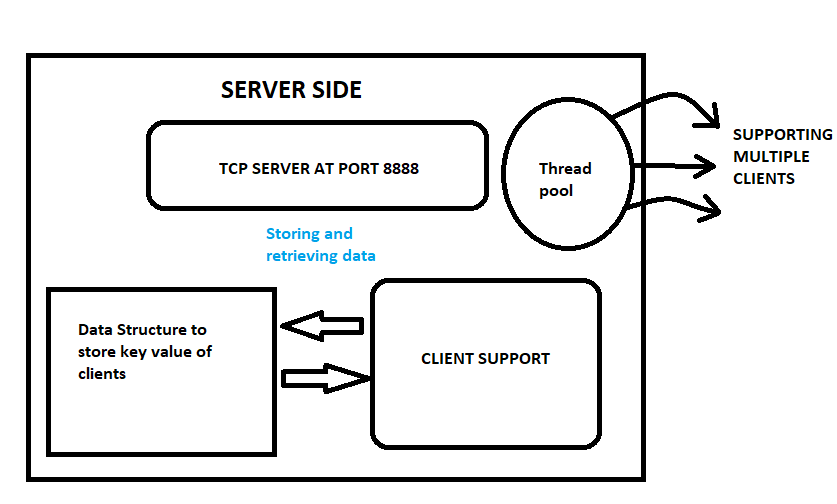
<blank>

The server should be running on a designated port no. The server should support multiple clients and maintain their key-value stores separately. Comment on the port nos used by the server and the clients.

Implement authorization so that only a few clients having the role “manager” can access other’s key-

value stores. A user is assigned the “guest” role by default. The server can upgrade a “guest” user to a “manager” user.

**Architecture Diagram:**

****

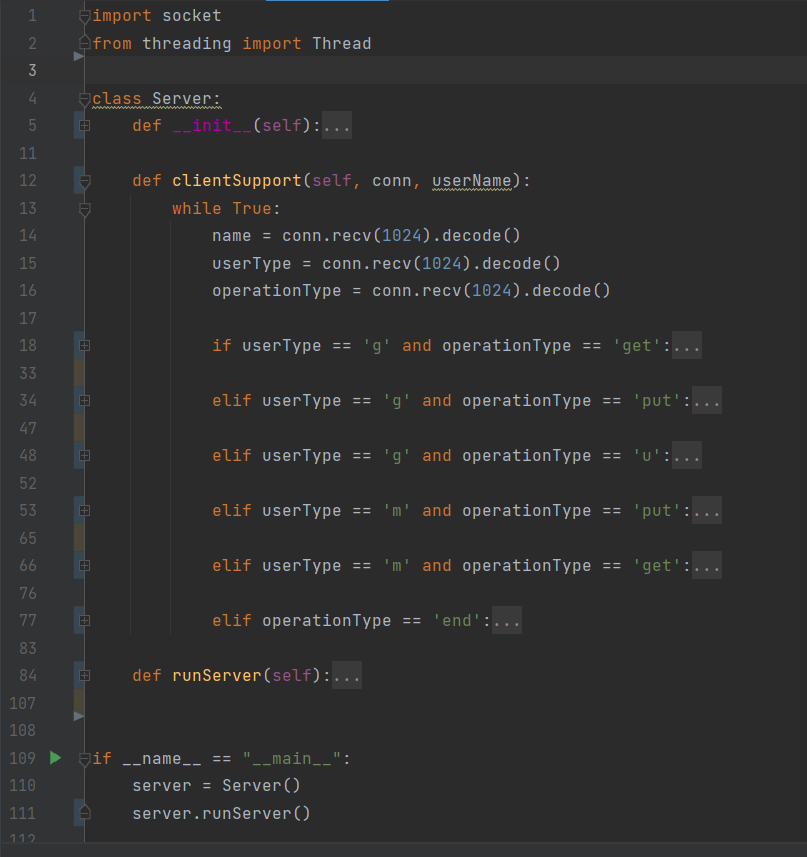
**Architecture Description :**

Server is running via a TCP socket connection on a speciﬁed port(8888) at localhost and has a pool of threads which work by deploying a client support method in thread for every incoming client request. For storing the data, I am maintaining two dictionaries named storeName and keyStore.

storeName will store the username of all the client and KeyStore is a mapping from username to their corresponding key value set. This is actually working as an in memory database.

**Server :**

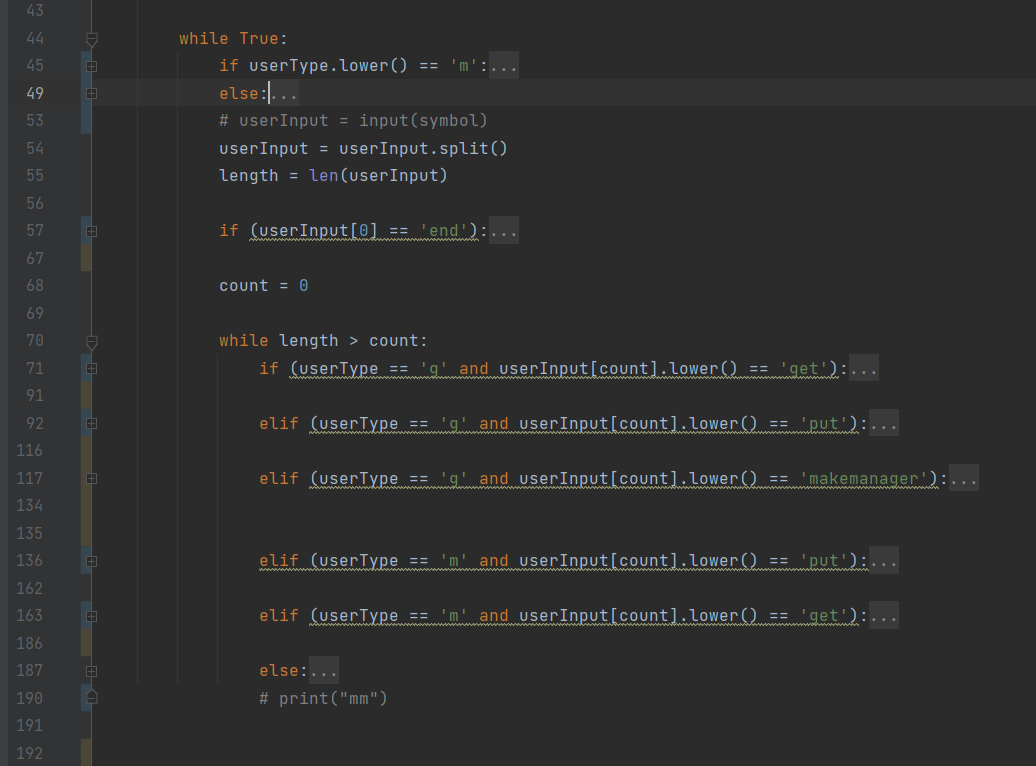
The server consists of two simple functions, namely ClientSupport and runServer . ClientSupport function deployes a new thread every time a new user binds with the server. Thus concurrency is handled.



**Client:**

Whereas the client side consist of only 1 main function namely start which split the input line into tokens and send some query to the server and get responses. The start method checks user input in a while loop and based on the input given by the user it performs some speciﬁc query over server fetch data and display them.





**Queries :**

At first every user need to perform a bind operation that will bind the client to the server over a speciﬁed port.

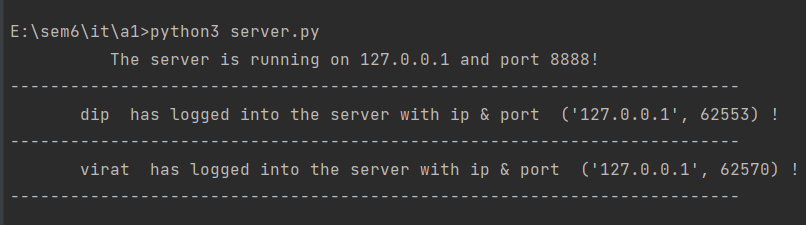
By default every user is a guest user. Guest user can get and put their own data and can perform upgrade command to escalate their privilege to a manager user.

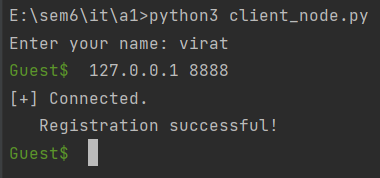
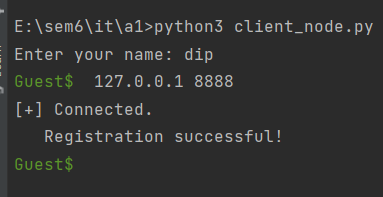
Manager user can access and modify data of any user registered in the in memory database.

**Output Screenshot:**

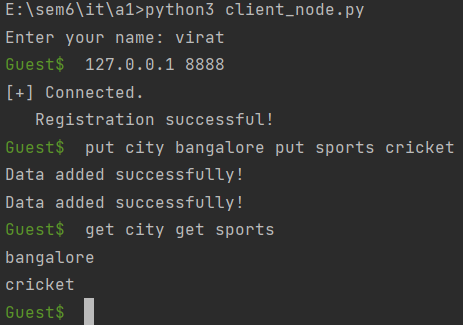
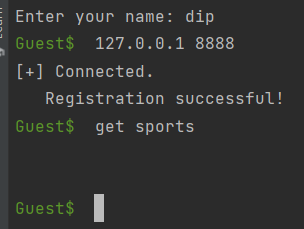
**Establishing connection :**

Clients and server are getting connected. By default every user is a guest (green color)

****

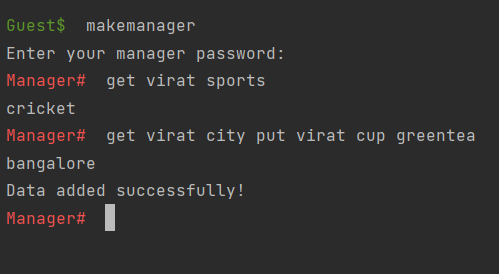
****

## Adding And Retrieving data :

## Manager Privilege :

Now we upgrade dip to privilege for the manager role (red colour). Manager can access and modify anyone's record.



# Conclusion :

After implementing the required features for a Key-Value store via TCP socket based server-client model , the concepts such as SERVER-CLIENT models , multithreading , synchronization , concurrency got cleared well . Also there might be a better approach to store each user name and their corresponding password to handle the case of different client having same name.