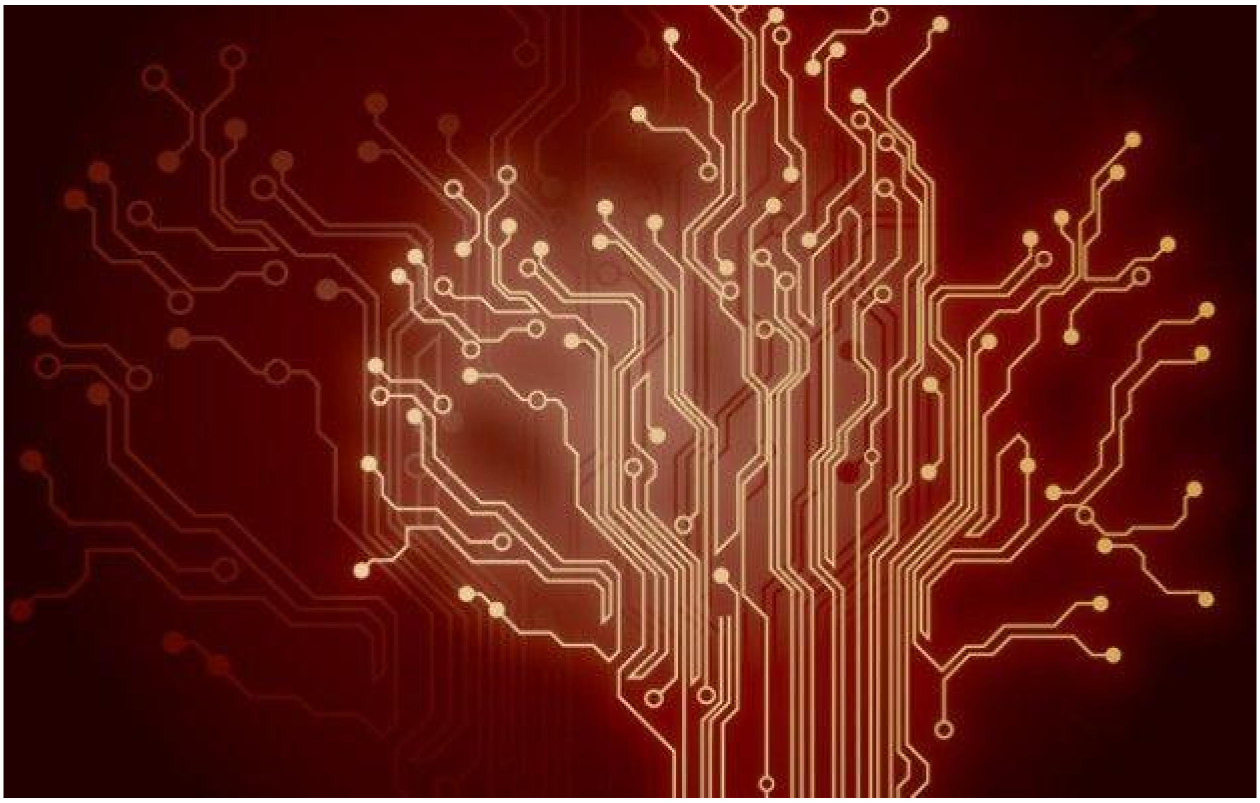
INTERNET TECHNOLOGY LAB REPORT

Assignment 1 : Implement a key-value store using socket programming.



Atanu Ghosh

BCSE-III (2019-2023) 6th sem, Section: A-1,

Roll: **001910501005**, Date: 22/01/2022

ASSIGNMENT-1

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 " and/or "put ". ./client 192.168.124.5 5555 put city Kolkata put country India get country get city get Institute India Kolkata 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 keyvalue stores. A user is assigned the "guest" role by default. The server can upgrade a "guest" user to a "manager" user.

.

DESIGN

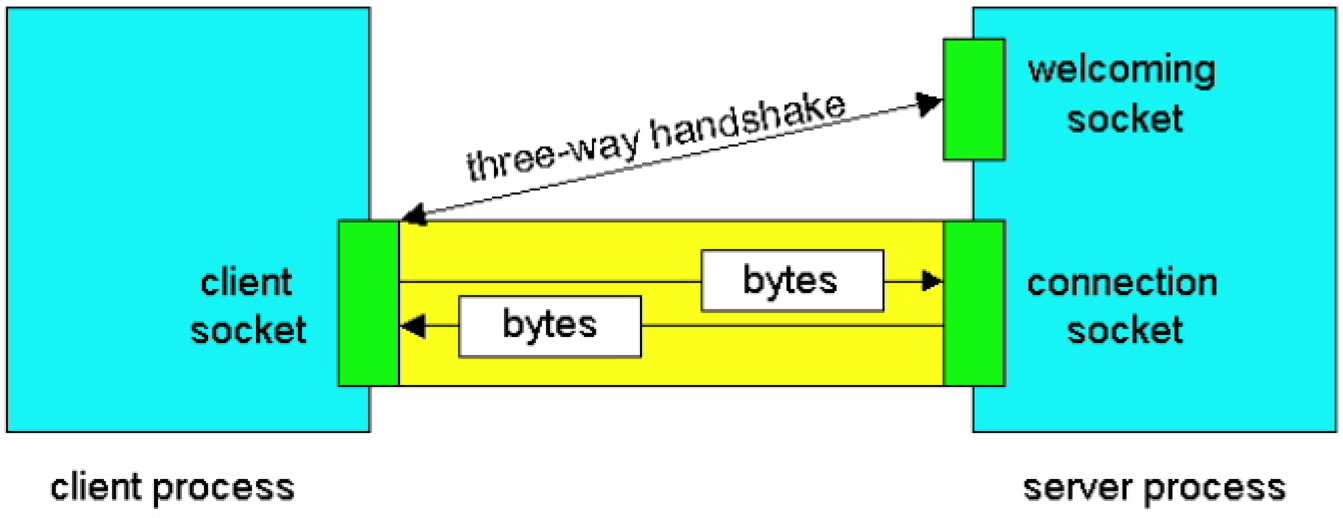


FIG : communicating through TCP sockets

For this assignment the language of choice is **Python**.

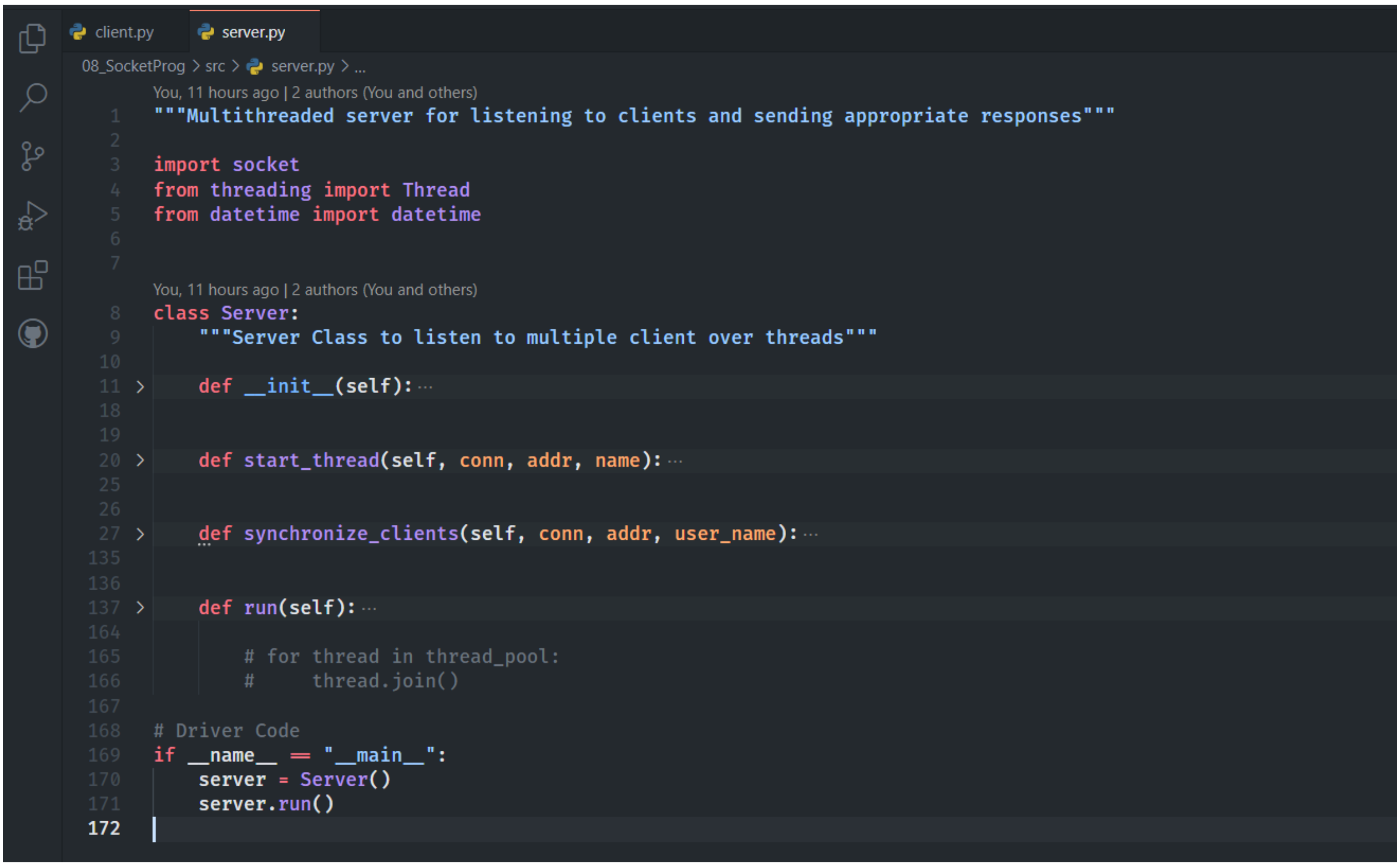
IMPLEMENTATION

Server is running via a TCP socket connection on a specified port (in my case it's 5050) at localhost and has a pool of threads which work by deploying a client handler thread for every incoming client request. For storing the data, I am maintaining two dictionaries named **`auth_dict`** and **`key_store`.**

- **`auth_dict`** keeps the username as key and password as it's value.
- **`key_store`** is basically a dictionary of dictionaries, for each user an inner dictionary is created where the key-value pair is maintained. This is actually working as an in-memory database.

Server

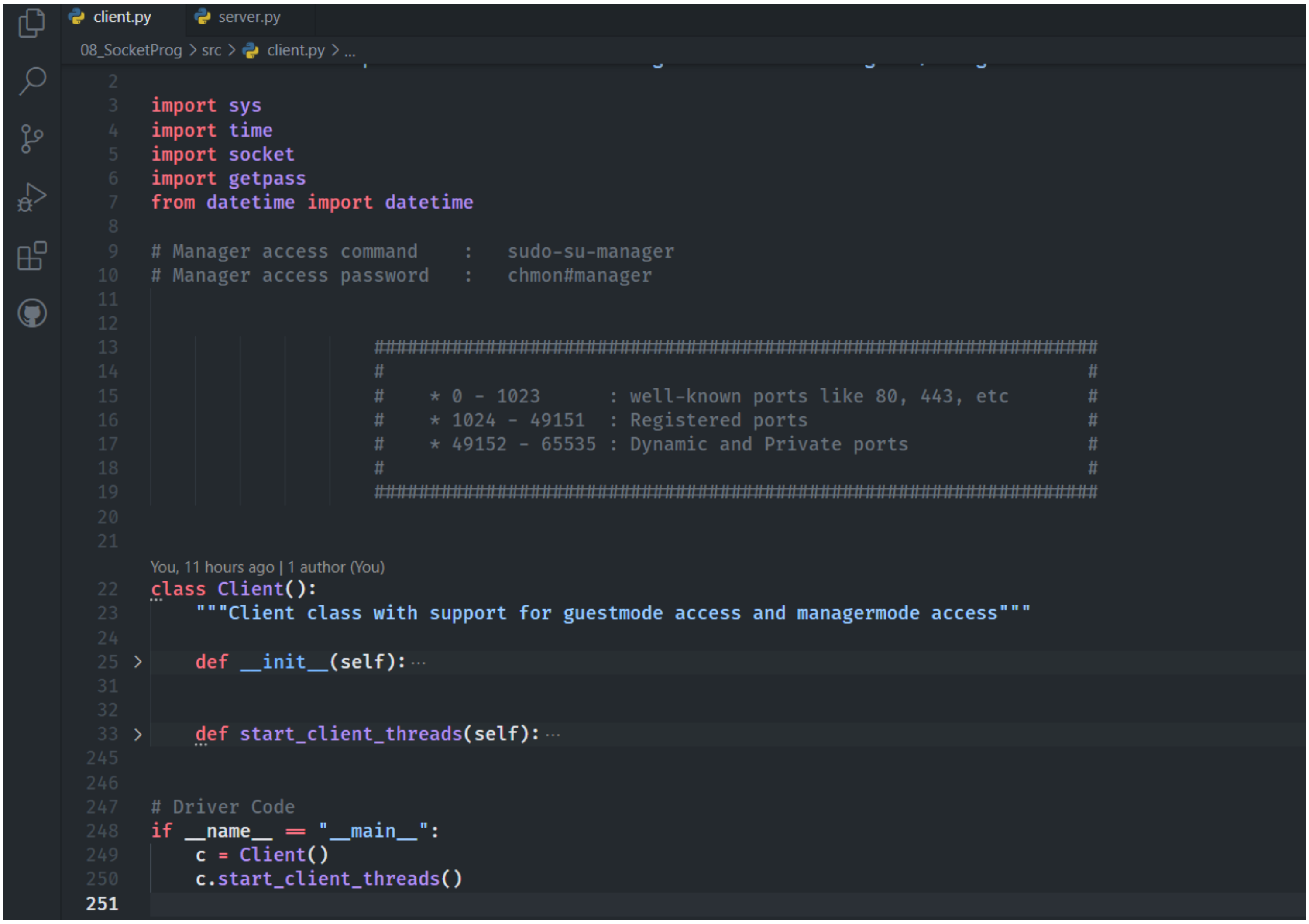
The server consists of 3 functions, namely **`start_thread()`**, **`synchronize_clients()`** and **`run()`**. The **`start_thread()`** initiates a server-thread to whose ip-address each client tries to connect to. The **`synchronize_clients()`** function deploys a new connection to the new thread every time a new user binds to the server and thus maintains concurrency.


client.py
server.py
08_SocketProg > src > server.py > ...
You, 11 hours ago | 2 authors (You and others)

```python
"""Multithreaded server for listening to clients and sending appropriate responses"""

import socket
from threading import Thread
from datetime import datetime


class Server:
    """Server Class to listen to multiple client over threads"""

    def __init__(self): ···


    def start_thread(self, conn, addr, name): ···


    def synchronize_clients(self, conn, addr, user_name): ···


    def run(self): ···

        # for thread in thread_pool:
        #     thread.join()

# Driver Code
if __name__ == "__main__":
    server = Server()
    server.run()
```

Client

The client consists of only one main function, named `start_client_threads()`,
This function parses user input, sanitizes it and performs some queries over the server and fetches data from it.

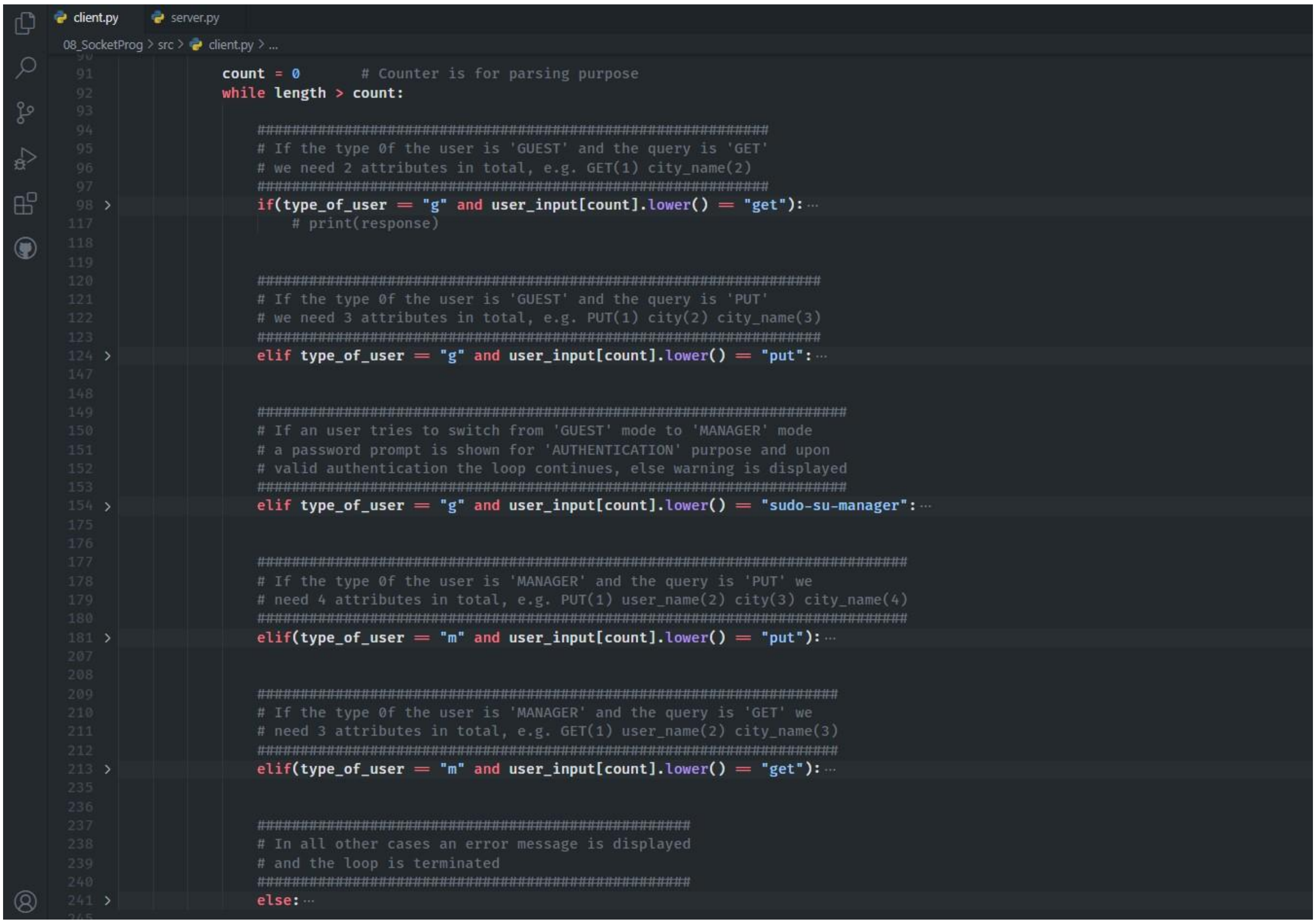


client.py   server.py

08_SocketProg > src > client.py > ...

```python

import sys
import time
import socket
import getpass
from datetime import datetime

# Manager access command    :   sudo-su-manager
# Manager access password   :   chmon#manager


                    ###################################################################
                    #                                                                 #
                    #    * 0 - 1023       : well-known ports like 80, 443, etc        #
                    #    * 1024 - 49151   : Registered ports                          #
                    #    * 49152 - 65535 : Dynamic and Private ports                  #
                    #                                                                 #
                    ###################################################################


You, 11 hours ago | 1 author (You)
class Client():
    """Client class with support for guestmode access and managermode access"""

    def __init__(self): ...


    def start_client_threads(self): ...


# Driver Code
if __name__ == "__main__":
    c = Client()
    c.start_client_threads()

```

The `start_client_threads()` method checks user input in a while loop and based on the input given by the user it performs some specific queries over server fetch data and displays them. The heart of the `start_client_threads()` function is shown below.


```python
count = 0        # Counter is for parsing purpose
while length > count:

    ###########################################################
    # If the type 0f the user is 'GUEST' and the query is 'GET'
    # we need 2 attributes in total, e.g. GET(1) city_name(2)
    ###########################################################
    if(type_of_user == "g" and user_input[count].lower() == "get"): ...
        # print(response)


    #################################################################
    # If the type 0f the user is 'GUEST' and the query is 'PUT'
    # we need 3 attributes in total, e.g. PUT(1) city(2) city_name(3)
    #################################################################
    elif type_of_user == "g" and user_input[count].lower() == "put": ...


    ####################################################################
    # If an user tries to switch from 'GUEST' mode to 'MANAGER' mode
    # a password prompt is shown for 'AUTHENTICATION' purpose and upon
    # valid authentication the loop continues, else warning is displayed
    ####################################################################
    elif type_of_user == "g" and user_input[count].lower() == "sudo-su-manager": ...


    ###########################################################################
    # If the type 0f the user is 'MANAGER' and the query is 'PUT' we
    # need 4 attributes in total, e.g. PUT(1) user_name(2) city(3) city_name(4)
    ###########################################################################
    elif(type_of_user == "m" and user_input[count].lower() == "put"): ...


    ###################################################################
    # If the type 0f the user is 'MANAGER' and the query is 'GET' we
    # need 3 attributes in total, e.g. GET(1) user_name(2) city_name(3)
    ###################################################################
    elif(type_of_user == "m" and user_input[count].lower() == "get"): ...


    ##################################################
    # In all other cases an error message is displayed
    # and the loop is terminated
    ##################################################
    else: ...
```

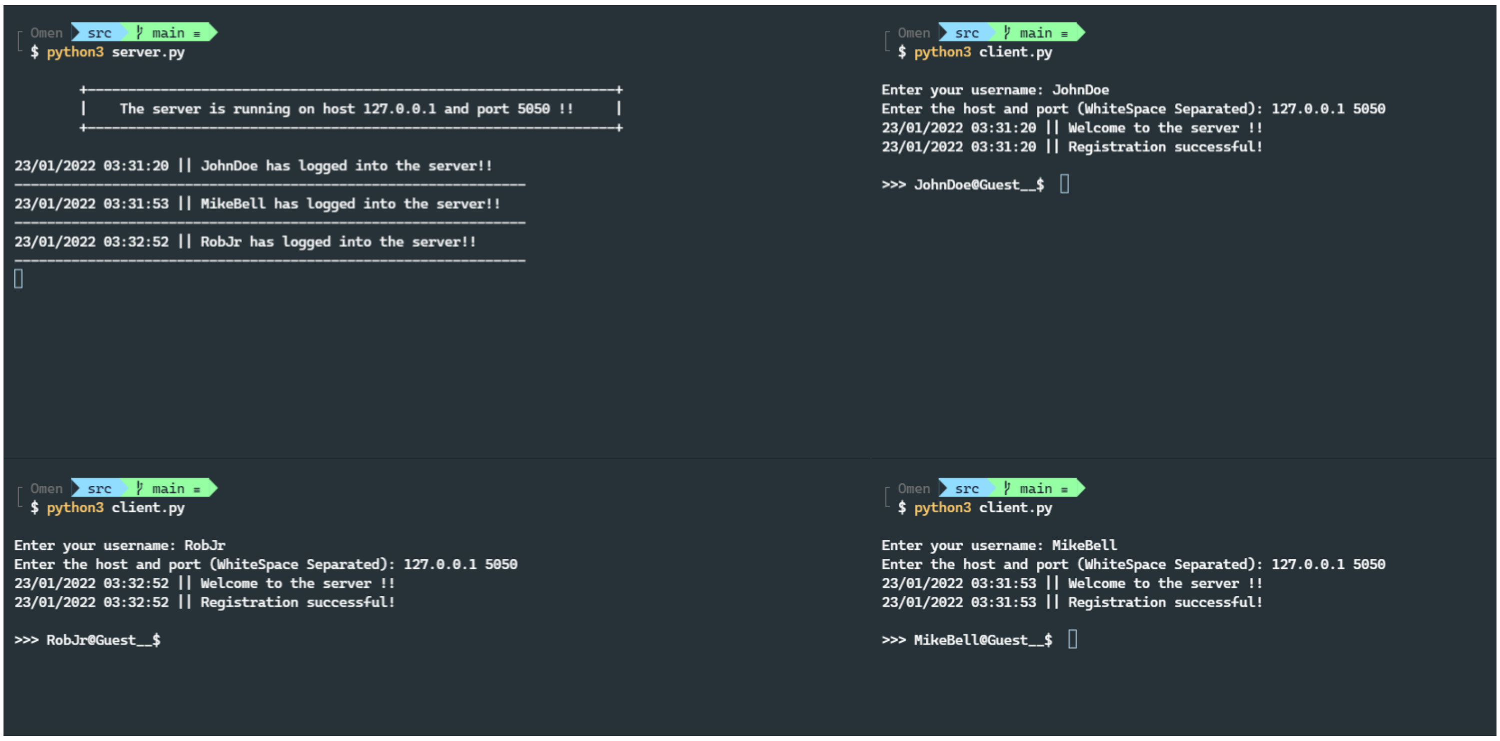
The queries

- First, in a terminal window, start the server by saying `python3 server.py`
- Now, in a different terminal window, start a client by saying `python3 client.py`
- Remember, you can open as many clients as you want, by simply running `python3 client.py` on separate terminal windows.
- You need to enter a username, and then the server host and port no. which is `127.0.0.1` and `5050`.
- Every signup/login is as a **guest** by default.
- Now, you can execute **GET** and **PUT** , as required in any order.
  ```
  GET city
  PUT city Kolkata
  ```
- To switch to `manager` mode, type `sudo-su-manager` and enter the password `chmon#manager`.
- As a manager, you can **GET** and **PUT** by following the command with the username, eg:
  ```
  GET [username] city
  PUT [username] city Kolkata
  ```
- Type `logout` to end the client session.

RESULTS

Initialization and Connection :

Clients and servers are getting connected. By default every user is a guest (`username@Guest__$` prompt).



```
Omen  src  main ≡
$ python3 server.py

       +-------------------------------------------------------------------+
       |    The server is running on host 127.0.0.1 and port 5050 !!      |
       +-------------------------------------------------------------------+

23/01/2022 03:31:20 || JohnDoe has logged into the server!!
----------------------------------------------------------------
23/01/2022 03:31:53 || MikeBell has logged into the server!!
----------------------------------------------------------------
23/01/2022 03:32:52 || RobJr has logged into the server!!
----------------------------------------------------------------
```

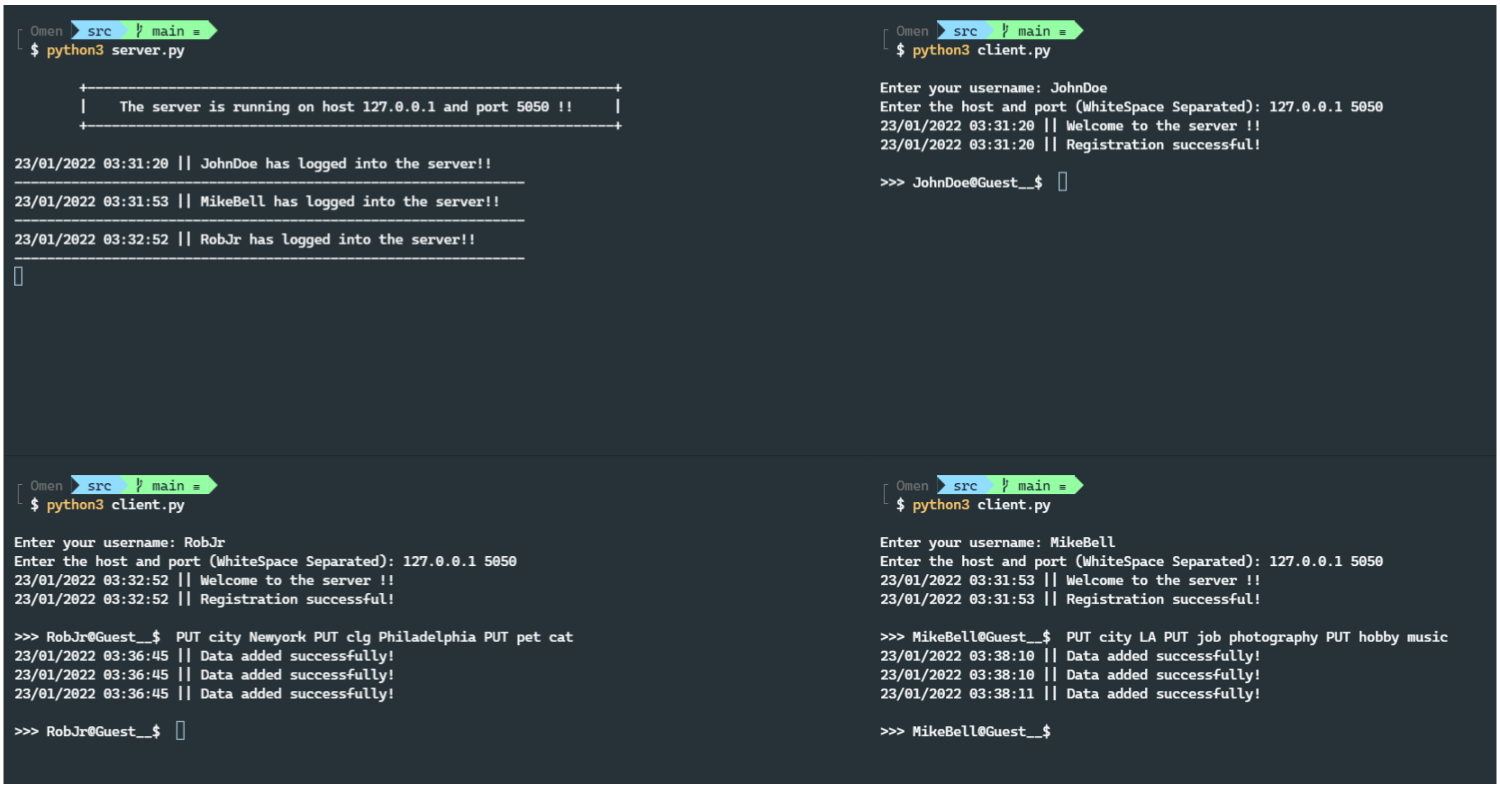
```
Omen  src  main ≡
$ python3 client.py

Enter your username: JohnDoe
Enter the host and port (WhiteSpace Separated): 127.0.0.1 5050
23/01/2022 03:31:20 || Welcome to the server !!
23/01/2022 03:31:20 || Registration successful!

>>> JohnDoe@Guest__$
```

```
Omen  src  main ≡
$ python3 client.py

Enter your username: RobJr
Enter the host and port (WhiteSpace Separated): 127.0.0.1 5050
23/01/2022 03:32:52 || Welcome to the server !!
23/01/2022 03:32:52 || Registration successful!

>>> RobJr@Guest__$
```

```
Omen  src  main ≡
$ python3 client.py

Enter your username: MikeBell
Enter the host and port (WhiteSpace Separated): 127.0.0.1 5050
23/01/2022 03:31:53 || Welcome to the server !!
23/01/2022 03:31:53 || Registration successful!

>>> MikeBell@Guest__$
```

Adding And Retrieving data (access control) :

Now we are adding data from the user terminals. We can easily use multiple commands in a single line



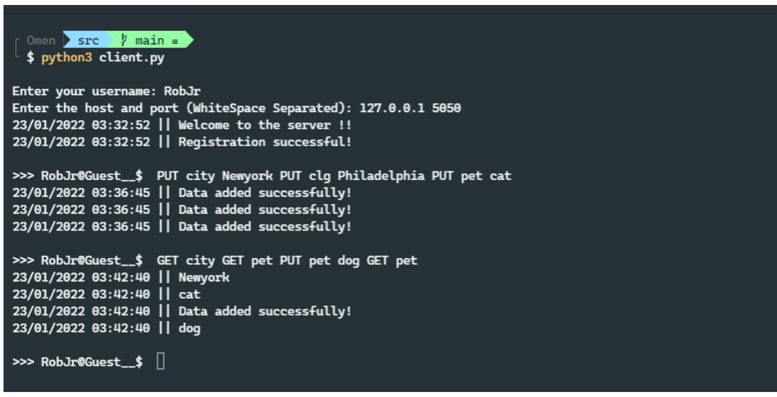
```
Omen  src  main ≡
$ python3 server.py

       +-------------------------------------------------------------------+
       |    The server is running on host 127.0.0.1 and port 5050 !!      |
       +-------------------------------------------------------------------+

23/01/2022 03:31:20 || JohnDoe has logged into the server!!
----------------------------------------------------------------
23/01/2022 03:31:53 || MikeBell has logged into the server!!
----------------------------------------------------------------
23/01/2022 03:32:52 || RobJr has logged into the server!!
----------------------------------------------------------------
```

```
Omen  src  main ≡
$ python3 client.py

Enter your username: JohnDoe
Enter the host and port (WhiteSpace Separated): 127.0.0.1 5050
23/01/2022 03:31:20 || Welcome to the server !!
23/01/2022 03:31:20 || Registration successful!

>>> JohnDoe@Guest__$
```

```
Omen  src  main ≡
$ python3 client.py

Enter your username: RobJr
Enter the host and port (WhiteSpace Separated): 127.0.0.1 5050
23/01/2022 03:32:52 || Welcome to the server !!
23/01/2022 03:32:52 || Registration successful!

>>> RobJr@Guest__$  PUT city Newyork PUT clg Philadelphia PUT pet cat
23/01/2022 03:36:45 || Data added successfully!
23/01/2022 03:36:45 || Data added successfully!
23/01/2022 03:36:45 || Data added successfully!

>>> RobJr@Guest__$
```

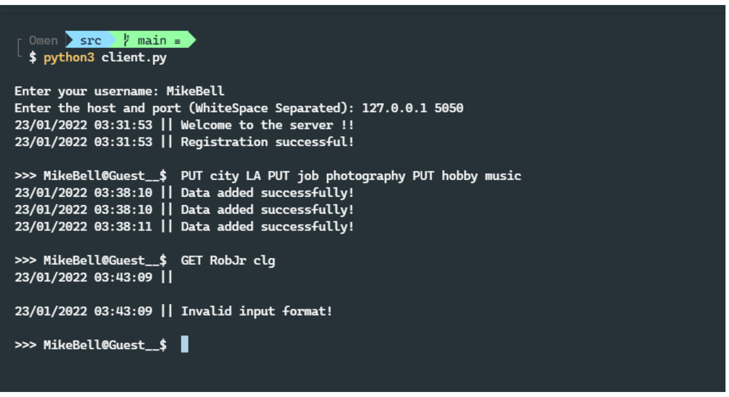
```
Omen  src  main ≡
$ python3 client.py

Enter your username: MikeBell
Enter the host and port (WhiteSpace Separated): 127.0.0.1 5050
23/01/2022 03:31:53 || Welcome to the server !!
23/01/2022 03:31:53 || Registration successful!

>>> MikeBell@Guest__$  PUT city LA PUT job photography PUT hobby music
23/01/2022 03:38:10 || Data added successfully!
23/01/2022 03:38:10 || Data added successfully!
23/01/2022 03:38:11 || Data added successfully!

>>> MikeBell@Guest__$
```

Now we will try to retrieve data. We can see in the next screenshot that one guest can access its own data and modify it, but we cannot get others' data.



```
 Omen  src  main ≡
 $ python3 client.py

Enter your username: RobJr
Enter the host and port (WhiteSpace Separated): 127.0.0.1 5050
23/01/2022 03:32:52 || Welcome to the server !!
23/01/2022 03:32:52 || Registration successful!

>>> RobJr@Guest__$  PUT city Newyork PUT clg Philadelphia PUT pet cat
23/01/2022 03:36:45 || Data added successfully!
23/01/2022 03:36:45 || Data added successfully!
23/01/2022 03:36:45 || Data added successfully!

>>> RobJr@Guest__$  GET city GET pet PUT pet dog GET pet
23/01/2022 03:42:40 || Newyork
23/01/2022 03:42:40 || cat
23/01/2022 03:42:40 || Data added successfully!
23/01/2022 03:42:40 || dog

>>> RobJr@Guest__$
```

Access Granted

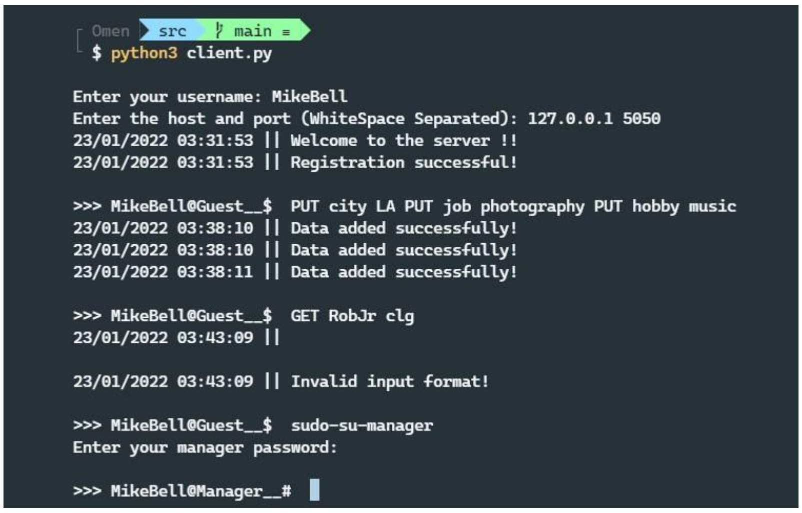


```
 Omen  src  main ≡
 $ python3 client.py

Enter your username: MikeBell
Enter the host and port (WhiteSpace Separated): 127.0.0.1 5050
23/01/2022 03:31:53 || Welcome to the server !!
23/01/2022 03:31:53 || Registration successful!

>>> MikeBell@Guest__$  PUT city LA PUT job photography PUT hobby music
23/01/2022 03:38:10 || Data added successfully!
23/01/2022 03:38:10 || Data added successfully!
23/01/2022 03:38:11 || Data added successfully!

>>> MikeBell@Guest__$  GET RobJr clg
23/01/2022 03:43:09 ||

23/01/2022 03:43:09 || Invalid input format!

>>> MikeBell@Guest__$
```

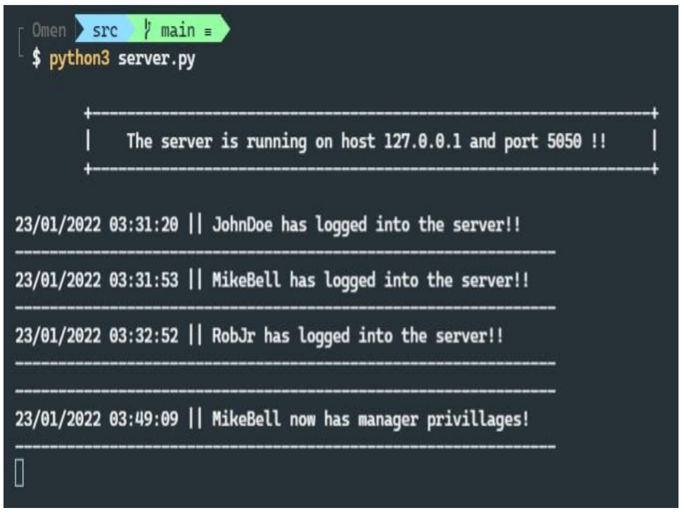
Access Denied

Manager Privilege

Now we can upgrade MikeBell's privilege to the manager role (**`username@Manager__#`** prompt). Manager can access and modify any other user's database.



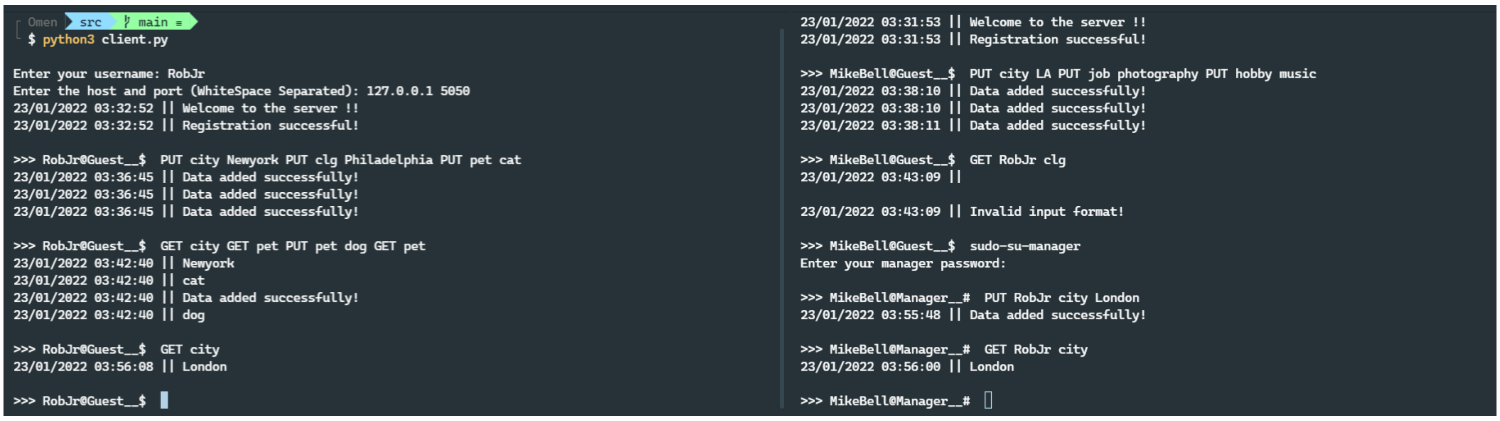
```
 Omen  src  main ≡
 $ python3 client.py

Enter your username: MikeBell
Enter the host and port (WhiteSpace Separated): 127.0.0.1 5050
23/01/2022 03:31:53 || Welcome to the server !!
23/01/2022 03:31:53 || Registration successful!

>>> MikeBell@Guest__$  PUT city LA PUT job photography PUT hobby music
23/01/2022 03:38:10 || Data added successfully!
23/01/2022 03:38:10 || Data added successfully!
23/01/2022 03:38:11 || Data added successfully!

>>> MikeBell@Guest__$  GET RobJr clg
23/01/2022 03:43:09 ||

23/01/2022 03:43:09 || Invalid input format!

>>> MikeBell@Guest__$  sudo-su-manager
Enter your manager password:

>>> MikeBell@Manager__#
```



```
 Omen  src  main ≡
 $ python3 server.py

   +--------------------------------------------------------------------+
   |    The server is running on host 127.0.0.1 and port 5050 !!        |
   +--------------------------------------------------------------------+

23/01/2022 03:31:20 || JohnDoe has logged into the server!!
----------------------------------------------------------------
23/01/2022 03:31:53 || MikeBell has logged into the server!!
----------------------------------------------------------------
23/01/2022 03:32:52 || RobJr has logged into the server!!
----------------------------------------------------------------
----------------------------------------------------------------
23/01/2022 03:49:09 || MikeBell now has manager privillages!
----------------------------------------------------------------
```

Manager Access

Now as we can see, MikeBell can access anyone's data easily. The upgrade of access also works as RobJr's city information is now changed to London. (it was Newyork before).



```
Omen  src  main ≡
$ python3 client.py

Enter your username: RobJr
Enter the host and port (WhiteSpace Separated): 127.0.0.1 5050
23/01/2022 03:32:52 || Welcome to the server !!
23/01/2022 03:32:52 || Registration successful!

>>> RobJr@Guest__$  PUT city Newyork PUT clg Philadelphia PUT pet cat
23/01/2022 03:36:45 || Data added successfully!
23/01/2022 03:36:45 || Data added successfully!
23/01/2022 03:36:45 || Data added successfully!

>>> RobJr@Guest__$  GET city GET pet PUT pet dog GET pet
23/01/2022 03:42:40 || Newyork
23/01/2022 03:42:40 || cat
23/01/2022 03:42:40 || Data added successfully!
23/01/2022 03:42:40 || dog

>>> RobJr@Guest__$  GET city
23/01/2022 03:56:08 || London

>>> RobJr@Guest__$
```

```
23/01/2022 03:31:53 || Welcome to the server !!
23/01/2022 03:31:53 || Registration successful!

>>> MikeBell@Guest__$  PUT city LA PUT job photography PUT hobby music
23/01/2022 03:38:10 || Data added successfully!
23/01/2022 03:38:10 || Data added successfully!
23/01/2022 03:38:11 || Data added successfully!

>>> MikeBell@Guest__$  GET RobJr clg
23/01/2022 03:43:09 ||

23/01/2022 03:43:09 || Invalid input format!

>>> MikeBell@Guest__$  sudo-su-manager
Enter your manager password:

>>> MikeBell@Manager__#  PUT RobJr city London
23/01/2022 03:55:48 || Data added successfully!

>>> MikeBell@Manager__#  GET RobJr city
23/01/2022 03:56:00 || London

>>> MikeBell@Manager__#
```

COMMENTS

After implementing the required features for a K-V store via server-client model , the concepts such as SERVER-CLIENT models , multithreading , synchronization and concurrency got cleared well. A better query processing mechanism could be implemented via considering all the error factors and informing the user in the case of multiple queries in a line as stated in the assignment.