INTERNET TECHNOLOGIES ASSIGNMENT REPORT

Name: ABHISHEK DE
Roll no: 001710501027
Group: A1
Assignment: 1

PROBLEM STATEMENT

Implement a TCP-based key-value store. 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 TCP port. The server should support multiple clients and maintain their key-value stores separately.
Implement authorization so that only 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.

SOLUTION APPROACH

The solution consists of 4 classes as follows:
Data:
Consists of key value store and methods to input/extract data.
TCPClient:
Client program, which connects to the server via TCP.
TCPServer:
Server program, which listens to the requests from client and creates a new thread for each client.
ClientThread:
A thread(part of server program) which is responsible for communicating with a single client.

The TCP connection is estabished using the socket library in Python. To simulate different IP addresses within the same device, the IP address is provided as a command line argument. This address is communicated to the server via a message. The key-value pairs are stored in a global dictionary, which also stores the owner(ip) of the key-value pair. Hence the sample data structure is as follows.

```
keyValueMap = {
      ‘city’: {
            ‘value’ : ‘kolkata’,
            ‘owner’: ‘192.34.56.76’

      }
      ‘country’: {
            ‘value’ : ‘india’,
            ‘owner’: ‘192.34.56.91’
      }
}
```

On establishing a connection the server program asks for the designation of the client(guest or manager).
On receiving the key-value pair, the server program, checks for the folllowing:

- If the command is **quit**, the connection to that client is closed, and a message is displayed on server side.
- If the command is of the form **put <key> <value>**
  - If key already exists send back a message **“Key already exists”**
  - If not store in key value pair and send **“Key value pair stored successfully”**
- If the command is of the form **get <key>**
  - If key already exists send **“Invalid key”**
  - If designation is manager, return the value.
  - If designation is guest, check the key for the owner.
    - If client is the owner, return the value
    - Else, send **“Unauthorised access”**
- If the command is of any other form, send **“Invalid request”**

SALIENT FEATURES UNIQUE TO MY SOLUTION

- Multiple clients are handled using threads.
- There is a global data store as compared to personal data stores for each client.
- To prevent unauthorised access the owner of a key-value pair is stored against the key.
- Same key cannot be overwritten with another value, ie the values are immutable.
- Whenever a client makes a connection with the server, server program asks to enter the designation as manager or guest.

SAMPLE INPUT/OUTPUT

Client 1 makes a request

```
$ python tcpClient.py 172.34.56.67
Starting client with IP address: 172.34.56.67
```

Server asks for designation -> Designation set to manager

```
New TCP client with IP 172.34.56.67 joined
Set designation: (1 for manager, 2 for guest) 1
```

Client 2 makes a request

```
$ python tcpClient.py 172.34.54.82
Starting client with IP address: 172.34.54.82
```

Server asks for designation -> Designation set to guest

```
New TCP client with IP 172.34.54.82 joined
Set designation: (1 for manager, 2 for guest) 2
```

Client 1 asks and retrieves a key-value pair

```
? put city kolkata
Key-value pair entered successfully
? get city
kolkata
```

Client 1 asks for invalid get request

```
? get school
Invalid key!
```

Client 2 enters a key-value pair

```
? put country india
Key-value pair entered successfully
```

Client 1 can access it since it is a manager

```
? get country
india
```

Client 2 cannot access Client 1's key-value pair

```
? get city
Unauthorised client
```

Client 2 enters a key which is already in use

```
? put city patna
Key already in use!
```

Client 2 makes an invalid request

```
? put college
Invalid request
```