Freshworks Assignment – Dinesh Balaji – CB.EN.U4CSE17113

Problem Statement:

Freshworks – Backend Assignment

Build a file-based key-value data store that supports the basic CRD (create, read, and delete) operations. This data store is meant to be used as a local storage for one single process on one laptop. The data store must be exposed as a library to clients that can instantiate a class and work with the data store.

The data store will support the following **functional requirements**.

1. It can be initialized using an optional file path. If one is not provided, it will reliably create itself in a reasonable location on the laptop.
2. A new key-value pair can be added to the data store using the Create operation. The key is always a string - capped at 32chars. The value is always a JSON object - capped at 16KB.
3. If Create is invoked for an existing key, an appropriate error must be returned.
4. A Read operation on a key can be performed by providing the key, and receiving the value in response, as a JSON object.
5. A Delete operation can be performed by providing the key.
6. Every key supports setting a Time-To-Live property when it is created. This property is optional. If provided, it will be evaluated as an integer defining the number of seconds the key must be retained in the data store. Once the Time-To-Live for a key has expired, the key will no longer be available for Read or Delete operations.
7. Appropriate error responses must always be returned to a client if it uses the data store in unexpected ways or breaches any limits.

The data store will also support the following **non-functional requirements**.

1. The size of the file storing data must never exceed 1GB.
2. More than one client process cannot be allowed to use the same file as a data store at any given time.
3. A client process is allowed to access the data store using multiple threads, if it desires to. The data store must therefore be thread-safe.
4. The client will bear as little memory costs as possible to use this data store, while deriving maximum performance with respect to response times for accessing the data store.

SOLUTION :

Code Explanation :

```
def manual():
    print("Hello User.")
    print()
    print("To access the library, we have a set of rules to be followed")
    print("1. For creating a record use, db.create_record(key,value,ttlvalue). "
    print("2. For reading/accessing a record use, db.read_record(key) .")
    print("3. For deleting a record use, db.delete_record(key) .")
    print()
    print("For creating a record, specifying ttlvalue is optinal. If you do not
    print("While creating a record, check if the key already exists by checking
    print("While creating a record, note that the key should not exceed 32 bits,
    print()
    print("Now you are good to go.")
```

➔ A manual to understand how to operate the library, the keywords to be used to access the functionality and the key points to note before interacting with the database.

```
def read_record(key): #Specify the key here, so that library will search for
    if(key in library):
        content = library[key]
        timetolive = content[1]
        if(timetolive != 0):
            if(time.time() < timetolive):
                jsonformat = str(key)+":"+str(content)
                return jsonformat
            else:
                print("ERROR : Time to live value has ended for this key.")
        else:
            jsonformat = str(key)+":"+str(content)
            return jsonformat

    else:
        print("ERROR : Key not present in library. Try a valid name.")
```

➔ Read record [read_record(key)] takes in the key as a parameter and displays the record contents in the json format as required. It checks if the timetolive value has exceeded or not and then prints the json formatted contents accordingly.

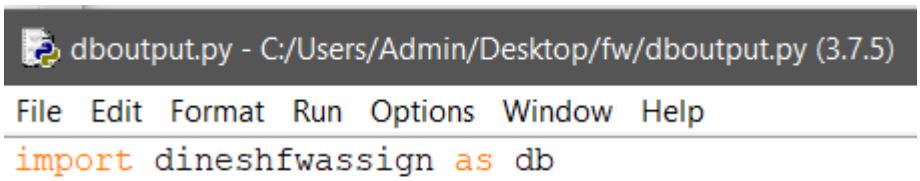
```
def create_record(key,value,ttlvalue): #Specify the key, value and ttlvalue for
    if(key in library):
        print("ERROR : Key already exists in library.")
    else:
        if(key.isalpha()):
            if(len(library) < 1024**3):
                if(value <= 16*(1024**2)):
                    if(ttlvalue==0):
                        record = [value,ttlvalue]
                    else:
                        record = [value,time.time()+ttlvalue]
                    if(len(key) <= 32):
                        library[key] = record
                        print("Record successfully inserted")
                        print(key,":",value)
                else:
                    print("ERROR : Content size should not be more than 16KB")
            else:
                print("ERROR : Library is currently full - 1GB reached.")
        else:
            print("ERROR : Key should only contain alphabets.")
```

➔ Create_record takes in key, its value, and its ttlvalue as its parameters and creates the record accordingly. Checks if the library is full at first, if not, checks if the value is within 16KB of memory, if yes, checks if the ttlvalue is 0, if not, adds the respective ttlvalue to the current time in seconds. It also checks if the key's size is within 32 characters. If all the conditions are satisfied, then the respective record will be created and added to the library. If the conditions are not satisfied properly, appropriate error messages will be displayed to the user.

```
delete_record(key): #Specify the key and the respective record will be d
if(key in library):
    content = library[key]
    ttlvalue = content[1]
    if(ttlvalue != 0):
        if(time.time() < ttlvalue):
            del library[key]
            print("MESSAGE : Key deleted.")
        else:
            print("ERROR : Time to live value has ended for this key.")
    else:
        del library[key]
        print("MESSAGE : Key deleted.")
else:
    print("ERROR : Enter a valid key.")
```

➔ Delete_record(key) takes in key as a parameter and checks if the key is present in the library or not, if present, deletes that respective record from the database, else, displays the appropriate error to the user.

Outputs with unit testing :



Created a new file to execute the CRD Database code. So, imported dineshfwassign file as db and the operations will be executed with db object.

Screenshots :

```
>>> db.manual()
Hello User.

To access the library, we have a set of rules to be followed
1. For creating a record use, db.create_record(key,value,ttlvalue).
2. For reading/accessing a record use, db.read_record(key) .
3. For deleting a record use, db.delete_record(key) .

For creating a record, specifying ttlvalue is optinal. If you do not need it, you can enter 0 in ttlvalue field.
While creating a record, check if the key already exists by checking the library, using the read_record(key) functionality.
While creating a record, note that the key should not exceed 32 bits, and the value should not exceed 16KB.

Now you are good to go.
```

➔ Executing the manual…

```
>>> db.create_record("Dinesh",100,0)
Record successfully inserted
Dinesh : 100
>>>
```

```
>>> db.create_record("PSBBKKN",1965,0)
Record successfully inserted
PSBBKKN : 1965
```

```
>>> db.create_record("2020cyclone",1965,0)
ERROR : Key should only contain alphabets.
>>> db.create_record("6464684888",1965,0)
ERROR : Key should only contain alphabets.
>>> db.create_record("@#$^*&^&**)((!@@#",1965,0)
ERROR : Key should only contain alphabets.
>>>
```

➔ The key should contain only strings in it and should not even be alphanumeric (as stated in the question). So the key should only contain alphabets.

```
>>> import datetime
>>> datetime.datetime.now()
datetime.datetime(2020, 12, 2, 1, 27, 51, 890204)
>>> db.create_record("endsintensecs",100,10)
Record successfully inserted
endsintensecs : 100
>>> db.read_record("endsintensecs")
'endsintensecs:[100, 1606852700.6471035]'
>>> datetime.datetime.now()
datetime.datetime(2020, 12, 2, 1, 28, 37, 704063)
>>> db.read_record("endsintensecs")
ERROR : Time to live value has ended for this key.
```

➔ As we can see, the record within its ttl value could display the contents inside the record, but while reading the contents of the record after 46 seconds, Started at 1:27:51, read the contents latest at 1:28:37, when it said the time to live value has ended for that particular key.

```
>>> db.read_record("PSBBKKN")
'PSBBKKN:[1965, 0]'
>>> db.delete_record("PSBBKKN")
MESSAGE : Key deleted.
>>> db.read_record("PSBBKKN")
ERROR : Key not present in library. Try a valid name.
>>>
```

➔ Here, the record containing PSBBKKN as its key is being deleted from the database and being checked again, and we can see that it displays an error stating the key is not present in the library.

The above unit testings have been done on the database and the CRD – create, read and delete operations have been clearly implemented and performed.

Thanking you,

Dinesh Balaji

CB.EN.U4CSE17113

Amrita Vishwa Vidyapeetham