The key value pair is represented by an arbitrary string such as a filename, URI or hash. In general, key-value stores have no query language. They provide a way to create, read, delete data using simple functions. The simplicity of this model makes a key-value store fast, easy to use, scalable, portable and flexible

The key value handle size well and are good at processing a constant stream of read/write operations with low latency making them perfect for Cache Replacement, Profile data, etc

The approach for this problem was to use a dictionary which will help to store the key value pair, values in a dictionary can be of any datatype and can be duplicated, whereas keys can't be repeated and must be immutable and also the keys in a dictionary doesn't allows Polymorphism and then finally a data file which will store the data in json format as it is easier to access the data and uses the atabase.json as the data source

I have used three functions: create, read, delete to create, read, delete a value corresponding to the key. Before every function I verified the key whether it is less than 32 chars and later checked the file size and value to be less than 1 GB and 16 kb respectively.

And then in all the three functions I have used the dictionary which is a hashfile in this case to compare the time to live value with the current time. ie if we create a key and provide a timeout of 60 sec and during that 60 sec if we invoke a read function corresponding to that key then it will display but after 60 sec it will through a message of time to live exceeded. I also check the memory limit of the key value pair

I have demonstrated the multi threading and thread safe
Thread safe where multiple user can use the shared data through multiple threads and in this by using os.getpid() function in leapth.py I have demonstrated that the data storage supports multi threading and thread safe and listed the different error messages in the leapexception.py