Basic Redis commands implemented in the project-

SET - It sets a key-value pair, value must be a string. If a key already has a value, its value is overwritten.

GET - It is used to get the value of key. If a key doesn't exist, nil is returned. GET accepts only string values.

EXPIRE - It sets a timeout on key. The key-value pair will be automatically discarded once the timeout has expired.

ZADD - It adds a member to the sorted set with the assigned score. We can add multiple score-member pairs at a time.

ZRANGE - It returns the elements of specified range (score-member pairs). The returned elements are ordered from the lowest to the highest score.

ZRANK - It returns the rank of member in the sorted set, with the scores ordered from low to high. The rank is 0-based (the lowest score has rank 0).

1. Why did you choose that language ?

Ans- I have used Node.js in the project. Node.js is single-threaded but with the help of the libuv library, it uses multiple threads in the background to execute asynchronous code.

2. What are the further improvements that can be done to make it efficient?

Ans- To increase efficiency Following improvements can be done :

- In get and set api’s I can also include json data type instead of string which consist of information in json format.
- In ZRANGE it can automatically decides the lower and upper bound of values based on the scores it stores.

3. What data structures have you used and why ?

Ans- String is used for SET, GET & EXPIRE commands, because in a key-value pair, value should be a string. SET, GET and EXPIRE accept only string values.

Sorted set is used for ZADD, ZRANGE & ZRANK commands. These commands operate only on sorted set.

4. Does your implementation support multi threaded operations? If No why can’t it be? If yes then how ?

Ans- Node.js is single-threaded but with the help of the libuv library, it uses multiple threads in the background to execute asynchronous code. Libuv handles something known as "thread pool". Each of those threads in this pool get a particular task assigned and they work on it concurrently.