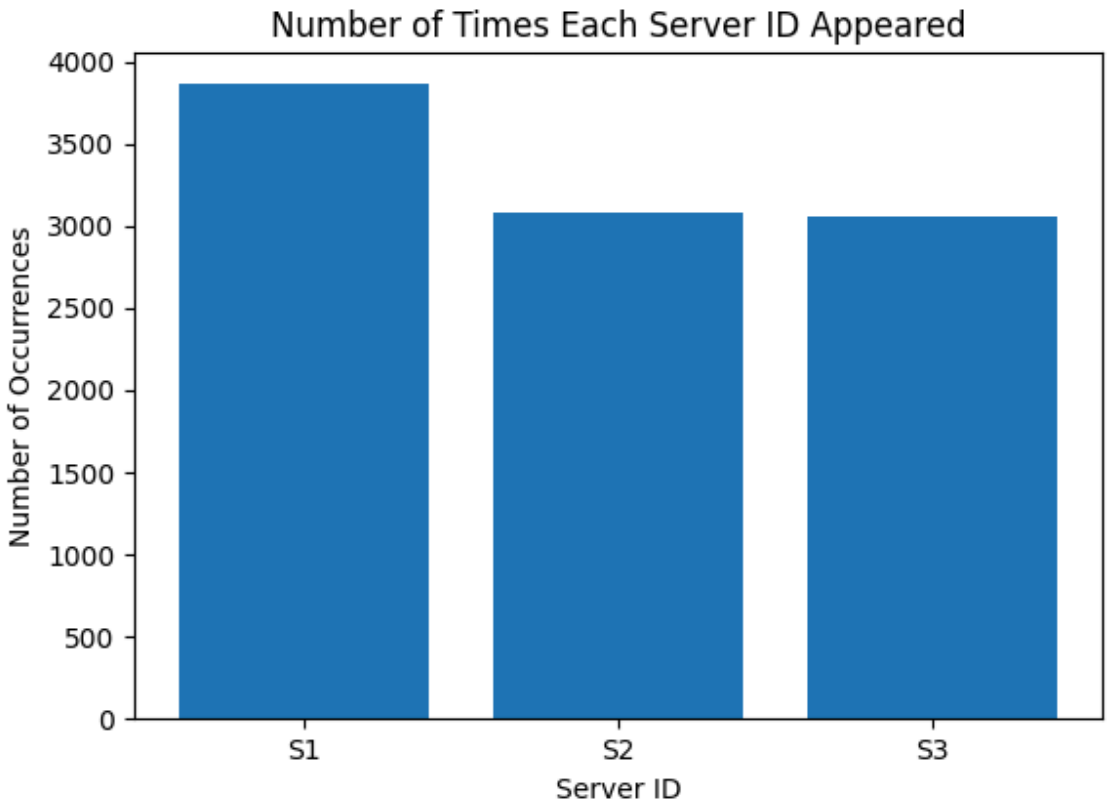
Run “make build” to start the loadbalancer.
Run “make add” to add the first 3 servers.

Run “make clean “ to stop all the containers and remove all the images

Experiment : A-1

Launch 10000 async requests on N = 3 server containers and report the request count handled by each server instance in a bar chart. Explain your observations in the graph and your view on the performance.

Upon sending 10,000 async requests to 3 servers : S1,S2,S3 the request count handled by each server is described the bar graph below:



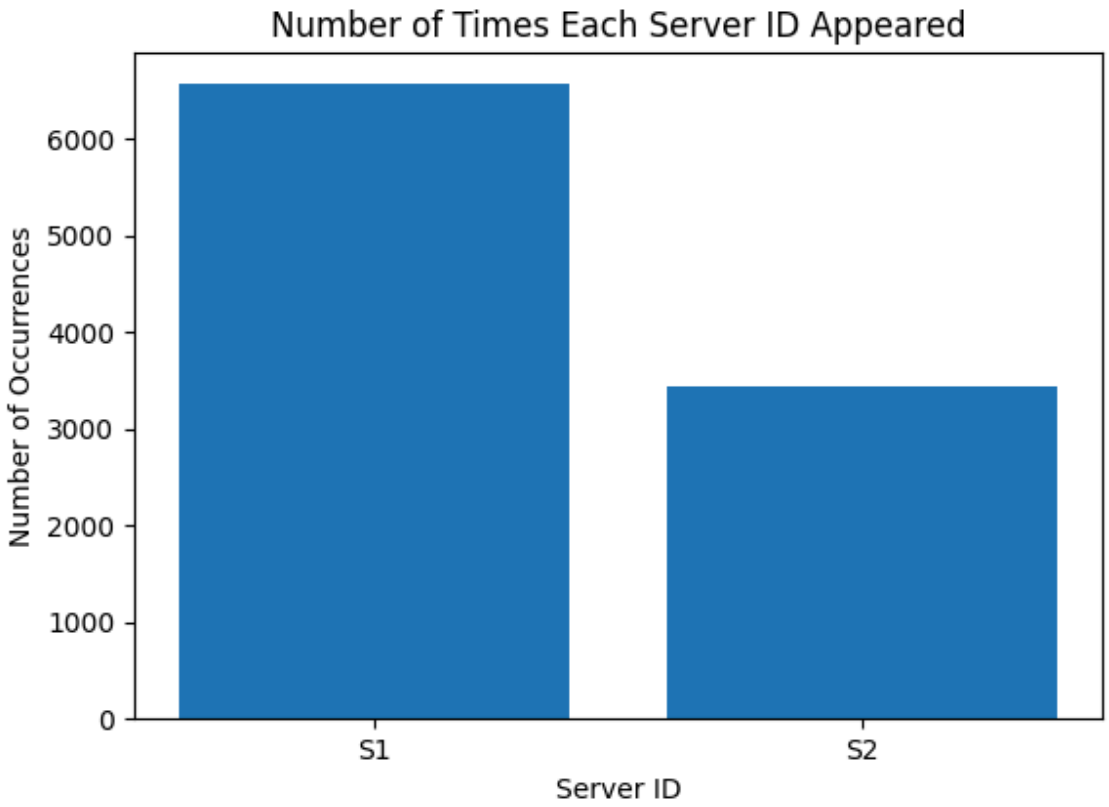
- Server S1 has served almost 3800 requests.
- Server S2 has served around 3100 requests.
- Server S3 has also served around 3100 requests.
- The load is well distributed among the 3 servers.

Experiment : A-2

Next, increment N from 2 to 6 and launch 10000 requests on each such increment. Report the average load of the servers at each run in a line chart. Explain your observations in the graph and your view on the scalability of the load balancer implementation.

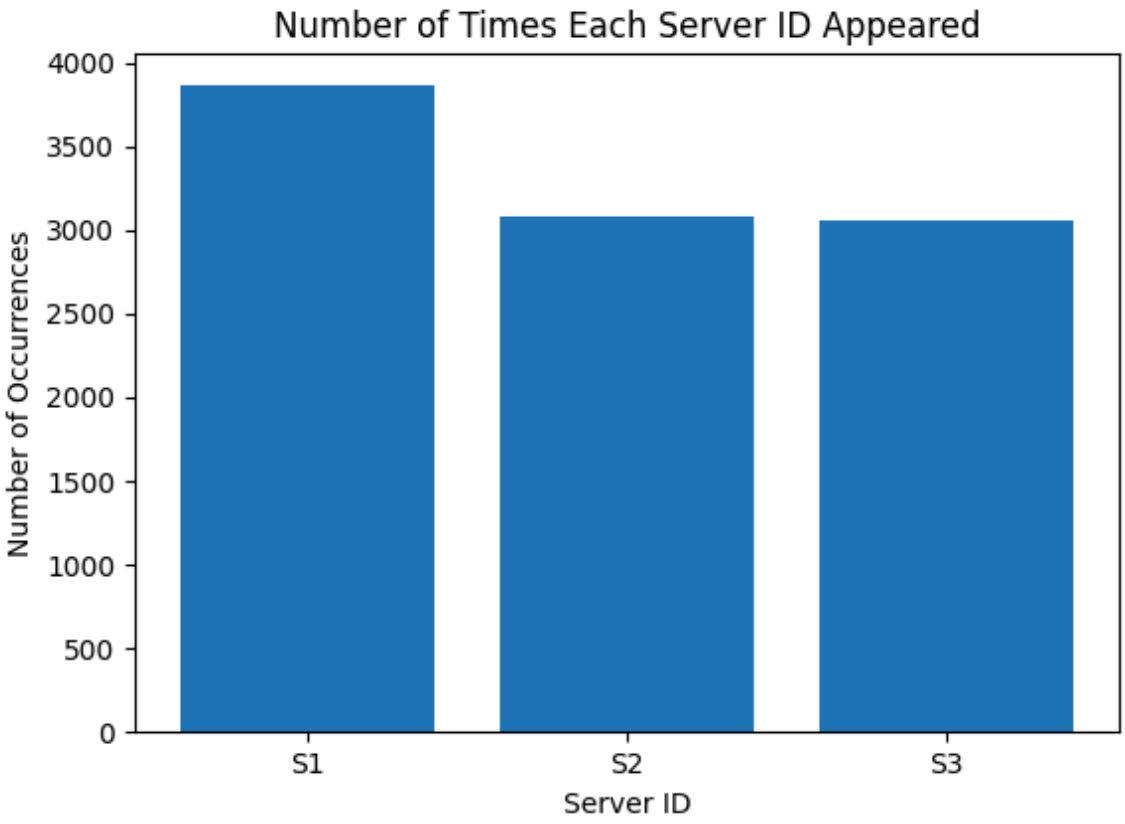
If N = 2:

- Server S1 has served around 6600 requests
- Server S2 has served around 3400 requests
- The load on server S1 > S2 .



If N = 3:

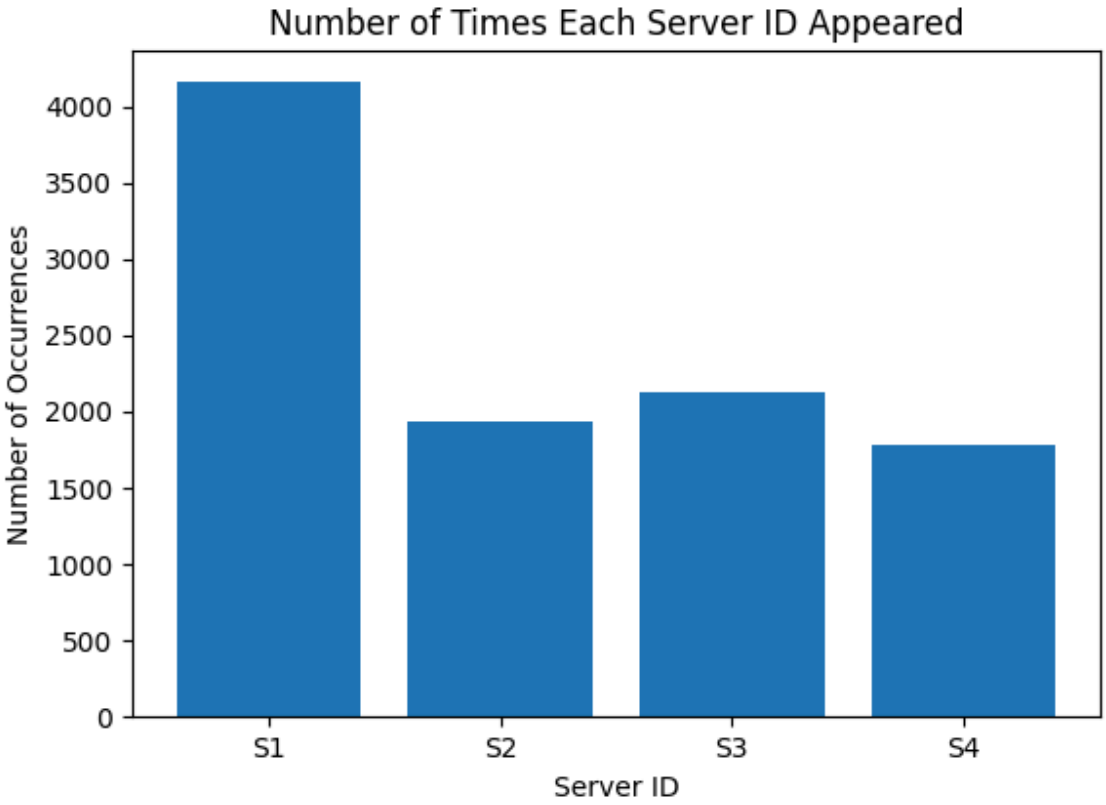
- Server S1 has served almost 3800 requests.
- Server S2 has served around 3100 requests.
- Server S3 has also served around 3100 requests.
- The load is well distributed among the 3 servers.



If N = 4:

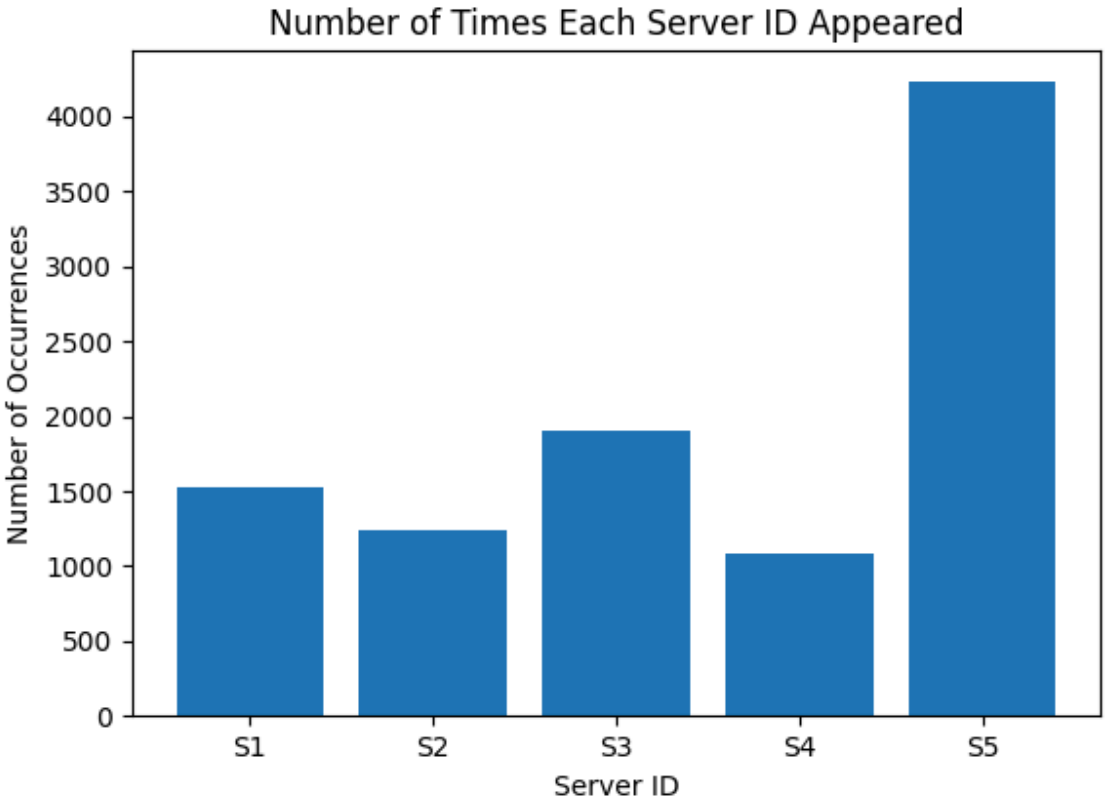
- Server S1 has served almost 4200 requests.
- Server S2 has served almost 1900 requests.
- Server S3 has served almost 2100 requests.
- Server S4 has served almost 1800 requests.
- The load on Server S1 is high

- The load on Server S2,S3,S4 is almost same and comparatively lower than S1.



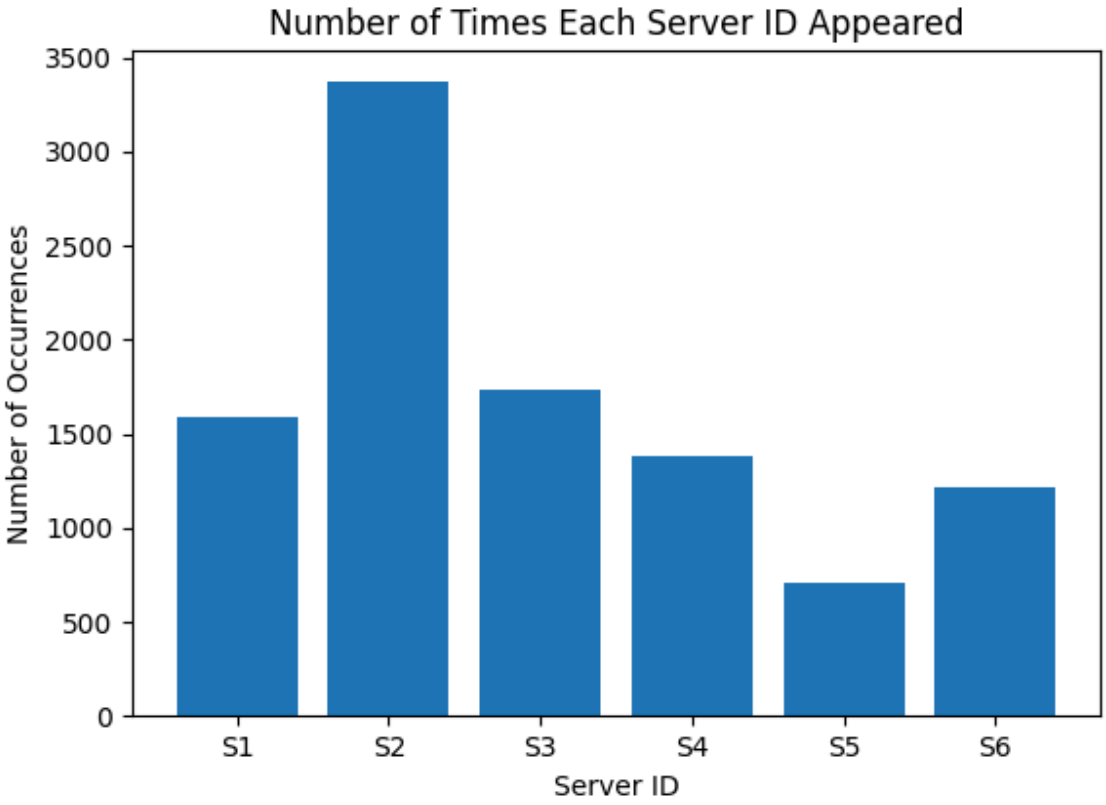
If N = 5:

- Server S1 has served almost 1500 requests
- Server S2 has served almost 1300 requests
- Server S3 has served almost 1900 requests
- Server S4 has served almost 1000 requests
- Server S5 has served almost 4300 requests
- The load on S5 is very high.
- The load on S4 is less.
- The load on S1,S2,S3 is almost same.



If N = 6:

- Server S1 has served almost 1600 requests.
- Server S2 has served almost 3400 requests.
- Server S3 has served almost 1600 requests
- Server S4 has served almost 1400 requests.
- Server S5 has served almost 700 requests.
- Server S6 has served almost 1300 requests.
- The load on server S2 is very high.
- The load on server S5 is low.
- The load on servers S1,S3,S4,S6 is almost the same.



Experiment A- 3:

Test all endpoints of the load balancer and show that in case of server failure, the load balancer spawns a new instance quickly to handle the load.

1) **End-point(/add):**
   We can add new servers from this endpoint. This is a post method. The payload should include the no.of hostnames and the names of the hostnames.

```
ranivenigalla@rani:~/DS_Assgn/DS_ASSIGNMENT_1$ curl -X POST -H "Content-Type: application/json" -d
'{"n": 6, "hostnames": ["S1","S2","S3","S4","S5","S6"]}' http://127.0.0.1:5000/add
{
  "message": {
    "N": 6,
    "replicas": [
      "S1",
      "S2",
      "S3",
      "S4",
      "S5",
      "S6"
    ]
  },
  "status": "successful"
}
ranivenigalla@rani:~/DS_Assgn/DS_ASSIGNMENT_1$
```

If the n value is more than the no. of hostnames mentioned in the list , it will automatically insert a server with some random name.

```
ranivenigalla@rani:~/DS_Assgn/DS_ASSIGNMENT_1$ curl -X POST -H "Content-Type: application/json" -d
'{"n": 2, "hostnames": []}' http://127.0.0.1:5000/add
{
  "message": {
    "N": 2,
    "replicas": [
      "S8789",
      "S3973"
    ]
  },
  "status": "successful"
}
ranivenigalla@rani:~/DS_Assgn/DS_ASSIGNMENT_1$
```

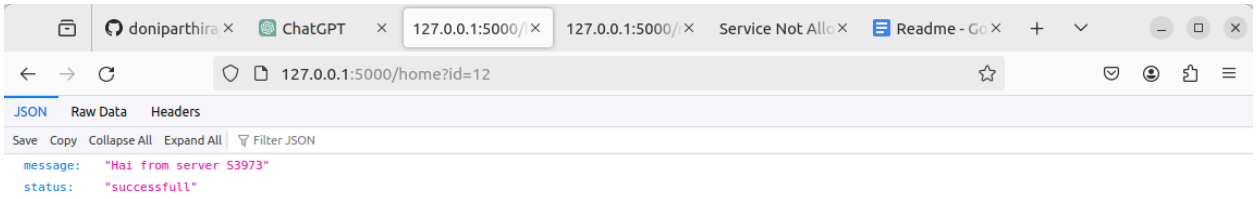
Else if the no.of hostnames are more than the value n mentioned , then we get a error :

```
ranivenigalla@rani:~/DS_Assgn/DS_ASSIGNMENT_1$ curl -X POST -H "Content-Type: application/json" -d
'{"n": 2, "hostnames": ["S45","S34","S56"]}' http://127.0.0.1:5000/add
{
  "message": "<Error> Length of hostname list is more than newly added instances",
  "status": "failure"
}
ranivenigalla@rani:~/DS_Assgn/DS_ASSIGNMENT_1$
```

2) **End-point(/home?id=12)**

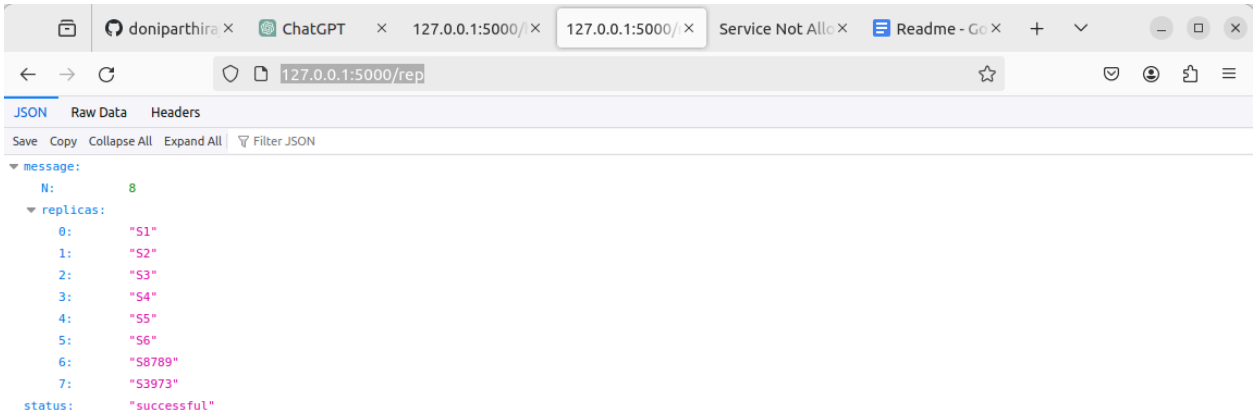
   Using this endpoint the client could send get-requests. The request returns a message specifying which server it got mapped to.

   here we have to send a parameter id which specifies the client-id.



3) End-point(/rep)

This endpoint gives the number and names of running servers .



4) End-point(/rm)

This endpoint removes (stops from running and removes the server)the servers specified in the payload.

If n = no. of hostnames specified. It removes the hostnames specified in the payload and returns the remaining list of replicas that are running.

If n > no. of hostnames specified, then all the hostnames specified will be removed, the remaining (n - len(hostnames)) are removed randomly.

```
ranivenigalla@rani:~/DS_Assgn/DS_ASSIGNMENT_1$ curl -X DELETE -H "Content-Type: application/json" -d '{"n": 2, "hostnames": []}' http://127.0.0.1:5000/rm
{
  "message": {
    "N": 4,
    "replicas": [
      "S4",
      "S5",
      "S8789",
      "S3973"
    ]
  },
  "status": "successful"
}
ranivenigalla@rani:~/DS_Assgn/DS_ASSIGNMENT_1$
```

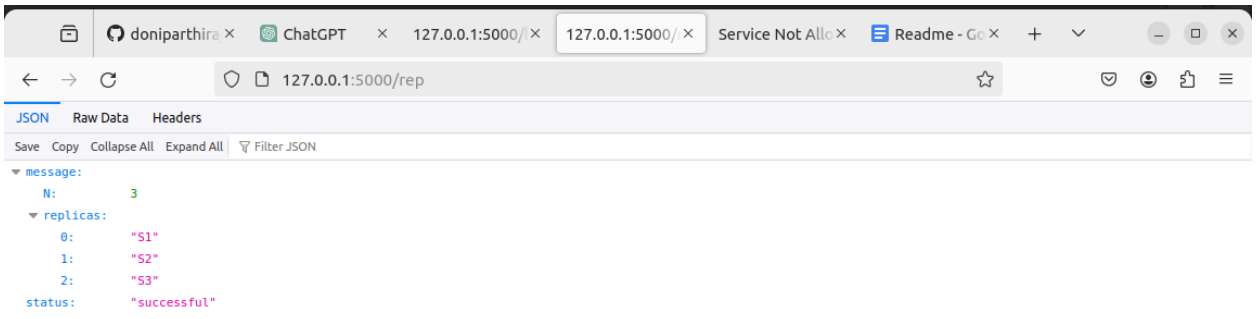
If n < no.of hostnames specified then we get an error message.

```
ranivenigalla@rani:~/DS_Assgn/DS_ASSIGNMENT_1$ curl -X DELETE -H "Content-Type: application/json" -d '{"n": 1, "hostnames": ["S4","S5"]}' http://127.0.0.1
:5000/rm
{
  "message": "<Error> Length of hostname list is more than removable instances",
  "status": "failure"
}
ranivenigalla@rani:~/DS_Assgn/DS_ASSIGNMENT_1$
```

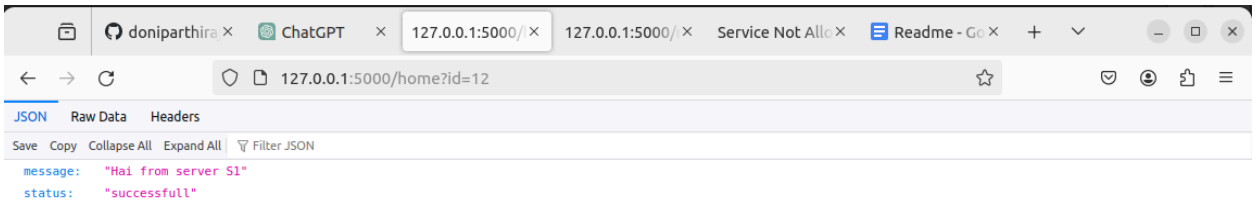
SPAWNING:

When a server is down(here , done it manually using stop command) then loadbalancer spawns a new server.

/rep:



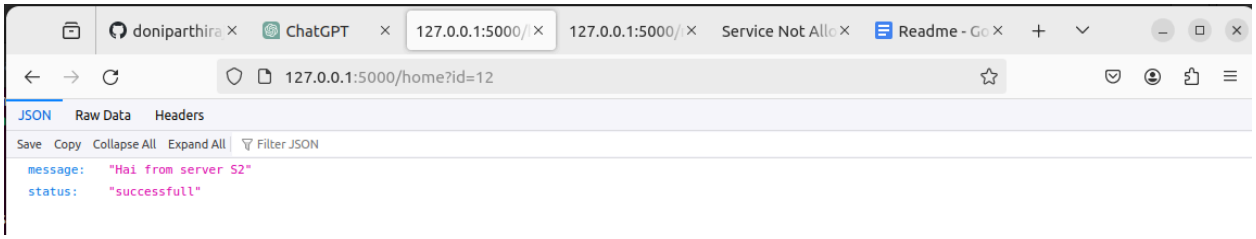
Sending a client request with id = 12.



Making S1 down.

```
}
ranivenigalla@rani:~/DS_Assgn/DS_ASSIGNMENT_1$ sudo docker stop S1
S1
ranivenigalla@rani:~/DS_Assgn/DS_ASSIGNMENT_1$
```

Sending the client request again with client id = 12. It got assigned to S2 now.



/rep :

Server S1_7 has been added once S1 is down.

Experiment A-4:

Finally, modify the hash functions H(i), Φ(i,j) and report the observations from (A-1) and (A-2).

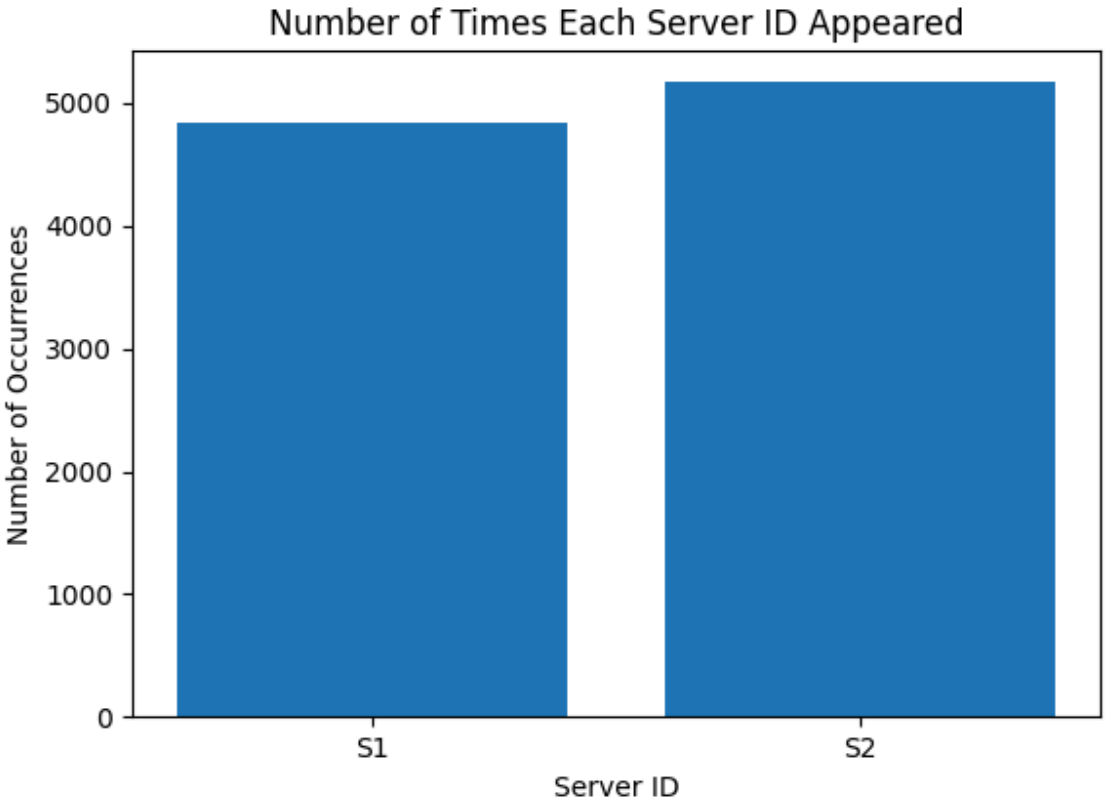
H(i) = i2 + 2i + 17 + k
Φ(i, j) = i2 + j2 + 2j + 25 + c

Where k , c are randomly generated integers in the range (0,10000).

N = 2:

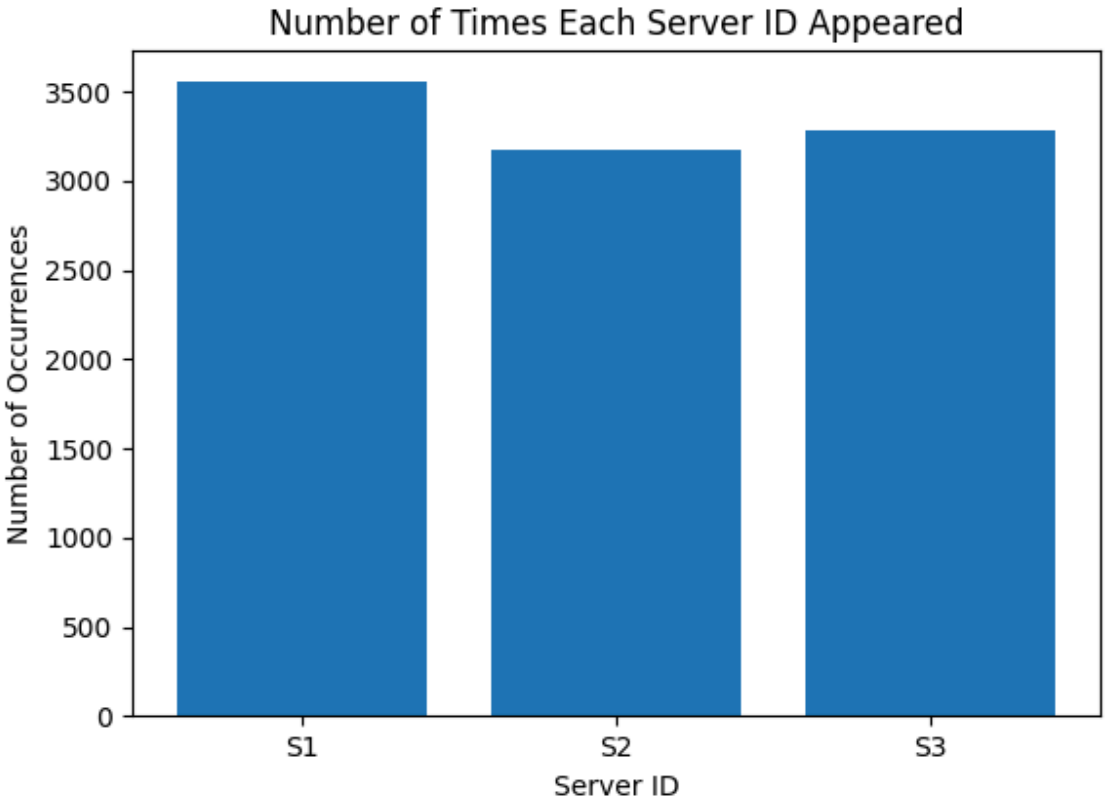
- Server S1 has served almost 4900 requests.
- Server S2 has served almost 5100 requests.

- Server S1 and S2 has almost the same load.



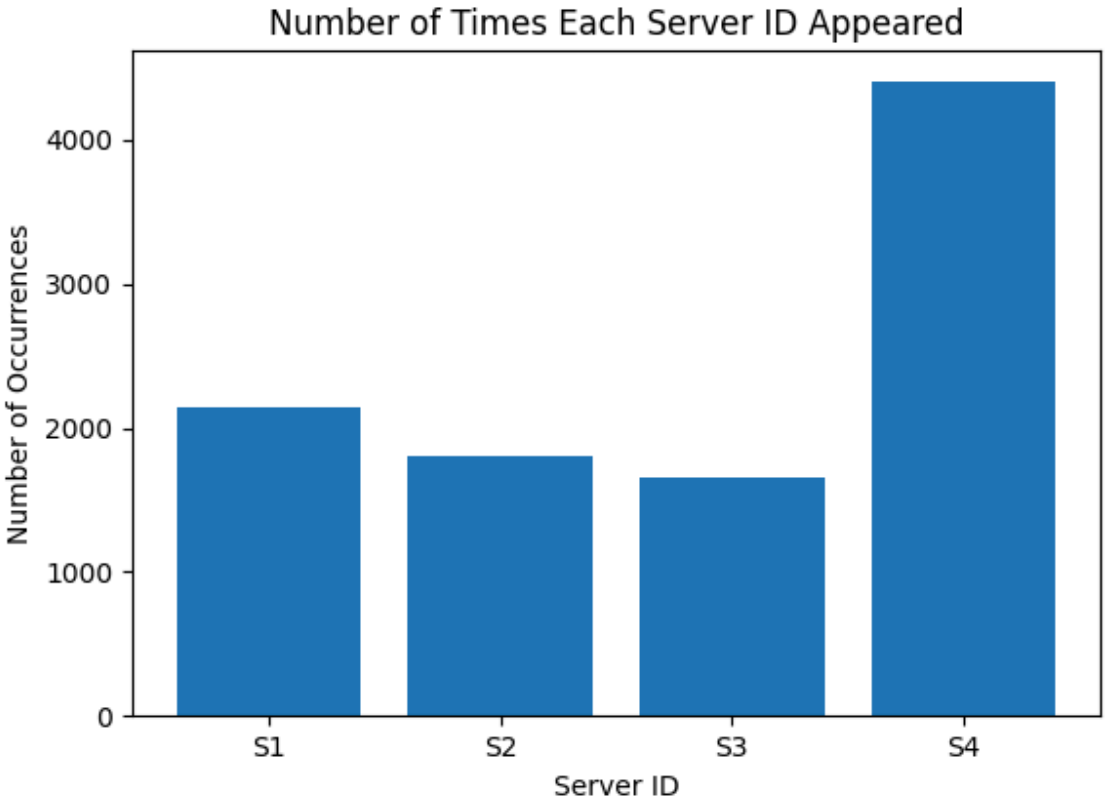
N = 3:

- Server S1 has served almost 3500 requests.
- Server S2 has served almost 3200 requests.
- Server S3 has served almost 3300 requests.
- All the 3 servers has almost the same load.



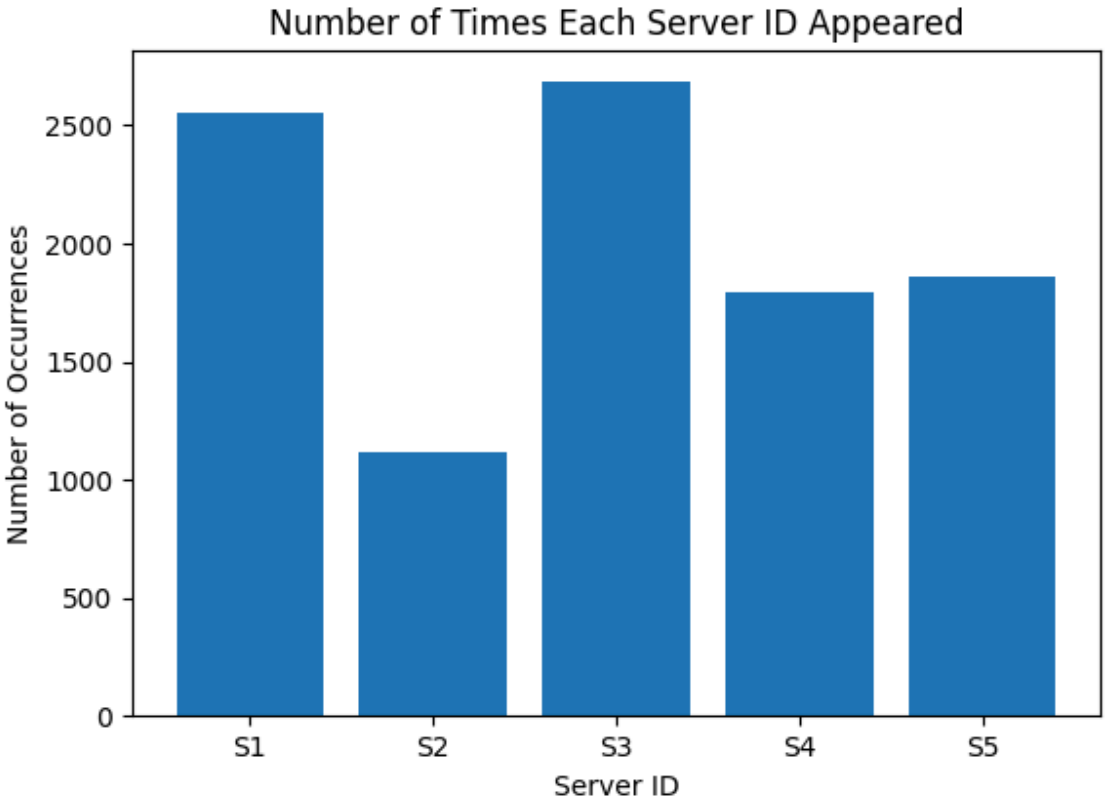
N = 4:

- Server S1 has served almost 2100 requests.
- Server S2 has served almost 1800 requests.
- Server S3 has served almost 1700 requests.
- Server S4 has served almost 4400 requests.
- Server S4 has the highest load.
- The remaining 3 servers S1,S2,S3 almost have the same load.



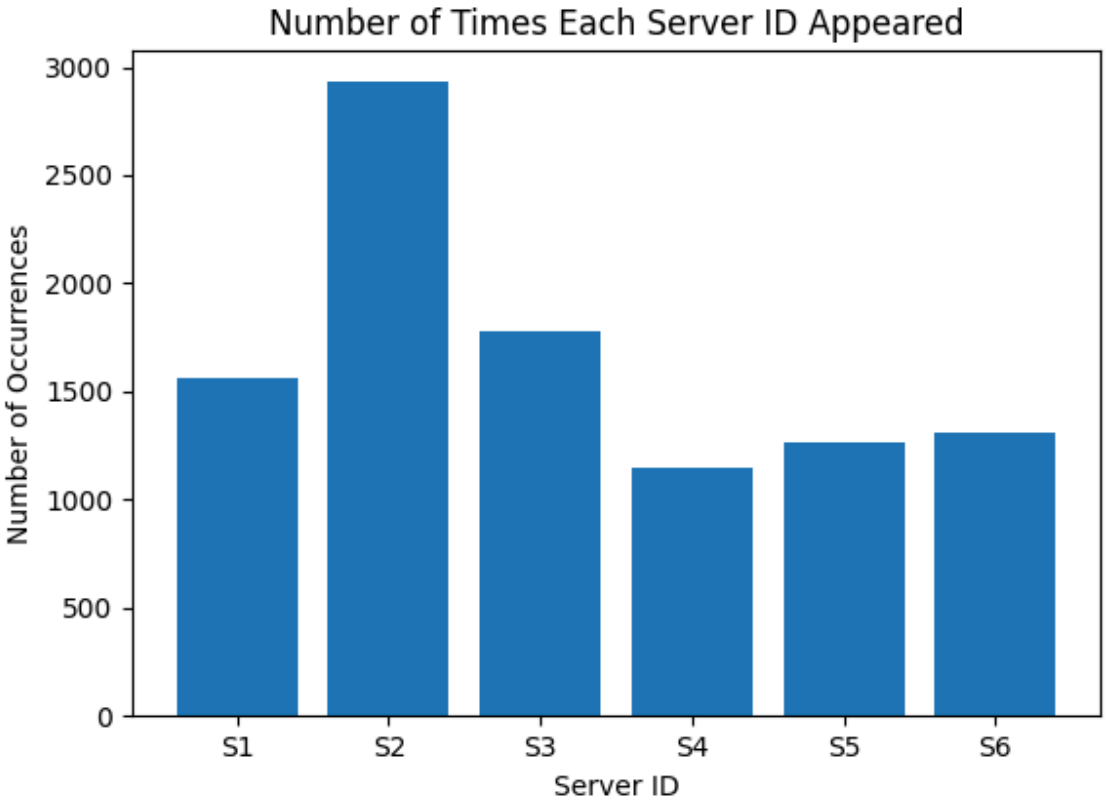
N = 5:

- Server S1 has served almost 2500 requests.
- Server S2 has served almost 1200 requests.
- Server S3 has served almost 2800 requests.
- Server S4 has served almost 1700 requests.
- Server S5 has served almost 1800 requests.
- The server S1,S3 have comparatively higher load compared to servers S2,S4,S5.



N = 6 :

- Server S1 has served almost 1600 requests.
- Server S2 has served almost 2900 requests.
- Server S3 has served almost 1800 requests.
- Server S4 has served almost 1100 requests.
- Server S5 has served almost 1200 requests.
- Server S6 has served almost 1400 requests.
- S2 has a higher load. Other servers have comparatively similar load.


Number of Times Each Server ID Appeared
Number of Occurrences
3000
2500
2000
1500
1000
500
0
S1
S2
S3
S4
S5
S6
Server ID