Short Url Generator:

Scale and Constraints:

Assumptions:
500 million new URL shortening requests per month
100:1 read/write ratio

Calculations:
Write Query per Month: 500 millions
Read Query per Month: 500 millions * 100 = 50 billions

Write query per second = 500 million / (30days * 24hours * 3600 seconds) = ~200 URLs
Read query per second = 200 * 100 = 20K / s

Encoding URL length:
Total number can be encoded = 62^7 (including 0) = 3,521,614,606,208

Time to exhaust all the counter = 3,521,614,606,208 / (500million * 12) = 586.9 years.

Even if 1000 (200*5) write queries are performed, it will take 117(586.9 / 5) years to exhaust.

Design Decisions:

1. *Selected SQL DB (PostgreSQL):*
   We have created two tables, one for users which is ***users*** and one for short urls which is ***shorturls,*** In stage-1 we decided to go with No-SQL database but we realized in later stage that in order to efficiently implement api-key queries and also implement favorite url endpoint we required relationship between both the tables, so we switched to PostgreSQL.

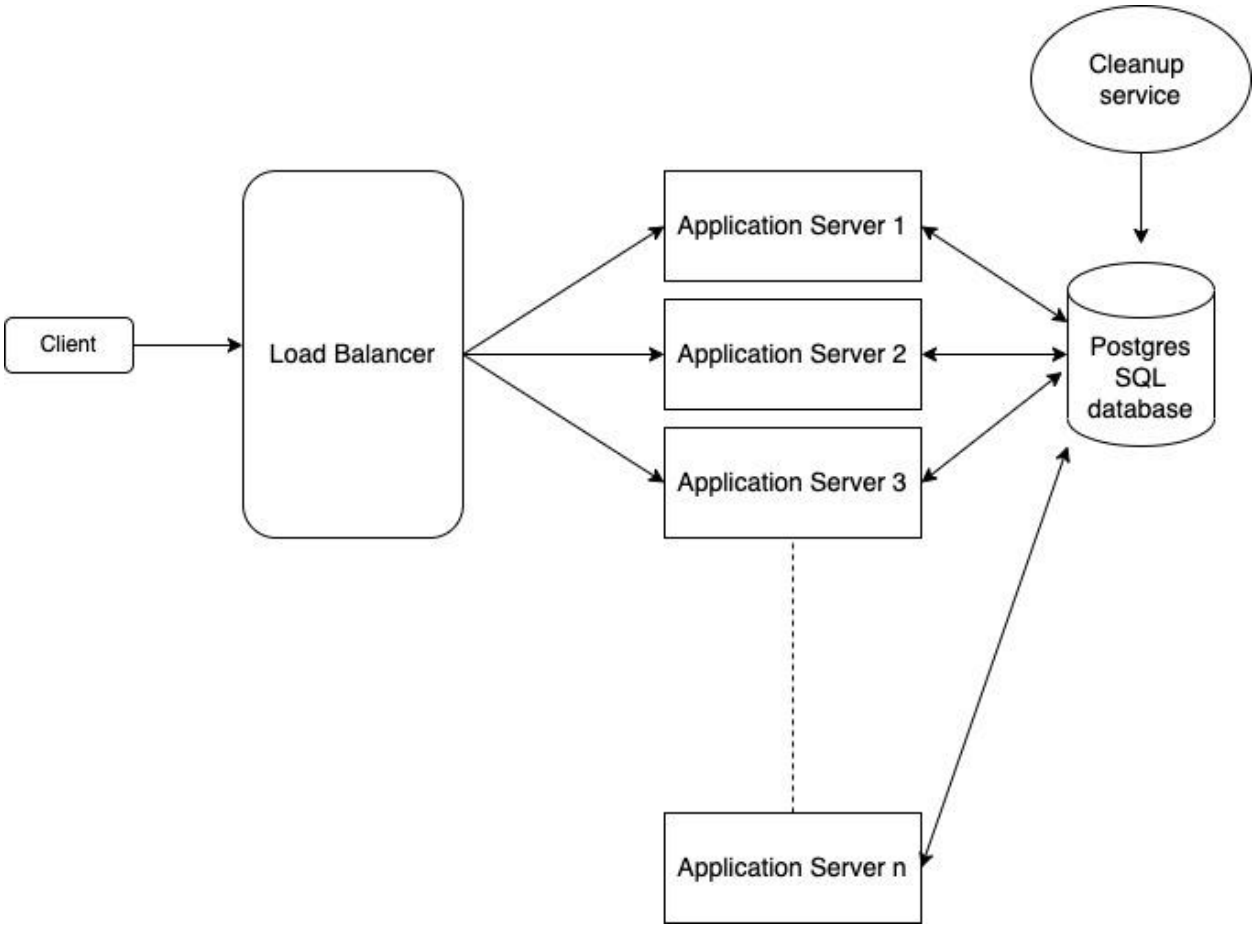
2. *Generating short-URL:*
   For generating short url we used base64 md5 digest using current timestamp, long_url and api_dev_key.

```
    md5encodedUrl=base64.b64encode(hashlib.md5(usernameWithLongurl.encod
e('utf-8')).digest()).decode("utf-8")
```

   We selected 7 characters from front of the generated md5encodeUrl.

3. *Cleanup Service:*
   In order to remove the expired urls, we ran a cleanup service which basically takes the current timestamp and does the query to select and delete all the urls which have expiry timestamp less than current timestamp.
4. *API call limiter*
   In order to limit the number of write calls per API key, we implemented a API call limiter, We basically check the quota for the user who have requested the write request, we only allow a write request when his quota is greater than zero, we also renew quota of the user based on a particular time period, right now we have limited our users to do 1000 write request per hour.

Skeleton of design doc:



Special Features:

- Users can mark/unmark a short url created by him as favorite and can access the list of favorite short url marked by him through a get request.
- We have created an api endpoint named /favorite which has two endpoints one of which is a POST request which mark/unmark a url as favorite, another one is GET request which gives all the favorite url related to a given api_key.

Hosted URL:

Base-URL: http://auto-scaling-group-1-1-1770288023.ap-south-1.elb.amazonaws.com/v1/

Documentation-Testing-URL:
http://auto-scaling-group-1-1-1770288023.ap-south-1.elb.amazonaws.com/v1/ui

How to use it ?

1. We can test the APIs using three methods, first by using swagger UI which is hosted at /v1/ui endpoint, another way is to use postman and finally we can use curl in our terminal.
2. Generate an API_KEY by giving user_name and password.
3. Use the API_KEY to generate short_url_key from long_url.
4. Now we can use the short_url_key to query the api, and it will be redirected to long_url.(Note that in order to test redirect-api in swagger, generate the response url and then paste it in your browser to test it, for postman and curl we can directly test the redirect api.)
5. Details documentation of the API calls can be found on API_Documentation.

Test-Credentials -:

```
{
      "user_name": "username",
      "password": "password"
}
```

```
api_dev_key : _EDCelhCKqquBQ-hUp6FhUg5crvNnunp0mwWLCoUtjA
```