Programming Assignment 1
(CS559: Computer Systems Design)

Instructions: This assignment can be done in pairs (group of 3 students). Each student in the group must understand the code and techniques used and be able to answer each question (don't divide the problems amongst yourselves). It is fine to use web-resources and help, provided you reference them and understand the code. Please don't copy blindly.

We will have viva or create Tierce exam questions which require knowledge of your implementation, so every member should understand the design thoroughly

Due Date:

[Stage 1] Jan 16, 2022 (End of the day)

[Stage 2] Jan 31, 2021 (End of day)

Late Days: You have total of 5 late days for the semester for all homework and assignments, which you can use as you need them

Submission: Submit your solution for each question as a Git-hub repo and provide us the URL for your service so we can test it using curl or an appropriate load generator.

Introduction

URL shortening is used to create shorter aliases for long URLs. We call these shortened aliases "short links." Users are redirected to the original URL when they hit these short links. Short links save a lot of space when displayed, printed, messaged, or tweeted. Additionally, users are less likely to mistype shorter URLs.

URL shortening is used to optimize links across devices, track individual links to analyze audience, measure ad campaigns' performance, or hide affiliated original URLs.

Functional Requirements:

1. Given a URL, our service should generate a shorter and unique alias of it. This is called a short link. This link should be short enough to be easily copied and pasted into applications.
2. When users access a short link, our service should redirect them to the original link.
3. Users should optionally be able to pick a custom short link for their URL.
4. Links will expire after a standard default timespan. Users should be able to specify the expiration time.

Non-Functional Requirements:

1. The system should be highly available. This is required because, if our service is down, all the URL redirections will start failing.
2. URL redirection should happen in real-time with minimal latency.
3. Shortened links should not be guessable (not predictable).

Extended Requirements:

1. Analytics; e.g., how many times a redirection happened?
2. Our service should also be accessible through REST APIs by other services.

Implementation Details

Suggested Reading:

https://github.com/donnemartin/system-design-primer/tree/master/solutions/system_design/pastebin

This is provided as a reference. Feel free to modify it according to the needs of your specific service.

Sample REST API specs:

```
createURL(api_dev_key, original_url, custom_alias=None, user_name=None, expire_date=None)
```

Parameters:
api_dev_key (string): The API developer key of a registered account. This will be used to, among other things, throttle users based on their allocated quota.
original_url (string): Original URL to be shortened.
custom_alias (string): Optional custom key for the URL.
user_name (string): Optional user name to be used in the encoding.
expire_date (string): Optional expiration date for the shortened URL.

Returns: (string)
A successful insertion returns the shortened URL; otherwise, it returns an error code.

```
deleteURL(api_dev_key, url_key)
```

Where "url_key" is a string representing the shortened URL to be retrieved; a successful deletion returns 'URL Removed'.

How do we detect and prevent abuse? A malicious user can put us out of business by consuming all URL keys in the current design. To prevent abuse, we can limit users via their api_dev_key. Each api_dev_key can be limited to a certain number of URL creations and redirections per some time period (which may be set to a different duration per developer key).

Grading Scheme

- Stage 1: [25 points] We will test if the basic service is working functional requirements
    - Evaluation using curl or a python script to measure latency and performance of your implementation
- Stage 1 and 2: [15+15 points] Creative features, interesting variations, analytics, visualizations
    - If x groups submit identical features, your credits will be 30/x
    - For example, you can create a service to store image thumbnails, documents, add logins, authentication
- Stage 1 and 2: [10+10 points] Quality of documentation, code
- Stage 2: [25 points] Fault tolerance and high availability
    - Evaluation: One or two VMs will be stopped randomly and the system should still work