

URL shortener

Authors

Anuja Kench

Reviewers

Rupalim Sarma

Introduction -

This document provides an overview of the URL shortening project. It describes the functional and non-functional requirements, the system design, design considerations, tools to develop a proof of concept.

Functional requirements -

This project should support the following requirements -

1. Generate a short URL from a long URL.
2. Redirect the user using a short URL to the actual long URL.
3. List the number of times that a user accesses a URL based on a timeframe.
4. Provide an interface to delete a short URL.
5. Support data persistence.
6. Provide data logging for troubleshooting.

Non-Functional requirements -

1. The system is read heavy

Assumptions and Design decisions -

1. **Storage assumptions** - The short URL will look like myshorturl.com/acfr10as. The URI suffix will have 8 characters. These characters can include digits from 0 - 9, lower case letters a - z and upper case letters A - Z.

- 2. Cache assumptions** - The system is **read heavy** so a cache should be used for **higher throughput, lower latency** and **higher response** time.

Whenever there is a write operation in the secondary storage, then a request would be made to update the cache(Lazy loading). Least frequently used(LFU) cache eviction policy can be implemented since users are most likely to access popular URLs like google.com frequently. There are around 1 million read requests generated a month.

- 3. Scalability assumptions** - An application level load balancer can be deployed between the client and the server to improve scalability. In the scope of the project, this is a future requirement.
- 4. High availability** - To support high availability, multiple instances of URL generator service could be run so that in case of failure the load balancer can redirect the request to another service.
- 5. Platform support** - This project is developed in a Windows environment. Ideally, a sandbox environment could be created using Vmware or other virtualization techniques so as to support multiple platforms like Linux apart from Windows OS.
- 6. Cloud support** - The URL generator service can be enhanced to be run as a microservice that could be run in a EC2 instance in AWS cloud for example. This is not scoped in the first version of the project.

System Design -

A URL shortening service will be designed as a HTTP-based application. REST API's will be used to expose the functionality. A long URL like <https://www.cloudflare.com/cybersecurity/> can be converted to short URL like <https://tinyurl.com/azxd60pa>

Tools -

Go1.22.0

Visual Studio Code

Github

Git bash