![educative] **educative**            ❓ Ask a Question

# System Design: TinyURL

Let's design a service similar to TinyURL for shortening the uniform resource locator (URL).

## We'll cover the following  ⌃

- Introduction
  - Advantages
  - Disadvantages
- How will we design a URL shortening service?

## Introduction

**URL** **shortening** is a service that produces short aliases for long URLs, commonly referred to as **short links**. Upon clicking, these short links direct to the original URLs. The following illustration depicts how the process works:
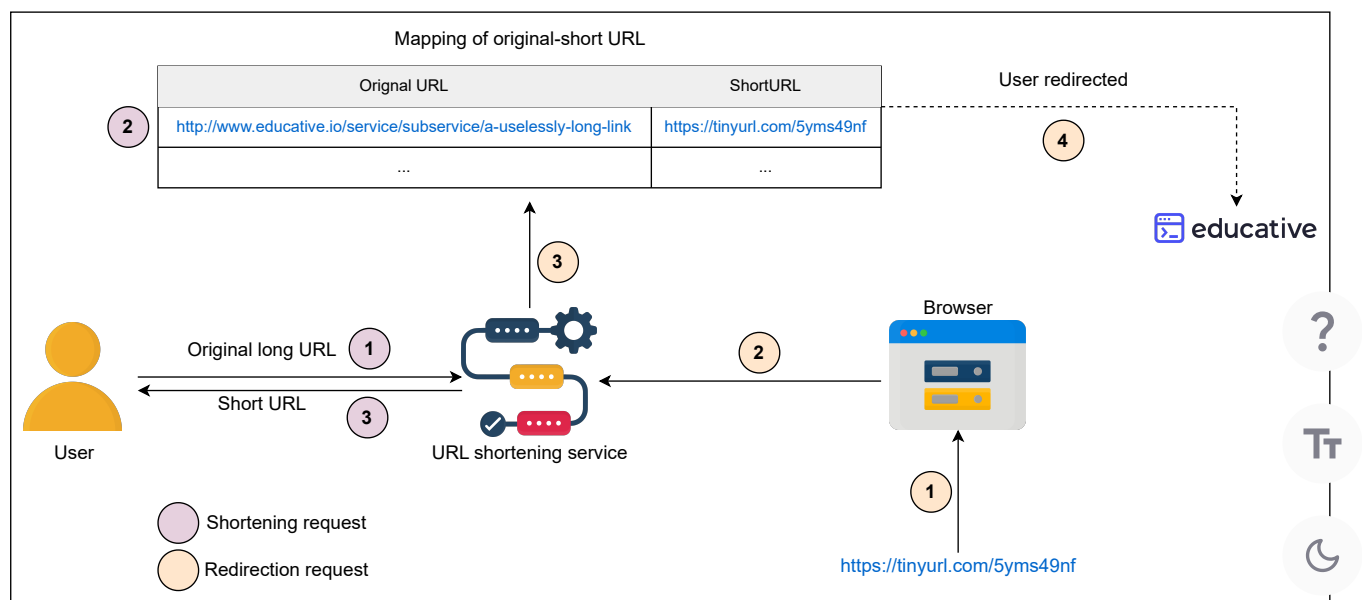
How URL shortening service works

## Advantages

> The key advantages of a URL shortening service are:

- Shortened URLs are convenient to use: they optimize link usage across different forms of devices due to their enhanced accessibility and non-breakability.
- They are visually professional, engaging and facilitate a higher degree of sharing possibilities.
- They are less error-prone while typing.
- They require smaller storage space at the user's end.

## Disadvantages

The URL shortening service has some associated drawbacks as well. Some of them are as follows:

- We lose the originality of our brand by using a short URL generated by a third-party service. Many different brands using the same service get short URLs containing the same domain.
- Since we're using a third-party service for URL shortening, the possibility of it getting shut down and wiping all our shortened URLs will always be there.
- Our business brand image depends on the trustworthiness of a URL shortening service. The wrong business move might negatively impact our business. The competition for acquiring the in-demand custom short URLs is immense, so it might be possible that the best custom URLs are already taken by the time we start generating short URLs.

# How will we design a URL shortening service?

We've divided the URL shortening service design into the following five lessons:

1. **Requirements:** This lesson discusses the functional and non-functional requirements of the URL shortening service, along with estimating the resources required to achieve these requirements. Moreover, it also lists down the fundamental building blocks needed to build such a service.

2. **Design and Deployment:** This explains the working and usage of each component, the linkage among them, and the overall working mechanism of them as a unit.

3. **Encoder:** This particular lesson unfolds the inner mechanism of the encoder used in the design, stating the reason we use it along with the mathematical explanation.

4. **Evaluation:** Lastly, we test our design by considering different dimensions of our design requirements and include the possibility of improving it.

5. **Quiz**: This lesson will test our understanding of the TinyURL design.

Let's start by defining the requirements for a TinyURL-like URL shortening service.

← **Back**

Quiz on Instagram's Design

**Next** →

Requirements of TinyURL's Design

✓ Completed