Kasu Journal of Computer Science Vol. 1 No. 3 [September, 2024], pp. 423-437
https://doi.org//10.47514/kjcs/2024.1.3.001

Online ISSN: 1597-2216
Print ISSN: 1597-2178
www.kjcs.edu.ng

A UNIFORM RESOURCE LOCATOR SHORTENER USING ACCENTED CHARACTERS

Oluwasogo Adekunle Okunade [1], Oluwaseyitan Osunade[2], Babatunde Seyi Olanrewaju [3], Ademola Abiodun Omilabu[4] and Opeyemi Akanbi[5]

[1] Department of Computer Science, National Open University of Nigeria, Abuja, Nigeria

[2] Department of Computer Science, University of Ibadan, Ibadan, Nigeria.

[3] Department of Computing, Wellspring University, Benin City, Nigeria.

[4] Department of Computer and Information Science, Tai Solarin University of Education, Ijagun, Ijebu-Ode, Nigeria.

[5] Department of Computer Science, University of Ibadan, Ibadan, Nigeria.

aokunade@noun.edu.ng[1], seyiosunade@gmail.com[2], Babatundeolanrewaju@wellspringuniversity.edu.ng[3], omilabuaa@tasued.edu.ng[4], akanbi.sunny@gmail.com[5]

Abstract

This study presents a novel methodology for enhancing URL shortening by utilizing accented characters to increase both the flexibility and security of shortened URLs. The methodology involves a multi-step process where users first register and log into the system. Upon submission of a long URL, a key creation service generates a unique, random key using an accented character set. This key is then used to produce a shortened URL, which is stored in a database along with the corresponding original URL. When a shortened URL is requested, the system queries the database to retrieve the original URL or returns an error message if the URL is not found. The design also includes load balancers to manage network traffic, an application server for URL submission, and a cleanup service to manage expired keys. Additionally, the system is designed to generate non-predictable and secure shortened URLs, with provisions for both functional and non-functional requirements, such as high availability and minimal latency. The findings suggest that this methodology improves the security and usability of URL shortening services, offering a significant reduction in the risk of URL-related fraud.

Keywords: Uniform Resource Locator (URL), URL Shortener, Redirect Link, Accented Characters, Log Database

1. Introduction

The texts in the URL (Uniform Resource Locator) is required to fit into a restricted number of characters, normally 140 characters in some social networking sites such as Twitter. This call for a service which would shorten a URL and yet automatically redirect the user to the original URL without misleading (Le-Khac & Kechadi, 2015a and Sankhala, et al., 2022). The usage of URL shortener has been intensified due to the security concern about vital personal or confidential information that are available on the web especially the widely used social media sites.

URL shortening is a technique used in the World Wide Web whereby a URL may be made substantially shorter as shown in Figure 1 and still direct to the required page.

Kasu Journal of Computer Science Vol. 1 No. 3 [September, 2024], pp. 423-437
https://doi.org//10.47514/kjcs/2024.1.3.001

Online ISSN: 1597-2216
Print ISSN: 1597-2178
www.kjcs.edu.ng

Figure 1. URL Shortening**, (**Source: www.socialpilot.co/blog/importance-url-shortener-social-media-marketing *(Mehta, 2016)).*

URL shortener is a website that creates abbreviated versions of web addresses, called shortlinks. In shortening of URL, it is necessary to make the shortened URL easier to track and remember (Shaheen, & Isaac, 2022). This is achieved by using a redirect which links to a web page that has a long URL. For example, the URL "https://example.com/assets/category_B/subcategory_C/Foo/" can be shortened to https://example.com/Foo. and http://www.example.com/my-super-awesome-blog-about-squirrels/2022/10/23/live-with-a-squirrel-fan--they%27re-awesome to be shortened to https://tinyurl.com/live-with-me. URL shortening services take as input a long URL to generate a short one in return and the generated Short URL redirects to the same long URL that looks random and unrelated to the actual link (Pattewar, et al., 2018).

It is hard and unexpected of visitors to understand or memorise long or complex URL, as such fraudsters can abuse this “trust” through a combination of social engineering and deliberately broken long or incorrect URL’s, to cause the visitors to follow a shortened URL that eventually led to a wrong destination, as a result of its simplicity (Ismail, 2014). As such to make a URL identifiable by humans, some organizations must be applied to make it more structured such as by using slashes, date of content creation, slugs, and keywords from articles and blog posts (Prasetyadi et al., 2018). With the difficulty in URL memorizing, manual typing, and distributing (especially via non-digital media). It can only be copied-pasted for reliability. URL shortener can shorten long URLs into shortened ones. Attackers abuse URL shortening services, perhaps to mask the final destination (their designed malicious site) where the victim will land after clicking on the malicious link (Page et al, 2018). Likewise, legitimacy assumption that the link will be secured without taking security into cognisant had tricked visitors to visit malicious web (Devesa, et al., 2011a). However, this paper presented accented characters as a URL shortener character set as a new model for URL shortening using accented characters as its base character set, for URL shortening flexibility.

Kasu Journal of Computer Science Vol. 1 No. 3 [September, 2024], pp. 423-437
https://doi.org//10.47514/kjcs/2024.1.3.001
Online ISSN: 1597-2216
Print ISSN: 1597-2178
www.kjcs.edu.ng

2. Related Works

The term "URL" refers to the address of a website (Shaheen & Isaac, 2022). A URL is essentially the address of a specific unique resource on the Web; theoretically, every valid URL points to a unique resource. These resources can be images, CSS documents, HTML pages, or any combination of these. The URL is typically displayed above a web page in the address bar of most web browsers. An example of a typical URL would be http://www.example.com/index.html, which would indicate the protocol (http), the hostname (www.example.com), and the file name (index.html). It was shown that over 10,000 unique blacklisted phishing and malware URLs were posted to Twitter during a 2-month timeframe in 2017 (Bell and Komisarczuk, 2020a). As stated by Simon et al. (2020), one way to assess how well a Twitter URL shortener shields users from malware and phishing is to look at its security rating. Anybody who clicks on a short link is redirected to the original long-form web address. Users of some URL shorteners can generate personalized, vanity or branded URLs. Marketers frequently use these technologies to enhance the visual appeal of lengthy URLs in collateral or communications. Short URLs are particularly common in mobile messaging with constrained screen real estate and on social media platforms that enforce character limits.

If the reader is copying a URL from a print source, they may want a friendly URL to make it easier to remember, to minimize typing, to serve as a permalink, or for messaging services that cap the character count in a message (like SMS). 2.1 billion times were links from the URL-shortening service Bitly viewed in November 2009 (Georgiev et al., 2016a). URL shortening can also be used to track clicks, "beautify" links, and hide the underlying address. While hiding the underlying address could be desirable for personal or professional reasons, there is potential for misuse (Le-Khac & Kechadi, 2015b). Due to websites using their redirect services to get around those same blacklists, certain URL shortening service providers have ended up on spam blacklists. Certain websites prohibit posting of short URLs that redirect. There's a character limit on Twitter and certain other instant messaging platforms for messages. It is no longer necessary to use a different URL shortening service in order to shorten URLs in a tweet, though, as Twitter now uses its own URL shortening service, t.co, to do so automatically. Using a URL shortener on other similar services enables linking to websites that would otherwise be prohibited. A string used to locate a web resource is called a web address. Azar & Associates, 2016b).

In order to both shorten URLs and guarantee sufficiently safe abbreviated URLs, researchers have created or suggested a variety of URL shorteners utilizing diverse techniques. The usage logs of a URL shortening service that has been running for roughly a year were examined by Klien & Stromhaier (2012). The analysis revealed the level of spamming occurring in the logs and offered preliminary information about the problem's likelihood. The findings are pertinent to scientists and engineers who want to learn more about the risks associated with spamming through URL shortening services and this new phenomenon. In 2014, Gupta and Kumaraguru conducted an analysis of the countermeasures currently in use by major shortening providers. Remarkably, it was discovered that these countermeasures are easily circumvented and ineffectual. In 2018, Sophie, Jordan, and Gregor gathered more than 7,000 harmful short URLs that were classified as malware or phishing.

Gottfried et al. (2018a) used the Python-based Django web framework to create a URL shortner that can be integrated and altered into any Django-based online project. The application was created using the Python programming language and standard character sets. In order to enable the user to determine if the destination is a file or a web document, Mun and Li (2017a) presented a method that writes the destination information when producing a short URL. The analysis they did was centered on the Twitter URL shortner's security level. Using a facade for links, such as a URL shortener that lets the user modify

Online ISSN: 1597-2216
Print ISSN: 1597-2178
www.kjcs.edu.ng

the shortened URL's target, is one potential option put out by Stephan (2017a). For linkages, they employed facade. This presents an additional dynamic targeting challenge. Bell and Komisarczuk (2020b) devised a method to gauge how well the tweeter URL shortener shields users from malware and phishing. They examined the efficacy of Twitter's URL shortening service (t.co) in shielding users against malware and phishing scams in their research. They demonstrated that during a two-month period in 2017, more than 10,000 distinct banned phishing and malware URLs were uploaded to Twitter. Over 1.6 million clicks resulted from this, all of which were directly from Twitter users, exposing individuals to potentially dangerous cyberattacks.

2.1 Recent Developments in URL Shortening

i. **Security Enhancements:** Recent studies have focused on improving the security aspects of URL shorteners. Mun and Li (2017b) proposed a secure short URL generation method that allows users to verify the destination of a shortened URL before visiting, significantly reducing the risk of redirection to malicious sites.
ii. **Advanced Shortening Algorithms:** Researchers like Gottfried et al. (2018b) have developed sophisticated URL shortening frameworks using modern web frameworks like Django, demonstrating the flexibility and customization possible with advanced programming techniques.
iii. **Phishing and Malware Detection:** Significant research has been conducted on the detection of malicious URLs generated through shortening services. Bell and Komisarczuk (2020c) analyzed over 10,000 blacklisted phishing URLs distributed via Twitter, highlighting the ongoing challenges in digital security related to URL shortening.

2.2 Theoretical and Practical Implications

i. **User Behavior Studies:** Investigations into how users interact with shortened URLs reveal patterns that can help in designing safer and more efficient services. Gupta and Kumaraguru (2014b) found that existing security measures in popular shortening services are often insufficient, suggesting a need for continuous improvement in security protocols.
ii. **Technical Innovations:** The adoption of non-traditional character sets, such as accented characters, in URL shortening presents a novel approach to enhancing the flexibility and capacity of shortening services. This method can potentially increase the number of unique URLs that can be generated, as explored by Stephan (2017b) in his development of a sustainable URL shortener for scientific use.

2.3 Additional References to Consider

i. **Security Studies:** Devesa et al. (2011b) discuss efficient security solutions for dealing with shortened URLs, which could provide a theoretical basis for the security measures proposed in your paper.
ii. **Technical Approaches:** The work by Georgiev and Shmatikov (2014b) on the vulnerabilities introduced by short URLs in cloud services can offer insights into the broader implications of URL shortening technologies.

Kasu Journal of Computer Science Vol. 1 No. 3 [September, 2024], pp. 423-437
https://doi.org//10.47514/kjcs/2024.1.3.001
Online ISSN: 1597-2216
Print ISSN: 1597-2178
www.kjcs.edu.ng

3. Methodology

To convert users' long URLs to short URL links with the ability to redirect to the actual URL and higher level of security, a methodology that uses accented character set was proposed. The proposed methodology as shown in Figure 2 comprises of:

i. Load balancers: these devices distribute network user requests among themselves.

ii. An application server that enables users to submit a URL for shortening

iii. A key creation service that will provide the required key at random to be used in order to shorten a URL that has been submitted,

iv. The database is used to map the created URL to the real URL, and the cleanup service is used to gather the generated keys and artifacts.

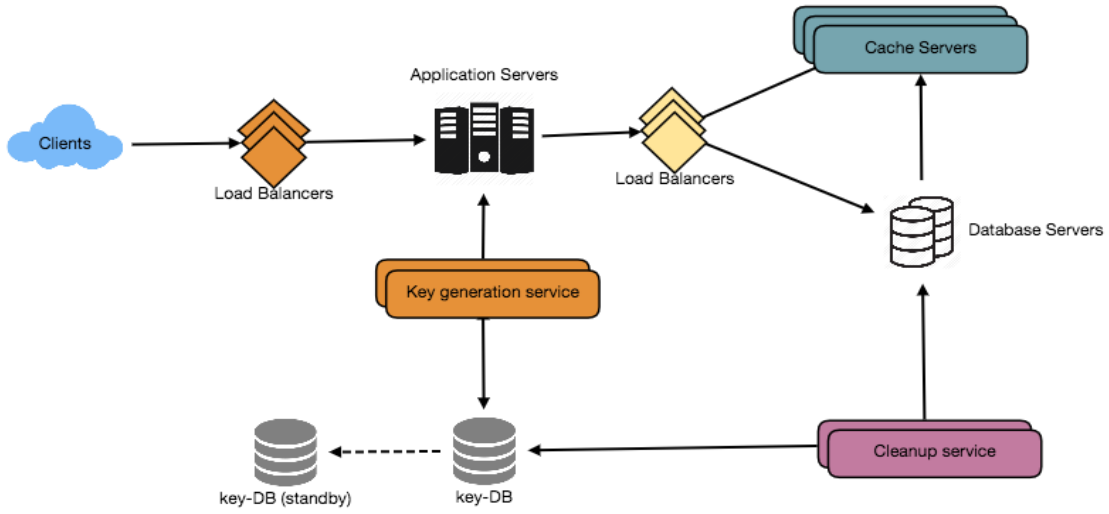

Figure 2. URL Shortening**, Source:** https://samirpaul.in/posts/url-shortening-service/

URL shortening involves generating concise alternatives for lengthy URLs, commonly referred to as "short links." When users click on these short links, they are directed to the original web address. Short links are particularly advantageous in terms of space conservation, whether in displays, prints, messages, or tweets. Moreover, shorter URLs tend to reduce the likelihood of user typing errors.

3.1 Functional Requirements

i. The system must create a concise and unique alias, known as a short link, for a given URL.
ii. Upon accessing a short link, the system must promptly redirect users to the original link.
iii. Users should have the option to select a custom short link for their URL.
iv. Short links will automatically expire after a predefined default time period, but users can specify a different expiration time.

Kasu Journal of Computer Science Vol. 1 No. 3 [September, 2024], pp. 423-437
https://doi.org//10.47514/kjcs/2024.1.3.001
Online ISSN: 1597-2216
Print ISSN: 1597-2178
www.kjcs.edu.ng

3.2 Non-Functional Requirements:

i. The system's availability should be exceptionally high to ensure uninterrupted URL redirection services.
ii. URL redirection must occur instantaneously with minimal latency to enhance user experience.
iii. Shortened links must be designed to be non-predictable or guessable, prioritizing security and privacy.

3.3 Design Process

The section on "Short URL Generation" describes how to respond to a system request to create a short URL for an incoming long URL, as seen in Figure 3. A long URL is sent to the URL shortening service, which converts it into a shorter URL and returns it as an output. The previously produced abbreviated URL will be provided if the provided long URL already exists in the system. If the current shortened URL is no longer valid, a new one will be created and sent as the reply.

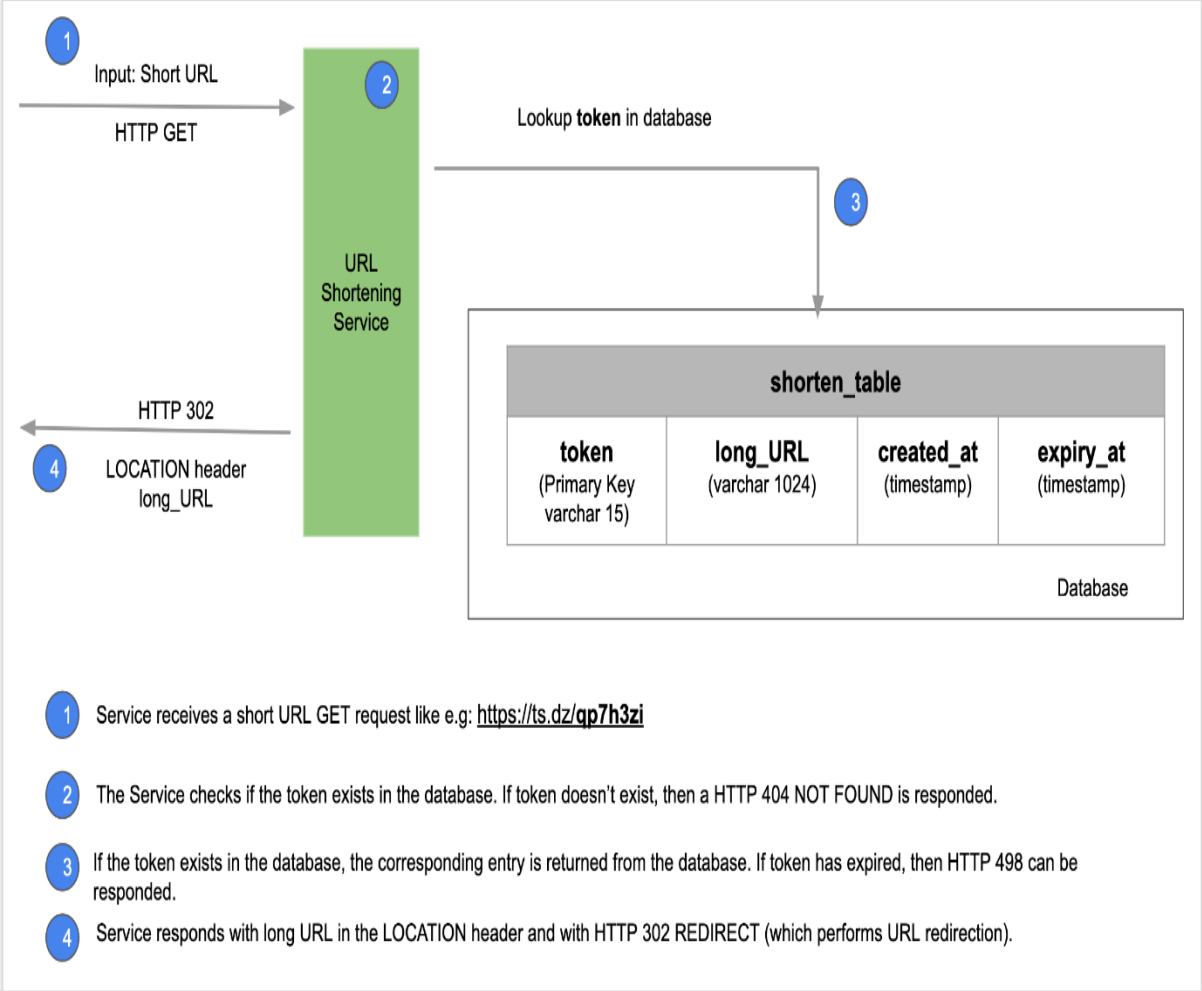

Figure 3. URL Generation, **Source:** https://dzone.com/articles/how-a-url-shortening-application-works

Short URL Redirection describes how to respond to a user's request to click on a short URL in order to redirect them to the lengthy URL displayed in Figure 4. In the event that a user chooses the shortened URL,

Kasu Journal of Computer Science Vol. 1 No. 3 [September, 2024], pp. 423-437
https://doi.org//10.47514/kjcs/2024.1.3.001
Online ISSN: 1597-2216
Print ISSN: 1597-2178
www.kjcs.edu.ng

the request ought to be sent to the URL Shortening Service, which will then send it to the actual long URL location. The service must be connected to the short domain in the brief URL as indicated below:

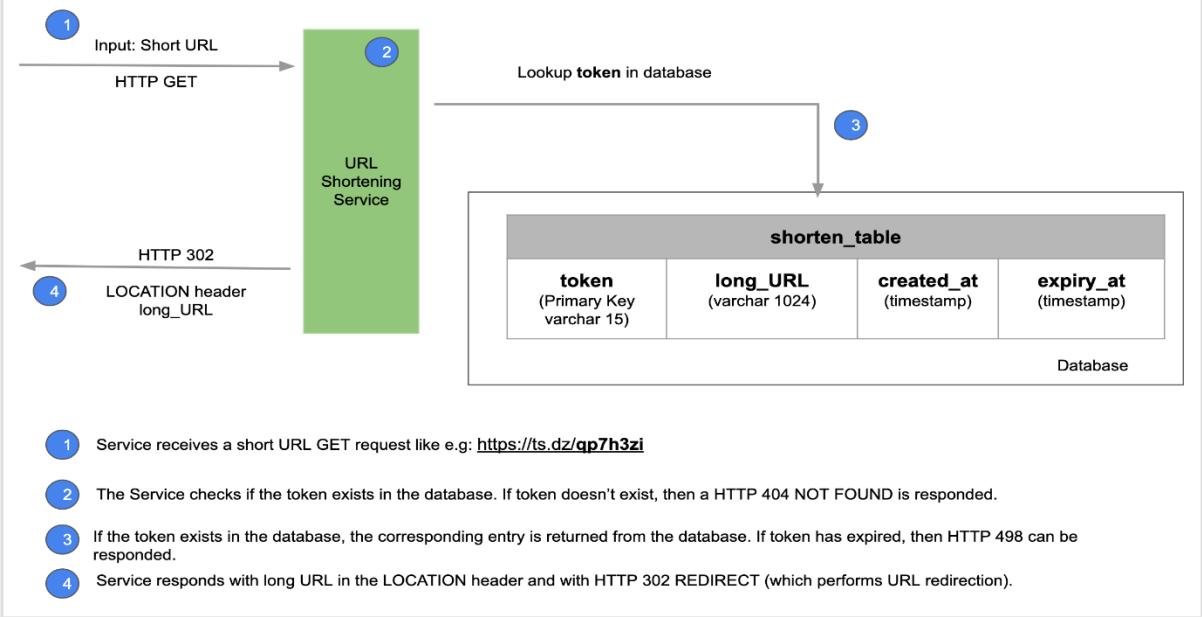

Figure 4. URL Redirection, **Sourc**e: https://dzone.com/articles/how-a-url-shortening-application-works

3.4 Token Generation

Creating unique, random tokens is a logical step. The URL shortening service has the responsibility to ensure that each token is associated with a single, distinct long URL. Failing to do so could result in the system incorrectly directing the abbreviated URL to an unintended long URL destination. Tokens can be generated using various methods, including the use of third-party libraries or traditional techniques like Base36 encoding.

3.5 Third-Party Library

Hashids, a highly dependable library available in over 30 languages, specializes in producing unique and random tokens or identifiers. It has the capability to generate concise and exclusive IDs from numeric values, including positive numbers and zero. Furthermore, Hashids provides the flexibility to utilize custom character sets and incorporate a "salt" to ensure that the IDs remain distinct to your specific application. The resulting IDs are intentionally obfuscated, rendering them challenging to predict. Notably, Hashids boasts an exceptionally small code footprint and operates without the need for external library dependencies.

Sample tokens generated for salt = "this is my salt" and character set = "abcdefghijklmnopqrstuvwxyz1234567890". The sample tokens provided are generated manually.

Kasu Journal of Computer Science Vol. 1 No. 3 [September, 2024], pp. 423-437
https://doi.org//10.47514/kjcs/2024.1.3.001
Online ISSN: 1597-2216
Print ISSN: 1597-2178
www.kjcs.edu.ng

The term "salt" typically refers to a random or semi-random value that is added to data before it is hashed. In the sample, the "salt" is given as a specific string, "this is my salt." The "character set" used is specified as "abcdefghijklmnopqrstuvwxyz1234567890." Automated token generation systems often use a predefined character set like this to create tokens. However, in this sample, the character set is explicitly defined, suggesting manual control over the token generation process.

Table 1: Token Generation

Hashid Input	Token Generated	Short URL
1	3joed16	https://test.com/3joed16
2	8lop31w	https://test.com/8lop31w
3	l6o3k1n	https://test.com/l6o3k1n
4	leo2418	https://test.com/leo2418

The given table demonstrates how Hashids can take input data, generate unique tokens, and use those tokens to create short URLs, which can be used to redirect users to specific web addresses. It is a practice for creating short, user-friendly links that are easier to share and manage. Here's an explanation of the columns:

Hashid Input: This column likely represents the input data, which could be a numerical value or some other identifier used as input to the Hashids library.

Token Generated: This column displays the tokens or unique identifiers that have been generated by the Hashids library based on the input data. These tokens are usually a combination of alphanumeric characters and are generated in a way that ensures uniqueness.

Short URL: In this column, the corresponding short URLs are displayed. These short URLs are constructed using the generated tokens and are typically used to redirect users to the original, longer URLs. The format of the short URLs appears to be "https://test.com/" followed by the generated token.

The given table demonstrates how Hashids can take input data, generate unique tokens, and use those tokens to create short URLs, which can be used to redirect users to specific web addresses. This is a common practice for creating short, user-friendly links that are easier to share and manage.

Base36 Encode

Base36 encoding a model shown in Figure 5 can be used to generate short URLs by converting a unique identifier, such as a numeric database record ID, into a shorter and more user-friendly representation.

Kasu Journal of Computer Science Vol. 1 No. 3 [September, 2024], pp. 423-437
https://doi.org//10.47514/kjcs/2024.1.3.001
Online ISSN: 1597-2216
Print ISSN: 1597-2178
www.kjcs.edu.ng

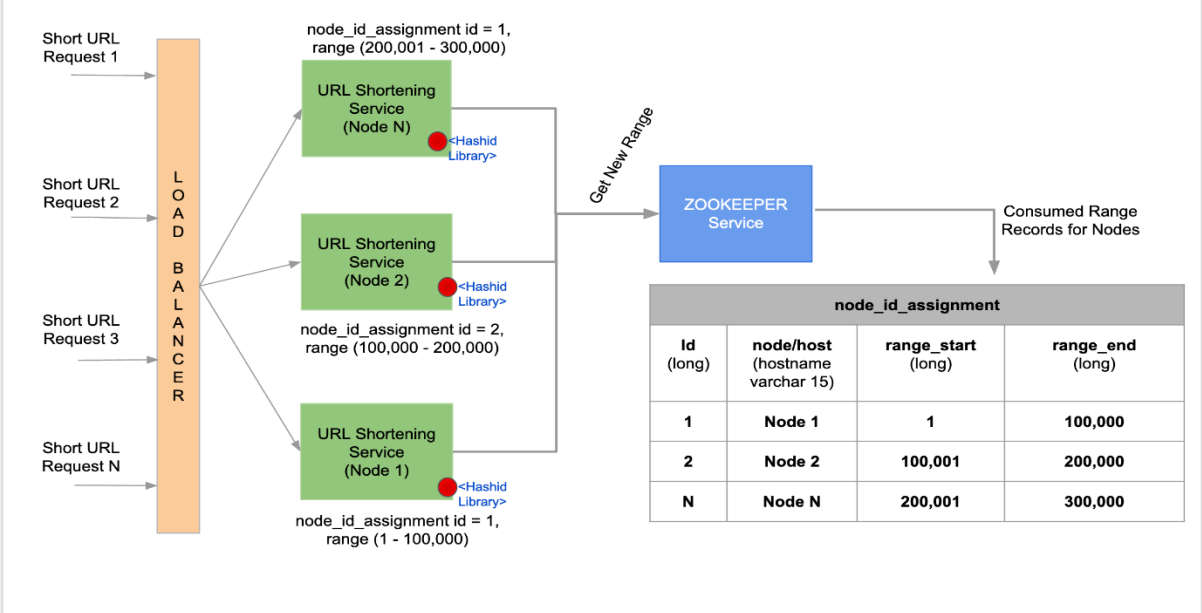

Figure 5. Base 36 Encoding, **Source:** https://dzone.com/articles/how-a-url-shortening-application-works

Here's how Base36 encoding can be applied to create short URLs:

1. Generate a Unique Identifier: Each resource, such as a web page or database record, has a unique identifier. For example, one might have a numeric database record ID like 12345.

2. Convert the Unique Identifier to Base36: Use a programming language or library that supports Base36 encoding to convert the unique identifier (e.g., 12345) to its Base36 representation. In Python, you can use the int and str functions for this purpose

```python
unique_id = 12345
short_url = base36encode(unique_id)
```

The short_url will now contain the Base36 representation of the unique identifier.

3. Construct the Short URL: Prepend the Base36-encoded identifier to your domain to create a short URL.

Kasu Journal of Computer Science Vol. 1 No. 3 [September, 2024], pp. 423-437
https://doi.org//10.47514/kjcs/2024.1.3.001
Online ISSN: 1597-2216
Print ISSN: 1597-2178
www.kjcs.edu.ng

```python
domain = "https://example.com/"
short_url = domain + short_url
```

The short_url now represents the short URL pointing to the original resource. For example, "https://example.com/3K7" if "3K7" is the Base36-encoded form of the identifier 12345.

Store and Redirect: Store the association between the short URL and the original resource in your system's database. When users access the short URL, retrieve the corresponding original resource and redirect them to it.

URL DATA

A token's length normally ranges from six to nine characters. The token's length and character count are taken into consideration while creating the URL shortening service. Smaller case characters a through z and numerals 0 through 9 are thought to provide a total character set of 36 for the security of the URL. Table 1 displays the total number of unique tokens that can be created with different token lengths. Seven characters was selected as the token length, allowing for the storage of over 78 billion distinct tokens. The token length can be set to 8 characters, or around 2 trillion, if a wider range is needed.

Table 2: URL DATA

Token Length	Unique Token Combination	Range
6	2,176,782,336	> 2 Billion
7	78,364,164,096	> 78 Billion
8	2,821,109,907,456	> 2 Trillion
9	101,559,956,668,416	> 100 Trillion

Storage Computation

20 Million (Mn) URLs are generated every day. For instances the system is designed to handle requests for the next five years. Total URLs to be stored = 20 Mn * 30 (days) * 12 *5 = *36 Bn.*

Long URL, short URL, created and modified fields will store in the data store. Every record will consume *0.5 Kb* of memory. Thus, the total storage needed will be 36 Bn * 0.5 Kb = *18 TB*

Kasu Journal of Computer Science Vol. 1 No. 3 [September, 2024], pp. 423-437
https://doi.org//10.47514/kjcs/2024.1.3.001
Online ISSN: 1597-2216
Print ISSN: 1597-2178
www.kjcs.edu.ng

Algorithm *(Base62 encoding and decoding)*

```
def idToShortURL(id):
    map = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789"
    shortURL = ""

    # for each digit find the base 62
    while(id > 0):
        shortURL += map[id % 62]
        id //= 62

    # reversing the shortURL
    return shortURL[len(shortURL): : -1]

def shortURLToId(shortURL):
    id = 0
    for i in shortURL:
        val_i = ord(i)
        if(val_i >= ord('a') and val_i <= ord('z')):
            id = id*62 + val_i - ord('a')
        elif(val_i >= ord('A') and val_i <= ord('Z')):
            id = id*62 + val_i - ord('Z') + 26
        else:
            id = id*62 + val_i - ord('0') + 52
    return id

if (__name__ == "__main__"):
    id = 12345
    shortURL = idToShortURL(id)
    print("Short URL from 12345 is : ", shortURL)
    print("ID from", shortURL, "is : ", shortURLToId(shortURL))
```

The algorithm Base62 encoding and decoding allows converting a numerical ID into a short URL and vice versa. It is a simple and effective algorithm for creating short and user-friendly URLs in web applications. It demonstrates how a numerical identifier can be transformed into a compact, alphanumeric string that is easy to share and understand. The algorithm is relatively easy to understand and implement, making it suitable for various use cases, particularly in URL shortening services. Here's a concise overview of the functioning of the algorithm:

A. idToShortURL Function:

i. This function takes a numerical ID as input and converts it into a short URL.
ii. It uses a character set defined in the "map" variable, which includes lowercase letters, uppercase letters, and digits (a total of 62 characters).
iii. The algorithm iteratively divides the ID by 62 and appends the corresponding character

Kasu Journal of Computer Science Vol. 1 No. 3 [September, 2024], pp. 423-437
https://doi.org//10.47514/kjcs/2024.1.3.001
Online ISSN: 1597-2216
Print ISSN: 1597-2178
www.kjcs.edu.ng

from the character set based on the remainder.

iv. The process continues until the ID becomes zero.

v. After generating the short URL, it reverses the order of characters in the string before returning it. Reversal is done to ensure that the most significant part of the ID maps to the first character in the short URL, providing a more uniform distribution of characters.

B. shortURLToId Function:

i. This function takes a short URL as input and converts it back into the original numerical ID.

ii. It initializes the ID as 0 and iterates through each character in the short URL.

iii. For each character, it calculates a value based on its position in the character set.

iv. The calculated value is multiplied by powers of 62 and summed to retrieve the original numerical ID.

A model known as ZooKeeper shown in Figure 6 is a powerful tool for maintaining order and coordination in distributed systems like URL shortening services. It enhances reliability, scalability, and the overall performance of such services by providing centralized configuration management, leader election, service discovery, distributed locks, and more. By utilizing ZooKeeper's services, URL shortening services can efficiently manage, scale, and coordinate their components, ensuring a seamless and reliable user experience.

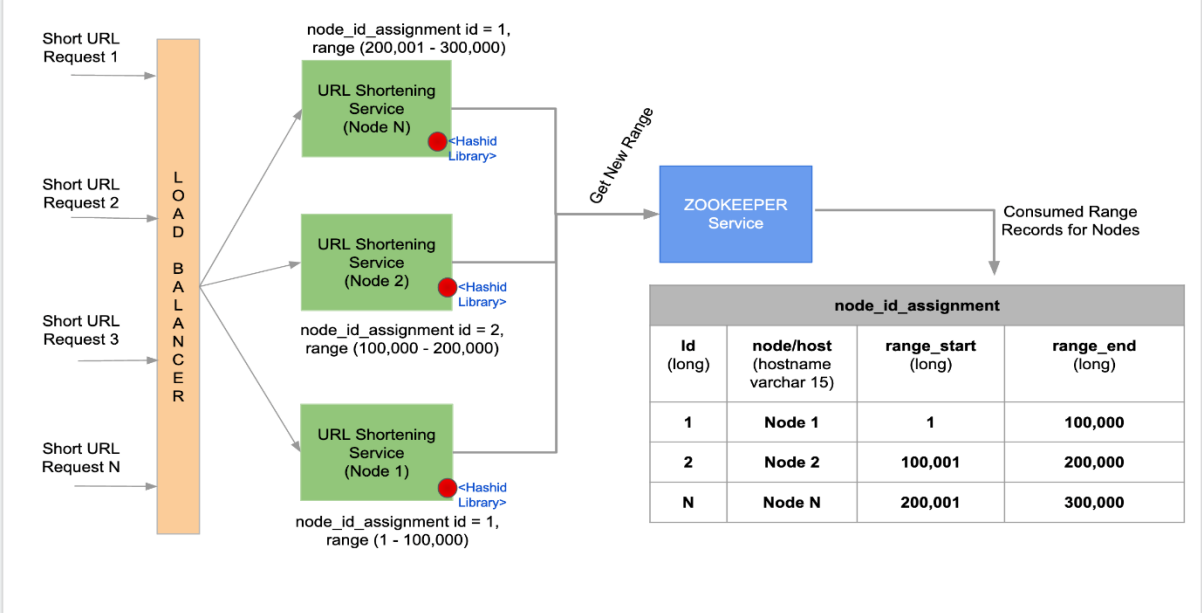

Figure 6. Final Model (ZooKeeper)**, (Source:** https://dzone.com/articles/how-a-url-shortening-application-works)

Kasu Journal of Computer Science Vol. 1 No. 3 [September, 2024], pp. 423-437
https://doi.org//10.47514/kjcs/2024.1.3.001
Online ISSN: 1597-2216
Print ISSN: 1597-2178
www.kjcs.edu.ng

4. Results and Discussions

Computer provided the link to access information located remotely in an interconnected environment. Following registration and login, a shortened link will be chosen that is linked to the registered URL. After that, a database is created and this URL and the shortened link that goes with it are entered. The log database is queried for a connected URL upon receipt of a request for a shortened link. The URL will be retrieved if the shortcut link is discovered to be connected to one; if not, an error notice will be displayed.

The application's first page is seen in Figure 7. It gives the user an interface via which they can communicate with the system and enter a lengthy URL. The created short URL may be seen in Figure 8. A modal interface displays the created link.

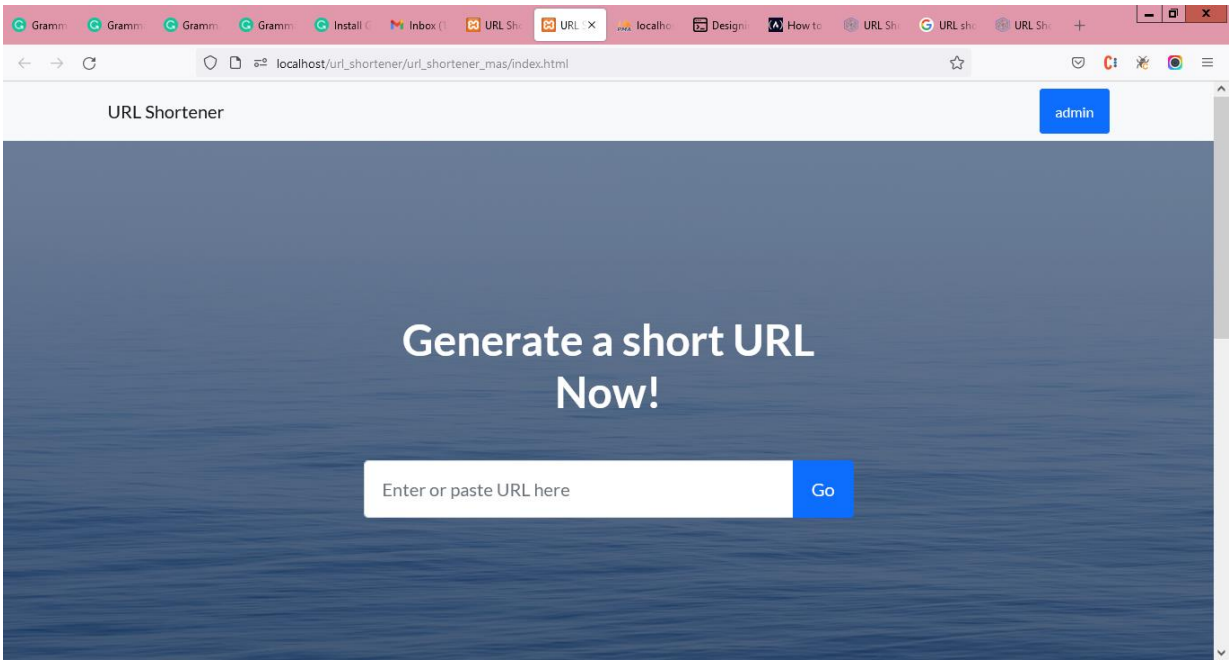

Figure 7. The image that shows the Long URL Service Page

Kasu Journal of Computer Science Vol. 1 No. 3 [September, 2024], pp. 423-437
https://doi.org//10.47514/kjcs/2024.1.3.001
Online ISSN: 1597-2216
Print ISSN: 1597-2178
www.kjcs.edu.ng

4.1 Short URL Generation Page

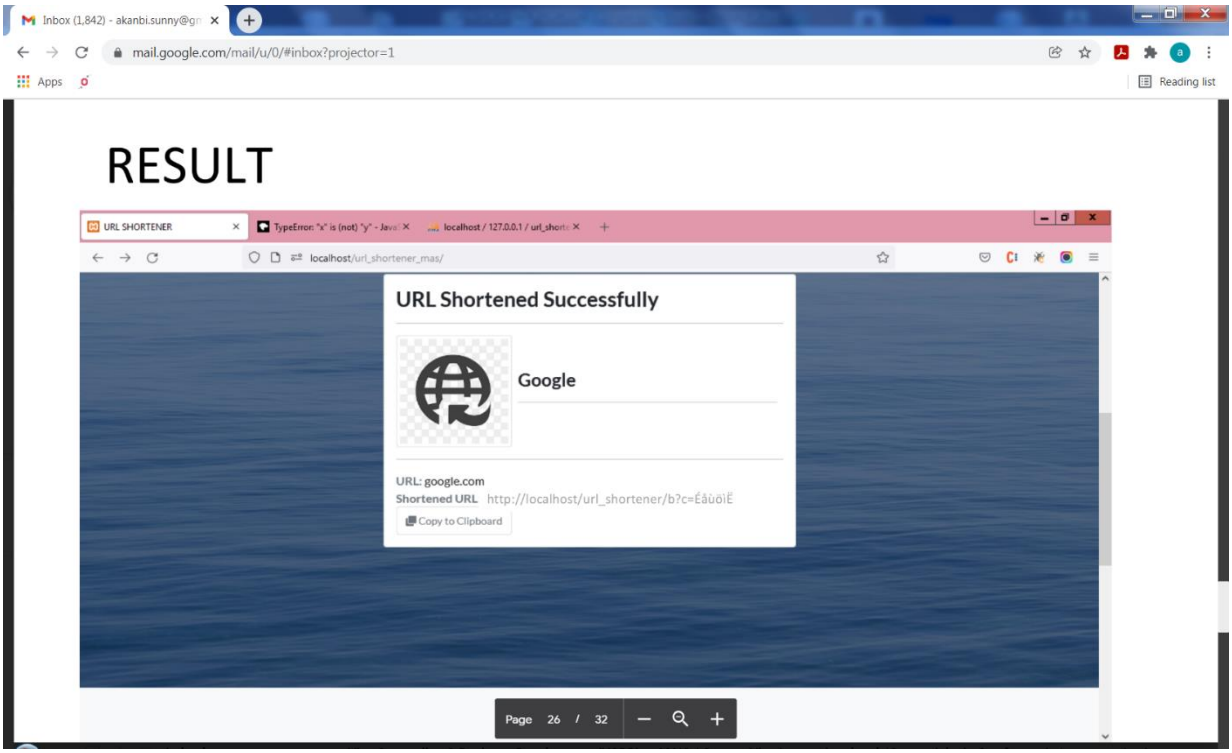

Figure 8. Screenshot displaying the generated short URL page

5. Conclusion

This study presents a novel approach to URL shortening by using accented characters as a base character set, effectively addressing limitations in flexibility, security, and user experience found in traditional methods. The proposed system generates secure and unique short URLs through a robust methodology that includes load balancing, key creation, and secure database management, with Base62 encoding ensuring unpredictable token generation. Features such as custom short link creation and automatic expiration further enhance security and usability.

The findings demonstrate that this approach significantly reduces URL-related fraud and improves the reliability and user experience of shortened URLs. The accented characters model offers a promising alternative to existing technologies, particularly where enhanced security is crucial.

Future research could explore integrating machine learning for threat detection and expanding the character set to increase the system's capacity. This study not only provides a practical solution to URL shortening challenges but also opens new directions for research and development in the field.

Online ISSN: 1597-2216
Print ISSN: 1597-2178
www.kjcs.edu.ng

References

Bell, S., & Komisarczuk, P. (2020). An analysis of phishing blacklists: Google safe browsing, OpenPhish, and PhishTank. In: Proceedings of the Australasian Computer Science Week Multiconference, Melbourne, VIC, Australia, p. 11.

Devesa, J., Cantero, X.,Alvarez, G. and G.BringasP. (2011). An efficient Security Solution for Dealing with Shortened URL Analysis. Proceedings of the 8th International Workshop on Security in Information Systems (WOSIS-2011), Science and Technology Publications, Lda.,70-79

Georgiev, M and Shmatikov, V. (2014). Gone in Six Characters: Short URLs Considered Harmful for Cloud Services. 1 – 11

Gottfried P., Utomo T. H. and Achmad B. M., (2018). "Singkat: A Keyword- Based URL Shorteners and Click Tracker Package for Django Web Application" International Journal of Advanced Computer Science and Applications (IJACSA), 9(9), 2018. http://dx.doi.org/10.14569/IJACSA.2018.090916

Gupta, N., & Kumaraguru, P. (2014). Exploration of gaps in Bitly's spam detection and relevant counter measures. *ArXiv, abs/1405.1511*.

Ismail, R. J. (2014). Proposing a Secure URL Shortening. Journal of Al Rafidain University College 209 ISSN Issue 34, 1681 – 6870.

Le-Khac, N. and Kechadi, M. T. (2015). Security Threats of URL Shortening: A User's Perspective. Journal of Advances in Computer Networks, Vol. 3(3), 213 – 219

Mun, HJ., Li, Y. Secure Short URL Generation Method that Recognizes Risk of Target URL. Wireless Pers Commun 93, 269–283 (2017). https://doi.org/10.1007/s11277-016-3866-8

Page, S. L., Jourdan, G., Bochmann, G. V., Flood, J. and Onut, I. (2018). Using URL Shorteners to Compare Phishing and Malware Attacks. IEEE

Pattewar, T., Mali, C., Kshire, S., Sadarao, M., Salunkhe, J. and Shah, M. A. (2019). Malicious Short URLs Detection: A Survey. International Research Journal of Engineering and Technology (IRJET) Vol. 6(11), 286 – 292

Pattewar, T., Wagh, M., Bisnariya, U. and Patil, L. (2018). Comparative Analysis of Malicious Detection of Short URLs from Tweets. Journal of Emerging Technologies and Innovative Research (JETIR). Vol. 5(4). 434 – 440

Prasetyadi, G., Hantoro, U., and Mutiara, A. (2018). Singkat: A Keyword-Based URL Shortener and Click Tracker Package for Django Web Application. (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 9, No. 9

Samir P. (2023). URL Shortening Service. https://samirpaul.in/posts/url-shortening-service/

Sankhala, R., Kharbanda, M., Yadav, A., Suthar, P. and Kaur, P. (2022). Sukshma - A URL Shortening Service Project. International Journal of Creative Research Thoughts (IJCRT). Vol. 10(4). 347 – 351

Shaheen, A. S and Isaac, N. M. (2022). Website Link Shortener. Journal of Global Scientific Research in Information Technology, ResearchGate. 7 (10), 2723 – 2727

Stephan, D. (2017). A sustainable URL shortener for science. Fourth Workshop on Sustainable Software for Science: Practice and Experiences (WSSSPE4), University of Manchester, Manchester, UK. Vol. 1686