IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

Sukshma - A URL Shortening Service Project

Rohit Sankhala, Manan Kharbanda, Ankit Yadav, Pardeep Suthar, Parampreet Kaur
School of Computer Science and Engineering
Lovely Professional University, Phagwara
Punjab, India

Abstract

In this paper, we have shown the importance of URL shortening at the time of sharing on various platforms. URLs appear to be long, unattractive, on most of the social platforms. They often get broken when shared on social media platforms like e-mail and short URLs can come in handy during these times. To shorten an Internet Long URL, we proposed a URL shortening service that will take a long Web URL/address and creates a shorter URL that will not break when we share on different platforms and make them more manageable by using flask framework base62 algorithm. In this paper, we have also provided a way that helps during digital marketing, social campaigning, and posting by giving the stats of the number of clicks made to the shortened URL as it shows the number of people interested in clicking the URL.

Keywords: *URL Shortening, Data Flow Architecture, SQLite database engine, URL Redirection, Tracking*

1. Introduction

URL shortening is a very common and widely used process nowadays. With the presence of social media networks where the text is required to fit in a restricted number of characters usually to 140 characters, therefore shortening of URLs became more and more crucial. So, to shorten the long URLs, a greater number of URL shortening services are being used [1]. When somebody needs to shorten their unsightly long URL they go to a URL shortening service and submit the long URL intended to get shortened [2]. These shortened URLs are not typically longer than 30 to 40 characters. These shortened URLs are then used instead of the long URLs to visit the respective websites.

In this paper, we have shown how we provide a custom URL shortening service. A custom URL shortener is connecting our custom domain to a URL shortener which acts as the foundation for all the short links that we create. Instead of a generic domain, we are combining the shortened one with our custom domain [3]. This custom URL shortening service is recommended because through the custom domain while sharing the links online, it leads to increased trust in the clients for the link, greater awareness to the brand, and increases click-through rate. For example, a person may have the following complex links, which is not understandable and looks ugly such as: https://privatebanking.mybank.com/privatebanking/ebankver2/secure/customersupport.aspx?messageID=3324341&Sess=asp04&passwordvalidate=true&changepassword=true
The shortened URL might look like this: http://domain.com/url/a23bcd7

The proposed URL shortening service will take the long URL link and gives a shorter link that will not break while sharing on social media platforms and also provides with the custom domain and by this we aim at user trust to use the service with availability.

2. Literature Review

In this section we have provided a brief overview of different security standards applicable to URL shortening services researched and mentioned under security threats of URL shortening in [4-5]. As the focus is more on shortening of URLs we will be referring the security standards as suitable to the research. The studies and research shows that the security is not only meant for the shortened URL but also to the users using the service. Many times we forget the question that, are people really interested in using these shortened URLs, if they are not what might be the reason behind it. When analyzed, it shows that these shortened URLs tend to be malicious, and cause spams, phishing etc. People tend to not trust the services which are malicious and not authorized for usage. Sharing the same kind of URLs, ending up on a different and irrelevant sites on redirection when the URL is clicked, and blocking the usage are few among the various security threats which result in data breach and thus resulting in issues [6].

Various studies and analyses of different URL shortening websites show that the shortening of URL is done and aimed at only reducing the amount of space that they take up when written or while sharing, but not at the URL that the user submits to shorten or at the amount of time taken to redirect to the original address. It shows that the URL submitted for shortening might be malicious and might go unrecognized while shortening and thus resulting in sharing of the same malicious one to many more other users causing inconvenience and resulting in the less usage of the service and reducing trust among users. In our proposed URL shortening service we are trying to provide a way such that it checks the users who should prove their identity before submitting any kind of URLs and in an indirect way checking the authenticity of the URLs. We also try to provide a customized domain which shows the user name in the shortened URL because many markets depend on brand value and loyal customers which means when a person sees the URL, if he recognizes the domain and has a trust on clicking the URL, he tends the use the service more often.

The main aspect here is to show that there has to be some kind of awareness among the users for trusting the application and increasing the usage of URL shortening services. The current state of the above overview shows that we have to deal with security problems from the point of view of the normal users. Using the existing knowledge of authenticating the users and various hashing techniques we tried to propose a service that includes the customized domain thus aiming at user trust and also trying to improve the dependability of both secure shortened URL and secure user for the future scope and analysis. In this section, we have discussed how the proposed URL shortening service directs the research to various fields such as transformation into social media services, insights or tracking, professionalism. These fields will be hugely benefitted because of the usage of the URL shortening service.

2.1 Transformation into social media services

From the URL Shortening services we can gather huge amount of data about how many people are really interested in clicking them and we can also whether the clicks are made by any automated machine or by human himself. The way these services are used can enhanced and be transformed into social media service for campaigning purposes.URL shorteners, in their way work as clusters of information. This can lead to some useful innovations in sharing links.

2.2 Tracking

The main benefit that we are aiming through this application is link tracking. By tracking the clicks we can tell which post got more attention and which region is interested in that particular information. By using this link tracking, the digital marketers can benefit in efficient way and by targeting the users at areas where the response is huge. This type of information can be gathered through this application.

2.3 Expertness

When the clients are getting used to the content shared in the form of shortened links, they start feeling more comfortable to share the links and might not use or click the URL, if we share a URL that hasn't been compressed. Long-form URLs can appear ugly and appear cleaner when shortened thus increasing the usage of more URL shortening services.

3. Proposed Framework for URL Shortener

The frameworks required are data flow architecture which shows the flow of data from one entity to another, flask which is used for the web development, SQLite database engine for the database operations. These frameworks complete the interface and operation of the proposed service.

3.1 Architecture for the proposed URL Shortening Service

The architecture that we have used for the URL shortening service is data flow architecture. In data flow architecture, the data and operations are independent of each other and there are transformations of data among components. In this architecture, the data enters the system and flows through the components and finally we get some kind of output. The main objective to use this architecture is to achieve reusability and modifiability. We use the pipe and filter architecture that comes under data flow architecture. This architecture provides the incremental transformation of data by successive components. Here, the system is divided into components such as, filters, pipes, etc. The filters process the data of one form to another and pipes move a data stream from one filter to another. This provides concurrency and high throughput, flexibility, and supports both sequential and parallel execution.

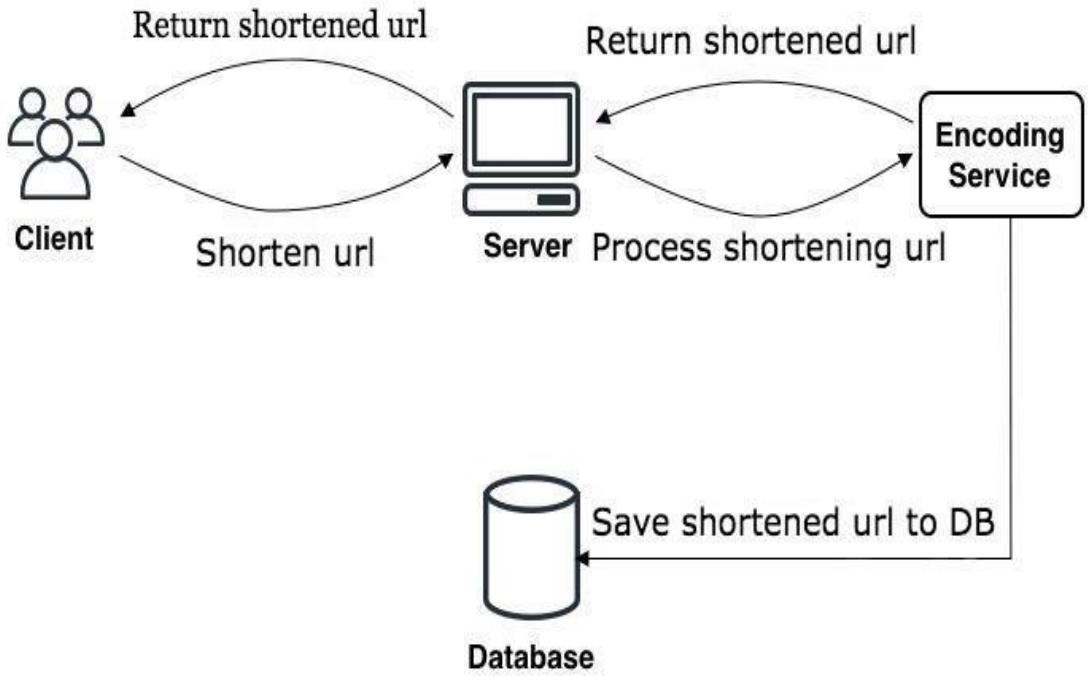

Fig.1. Architecture of URL shortening service

Figure 1 is showing how the data flows from one entity to another while carrying out the operations. The pipes are arrow marks and filters are entities like server, client, etc.

3.2 Flask Framework for web development

Flask is an API of Python, we used this framework for building the URL shortening application. It was developed by Armin Ronacher. As this framework is easier to learn and also has a collection of modules and libraries. This framework is based on WSGI (Web Server Gateway Interface) and Jinja2 . There is no need of implementing the low level code here. We have used Werkzeug and WSGI library. This works along with flask and it doesn't enforce any dependencies. It is up to us, the developers to choose the handling of requests, template engine and other necessary works. The flask framework looks for HTML

files and these HTML files should be stored in a folder called templates. When python code is executed the variables, code, etc are loaded, before sending the template over. The Flask framework is a lightweight micro-framework. Thus it is more than suitable for creating this kind of web application.

3.3 SQLite database engine for the data related operations

SQLite database engine has many advantages for application which rely on server-less configuration. It gives the implementation of auto-configuration. Whatever the tables, views, triggers that we create using the application are stored in the ordinary disk files. This engine does not need any separate server process as it is an embedded SQL database engine. As there is no setup procedure and no configurations regarding the server protocols, there is also no need of prior instalment procedure. The programs that are execute or the processes that run on the device directly interact with the disk files for reading and writing operations. The data will be stored and can be retrieved easily even if there is a power failure. The main advantage is that any program that is able to access the disk on the computer can use this SQLite database. No separate server process even for troubleshooting.

4. Result Analysis

When a user is registered with a URL shortening service, he can keep track of the stats of the shortened URLs that he has created using this application. The service provides the user with the information of the date and time of the creation of the short URL and also the number of clicks made to that particular short URL. When the user visits the application to check the stats of created short URLs he visits the dashboard, and he can also edit the shortened URL if the lifetime of that URL is expired. Through the stats, the user can check how many visitors are coming to their site and can get a daily breakdown. This tracking of the clicks made to the URLs is for digital marketers for tracking large campaigns.

Table 1 Capacity estimations and constraints.

Constraint	Estimation
Incoming URLs	100/s
The URL Redirections to be made	10K/s
Incoming Data to the Service	50KB/s
Outgoing request	5MB/s
Memory for storing frequent URLs	85GB

In table 1 we have shown the expected capacity estimations and constraints data which show the required elements necessary for the maintenance of the URL shortening service and its related data to maintain smooth user experience. Since we are planning to store a huge number of URLs.

Conclusion

In this paper, we have provided a brief overview of how the URL shortening service works. The proposed shortening has features such as, Putting the domain name in the shortened URL that gives the user trust to use the service, providing the user with the option of editing the URL when needed, maintaining the record of shortened URLs and their data. The application also provides authentication of the user by hashing the password and confirming it with the one stored in the database, thus maintaining the application secure from the people who are not registered. The advantages of using a URL shortening service includes attractiveness to the URL which gives the user a clutter-free view of the content while seeing any webpage containing the short URL, ease of sharing on different platforms, and avoiding the problem of breakage of URLs when shared traditionally, ease of maintaining as we are having a shorter version of URL, etc. The service also comes with few disadvantages like privacy breach, dead links that is they become inaccessible after some time, but here in our application, we are giving the user a way to

check the creation details of the URL and also allowing him to edit it when needed. We can get the best of the service when used properly and ethically.

ACKNOWLEDGMENT

The completion of this undertaking could not have been possible without the participation and assistance of our team. Their contributions are sincerely appreciated and gratefully acknowledged. However, the group would like to express their deep appreciation and indebtedness particularly to the following:

Manan Kharbanda, Rohit Sankhala, Ankit Yadav for their endless support, kind and understanding spirit during our case presentation

To all members of our group who in one way or another shared their support, either morally, financially and physically, thank you. Above all, to the Great Almighty, the author of knowledge and wisdom.

References

[1] Banker, K. (2012). *MongoDB in Action.* Shelter Island, New York, United States: Manning Publications.

[2] Bearnes, B. (2016, May 12). How To Set Up a Node.js Application for Production on Ubuntu 16.04. *How To Set Up a Node.js Application for Production on Ubuntu 16.04*, p. 13.

[3]Brown, E. (2019). *Web Development with Node & Express Leveraging the JavaScript Stack.* O'Reilly Media.

[4]Choudhary, H. (2021). Hitesh Choudary JavaScript Youtube. *Hitesh Choudhary.* Jaipur, India: LearnCodeOnline.in.

[5]Reem, I. (2014). Proposing a Secure URL Shortening Service by using Blackboard Architecture. *JRUCS.*

[6]Wieruch, R. (November 15, 2017). *The Road to React: Your journey to master React.js in JavaScript (2021 Edition).* Berlin, German: Independently Published.