REPORT

ON

URL SHORTENER WEB APPLICATION

Submitted by: - Dipankar Medhi

To: - Innomatics Research Labs

Introduction

Sharing urls with friends, family, colleagues, etc. is part of the daily life of a modern human being. We share all kinds of links, from movies to important research materials. And we want the sharing process to be efficient, easy and reliable. But some urls are way longer than the usual ones, and sharing these urls are very tedious and untidy.

So, to overcome this issue, we have URL SHORTNERs. Their job is to turn these long and ugly urls into short and lovely user-friendly urls.

Some of the common and popular URL SHORTENS are: -

- Bitly
- TinyURL
- Ow.ly
- Rebrandly

Although we have such cool and awesome URL SHORTNERs, coding our own Url Shortener from scratch and working on its functionality is the main content and purpose of this report. This report contains all the necessary information about Tech Stack, Functions, Libraries, etc. used while developing the Url Shortener.

Tech Stack

Frontend: -

- HTML
- CSS

Backend: -

- Flask
- SQLAlchemy - database

Project

The project starts with the creating of the skeleton i.e. the HTML layout. For generating the HTML pages using python, the JINJA template is being used.

Jinja is a modern and designer-friendly templating language for Python, modelled after Django's templates. It is fast, widely used and secure with the optional sandboxed template execution environment.

With the layout in place, it is time to implement the backend to route and view the web pages in the browser. For the backend, the **Flask** library is being used. Flask depends on the Jinja template engine.

For storing the URL data, **SQLAlchemy** is used. SQLAlchemy is the Python SQL toolkit and Object Relational Mapper that gives application developers the full power and flexibility of SQL. It provides a full suite of well-known enterprise-level persistence patterns, designed for efficient and high-performing database access, adapted into a simple and Pythonic domain language.

Files: -

- An **app.py** is created, where all the backend related functions are written for routing and storing the URLs in the table inside the database.
- Then a **template** folder is created, where all the HTML files are kept.
- HTML files include **layout.html, index.html** and **history.html.**
- **layout.html** includes the layout of the webpage which is then used in the index and history page using JINJA templating.
- All the routing and database related logic are defined inside the app.py file.

Main Logic behind the shortener app: -

The main job of the application is to convert long urls into shorter urls. And these automatically generated shorter urls must be unique. So, for making the shorter urls unique, **random** python library is used. Using the random library, a random set of character is generated containing letters from a- z and A – Z.

If the generated short url is not present in the database, the shorten function returns the url and the same is displayed in the browser window. From there, a user can copy that shortened url using the copy button and use it as per their needs.

Database: -

The database schema contains **id**, **original_url** and **shorten_url**.

Functionality: -

User types/pastes their long url inside the given input field in the web application and click on the **shorten** button to run the shortening function and output the shortened url in the browser window itself. To copy the shortened url, a user needs to click on the **copy** button. And upon clicking the copy button, an **alert** dialogue box appears stating the completion of coping the shortened text.