URL SHORTNER APPLICATION

Project Description

This project's objective was to create a URL shortener with Flask and SQLite. A programme known as a URL shortener takes a long URL and creates a shorter, one-of-a-kind URL that, when clicked, leads to the long URL in question.

A home page, a history page, and a result page are the three main pages of the project. The user inputs the lengthy URL they wish to abbreviate on the home page. All of the shortened URLs are listed in a table on the history page. The shortened URL that was created for the long URL entered is shown on the result page.

Technologies Used

- Flask - a Python web framework
- SQLite - a lightweight relational database management system
- Bootstrap - a CSS framework for creating responsive web pages
- Jinja - a templating language for Python

The URL shortener application has been developed using Flask and Python programming language. Flask is a micro web framework that is used to develop web applications in Python. Flask provides a simple and easy-to-use interface to develop web applications.

Bootstrap has been used to style the application's pages. Bootstrap is a popular CSS framework that provides pre-designed CSS classes to develop responsive web applications.

Application Flow

The script app.py is executed to launch the programme. The Flask application is configured and the SQLite database is connected using this script.

The user is offered with a field where they can input a lengthy URL when they go to the home page. The process view function is triggered by the user's submission of the form. This function determines whether the lengthy URL is present in the database already. If it does, the user is shown the relevant short URL on the result page after retrieving it. A new short URL is generated and

stored in the database if the long URL is not already there. The user is then shown the updated short URL on the result page.

The person can view their history.

Functionality in Project

Shortening a URL, routing the abbreviated URL back to its original URL, and presenting a list of shortened URLs are the three major features of the URL shortener programme. The application uses the built-in Flask routines to create a shorter URL when a user submits a long URL in the input field. The original URL and the shortened URL are then both shown on the result page.

The programme leads users to the original Website when they click on the truncated URL. To send the user back to the original URL, the application makes use of Flask's routing functionality.

A list of all the URLs that have been abbreviated is shown on the application's history page. The original URL, the abbreviated URL, and the time and date that the URL was shortened are all displayed on this page.

Files and their Functions

The following files and functions make up the project:

app.py: The primary Flask application is contained in this file. It carries out the following duties:

- **Process():** This function handles the home page's form data. If the long URL is not already in the database, it generates a short URL and enters the new URL. Following that, the short URL that was generated renders the result page.
- **history():** This function renders the history page with a table of all the URLs after retrieving them all from the database.
- **redirect_url(short_url)**: This function is called when a short URL is clicked on the history page. It retrieves the corresponding long URL from the database and redirects the user to that URL.

The HTML code for the home page is in the file named **home.html**. It has a form where a long URL can be entered to be shortened.

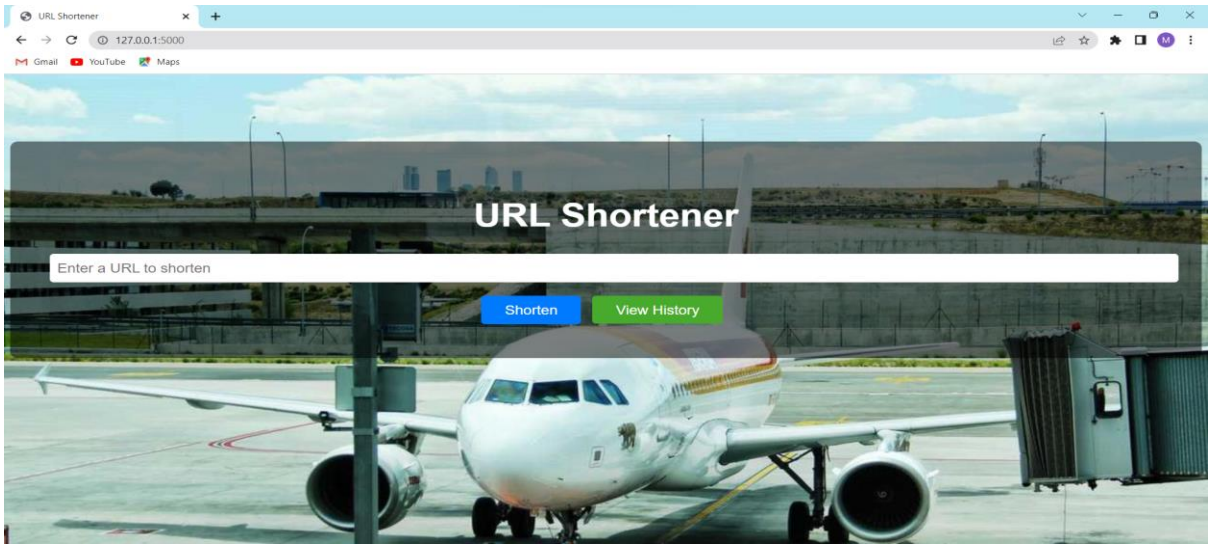
The HTML code for the result page is in the file named **result.html**. It shows the user the short URL that was generated.

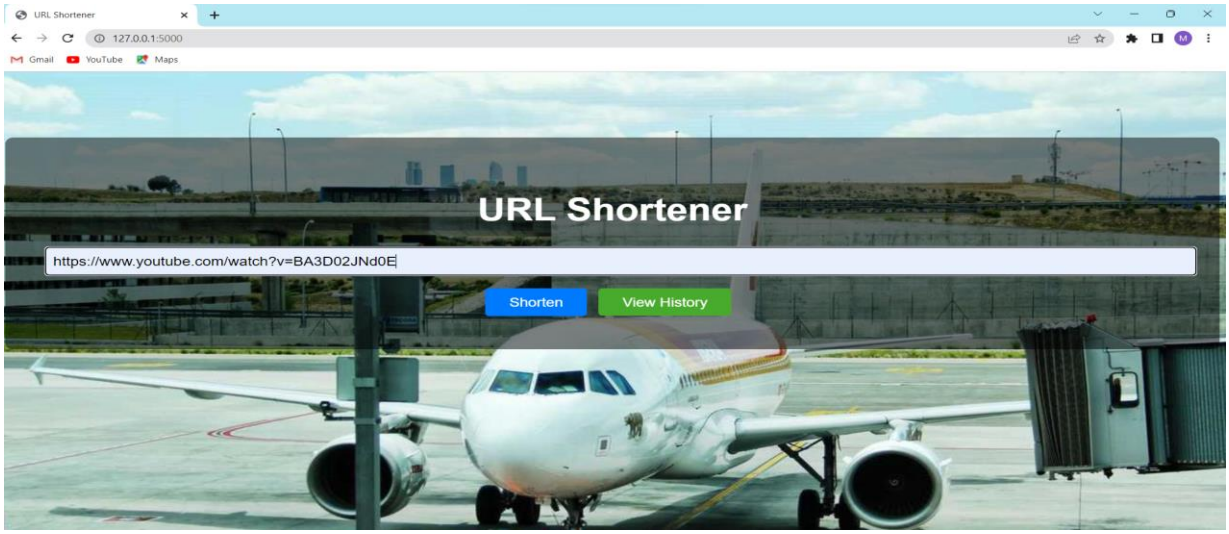
The HTML code for the history page is included in the file **history.html**. It shows a table of all the shortened URLs along with links to their full counterparts.

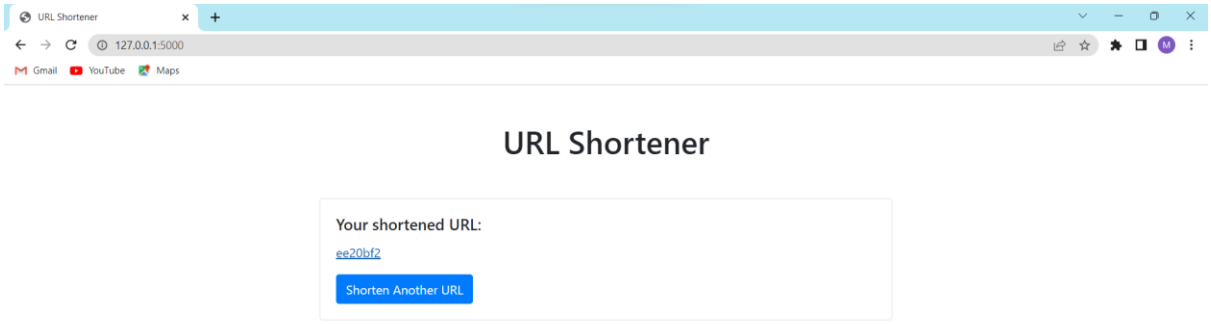
Further CSS styling for the application may be found in the file **style.css**.

Implementation & Output of Project

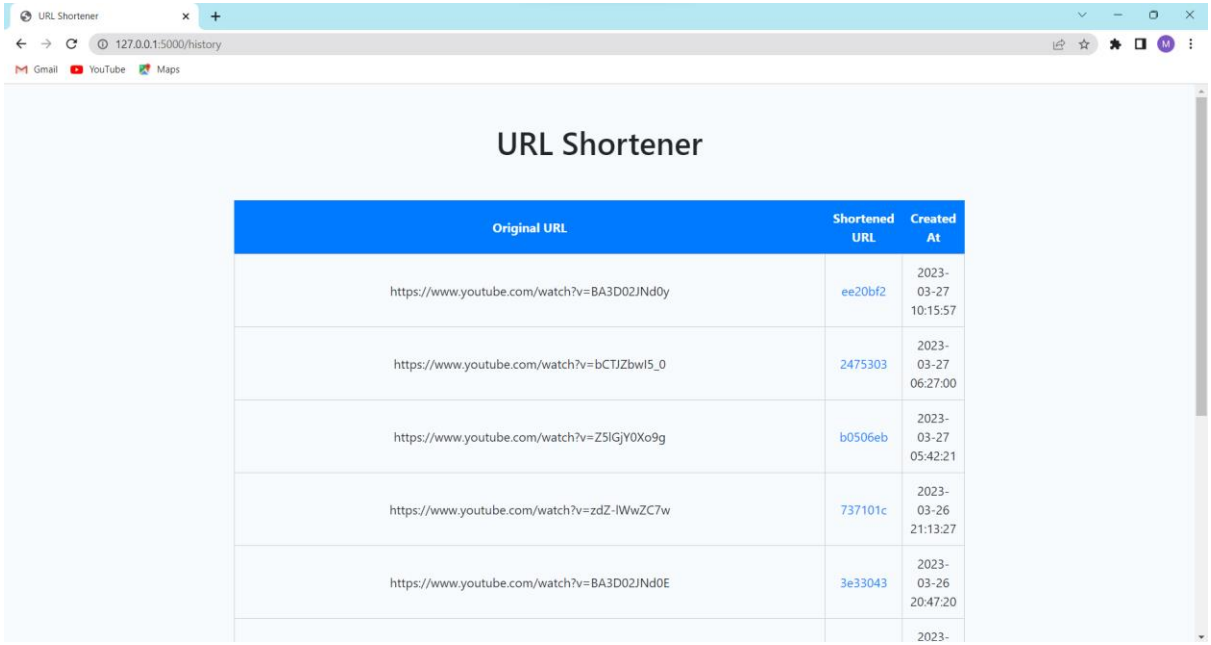
The routing system in Flask has been used to construct the URL shortener application. The home, result, and history buttons are the application's three primary pathways. The long URL can be entered into a form on the home route, which displays the home.html template. The result route creates a shorter URL after the form is submitted and presents the result.html template, which shows both the long URL and the short URL. The result route also updates the database with the long URL and the shortened URL.







The history route generates a list of all the shortened URLs using the history.html template. The history route searches the database and retrieves all of the abbreviated URLs.



SQLite is the database utilised in the project. As it doesn't need a separate server, SQLite is a compact database engine. It is ideal for little web applications that need for a straightforward database engine.

CONCLUSION

As a result, we have created a URL shortener web application utilising Flask and SQLite, accomplishing our goal of giving consumers a quick and easy way to shorten and manage URLs. Shortening URLs, rerouting abbreviated URLs to their original source, and showing a history of previously shortened URLs are the three key features of the application. We were able to develop a quick-loading, responsive application that is simple to use and maintain thanks to the use of Flask and SQLite. The user interface was improved and the user experience was flawless because to the integration of Bootstrap and Jinja. This project is a great illustration of how to use Flask and SQLite to create robust and efficient web apps in Python.

THANKYOU

MADAMANCHI CHANDRA VARDHAN