URL shortener project (Basic)

Presented and Developed By Vamsi Tallapudi

Objective

The aim of this project is to create a web application that shortens long URLs without losing their functionality. The application has two main sections, the frontend, which uses HTML and CSS, and the backend, which uses Flask (Python) and SQLAlchemy Database ORM. To develop the application, we used a code editor called VScode and ensured that all the necessary requirements were met to run the Flask app successfully. The goal is to make it easy and less annoying for users to copy and share URLs.

Prerequisites

1. Python: The project is written in Python programming language. Therefore, you need to have Python installed on your system. You can download the latest version of Python from the official website https://www.python.org/.
2. Flask: The project is built using the Flask web framework. You can install Flask using the pip package manager by running the command **pip install flask**.
3. SQLAlchemy: The project uses SQLAlchemy as an Object-Relational Mapping (ORM) tool. You can install it using the pip package manager by running the command **pip install flask_sqlalchemy**.
4. Flask-Migrate: The project uses Flask-Migrate to handle database migrations. You can install it using the pip package manager by running the command **pip install flask_migrate**.
5. Validators: The project uses validators to validate URLs. You can install it using the pip package manager by running the command **pip install validators**.
6. SQLite: The project uses SQLite as a database management system. SQLite comes pre-installed with most of the operating systems.

Routes

The application has four routes:

The Home route (/) is the landing page for the application. It serves an HTML form that allows the user to input a URL to be shortened. If the form is submitted, the application checks if the URL already exists in the database. If it does, the existing short URL is retrieved and displayed. If not, a new short URL is generated and added to the database.

The History route (/history) displays a list of all previously shortened URLs, along with their corresponding long URLs.

The redirection route (/<finalurl>) is used to redirect short URLs to their corresponding long URLs. When a user clicks on a short URL, the application retrieves the corresponding long URL from the database and redirects the user to that URL.

The delete route (/delete/<int:id>) allows the user to delete a previously shortened URL from the database. When the user clicks on the "Delete" button next to a URL in the History page, the corresponding record is deleted from the database.

Html templates that are used for rendering

The home page ("/"): This page allows users to shorten a URL. It accepts a URL through a form submission and generates a new short URL for it. The user can also view the history of the shortened URLs.

```
{% extends 'Base.html' %}

{% block title %}URL Shortener{% endblock %}

{% block content %}
    <form method="post">
        <label for="url" style="color: #555; font-weight: bold;">Enter your URL:</label>
        <input type="text" name="url" id="url" placeholder="https://Demosite.com" required>
        <input type="submit" value="Shorten">

    </form>


    {% if error == "not valid" %}
        <p class="error">Please enter a valid URL</p>
    {% endif %}
    {% if finalurl %}
        <p>Shortened URL is:</p>
        <p class="short-url"><a href="{{ url_for('redirection', finalurl=finalurl) }}">{{ url_for('redirection', finalurl=finalurl, _external=True) }}</a></p>
    {% endif %}


{% endblock %}
```

The history page ("/history"): This page displays the history of all shortened URLs that have been generated.

```
{% extends 'Base.html' %}

{% block title %}URL Shortener History{% endblock %}

{% block content %}
    <h2 style="text-align: center;"> History of URL Shortener </h2>
    {% if saved %}
        <table>
            <thead>
                <tr>

                    <col style="width:50%">
                    <th >Long URL</th>
                    <th>Short URL</th>
                    <th>Fun( )</th>

                </tr>
            </thead>
            <tbody>
                {% for i in saved %}
                    <tr>
                        <td><a href="{{ i.long_url }}">{{ i.long_url }}</a></td>
                        <td><a href="{{ url_for('redirection', finalurl=i.short_url) }}">{{ url_for('redirection', finalurl=i.short_url, _external=True) }}</a></td>
                        <td><a href="{{ url_for('delete', id=i.id) }}" class="delete-link">Delete</a></td>

                    </tr>
                {% endfor %}
            </tbody>
        </table>
    {% else %}
        <br> <br> <br>
        <p style="text-align: center;color: brown;"><b>URLs are yet to be shortened.</p>
    {% endif %}

{% endblock %}
```

The redirection page ("/<finalurl>"): This page takes a shortened URL as input and redirects the user to the original URL associated with it. If the shortened URL does not exist in the database, it displays an error message.

Frontend and Backend operations

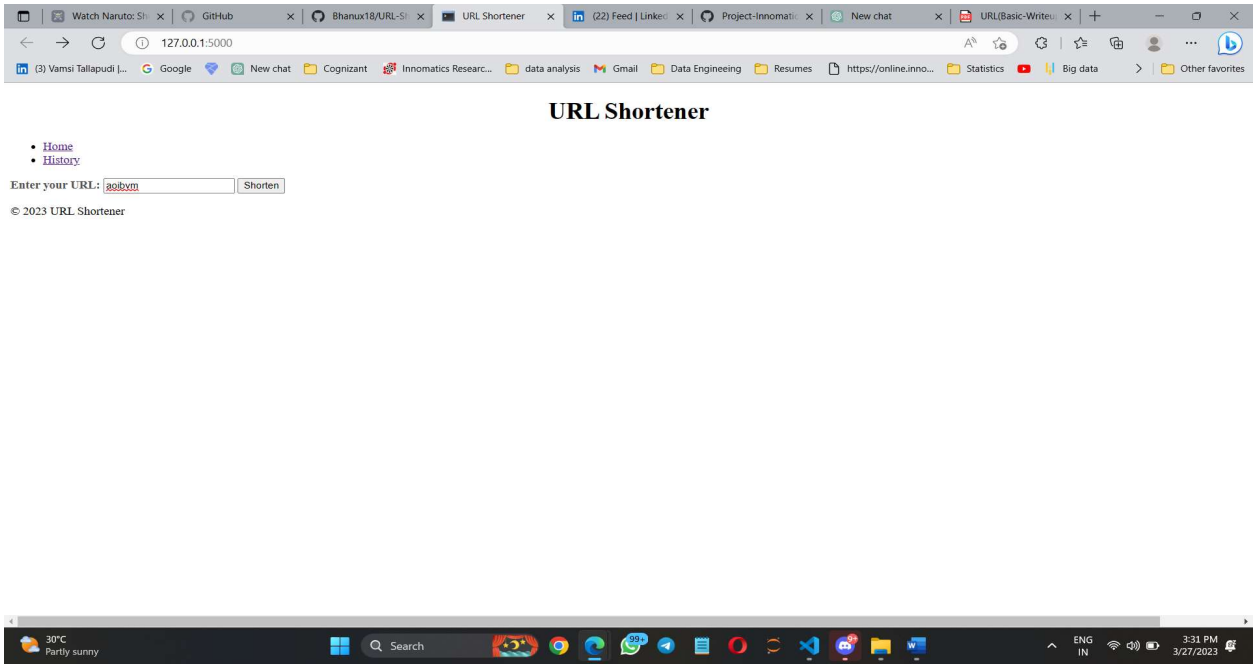
Frontend and backend operations refer to different aspects of web development.

The frontend, also known as the client-side, is the part of the website that users can see and interact with directly. It involves the creation and design of the user interface, such as the layout, buttons, forms, and graphics. This is done using programming languages such as HTML, CSS, and JavaScript, and is executed by the user's web browser.

The backend, also known as the server-side, is the part of the website that users cannot see, but which performs the behind-the-scenes operations necessary for the website to function. It involves the creation of the server-side logic, such as the database management, server configuration, and other server-side scripts. This is done using programming languages such as PHP, Python, Ruby, and Java, and is executed by the web server.

In the code provided, the frontend operations involve rendering the HTML templates for the home page, history page, and error page, as well as handling user input from the form on the home page using Flask. The backend operations involve setting up the database using SQLAlchemy, defining the Url model, and performing database operations such as adding new URLs, querying the database for existing URLs, and deleting URLs from the database.

Relevant screen shots for further use



URL Shortener

Home History

History of URL Shortener

Long URL	Short URL	Fun( )
https://en.wikipedia.org/wiki/Timeline_of_chemical_element_discoveries	http://127.0.0.1:5000/GKv3D	Delete
https://www.easybiologyclass.com/biostatistics-free-lecture-notes-online-tutorials-ppts-and-mcqs/	http://127.0.0.1:5000/3GvTc	Delete