Project: URL Shortener

By- Kishore Chandra U, Kurumurthy P

Problem Statement: Lengthy URLs are very difficult to handle while sharing the URL with multiple users. So shortening them and redirecting them to the original url helps us with easy handling and with short and crisp communication. So we need to **build an application that shortens the given URL.**

Description of the project:

1. The web page(home page) should be loaded with input and a button, which takes the input of the URL.
2. The shortened version of the URL should be reflected on the same page and it should be copyable to the clipboard.
3. The list of Original URLs and the Shortened URLs should be stored in the table format in a database.
4. The copied shortened url should be able to redirect to the original url and it's respective page.

Steps:

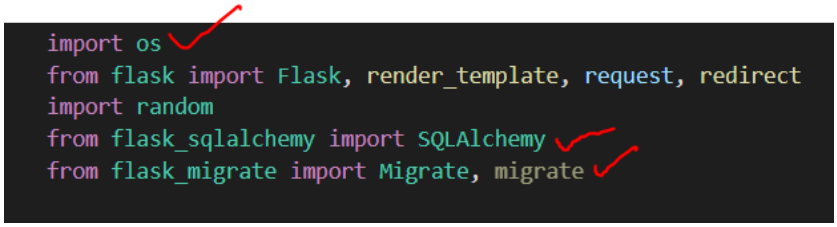
1. To create a database that can store the Original URLs and the shortened URL in a table.
2. To build a web page and the corresponding backend flask model that can take the URL as input and result in a shortened url with a copy button and stores the URLs history.
3. To build a web page that displays the search history of the URLs and the corresponding shortened URLs
4. To make the application capable of redirecting the shortened URL to the original URL entered by the user.

Procedure:

Step#1

To create a database that can store the Original URLs and the shortened URL in a table.

a. Import the required libraries to set up the database system within the server computer.



```
import os
from flask import Flask, render_template, request, redirect
import random
from flask_sqlalchemy import SQLAlchemy
from flask_migrate import Migrate, migrate
```

Project: URL Shortener

b. Configure SQL alchemy

```
app = Flask(__name__)
#SQL Alchemy Configuration
basedir = os.path.abspath(os.path.dirname(__file__))
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///'+ os.path.join(basedir,'data.sqlite')

app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False

db = SQLAlchemy(app)
Migrate(app, db)
```

c. Database Model Creation: A database of table type that has column names id which is also a primary key, original and Shortened columns.
The database can take the inputs of two arguments which can be stored in a row and also creates an id for the corresponding row.
On calling Url Shortener can result in a database table which can be added, deleted and updated with new rows of URL data..

```
#Model Creation
class UrlShortner(db.Model):
    __tablename__ ="URL_Table"
    id = db.Column(db.Integer, primary_key = True)
    original = db.Column(db.Text)
    Shortened = db.Column(db.Text)

    def __init__(self, original, Shortened):
        self.original = original
        self.Shortened = Shortened
```

Step -2

To build a web page and the corresponding backend flask model that can take the URL as input and result in a shortened url with a copy button and stores the URLs history.

1. **Import the required libraries to set up the application**

Project: URL Shortener

```
from flask import Flask,render_template,request
import random
from flask_sqlalchemy import SQLAlchemy
from flask_migrate import Migrate
import os
from werkzeug.utils import redirect

app =Flask(__name__)
```

2. creating routes:

As soon as the user enters the localhost server url with port, in a browser ,the request goes to the application. In the application the GET route will respond and render the html file which is created in the templates.

```
@app.route('/')
def fun_1():
    return render_template("home.html")
```

3.Creating POST request

When the user enters the url and clicks on shorten button →

It sends the post request→

```
<form method='post'>
<input type="URL" name="in_1" placeholder="Enter original URL">
<button>Shorten</button>
</form>
```

If the request is POST , it takes the user to the web form which is created in the html file and the user needs to enter the required original URL(the input field takes only URL type of data) in the web form.

```
<form method='post'>
<input type="URL" name="in_1" placeholder="Enter original URL">
<button>Shorten</button>
</form>
```

4.shortening the original URL:

As soon as the user clicks the shorten button, home.post() function creates a random alphanumeric string of length 4.

Project: URL Shortener

This function verifies if the url exists within the database, if it doesn't exist it adds both the original url and shortened url to the database.

```
@app.route('/', methods=['POST'])
def home_post():
    original_url = request.form.get('in_1')

    if original_url != None:
        if original_url !="":
            short_url = random.choice(dummylist) + random.choice(dummylist) +random.choice(dummylis
            k = URLS.query.filter_by(original_url = original_url)
            for i in k:
                if i.original_url ==original_url:
                    short_url= i.shorten_url
                    return render_template('home.html', data=short_url)
```

5.copy the shorten URL:

The 4 letter alphanumeric string is sent to home.html and results in a new url and it is allowed to be copied through clipboard.

```
<form>
    <input value= "127.0.0.1:5000/sh/{{data}}" id="myInput">
    <button onclick="myFunction()"> Copy </button>
</form>
```

Step -3

1.creating history table of the original and shorten URLs

In the database the id ,original and shortened urls are stored in the table.

To create all the original and shortened urls in the table, we have to create the rows that contain ids ,original and shortened urls. The data should be stored in the table as soon as the user copied the shortened url.

```
            new_row = URLS(original_url, short_url)
            db.session.add(new_row)
            db.session.commit()
            return render_template('home.html', data=short_url)
    return render_template('home.html')
```

Project: URL Shortener

```
@app.route('/history')
def fun_history():
        url_list = URLS.query.all()
        return render_template("history.html",data = url_list)
```

2.dynamic searching

When user search for the website using shortened url, we use dynamic routing technique that redirects the original URL.

```
@app.route('/sh/<short>')
def fun(short):
    url_list1 = URLS.query.filter_by(shorten_url = short)
    for i in url_list1:
        if (i.shorten_url) == short:
            return redirect(i.original_url)
        else:
            return "incorrect URL"
```

`

-------- 0 --------