FLASK WEB APPLICATION FINAL PROJECT

URL Shortener app - ShortiGO

Overview:

The flask URL Shortener application allows users to input a long URL and generate a shortener version of the URL that can be shared easily. The application stores the long and short URLs in a SQLite database, and users can view a list of all stored URLs or delete specific URLs.

The goals of this project were to:

- Develop a web application using flask
- Implement URL shortening functionality
- Store URLs in a database using SQLAlchemy

The application uses Flask, SQLAlchemy and Flask-migrate, URLs are stored in a SQLite database, and the application generates a random 3- letter string for each short URL. User can access the application through the web browser, and the application.

DEVELOPMENT PROCESS

1. Set up a virtual environment for the project and install the required packages.

```
   python -m Venv "name of virtual environment"
```

   To activate the environment:
   environment name\Scripts\activate

2. Imported required libraries

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

3. Then created a flask application instance called “app”.

```
app = Flask(__name__)
```

   “Flask(__name__)” -- this function creates the application instance with the name of the application’s module or package as its argument.
   “__name__” – it is a special variable that refers to the name of the current module or package, it allows Flask to determine the root path of the application.

4. 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['SOLALCHEMY_TRACK_MODIFICATIONS'] = False

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

5. Creating the table class

```
class URL(db.Model):
    __tablename__ = "urls"
    id = db.Column ("id",db.Integer, primary_key=True)
    long = db.Column ("long",db.String())
    short = db.Column ("short",db.String(3))
    def __init__(self, long, short):
        self.long = long
        self.short = short
```

Created 3 columns “id”,”long”,”short” to save the data to the class.

After this I created a decorator with a function to be executed once, before the first request to the application.

```
@app.before_first_request
def create_tables():
    db.create_all()
```

6. Created a function to chose random letters

```
def shorten_url():
    #letters = string.ascii_lowercase + string.ascii_uppercase

    letters = "qwertyuiopasdfghjklzxcvbnmQWERTYUIOPASDFGHJKLZXCVBNM"
    while True:
        rand_letters = random.choices(letters,k=3)
        rand_letters = ''.join(rand_letters) #converting list to string
        short_url = URL.query.filter_by(short=rand_letters).first()
        if not short_url:
            return rand_letters
```

From the letters using while loop choosing random 3 letters, then converting it into string, and giving the random letters to the table column “short”, “if- statement” is used to store that random letter once only.

7. Creating routes

```
@app.route('/' , methods=["GET","POST"])
def index():
```

```
    if request.method == 'POST':
        url_received = request.form["long_url"]
        # check if url already exists in db
        found_url = URL.query.filter_by(long=url_received).first()
        if found_url:
            #return short url if found
            return redirect(url_for('display_short_url',
url=found_url.short))
        else:
            #create short url if not found
            short_url = shorten_url()
            new_url = URL(url_received, short_url)
            db.session.add(new_url)
            db.session.commit()
            return redirect(url_for('display_short_url',url=short_url ))
    else:
        return render_template('home.html')
```

Home Page is displayed using this route and if the method is “POST” the long URL that user enters will be saved in the database if that URL is not already exists in the database.
This is the html frontend of Home Page:

```
<form method="post" action="{{url_for('index')}}">

    <label for="long_url">Enter URL:</label><br>
    <input type="url" name="long_url" placeholder="Enter Here"><br>
    <button type="submit">Submit</button>
</form>
```

And another page will be shown with the short URL.

```
@app.route("/display/<url>")
def display_short_url(url):
    return render_template("shorturl.html", short_url_display=url)
```

shorturl.html:

```
<h3>Short URL : http://127.0.0.1:5000/{{short_url_display}}</h3>

<h4>Click Here : <a
href="http://127.0.0.1:5000/{{short_url_display}}">http://127.0.0.1:5000/{{short_
url_display}}</a></h4>
```

Using the below route the created short URL will be redirect to the original URL and it goes to the same webpage.

```
@app.route("/<short_url>")
def redirection(short_url):
```

```
    long_url = URL.query.filter_by(short=short_url).first()
    if long_url:
        return redirect(long_url.long)
    else:
        return f'<h1>url dosent exists</h1>'
```

By this route we can see the history of the URLs that the user has shortened using this application.

```
@app.route("/display")
def display():
    urls = URL.query.all()
    return render_template("display.html", urls=urls)
```

In display.html page I created a table which shows the original URL and short URLs and we can delete rows by using the below route

```
@app.route("/delete/<int:id>")
def delete(id):
    row = URL.query.get_or_404(id)
    db.session.delete(row)
    db.session.commit()
    return redirect("/display")
```

After deleting it will redirect into the same page.

Also I created a delete button to delete the data in display.html.

8. Created an style.css file in static folder to give some style to the application. And connected “style.css” file to the “layout.html” file

```
    <link rel="stylesheet" href="{{ url_for('static',
filename='css/style.css') }}">
```

9. Finally command “python app.py” in vs code command promt. Flask Web application will be opened In browser.

SCREENSHOP OF RUNNING APPLICATION