INNOMATICS
RESEARCH LABS

URL SHORTENER APPLICATION PROJECT

URL Shortener Web Application

Done by : Piyush Saini

Introduction :

URL shortener web application is a simple tool that converts the long URL into simple and short URL. And when you enter the short URL it redirects to the original URL.

Sometimes we need to share or send links and this can be tiresome and annoying to copy and paste long URLs. That is where URL shorteners come in. Not only it helps in shortening the URL but it also allows the user to copy the shortened URL with a click of a button.

Explanation :

To make a URL shortener we need a framework. The framework that we used is Flask for backend application server. This project is built is using flask where we created python file named app.py **Importing Modules :**

Importing of modules in Python are **important for structuring your code effectively**. Using imports properly will make you more productive, allowing you to reuse code while keeping your projects maintainable.

- We have to import os to create and access the different files in operating system.
- We have to import render_template for rendering templates and request for post and get methods, redirect for redirecting the url's .
- SQlAlchemy is a sql-toolkit which gives power and flexibility to sql database.
- Flask – migrate is used to migrate the files in database and binding.

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

Database configuration :

After importing Libraries the next step is configuring the SQLAlchemy

```
11
12
13  base_dir=os.path.abspath(os.path.dirname(__file__))
14  app.config['SQLALCHEMY_DATABASE_URI']='sqlite:///'+ os.path.join(base_dir,'data.db')
15  app.config['SQLALCHEMY_TRACK_MODIFICATIONS']=False
16
17
18
19
```

Creating a Model :

The next step is to create a model class for creating a table in database. And provide def __init__ method for initialization and __repr__ is a special method used to represent a string.

```python
##########  CREATE A MODEL  ##################
class url(db.Model): #blueprint to create an object
    __tablename__= 'urls'
    id=db.Column(db.Integer,primary_key=True)
    user_url=db.Column(db.Text)
    short_url=db.Column(db.Text)
    def __init__(self, user_url, short_url):
        self.user_url = user_url
        self.short_url = short_url
    def __repr__(self): #special method used to represent a class object as a string
        return "Original_url ::: {}   and  short_url  ::: {}".format(self.user_url , self.short_url)
```

Routes :

App routing is used to map the specific URL with the associated function that is intended to perform some task. It is used to access HTML pages in web application.

- Firstly we created a “/” route by defining two functions for get and post methods which are binded to HTML pages which render the form from “index.html” template.
- In the post method we created short url by using random module. And binded the original url and short url to the class URL. • Nextly we created a “/history” route to see the history of the user which shows the original url and short url .
- Final route is for redirecting the url to its page by using dynamic routing . If the original url is not there it return the “indirect url”

```
data ={}
@app.route("/") def
home_get():
    return render_template("index.html")

@app.route("/" , methods = ['POST']) def
home_post():    himalayan =  5
        original_url = request.form.get("inp_1")         short_url_1
=
"".join(random.choices(string.ascii_letters+string.digits ,k = himalayan
))         data[short_url_1] = original_url         new_url =
url(original_url , short_url_1)         db.session.add(new_url)
db.session.commit()         return render_template("index.html" , k =
short_url_1 )
@app.route("/history") def
fun_2():
    all_urls = url.query.all()     return render_template("history.html" , data
= data , all_urls = all_urls )

@app.route("/robhra/<short>")
def fun_3(short):
if(short) in data :
        return
redirect(data[short])     return
"incorrect url"       if __name__ ==
"__main__":
app.run(debug =True)
```

Templates :

“index.html” : This is the homepage of url shortener where the original url is entered and it is shortened by clicking shorten button and then the short url which is generated can be copied using copy button.

```
<form action = "" method = "post">
        Enter the Original Url : <input type = "url" name = "inp_1" class =
"form-control" placeholder="enter the original url">
        <button>SHORTEN</button><br><br>


    </form> Shortened
Url :
               <input class = 'form-control'    value =
"127.0.0.1:5000/robhra/{{k}}" id = "myInp"  >
```

```
                <button id = "btncopy">Copy</button>

                <script type = "text/javascript">
const myInp = document.getElementById("myInp");
const btncopy = document.getElementById("btncopy");
btncopy.onclick = function(){
myInp.select();
document.execCommand("Copy");
                } ;

            </script>         </div>
```

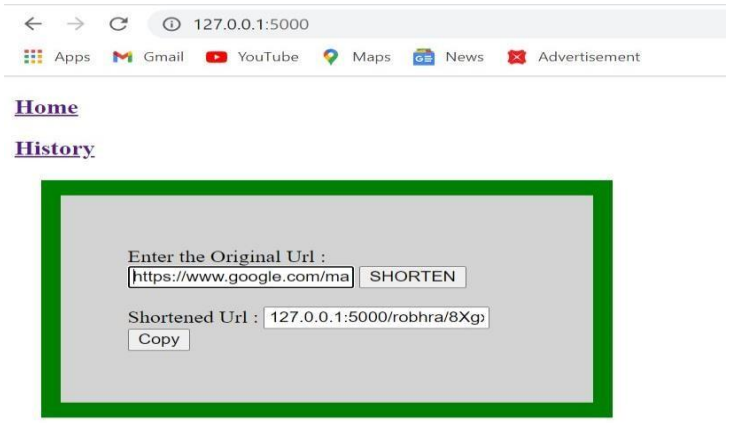
“History.html” : This page displays the collection of all the original url’s entered by user and short url’s **.**

```
<h1>History of URL's</h1>
    <table>
        <tr>
            <th>original url</th>
            <th>shorten url </th>
        </tr>


    {% for short_url in  data %}
        <tr>
      <td>  {{ data[short_url]}} </td>
       <td> 127.0.0.1:5000/robhra/{{short_url}} </td>
        </tr>     {% endfor %}
```

OUTPUT :

URL : 127.0.0.1:5000



URL : 127.0.0.1:5000/history