INNOMATICS RESEARCH LABS-INTERNSHIP(OCT'21)

PROJECT NAME: URL SHORTENER

D SIKHA DATTA REDDY

Introduction

URLs can get long, unattractive, and break when sent via e-mail. Them being too long would be problematic when someone must share that link with their peers. In this project, I built an URL Shortener, a service that takes any URL and generates a shorter, more readable version. I have used some basic Flask concepts such as creating routes, rendering HTML templates, and connecting to an SQLite database. This application will allow users to enter a URL and generate a shorter version, in addition to a history page where users can view the shortened URL and date and time it was created on. I used CSS and JavaScript to give some basic styling to the table generated on the history page.

PROCESS OF BUILDING THE PROJECT

Step 1) Setting Up Dependencies:

In this step, I activated Python environment and installed Flask and sqlachemy.

Step 2) app.py file

In this python file I started off by writing the basic structure of a flask application file. I imported Flask, render_template, request, redirect and abort from flask. I have imported the following:
datetime, SQLAlchemy from flask_sqlalchemy, random, os.

```
import os
import string
from flask import Flask, render_template, request,redirect,abort
import random
from flask_sqlalchemy import SQLAlchemy
from sqlalchemy.orm import column_property
from datetime import datetime
```

Creating a SQLite Database in Flask with SQLAlchemy

SQLALCHEMY_DATABASE_URI should equal the path of the app
I'm going to start off by creating a variable here called **application_dir.** This is going to stand for the base directory of our application because if we don't know what the base directory is then flask isn't going to know where to save our SQLite table to. I'm going to start off with this, and I'm going to call OS, our operating system library, and then say:

```
application_dir =
os.path.abspath(os.path.dirname(__file__))
```

SQLALCHEMY_DATABASE_URI. Now, this is a specific name. This is something inside of our system, and it has to be set exactly like this or else the SQLite database is not going to know where to go or what the values are.
Now that you have that, we're going to pass in a string here. So, I'm going to say:

```
app.config['SQLALCHEMY_DATABASE_URI'] =
'sqlite:///'+os.path.join(application_dir,'db1.sqlite')
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
```

now, we will create a database object:

```
table= SQLAlchemy(app)
```

Next thing I did was making routes for my file. The routes I created are:

- @app.route('/')
- @app.route('/', methods=['POST'])
- @app.route('/history')
- @app.route('/<shorten_url>')

  In the route @app.route('/', methods=['POST']), I made a function home_post() which will first get the input given by the user and then will shorten the URL. Then the shortened URL will be sent to the index1.html page with the help of render_template.
  For getting the shortened URL, we will be using **random** which will generate a random 4-digit number and attach it to the local host URL.

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

    original_url = request.form.get('in_1')
    if 'http' not in original_url:
      original_url = f'http://{original_url}'
    if original_url is None:
     return "<h2 style='color:red'> Invalid URL </h2>"
    shorten_url = random.randint(1000, 9999)
    data[shorten_url] = original_url
    DATA = link(shorten_url,original_url)
    table.session.add(DATA)
    table.session.commit()
    return render_template('index1.html', short=shorten_url)
```

In the route @app.route('/history'), I made a function history_get() which would fetch the data from the SQLite database and send it to the history1.html page where all the URLs would be displayed.

```
@app.route('/history')

def history_get():
    DATA = link.query.all()
    return render_template('history1.html', DATA=DATA)
```

In the route @app.route('/'), I made a class which would make the columns for the table link in the database. The columns made are id, original_url, shorten_url, date. So right here I'm going to say class link, this is going to inherit from (table.Model). You may notice this table call right here is calling this object, the SQL Alchemy object that I created before.

```
@app.route('/')

def home_get():
    return render_template('index1.html')

class link(table.Model):
    __tablename__ = 'link'
    id= table.Column(table.Integer, primary_key=True)
    original_url = table.Column(table.String(500))
    shorten_url= table.Column(table.String(5), unique=True)
    date= table.Column(table.DateTime,default=datetime.now)

    def __init__(self,shorten_url,original_url):
        self.shorten_url = shorten_url
        self.original_url = original_url

    def __repr__(self):
        return '{} => {}'.format(self.original_url,self.shorten_url)
```

We are making this table database with the help of SQLAlchemy and all the data which will be given as input and the required output will be stored in a file called “db1.sqlite”.

In the route @app.route('/<shorten_url>'), we have a function named as redirect_to_url which helps us to get the shortened URL to the database with use_for in the history1.html page.

```
@app.route('/<shorten_url>')
def redirect_to_url(shorten_url):
    Link = link.query.filter_by(shorten_url=shorten_url).first_or_404()

    Link.id = Link.id + 1
    table.session.commit()

    return redirect(Link.original_url)
```

I am applying the given filtering criterion to a copy of the Query, using keyword expression shorten_url.

```
 Link = 
link.query.filter_by(shorten_url=shorten_url).first_or_404
()
```

Step 3) layout.html file

This file will contain a JavaScript code which will take clipboard.min.js. This will help us to copy the URL directly to clipboard.
Then we have home and history with href. When we click the history page, we will be redirected to a new page which will contain the table with all the information. Home and history will be put up in the navigation bar.

```
<title>URL shortener app</title>

      <script 
src="https://cdnjs.cloudflare.com/ajax/libs/clipboard.js/1.5.5/clipboard.min.j
s"></script>




</head>
<body>
  <h2>Create your short link</h2><hr>
  <nav>
   <a href='/'>Home</a>
   <a href='/history'>History</a>
   </nav>
```

Step 4) index1.html file

This file will be inherited from layout.html
In the index.html, we have a JavaScript function named function() which helps us to copy the generated URL when the copy button is pressed. It uses clipboard.min.js This will convert the button (with id="copy-button") into a clipboard copy button. And pressing it will put the value of data-clipboard-text on the clipboard.

```
(function(){
    new Clipboard('#copy-button');
})();
```

We will have the basic heading which says, “Welcome to the URL Shortener web app”. There is a form included in this page which would take the original URL as input and would automatically put the URL in the copy field so that we can copy the URL by just clicking the button.
We have an enter URL field and after submitting, we can copy the URL to the clipboard with the help of copy button.

```
<form method='post'>


    <label>Enter URL</label>
    <input type="text" name="in_1" placeholder="Enter the URL" >
       <button>Submit</button>
</form>


<form  method="post">

  <input id='short-link' value= "http://127.0.0.1:5000/{{ short }}" />

   <button class="btn ui button" type='button' id="copy-button" data-
clipboard-target="#short-link">Copy</button>
```

Step 5) history1.html file

this file will have a table with columns as id, original_url, short_url, date created.

```
{% for link in DATA %}
<tr>
<td class="countercell"></td>
<td>{{ link.original_url }}</td>
<td>{{ url_for('redirect_to_url', shorten_url=link.shorten_url,
_external=True) }}
```

```
</td>
<td>{{ link.date }}</td>
</tr>
{% endfor %}
```

The above code will put the values in the table. With the help of **url_for,** we will get the shortened URL by using the function redirect_to_url which is there in the route @app.route('/<shorten_url>'). We will get all the information stored in the database. The id is increment and the logic to that is there in a CSS sheet named as styles.css.

Step 6) styles.css file

```
body {
    background-color: powderblue;

  }
  table {
    border-collapse: collapse;

  }
  .table1 {
    counter-reset: serial-number;
  }

  .table1 td:first-child:before {
    counter-increment: serial-number;  /* Increment the serial number counter
*/
    content: counter(serial-number);  /* Display the counter */
  }

  td, th {
    border: 1px solid #999;
    padding: 0.5rem;
    text-align: left;
  }
```

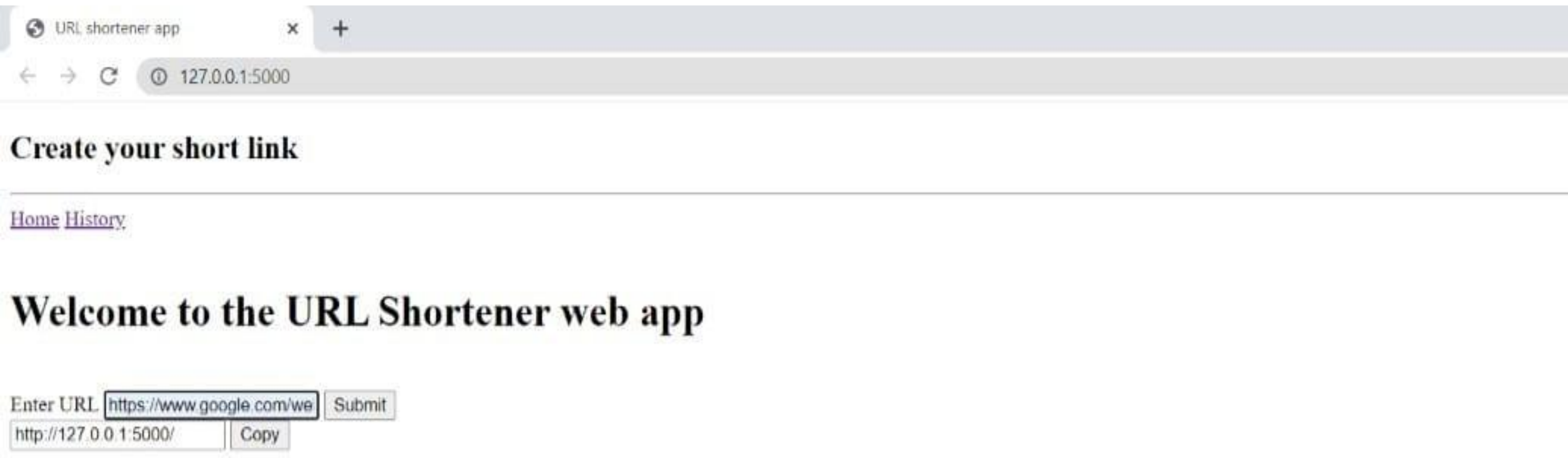
The above code was used to fit the information in their respective column. I have made the background colour of the page as powder blue. With the help of the code given below, we were able to increment the id column every time a new input was given.

```
.table1 {
   counter-reset: serial-number;
  }
```

```
 .table1 td:first-child:before {
   counter-increment: serial-number;  /* Increment the serial number counter */
   content: counter(serial-number);  /* Display the counter */ }
```

OUTPUT-

Step -1: Enter the Original URL of your choice



Create your short link

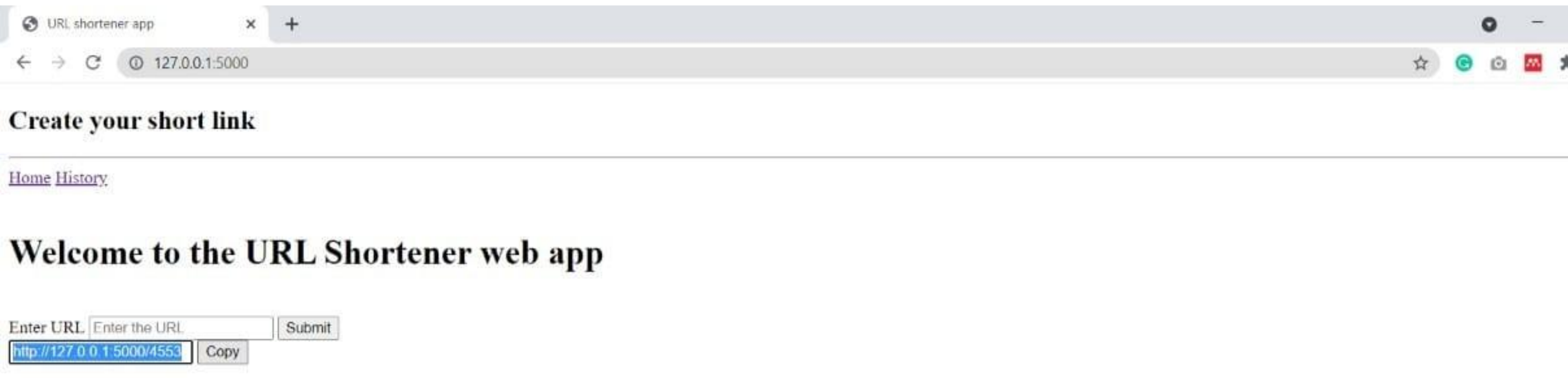
Welcome to the URL Shortener web app

Enter URL https://www.google.com/we

http://127.0.0.1:5000/

After clicking the submit button, the short ul will be automatically displayed in the copy field.

Step-2: Click copy button to copy the URL



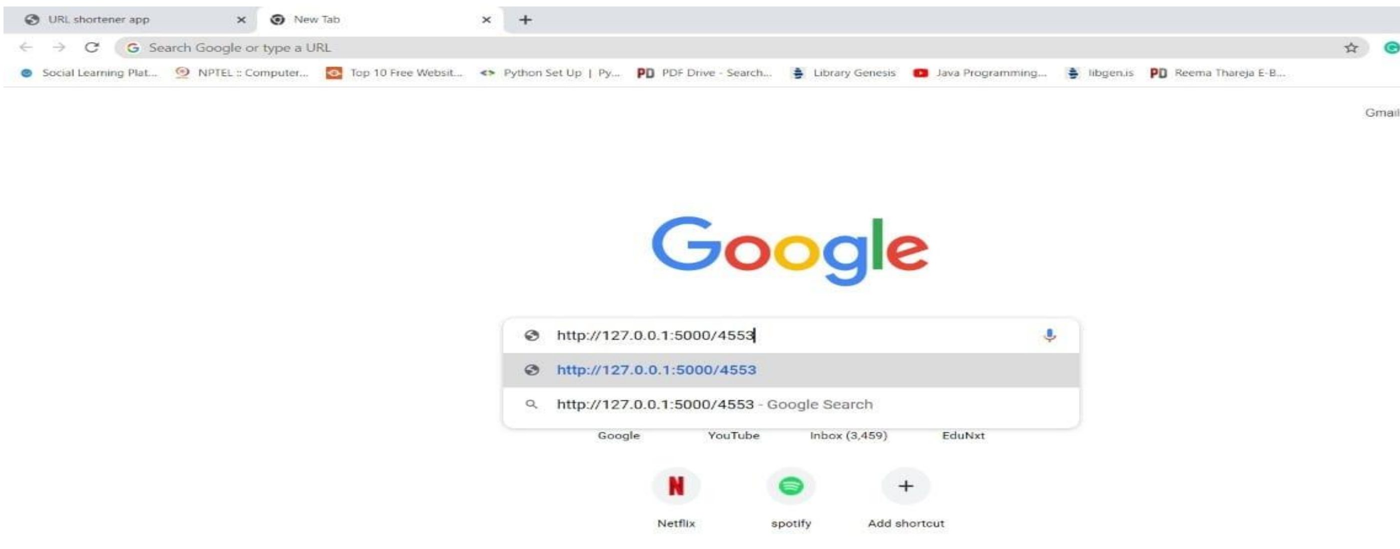
Create your short link

Home History

Welcome to the URL Shortener web app

Enter URL [Enter the URL] Submit

http://127.0.0.1:5000/4553 Copy

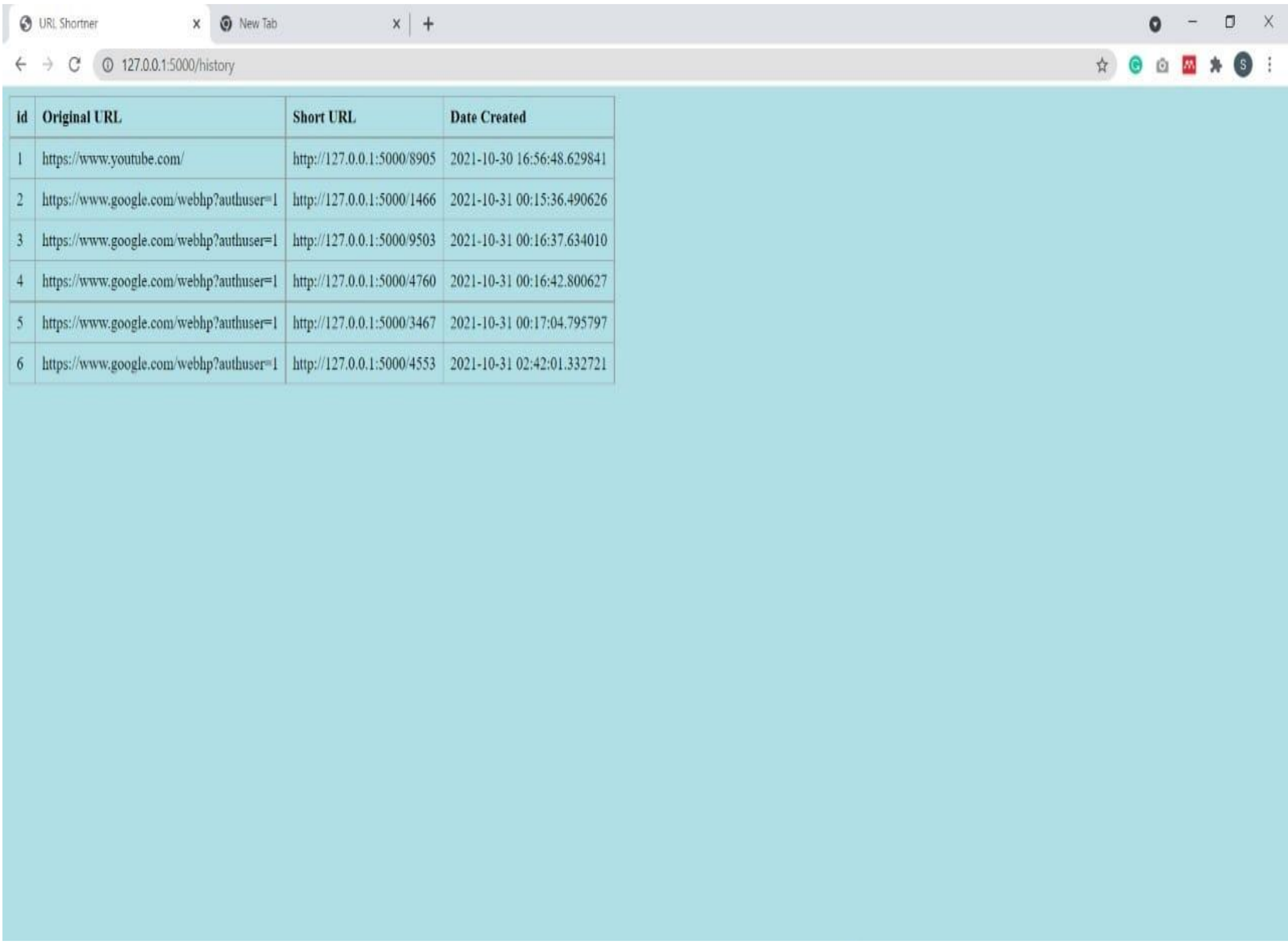


http://127.0.0.1:5000/4553

http://127.0.0.1:5000/4553

http://127.0.0.1:5000/4553 - Google Search

Step-3: click on the history to go to the history page which would give you the information regarding the URLs which were used before to shorten



id	Original URL	Short URL	Date Created
1	https://www.youtube.com/	http://127.0.0.1:5000/8905	2021-10-30 16:56:48.629841
2	https://www.google.com/webhp?authuser=1	http://127.0.0.1:5000/1466	2021-10-31 00:15:36.490626
3	https://www.google.com/webhp?authuser=1	http://127.0.0.1:5000/9503	2021-10-31 00:16:37.634010
4	https://www.google.com/webhp?authuser=1	http://127.0.0.1:5000/4760	2021-10-31 00:16:42.800627
5	https://www.google.com/webhp?authuser=1	http://127.0.0.1:5000/3467	2021-10-31 00:17:04.795797
6	https://www.google.com/webhp?authuser=1	http://127.0.0.1:5000/4553	2021-10-31 02:42:01.332721

THE END