Project - URL Shortner Web Application

Objective:

1. As the name suggests, it shortens URLs.

2. Users can also save URL's by coming to the web app.

Need for URL Shortener:

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 does it help in shortening the URL but it also allows the user to copy the shortened URL with a click of a button.

The project consists of 3 parts:

1. Frontend (done with HTML, CSS, and Bootstrap)

2. Backend - Flask (Python)

3. Backend - Database ORM

Front-End Information:

The front end consists of 2 web pages:

1. Home Page - A page will be shown where the user can enter the URL he/she wants to shorten. After the 'shorten' button is clicked, the shortened URL is displayed in the text field which the user can copy using the copy button.

2. History Page - Containing all the Original URLs along with the Shortened URLs.

Project Workflow:

1. Users can enter the URL they want to shorten. After entering a URL, click on the 'Shorten' URL button to display the shortened URL in the following text field which can be copied by clicking on the copy button.

2. After the ‘Shorten’ button is clicked, the URL that is entered is saved in our database with the shortened URL. It is saved in the database so that the user can look into the previous URLs he entered in our web app with their shortened URL.

3. Try to verify whether the URL entered by the user is correct or not.

Packages:

To build the app following packages were used:

- Flask
- Flask_SQLAlchemy
- Flask_Migrate
- Os

Steps involved in building this application:

Step-1: Setup SQLite Database in Flask-App

i. Create a basic app.py and import the required modules from the given packages and code the basic flask template and also create basic templates files.

ii. Configure SQLAlchemy -- by using the below commands install sqlalchemy and migrate.

```
Pip install flask-sqlalchemy
Pip install flask-migrate
```

```
basedir=os.path.abspath(os.path.dirname(__file__))
app.config['SQLALCHEMY_DATABASE_URI']='sqlite:///'+os.path.join(basedir,'data.
sqllite')
app.config['SQLALCHEMY_TRACK_MODIFICATIONS']=False
```

iii. Pass the application to the database by using the commands

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

```

Step-2:To create a model in the Flask-App

i. Create a model class

ii. Inherit from db. model

iii. Provide a table it contains the table name, add columns with column names with its types and primary key.

iv. __init__ and __rep__

```
class URLData(db.Model):
    __tablename__='url_shortener'
    id=db.Column(db.Integer,primary_key=True)
    long_url=db.Column(db.String(100))
    short_url = db.Column(db.String(50))

    def __init__(self,long_url,short_url):
        self.long_url=long_url
        self.short_url=short_url


    def __repr__(self):
        return "Long url - {} and Short url-
{}".format(self.long_url,self.short_url)

```

Step-3:Final set up with some commands

By implementing some commands in the command prompt it will be used in administrator mode, and the final setup is established to proceed.

i. The first command is **flask db init** after running this command we get the Migrations folder.

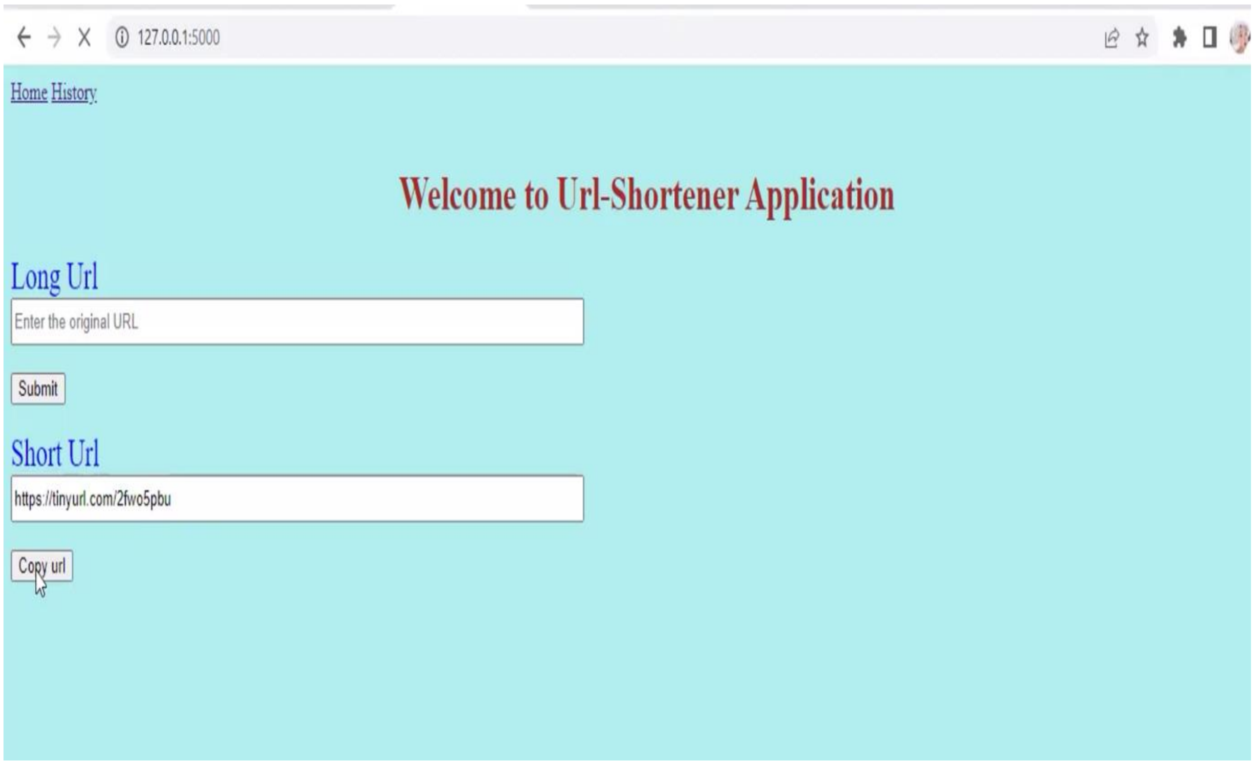
ii. The second command is **flask db migrate -m “First Migration”** after running this command data.sqlite file is generated automatically in the folder.

iii. The third command is **flask db upgrade**

Step-4:Create routes and Html files

- Create basic route with methods GET and POST, and define the function for performing the url shortner operation.
- pyshorteners is a **Python lib** to help you short and expand URLs using the most famous URL Shorteners available. With the help of pyshorteners, you can generate a short URL or expand another one which is as easy as typing.
- Create 3 HTML files inside the templates folder,“layout.html”, “home.html”, and “history.html”.
- In the “layout.html” file containing all the basic template code with the navigation bar and should contain blocks for inheritance.
- Build “home.html” and “history.html” using the concept of inheritance from “layout.html” and should contain relevant requests, forms and display content.
- Give a CSS styling to both the HTML files.
- In “home.html” add a button to copy the shortened URLs using JavaScript code snippets
- Create another route for history for adding all URLs to the table present in the database.
- In “history.html” add a table for displaying all URLs.
- “history()” - this fetches all the rows of database and finally displays each original URL and Shorten URL ever created row-by-row in a table in Frontend.

The Home page is displayed as



127.0.0.1:5000

Home History

Welcome to Url-Shortener Application

Long Url

Enter the original URL

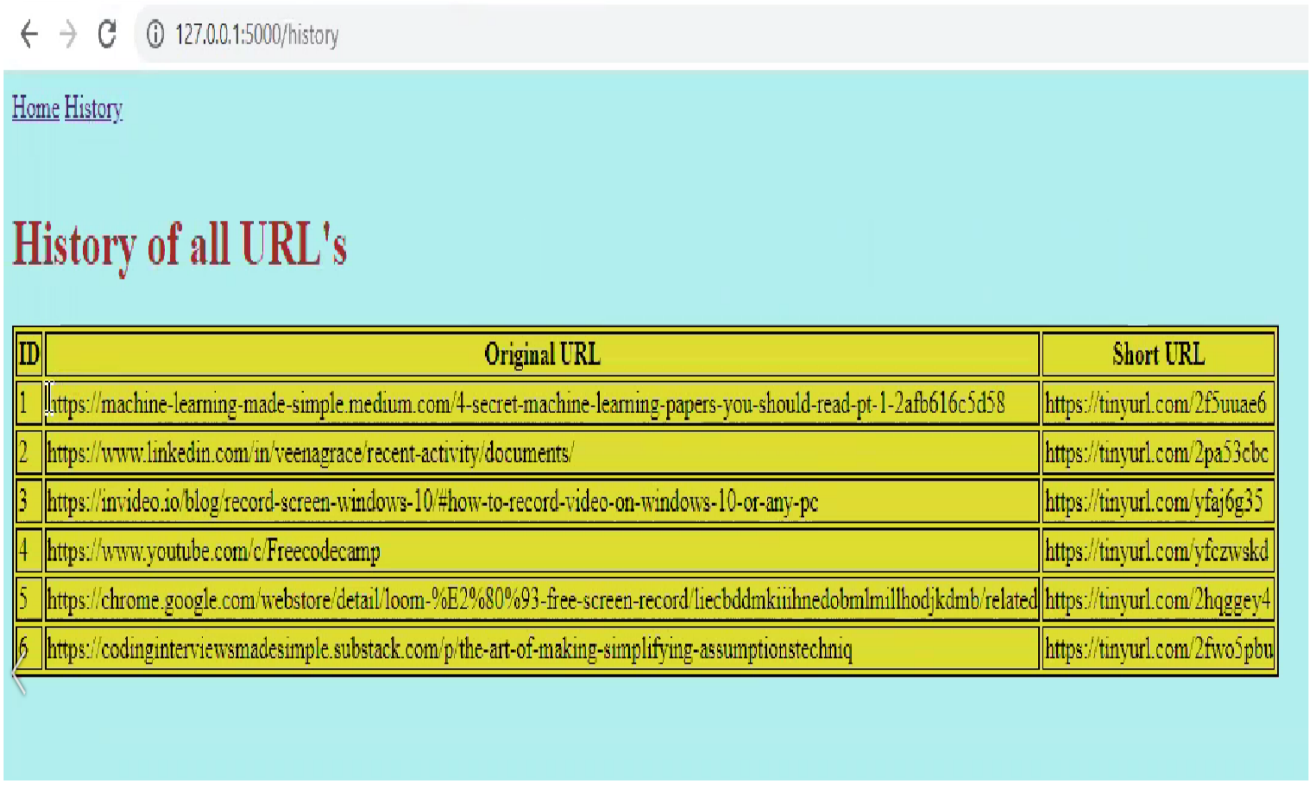
Submit

Short Url

https://tinyurl.com/2fwo5pbu

Copy url

The History page is displayed as



127.0.0.1:5000/history

Home History

History of all URL's

ID	Original URL	Short URL
1	https://machine-learning-made-simple.medium.com/4-secret-machine-learning-papers-you-should-read-pt-1-2afb616c5d58	https://tinyurl.com/2f5uuae6
2	https://www.linkedin.com/in/veenagrace/recent-activity/documents/	https://tinyurl.com/2pa53cbc
3	https://invideo.io/blog/record-screen-windows-10/#how-to-record-video-on-windows-10-or-any-pc	https://tinyurl.com/yfaj6g35
4	https://www.youtube.com/c/Freecodecamp	https://tinyurl.com/yfczwskd
5	https://chrome.google.com/webstore/detail/loom-%E2%80%93-free-screen-record/liecbddmkiiihnedobmlmillhodjkdmb/related	https://tinyurl.com/2hqggey4
6	https://codinginterviewsmadesimple.substack.com/p/the-art-of-making-simplifying-assumptionstechniq	https://tinyurl.com/2fwo5pbu