URL SHORTNER APP

USING FLASK

1. Creating the required Files and Libraires:
   - Creating the main python file (app.py)
   - Creating a virtual environment.
   - Importing Flask, Sql Alchemy, Random and String.

2. Creating the Basic website:
   - Creating the templates folder.
   - Inside the templates folder creating the base history and homepage html files
   - Importing Bootstrap for better CSS look
   - The Base html contains the basic template file of the website.
   - The Homepage contains the Text field to enter the Long URL and display the shortened form of the URL
   - The History page contains all the URLs both long and shortened form

3. Starting to create the server side i.e., app.py:
   - Creating the required database.

```python
# creating the database uri (using SQL Lite)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///URL_Directory.db'
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False

# creating the database
url_db = SQLAlchemy(app)

@app.before_first_request
def create_table():
    url_db.create_all()

# creating the schema of the database
class URLS(url_db.Model):
    id_ = url_db.Column(url_db.Integer , primary_key = True)
    long_url = url_db.Column(url_db.String(), nullable=False)
    short_url = url_db.Column(url_db.String(20), nullable=False)

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

- The data base is created using a class (URLS). It has 3 columns id long_url and short_url.
- Using the init method(constructor) we have initialised the variables.
- The table is created using the create table function.

- Creating the main home route. The home gets the Long URL from the form in the homepage.
  - It checks is the URL present in the database or not.
  - If the URL is already present then it returns the corresponding short URL directly to the text field in the home page
  - If the URL is not present the shorten_url function is called which returns a random string of characters.
  - Then both the long and short URL is passed to the database adding the new data into the database and hence returning the shortened URL to the home page.
  - If nothing is passed in the form it would just display the homepage directly.

```
@app.route("/home", methods=['POST', 'GET'])
def home():
    if request.method == 'POST':
        url_long = request.form['long_url']

        found_url = URLS.query.filter_by(long_url=url_long).first()

        if found_url:
            return render_template('HomePage.html', url=found_url.short_url)
        else:
            url_short = shorten_url()
            n_url = URLS(url_long, url_short)
            url_db.session.add(n_url)
            url_db.session.commit()
            return render_template('HomePage.html', url=url_short)
    else:
        return render_template('HomePage.html')
```

- Creating the shorten URL function.
  - Creating a set of characters using string printable
  - Choosing 8 random characters from the set and joining them to form the short URL.

- Checking if the URL already exists or not , if the URL already exists changing the combinations else returning the short URL to the database object in home route.

```
def shorten_url():
    characterSet = string.ascii_lowercase + string.ascii_uppercase
    while True:
        letters = random.choices(characterSet, k=8)
        url = ''.join(letters)
        short_found = URLS.query.filter_by(short_url=url).first()
        if not short_found:
            url = url.replace(' ', '0')
            return url
```

- Creating the History route.
  - This page gives the output of all the URLs present in the database

```
@app.route("/history")
def history():
    urls = URLS.query.all ()
    return render_template('History.html' , url = urls)
```

- Creating the route to set the long URL as the short URL.
  - Open URL function checks is the short URL is present in the database or not.
  - If the URL is present it redirects the page to the original long URL which the short URL is attached to.

```python
@app.route('/<s_url>')
def open_url(s_url):
    main_url = URLS.query.filter_by(short_url = s_url).first()

    if main_url:
        return redirect(main_url.long_url)
    else:
        return f'<h1>URL Doesnot Exist</h1>'
```

- Finally, at last the delete route
  - This function is used to clear all the elements in the database .

```python
@app.route('/delete')
def delete():
    url_db.session.query(URLS).delete()
    url_db.session.commit()
    return redirect('/history')
```