Final Project (Basic Version) - URL Shortener Application

- Why do we need a 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 it helps in shortening the URL but it also allows the user to copy the shortened URL with a click of a button.

- 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.

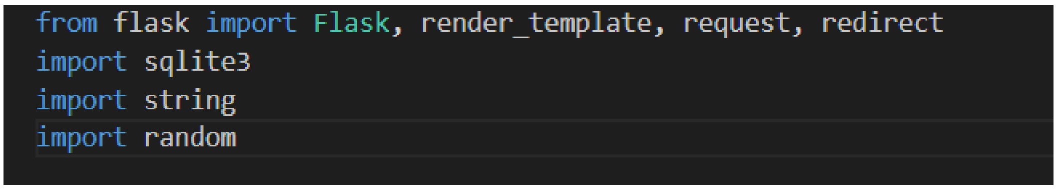
2. After the 'Shorten' button is clicked, the URL that is entered is saved in our database with the shortened URL. It is saved in the database so that the user can look into the previous URLs he entered in our web app with their shortened URL.

3. Try to verify whether the URL entered by the user is correct or not. (Do some googling to find out how to make it possible)

- Approach:

To build a URL shortener using Flask, SQLite, Python, and HTML with the features described, you would need to follow these basic steps:

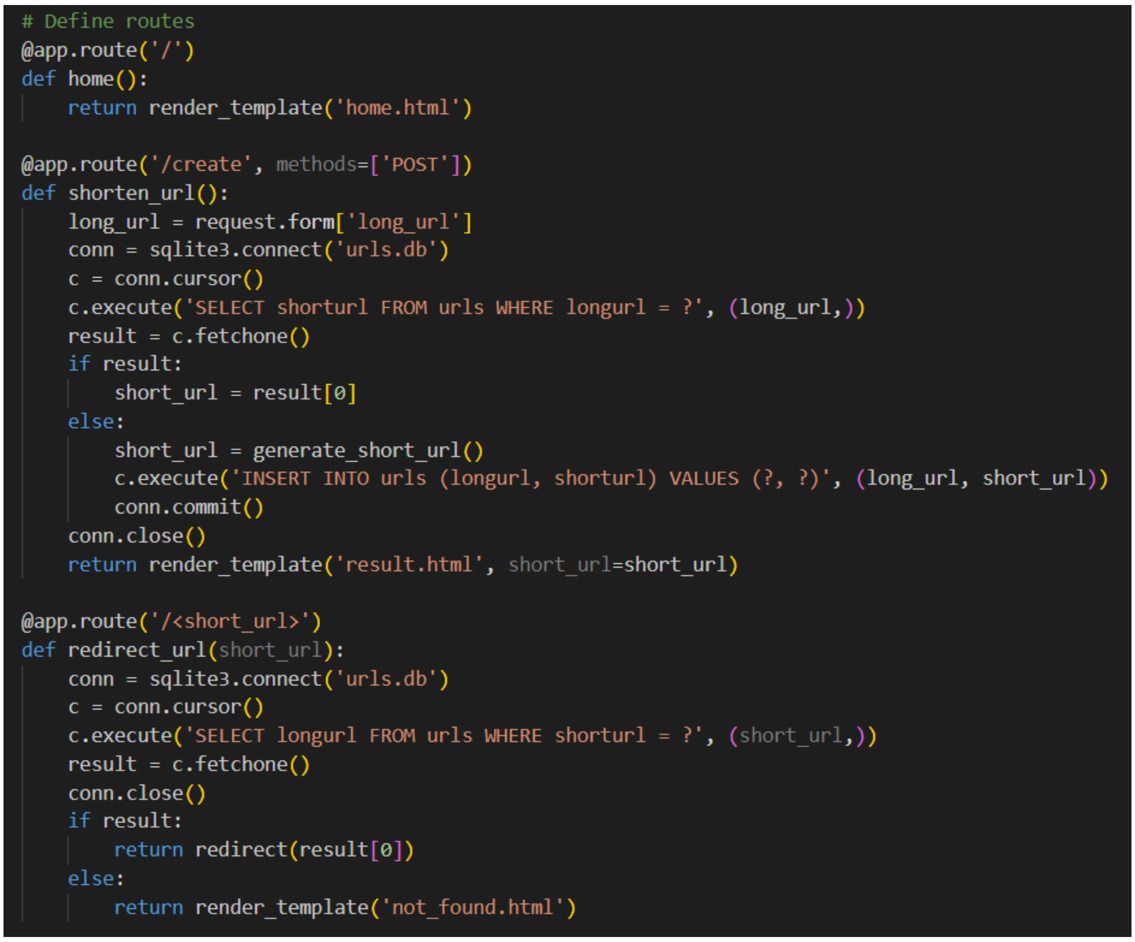
- Importing the required libraries:



```
from flask import Flask, render_template, request, redirect
import sqlite3
import string
import random
```

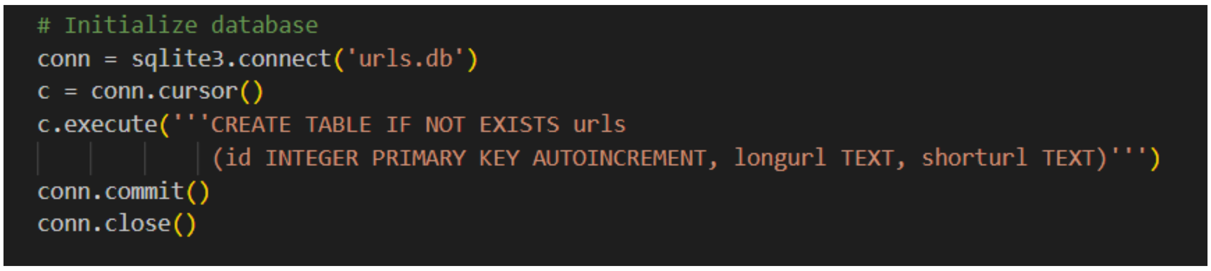
1) Create a Flask app and define routes for handling incoming requests. This includes routes for the homepage, creating new URLs, and redirecting short URLs to their original destinations.

   Here I have specified 3 routes:
   a) /
   b) /create
   c) /<short_url>

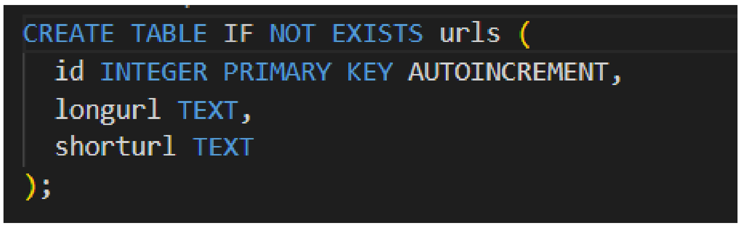


```python
# Define routes
@app.route('/')
def home():
    return render_template('home.html')

@app.route('/create', methods=['POST'])
def shorten_url():
    long_url = request.form['long_url']
    conn = sqlite3.connect('urls.db')
    c = conn.cursor()
    c.execute('SELECT shorturl FROM urls WHERE longurl = ?', (long_url,))
    result = c.fetchone()
    if result:
        short_url = result[0]
    else:
        short_url = generate_short_url()
        c.execute('INSERT INTO urls (longurl, shorturl) VALUES (?, ?)', (long_url, short_url))
        conn.commit()
    conn.close()
    return render_template('result.html', short_url=short_url)

@app.route('/<short_url>')
def redirect_url(short_url):
    conn = sqlite3.connect('urls.db')
    c = conn.cursor()
    c.execute('SELECT longurl FROM urls WHERE shorturl = ?', (short_url,))
    result = c.fetchone()
    conn.close()
    if result:
        return redirect(result[0])
    else:
        return render_template('not_found.html')
```

2) Create a SQLite database table to store the original and short URLs.
Use Python's built-in sqlite3 module to create the database and define the table. This code creates a new SQLite database file called urls.db, and then creates a table called shortened_urls with three columns: id (an integer primary key), original_url (a text field to store the original URL), and short_code (a text field to store the shortened code). ORM that provides a high-level interface to various databases, including SQLite.

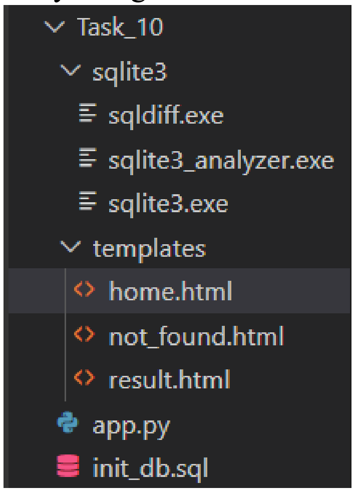


```python
# Initialize database
conn = sqlite3.connect('urls.db')
c = conn.cursor()
c.execute('''CREATE TABLE IF NOT EXISTS urls
             (id INTEGER PRIMARY KEY AUTOINCREMENT, longurl TEXT, shorturl TEXT)''')
conn.commit()
conn.close()
```



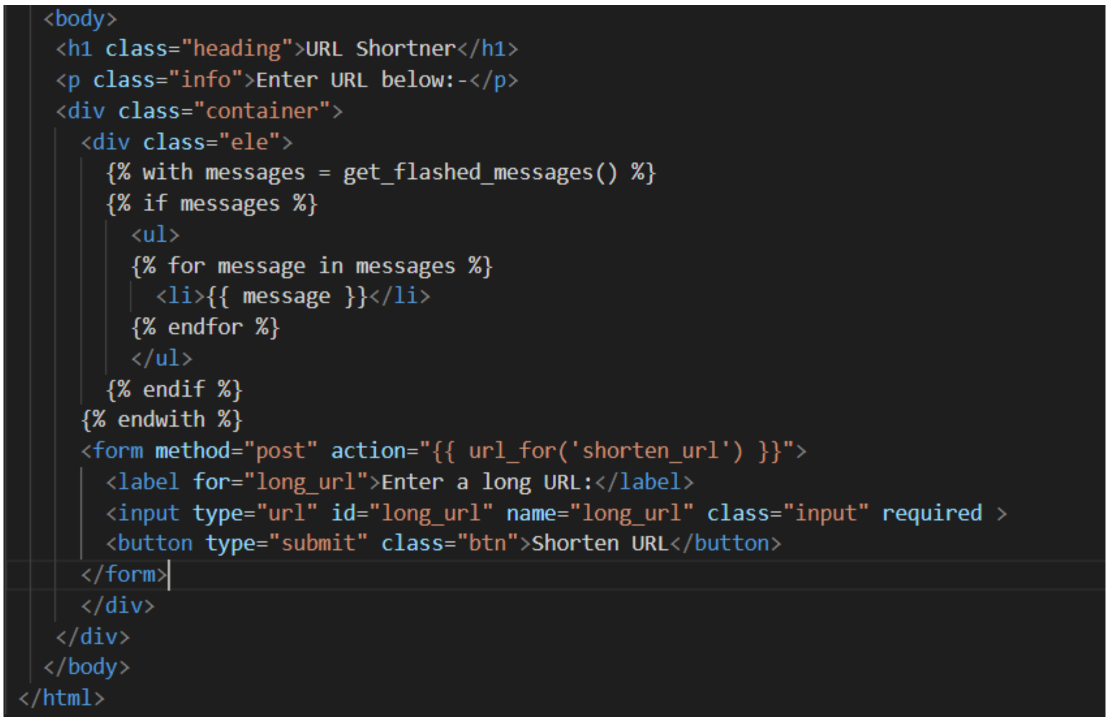
```sql
CREATE TABLE IF NOT EXISTS urls (
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  longurl TEXT,
  shorturl TEXT
);
```

3) Create an HTML form for users to enter URLs and display the short URL for each new entry using Flask's built-in support for HTML templates.



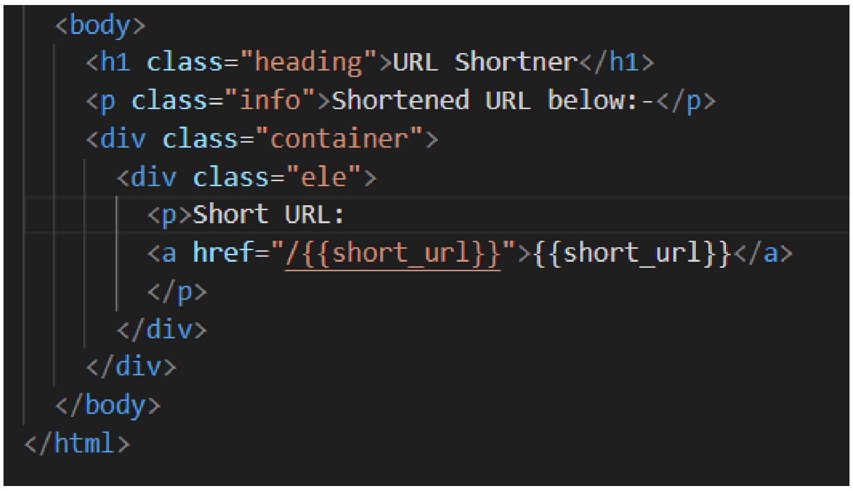
```
Task_10
  sqlite3
    sqldiff.exe
    sqlite3_analyzer.exe
    sqlite3.exe
  templates
    home.html
    not_found.html
    result.html
  app.py
  init_db.sql
```

Home.html



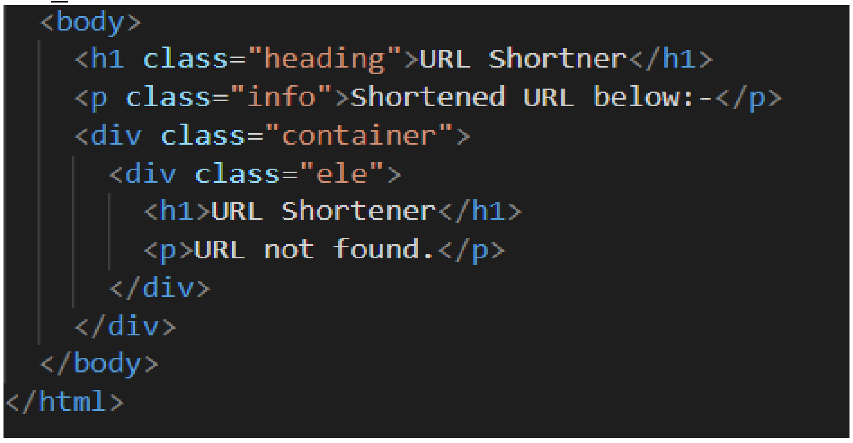
```
  <body>
   <h1 class="heading">URL Shortner</h1>
   <p class="info">Enter URL below:-</p>
   <div class="container">
     <div class="ele">
       {% with messages = get_flashed_messages() %}
       {% if messages %}
         <ul>
         {% for message in messages %}
           <li>{{ message }}</li>
         {% endfor %}
         </ul>
       {% endif %}
     {% endwith %}
     <form method="post" action="{{ url_for('shorten_url') }}">
       <label for="long_url">Enter a long URL:</label>
       <input type="url" id="long_url" name="long_url" class="input" required >
       <button type="submit" class="btn">Shorten URL</button>
     </form>
     </div>
   </div>
  </body>
</html>
```

Result.html:



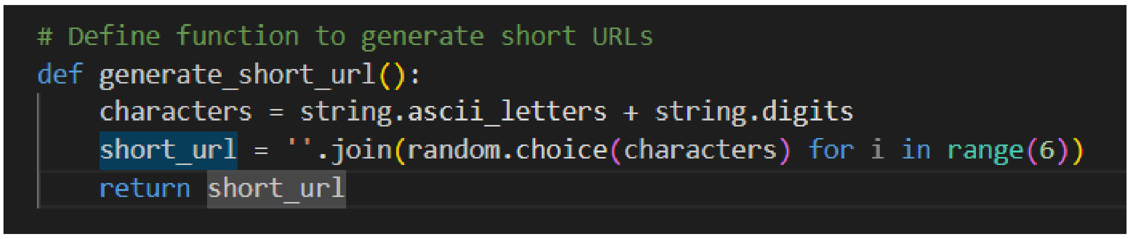
```
  <body>
    <h1 class="heading">URL Shortner</h1>
    <p class="info">Shortened URL below:-</p>
    <div class="container">
      <div class="ele">
        <p>Short URL:
        <a href="/{{short_url}}">{{short_url}}</a>
        </p>
      </div>
    </div>
  </body>
</html>
```

Not_found.html:



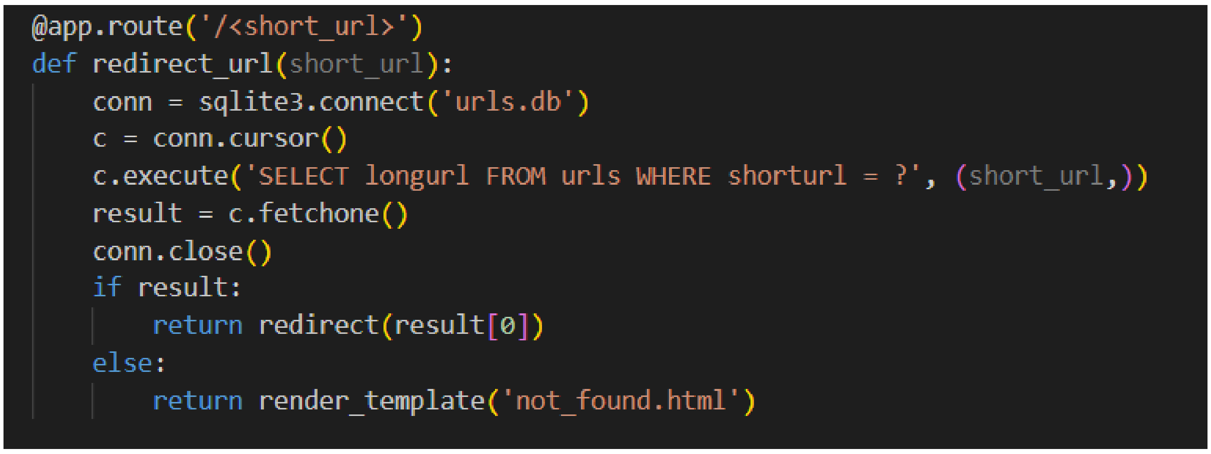
```
  <body>
    <h1 class="heading">URL Shortner</h1>
    <p class="info">Shortened URL below:-</p>
    <div class="container">
      <div class="ele">
        <h1>URL Shortener</h1>
        <p>URL not found.</p>
      </div>
    </div>
  </body>
</html>
```

4) Use Python to generate unique short codes for each new URL using a hashing function that generates unique values for each URL



```
# Define function to generate short URLs
def generate_short_url():
    characters = string.ascii_letters + string.digits
    short_url = ''.join(random.choice(characters) for i in range(6))
    return short_url
```

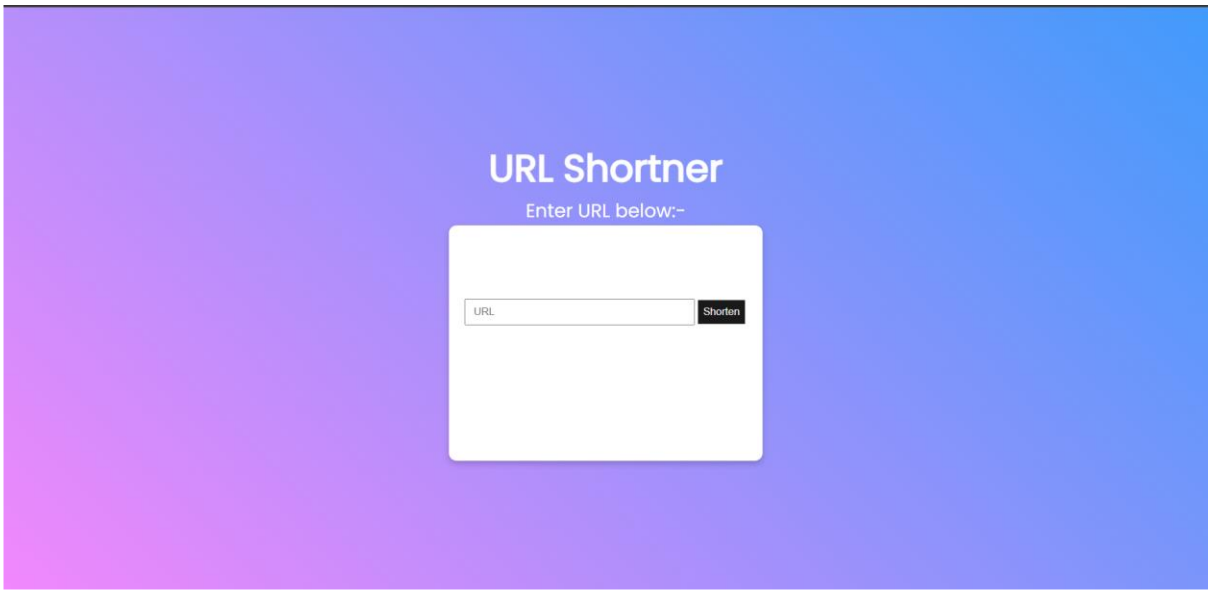
5) Save the original and short URLs to the SQLite database using sqlite3, and then display the short URL to the user.

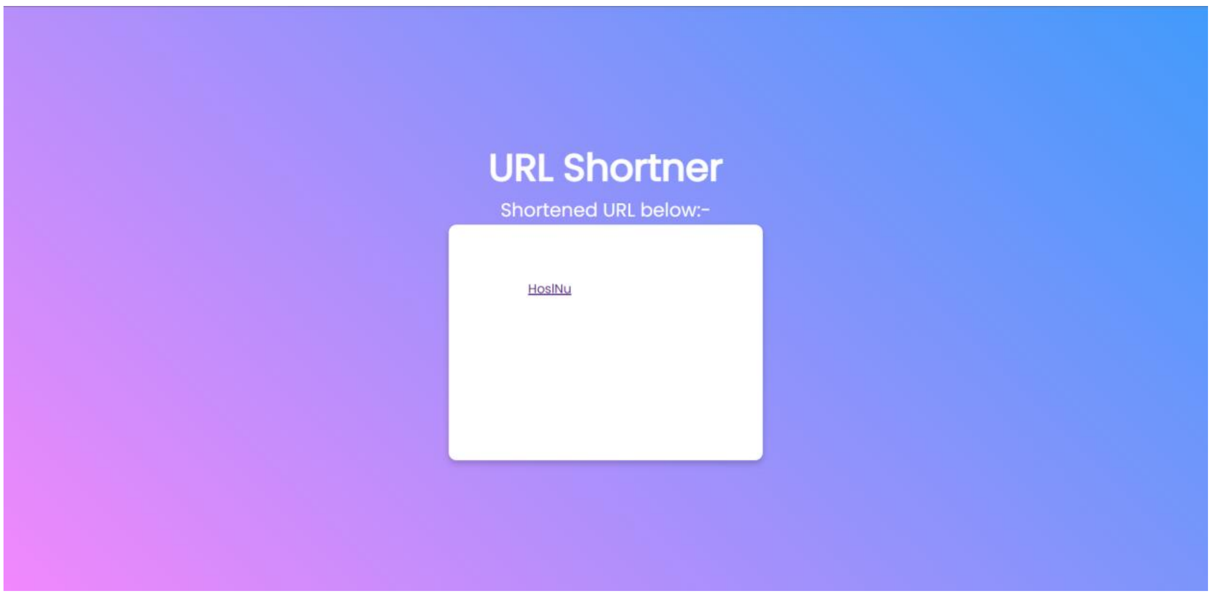


```
@app.route('/<short_url>')
def redirect_url(short_url):
    conn = sqlite3.connect('urls.db')
    c = conn.cursor()
    c.execute('SELECT longurl FROM urls WHERE shorturl = ?', (short_url,))
    result = c.fetchone()
    conn.close()
    if result:
        return redirect(result[0])
    else:
        return render_template('not_found.html')
```

6) Finally Run the application by typing “flask run” or “python app.py” in vs command prompt

Output:


URL Shortner
Enter URL below:-
URL
Shorten


URL Shortner
Shortened URL below:-
HosINu