# URL Shortener Web Application

## Summary of the Project

This project is a Flask web application that generates shortened URLs for long URLs provided by users. The application uses an SQLite database to store the URLs and their shortened versions.

## Project Workflow

### 1. Importing modules:

The first few lines of code import the necessary modules - Flask, render_template, request, redirect, url_for, random, string, sqlite3, SQLAlchemy, Migrate, and os.

```
from flask import Flask, render_template, request, redirect, url_for
import random
import string
import sqlite3
from flask_sqlalchemy import SQLAlchemy
from flask_migrate import Migrate
import os
```

### 2. Flask setup:

The next few lines of code create a Flask instance and configure it to use an SQLite database. The 'SQLALCHEMY_DATABASE_URI' variable sets the path to the database file, and 'SQLALCHEMY_TRACK_MODIFICATIONS' is set to False to suppress warning messages.

```
app = Flask(__name__)

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

**3. Database setup:**

The 'db' variable is initialized with SQLAlchemy and the 'Migrate' function is called to link the application and the database.

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

**4. Model setup:**

The 'Urls' class is defined with three attributes - 'id_', 'long', and 'short'. The 'id_' attribute is the primary key, and 'long' and 'short' are the original URL and its shortened version, respectively. The 'init' method initializes the attributes with the values passed in as parameters.

```
class Urls(db.Model):
    id_ = db.Column("id_", db.Integer, primary_key=True)
    long = db.Column("Long_URL", db.String())
    short = db.Column("Short_URL", db.String(20))

    def __init__(self, long, short):
        self.long = long
        self.short = short
```

**5. URL shortening function:**

The 'shorten_url' function generates a random string of three characters from the set of lowercase and uppercase letters. It checks if the string already exists in the database, and if not, returns the string.

```
def shorten_url():
    letters = string.ascii_lowercase + string.ascii_uppercase
    while True:
        rand_letters = random.choices(letters, k=3)
        rand_letters = "".join(rand_letters)
        short_url = Urls.query.filter_by(short=rand_letters).first()
        if not short_url:
            return rand_letters
```

**6. Home route:**

The '/' route handles GET and POST requests. If the request method is POST, it receives the long URL from the form data, checks if the URL already exists in the database, and redirects the user to the shortened URL if it does. If not, it generates a new shortened URL using the 'shorten_url' function, saves the URL in the database, and redirects the user to the shortened URL. If the request method is GET, it renders the 'url_page.html' template, which displays the form for submitting a long URL.

```
@app.route('/', methods=['POST', 'GET'])
def home():
    if request.method == "POST":
        url_received = request.form["nm"]
        found_url = Urls.query.filter_by(long=url_received).first()

        if found_url:
            return redirect(url_for("display_short_url", url=found_url.short))
        else:
            short_url = shorten_url()
            print(short_url)
            new_url = Urls(url_received, short_url)
            db.session.add(new_url)
            db.session.commit()
            return redirect(url_for("display_short_url", url=short_url))
    else:
        return render_template('url_page.html')
```

**7. Redirection route:**

The '/<short_url>' route handles requests for shortened URLs. It looks up the shortened URL in the database and redirects the user to the corresponding long URL if it exists. If not, it displays an error message.

```
@app.route('/<short_url>')
def redirection(short_url):
    long_url = Urls.query.filter_by(short=short_url).first()
    if long_url:
        return redirect(long_url.long)
    else:
        return f'<h1>Url doesnt exist</h1>'
```

**8. Display shortened URL route:**

The '/display/<url>' route displays the shortened URL to the user after it has been generated.

```
@app.route('/display/<url>')
def display_short_url(url):
    return render_template('shorturl.html', new_url=url)
```

**9. Display all URLs route:**

The '/history' route displays a list of all URLs in the database.

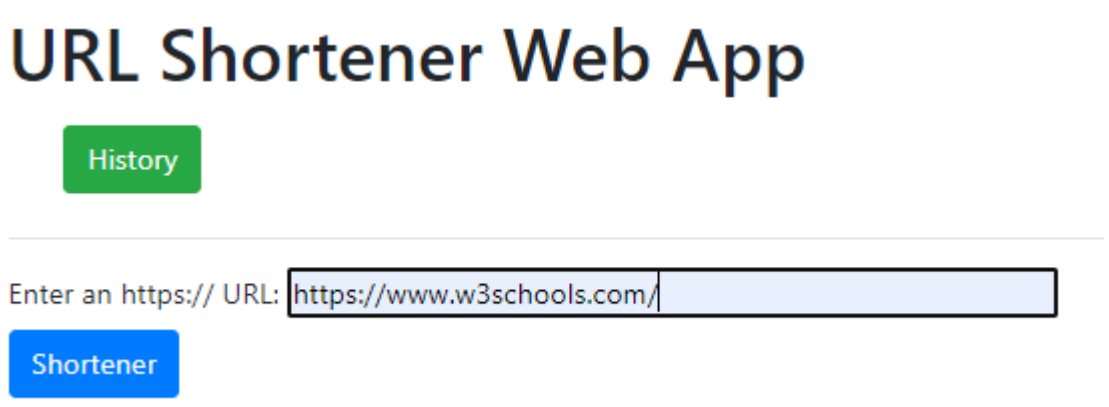
```
@app.route('/history')
def display_all():
    return render_template('history.html', vals=Urls.query.all())
```

**10. Main function:**

The main function runs the application on port 5000 in debug mode.

```
if __name__ == '__main__':
    app.run(port=5000, debug=True)
```

**11. Web App**



## New URL Generation


## DataBase

| Original URL | Short URL |
|---|---|
| https://www.w3schools.com/ | http://127.0.0.1:5000/QHl |
| https://www.sciencedirect.com/search?qs=advantages%20of%20panchgavya&show=50&articleTypes=FLA&lastSelectedFacet=articleTypes | http://127.0.0.1:5000/UUE |
| https://www.youtube.com/ | http://127.0.0.1:5000/apV |
| https://online.innomatics.in/ | http://127.0.0.1:5000/NhF |
| https://www.linkedin.com/feed/ | http://127.0.0.1:5000/fwk |
| https://mail.google.com/mail/u/0/#inbox | http://127.0.0.1:5000/LyE |
| https://chat.openai.com/auth/login | http://127.0.0.1:5000/hJi |
| https://drive.google.com/drive/my-drive | http://127.0.0.1:5000/JNp |
| https://github.com/ | http://127.0.0.1:5000/Slq |

Back to Home

## Conclusion

Overall, the code demonstrates the use of Flask and SQLAlchemy to create a web application that generates, and stores shortened URLs into the database.