Proof of Concept (POC) - URL Shortener

Intern Name: Sayali Vijay Pol

Intern ID: 345

Organization: Digisuraksha parhari foundation

1. Introduction

In the digital era, URLs can often be long, complex, and difficult to share. A URL shortener converts long web addresses into concise, easy-to-remember links that redirect to the original resource. This tool allows users to input a long URL, generate a unique short code, store it in a database, and redirect from the short code to the original page.

2. Objective

The objective of this PoC is to demonstrate a web-based URL shortener that takes a long URL, generates a short code, stores it in SQLite, and redirects to the original URL when accessed.

3. Background

Sharing long URLs can be inconvenient, especially in contexts where space is limited or aesthetic appearance matters. URL shorteners make sharing cleaner and easier, improving usability in social media, emails, and printed materials.

4. Scope

This tool accepts long URLs from a user, generates a 6-character alphanumeric slug, stores it in a SQLite database, and returns a short URL that redirects to the original link.

5. Tools & Technologies Used

Component	Technology
Language	Python 3
Framework	Flask
Database	SQLite
Frontend	HTML
Slug Generation	Random alphanumeric strings

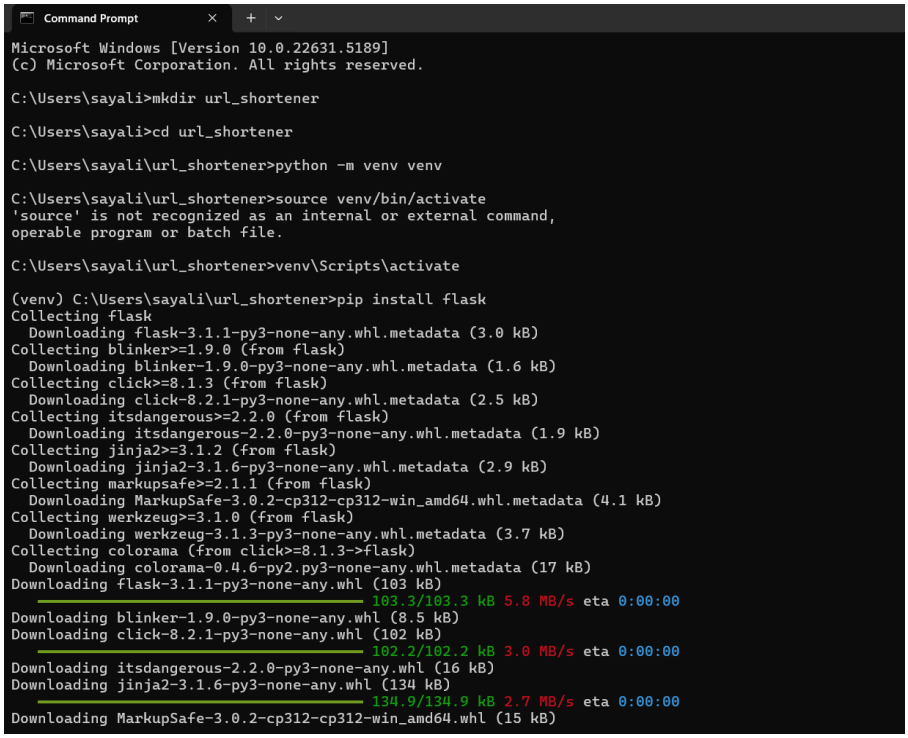
6. Workflow Diagram

User → Form Input → Flask Backend → Generate Slug → Store in DB → Return Short URL →

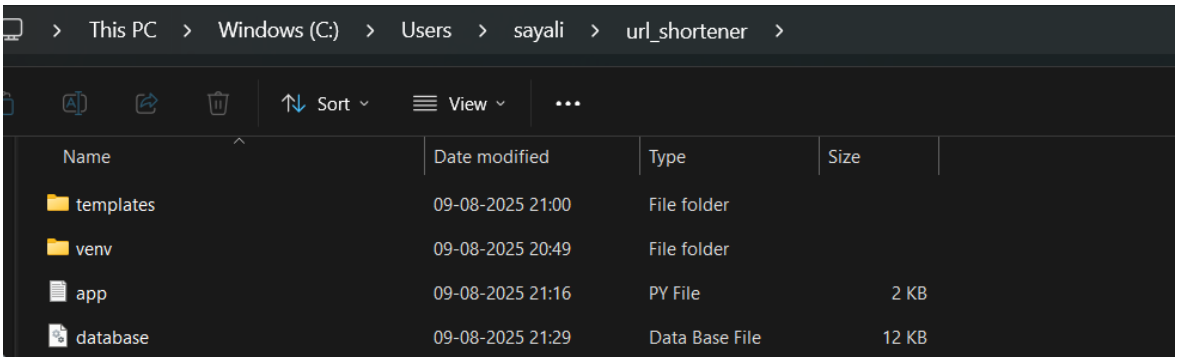
Redirect to Original URL7.

7.Implementation Steps

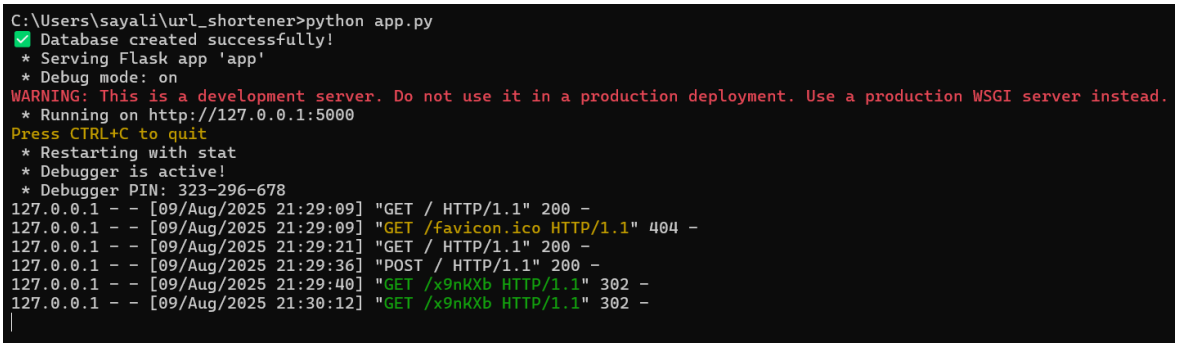
Step 1: Setup the environment.



Step 2: Create project structure.



Step 3: Database setup.



Step 4: Flask application backend code.

```
from flask import Flask, request, redirect, render_template

import string, random, sqlite3, os


app = Flask(__name__)


def init_db():
    if not os.path.exists('database.db'):
        conn = sqlite3.connect('database.db')
        conn.execute('CREATE TABLE urls (id INTEGER PRIMARY KEY AUTOINCREMENT, slug TEXT, original_url TEXT)')
```

```
        conn.commit()
        conn.close()
        print("☑ Database created successfully!")

# Database connection
def get_db_connection():
    conn = sqlite3.connect('database.db')
    conn.row_factory = sqlite3.Row
    return conn

def generate_slug():
    return ''.join(random.choices(string.ascii_letters + string.digits, k=6))

# Routes
@app.route('/', methods=['GET', 'POST'])
def index():
    short_url = None
    if request.method == 'POST':
        original_url = request.form['url']
        slug = generate_slug()

        conn = get_db_connection()
        conn.execute('INSERT INTO urls (slug, original_url) VALUES (?, ?)', (slug, original_url))
        conn.commit()
        conn.close()
```

```
        short_url = request.host_url + slug

    return render_template('index.html', short_url=short_url)

@app.route('/<slug>')
def redirect_to_url(slug):
    conn = get_db_connection()
    url_data = conn.execute('SELECT original_url FROM urls WHERE slug = ?', (slug,)).fetchone()
    conn.close()

    if url_data:
        return redirect(url_data['original_url'])
    else:
        return "URL not found", 404

# Main entry point
if __name__ == '__main__':
    init_db()  # Create DB if missing
    app.run(debug=True)
```

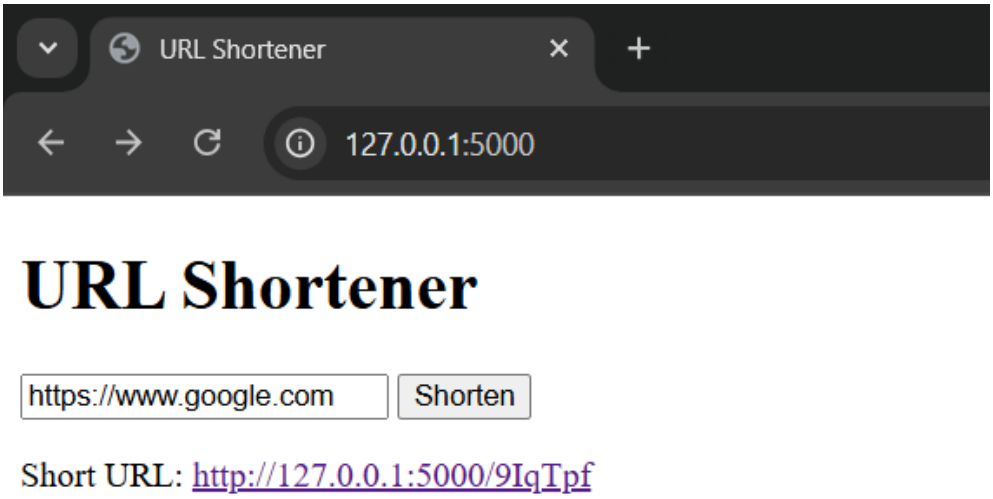
Step 5: HTML form creation.

```
<!DOCTYPE html>
<html>
<head>
```

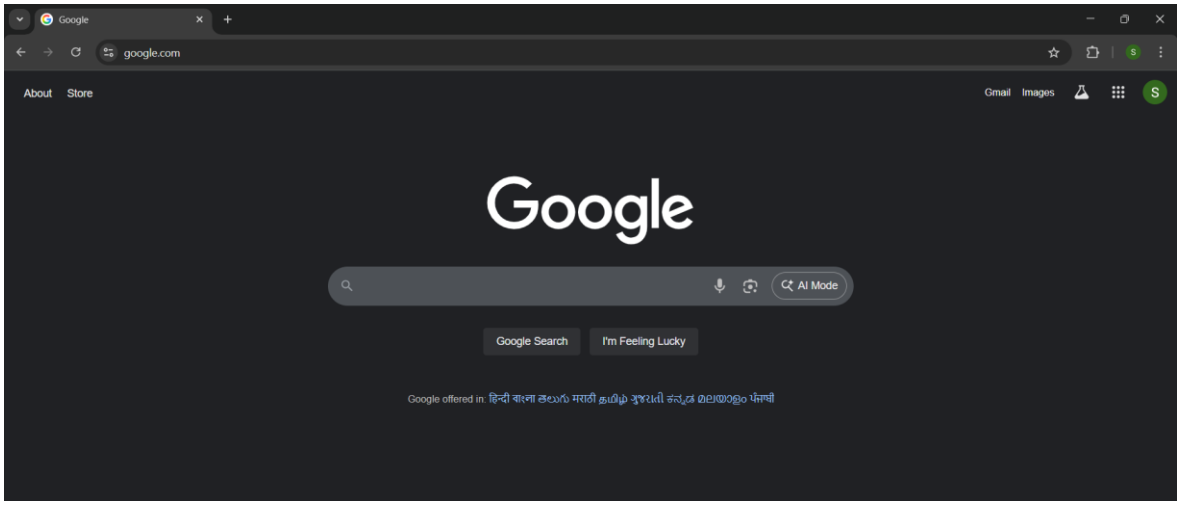
```
    <title>URL Shortener</title>
</head>
<body>
    <h1>URL Shortener</h1>
    <form method="POST">
        <input type="text" name="url" placeholder="Enter your long URL" required>
        <button type="submit">Shorten</button>
    </form>

    {% if short_url %}
        <p>Short URL: <a href="{{ short_url }}">{{ short_url }}</a></p>
    {% endif %}
</body>
</html>
```

Step 6: Short code generation.



Step 7: Testing redirection.



8. Conclusion

This PoC demonstrates a functional URL shortener capable of generating short, shareable links that redirect to original URLs. With enhancements, it can be deployed for production use.