

# **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 URL.

## 7. Implementation Steps

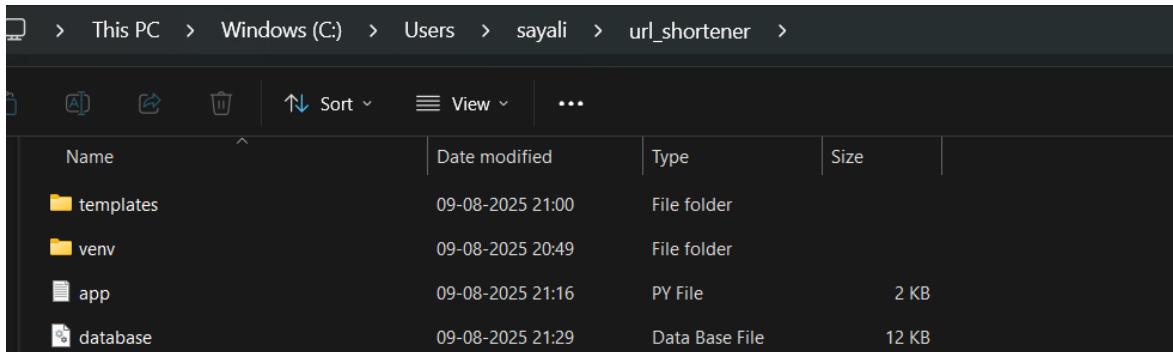
Step 1: Setup the environment.

```
Command Prompt
Microsoft Windows [Version 10.0.22631.5189]
(c) Microsoft Corporation. All rights reserved.

C:\Users\sayali>mkdir url_shortener
C:\Users\sayali>cd url_shortener
C:\Users\sayali\url_shortener>python -m venv venv
C:\Users\sayali\url_shortener>source venv/bin/activate
'source' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\sayali\url_shortener>venv\Scripts\activate
(venv) C:\Users\sayali\url_shortener>pip install flask
Collecting flask
  Downloading flask-3.1.1-py3-none-any.whl.metadata (3.0 kB)
Collecting blinker>=1.9.0 (from flask)
  Downloading blinker-1.9.0-py3-none-any.whl.metadata (1.6 kB)
Collecting click>=8.1.3 (from flask)
  Downloading click-8.2.1-py3-none-any.whl.metadata (2.5 kB)
Collecting itsdangerous>=2.2.0 (from flask)
  Downloading itsdangerous-2.2.0-py3-none-any.whl.metadata (1.9 kB)
Collecting jinja2>=3.1.2 (from flask)
  Downloading jinja2-3.1.6-py3-none-any.whl.metadata (2.9 kB)
Collecting markupsafe>=2.1.1 (from flask)
  Downloading MarkupSafe-3.0.2-cp312-cp312-win_amd64.whl.metadata (4.1 kB)
Collecting werkzeug>=3.1.0 (from flask)
  Downloading werkzeug-3.1.3-py3-none-any.whl.metadata (3.7 kB)
Collecting colorama (from click>=8.1.3->flask)
  Downloading colorama-0.4.6-py2.py3-none-any.whl.metadata (17 kB)
Downloading flask-3.1.1-py3-none-any.whl (103 kB) ━━━━━━━━ 103.3/103.3 kB 5.8 MB/s eta 0:00:00
Downloading blinker-1.9.0-py3-none-any.whl (8.5 kB)
Downloading click-8.2.1-py3-none-any.whl (102 kB) ━━━━━━━━ 102.2/102.2 kB 3.0 MB/s eta 0:00:00
Downloading itsdangerous-2.2.0-py3-none-any.whl (16 kB)
Downloading jinja2-3.1.6-py3-none-any.whl (134 kB) ━━━━━━━━ 134.9/134.9 kB 2.7 MB/s eta 0:00:00
Downloading MarkupSafe-3.0.2-cp312-cp312-win_amd64.whl (15 kB)
```

## Step 2: Create project structure.



A screenshot of a Windows File Explorer window. The path is displayed at the top: This PC > Windows (C:) > Users > sayali > url\_shortener >. The main area shows a list of files and folders:

Name	Date modified	Type	Size
templates	09-08-2025 21:00	File folder	
venv	09-08-2025 20:49	File folder	
app	09-08-2025 21:16	PY File	2 KB
database	09-08-2025 21:29	Data Base File	12 KB

## Step 3: Database setup.

```
C:\Users\sayali\url_shortener>python app.py
✓ Database created successfully!
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 323-296-678
127.0.0.1 - - [09/Aug/2025 21:29:09] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [09/Aug/2025 21:29:09] "GET /favicon.ico HTTP/1.1" 404 -
127.0.0.1 - - [09/Aug/2025 21:29:21] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [09/Aug/2025 21:29:36] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [09/Aug/2025 21:29:40] "GET /x9nKxb HTTP/1.1" 302 -
127.0.0.1 - - [09/Aug/2025 21:30:12] "GET /x9nKxb HTTP/1.1" 302 -
```

## 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 %>

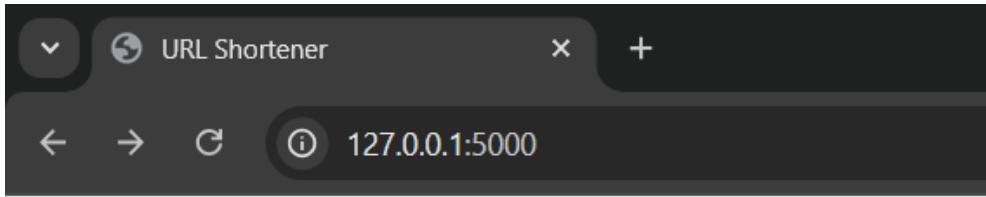
<p>Short URL: <a href="{{ short_url }}>{{ short_url }}</a></p>

<% endif %>

</body>

</html>
```

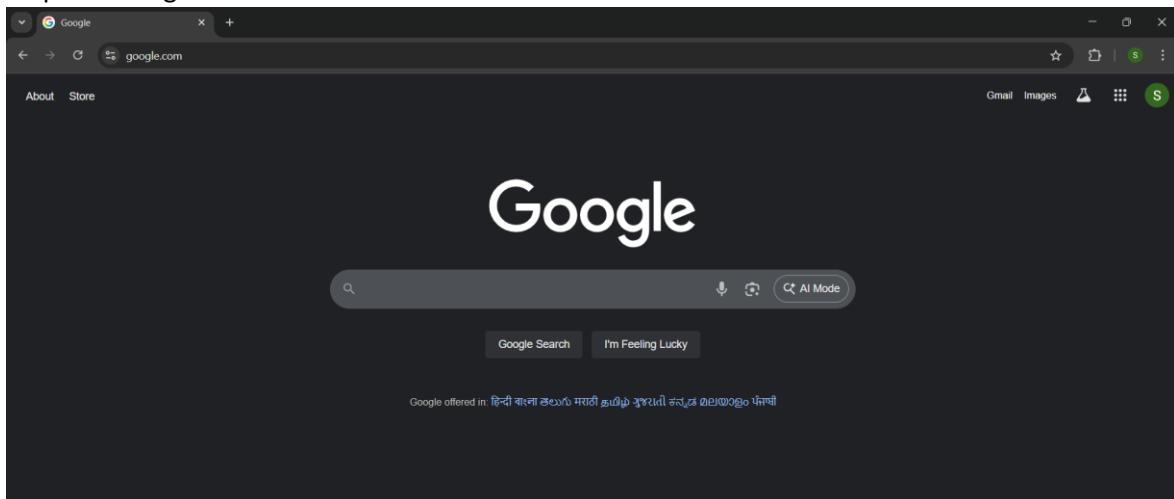
Step 6: Short code generation.



## URL Shortener

Short URL: <http://127.0.0.1:5000/9IqTpF>

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.