URL Shortener Tool

Analyst: Vinit Chaskar
Intern ID: 395
Date: 03/08/2025
Tool Name: URL Shortener
Category: Web Tools / Backend Development / Productivity
Platform: Web-based (Python Flask or Node.js)

■ Objective

To design and implement a simple web-based URL shortener that takes a long URL and returns a unique, shorter version. This functionality mimics popular services like bit.ly or tinyurl.com.

■ Use Case

URL shorteners are commonly used in social media, SMS marketing, and analytics tracking to:

- Make long URLs user-friendly and shareable.
- Hide complex query parameters.
- Track clicks and redirections.
- Integrate with branding and campaign monitoring.

■ Technology Stack

- **Backend Framework**: Flask (Python) / Express (Node.js)
- **Database**: SQLite (lightweight) or in-memory dictionary
- **Short Code Generation**: Base62 encoding or random string generator
- **Frontend**: Basic HTML form for user input

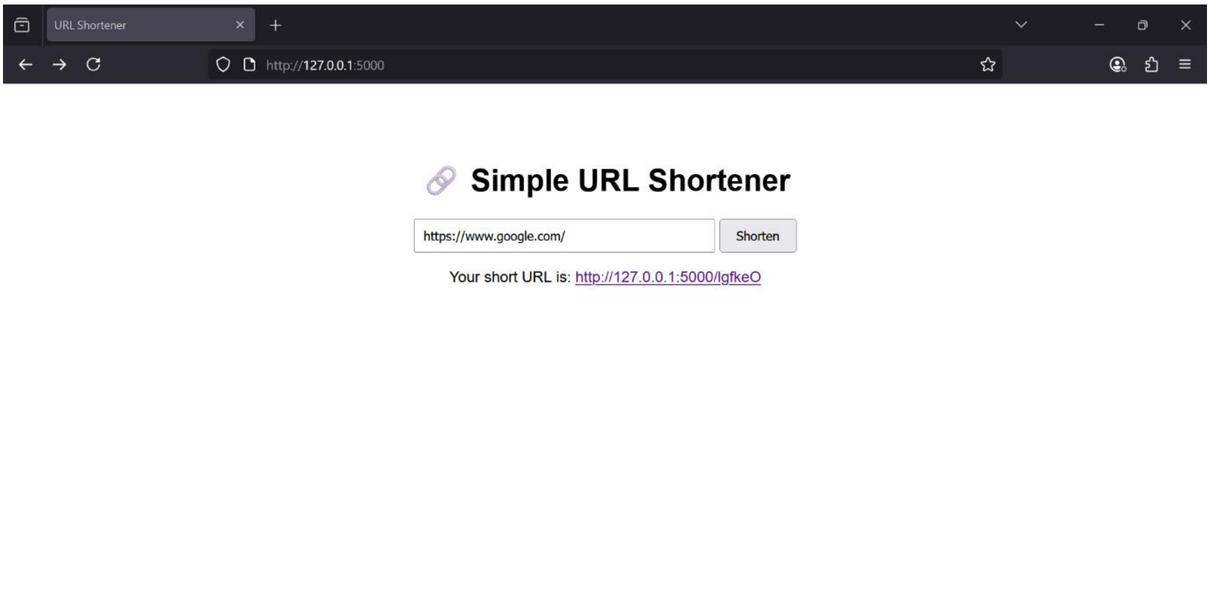
■ How It Works

1. **Input**: User enters a long URL in a web form.
2. **Short Code Generation**:
   - Either randomly generates a 6–8 character string OR
   - Encodes an auto-incremented ID using base62.
3. **Mapping**:
   - Stores the long URL and its corresponding short code in a database/dictionary.
4. **Redirection**:
   - When a user visits the short URL, it redirects them to the original long URL.

■ Features

- Auto-generates unique short codes
- Basic input validation (to avoid malformed URLs)
- Permanent redirect (HTTP 301 or 302)
- Option to store click counts (optional extension)

■ Screenshot



🔗 Simple URL Shortener

https://www.google.com/

Your short URL is: http://127.0.0.1:5000/lgfkeO

■ Code Snippet

app.py

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

app = Flask(__name__)

DATABASE = "database.db"
```

```
# ----- Utility Functions -----

def get_db_connection():
    conn = sqlite3.connect(DATABASE)
    conn.row_factory = sqlite3.Row
    return conn

def init_db():
    if not os.path.exists(DATABASE):
        conn = get_db_connection()
        conn.execute('''
            CREATE TABLE urls (
                id INTEGER PRIMARY KEY AUTOINCREMENT,
                original_url TEXT NOT NULL,
                slug TEXT UNIQUE NOT NULL
            );
        ''')
        conn.commit()
        conn.close()

def generate_slug(length=6):
    characters = string.ascii_letters + string.digits
    return ''.join(random.choice(characters) for _ in range(length))

def get_slug_for_url(url):
    conn = get_db_connection()
    cur = conn.execute('SELECT slug FROM urls WHERE original_url = ?', (url,))
    row = cur.fetchone()
    conn.close()
    return row['slug'] if row else None
```

```
def store_url(original_url):
    # Check if URL already exists
    existing_slug = get_slug_for_url(original_url)
    if existing_slug:
        return existing_slug

    conn = get_db_connection()
    while True:
        slug = generate_slug()
        try:
            conn.execute('INSERT INTO urls (original_url, slug) VALUES (?, ?)', (original_url, slug))
            conn.commit()
            conn.close()
            return slug
        except sqlite3.IntegrityError:
            # Collision — regenerate slug
            continue

def get_original_url(slug):
    conn = get_db_connection()
    cur = conn.execute('SELECT original_url FROM urls WHERE slug = ?', (slug,))
    row = cur.fetchone()
    conn.close()
    return row['original_url'] if row else None

# ----- Flask Routes -----

@app.route('/', methods=['GET', 'POST'])
def index():
    if request.method == 'POST':
        original_url = request.form['url']
        slug = store_url(original_url)
        short_url = request.host_url + slug
        return render_template('index.html', short_url=short_url)
    return render_template('index.html', short_url=None)

@app.route('/<slug>')
def redirect_to_original(slug):
    original_url = get_original_url(slug)
    if original_url:
        return redirect(original_url)
    else:
        return "❌ Invalid or expired short URL.", 404

# ----- App Start -----

if __name__ == '__main__':
    init_db()
    app.run(debug=True)
```

templates/index.html

```
<!DOCTYPE html>
<html>
<head>
    <title>URL Shortener</title>
    <style>
        body { font-family: Arial; text-align: center; padding: 50px; }
        input[type=text] { width: 300px; padding: 8px; }
        input[type=submit] { padding: 8px 16px; }
    </style>
</head>
<body>
    <h1>🔗 Simple URL Shortener</h1>
    <form method="POST">
        <input type="text" name="url" placeholder="Enter long URL" required>
        <input type="submit" value="Shorten">
    </form>

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