Weekly Task Document 2: URL Shortener

Intern Name: Sumith Bangera

Intern ID: 130

Objective

Design and implement a basic web-based URL Shortener that accepts a long URL and returns a shortened link, allowing redirection using the shortened URL. The project simulates services like bit.ly or tinyurl.com.

Task Overview

The URL shortener consists of a frontend HTML form, a Flask backend API, and a short code generation system. When a user submits a long URL, the backend generates a unique alphanumeric code and stores it in a mapping table. Accessing the short link redirects the user to the original long URL.

Key Concepts Used

• Flask web framework for backend routing and handling requests. • In-memory dictionary for temporary storage of mappings. • Random alphanumeric slug generation for unique short codes. • HTML form for user input and displaying shortened URLs.

Code Snippet

```
from flask import Flask, request, redirect, render_template_string
import string, random

app = Flask(__name__)

# In-memory storage for URL mappings
url_mapping = {}

# Function to generate short slug
def generate_short_slug(length=6):
    characters = string.ascii_letters + string.digits
    return ''.join(random.choice(characters) for _ in range(length))

# HTML template
form_html = """
<!doctype html>
<title>URL Shortener</title>
<h2>Simple URL Shortener</h2>
<form method="POST" action="/shorten">
    <input type="url" name="long_url" placeholder="Enter long URL" required style="width:300px">
    <input type="submit" value="Shorten">
</form>
{% if short_url %}
    <p>Shortened URL: <a href="{{ short_url }}">{{ short_url }}</a></p>
{% endif %}
"""

@app.route("/", methods=["GET"])
def index():
    return render_template_string(form_html)

@app.route("/shorten", methods=["POST"])
def shorten_url():
    long_url = request.form.get("long_url")
    slug = generate_short_slug()
    url_mapping[slug] = long_url
    short_url = request.host_url + slug
    return render_template_string(form_html, short_url=short_url)

@app.route("/<slug>")
def redirect_to_url(slug):
    if slug in url_mapping:
        return redirect(url_mapping[slug])
    return "Invalid URL", 404

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

Challenges Faced

• Ensuring short code uniqueness to prevent collisions. • Maintaining mappings between sessions (requires database integration for persistence). • Handling invalid or malicious URLs safely. • Designing a minimal but functional frontend.

Conclusion

A functional proof-of-concept URL Shortener was created using Flask and in-memory storage. The system successfully generates short links and redirects users to the original URLs. This can be extended with persistent storage, user authentication, and analytics tracking.