Proof of Concept (POC): Custom URL Shortener Web App

Name: Akkshat shah

Intern id: 261

🧠 Objective

To build a lightweight and user-friendly web application that converts long URLs into short and manageable links. The tool also allows users to generate **custom slugs**, similar to services like `tinyurl.com`, so that the shortened links are easy to remember and share.

🗒️ Overview

This web tool is built using **Python Flask** and a simple HTML frontend. It allows users to:

- Enter a **long URL** (e.g., https://example.com/some/very/long/link)
- Optionally add a **custom slug** (e.g., `mycustomlink`)
- Generate a shortened version like: `http://localhost:5000/mycustomlink`

When a user clicks on this short link, they are automatically **redirected** to the original long URL.

💡 Key Features

- Clean, minimal **HTML UI**
- **Random short code** generation if custom slug not provided
- **Custom slug** support with duplication checks

- **In-memory storage** (can be extended to SQLite)
- URL **redirection route**
- Fully working on localhost via Flask
- Lightweight and easily deployable

🧪 How It Works

🔷 Input:

User enters:

- A long URL
- (Optional) a custom short code (slug)

🔷 Backend:

- If a **custom slug** is provided, it checks if it's already used.
- If it's taken → shows error.
- If not → stores it in a Python dictionary (`url_map`).
- If no custom slug is given → generates a **random 6-character string**.

🔷 Output:

- Displays the final short link (e.g., `http://localhost:5000/mylink`)
- Clicking the link redirects the user to the original URL

🔐 Core Logic

```
# Check if custom slug is provided and available
if custom_slug:
    if custom_slug in url_map:
```

```
        return "Slug already taken"
    else:
        url_map[custom_slug] = long_url
```

If no slug is provided:

```
# Generate random short code
slug = generate_slug()
while slug in url_map:
    slug = generate_slug()
url_map[slug] = long_url
```

When accessing the short URL:

```
@app.route("/<slug>")
def redirect_to_long(slug):
    return redirect(url_map.get(slug))
```

🖥️ UI Preview (HTML Form)

```
<form method="POST">
    <input type="text" name="long_url" placeholder="Enter long URL">
    <input type="text" name="custom_slug" placeholder="Optional custom slug">
    <button type="submit">Shorten</button>
</form>
```

✅ Sample Run

Example Input:

Long URL: https://instagram.com
Custom Slug: insta

Output:

Short URL: http://localhost:5000/insta

Clicking this will redirect to: https://instagram.com

🔮 Future Upgrades

- Save data in SQLite or Firebase
- Track click analytics (views, timestamps)
- Add QR code generation
- Build a web dashboard to manage links
- Deploy online using Replit, PythonAnywhere, or Render

🧾 Conclusion

This project shows how simple it is to implement a working custom URL shortener using Flask. It helps users clean up long URLs, create custom branded links, and provides a foundation for more advanced URL tracking and redirection systems. It’s ideal for internal tools, portfolio projects, and educational demos.
