Proof of Concept - URL Shortner

Name : Shivani Yadav
Intern ID - 279

Objectives:

To convert long, complex URLs into short, easy-to-share links while keeping a mapping between the short link and the original URL, so that when users click the short link, they are redirected to the original address.

Key points in the objective:

- Simplification → Makes links shorter and easier to remember or share.
- Redirection → Short links still take users to the same destination.
- Tracking (optional) → Can collect analytics like clicks, location, and time.
- Storage → Maintains a mapping between the short code and the original URL.

Scope:
Core Functions

- Accept a long URL from the user.
- Generate a unique short code (slug).
- Store the mapping between the short code and the original URL in a database.

Possible Enhancements

- Custom short codes chosen by the user.
- Analytics tracking (number of clicks, user location, device type).
- Expiry dates for links (temporary access).

Real-World Usage

- Social media (Twitter, Instagram bios) where space is limited.
- Marketing campaigns (track link clicks).
- Masking long affiliate/referral URLs.

Technology Stack:
Backend

- **Language:** Python (Flask framework) – handles routing, link creation, and redirection.
- **Alternative:** Node.js (Express), Java (Spring Boot), PHP (Laravel) — depending on preference.

Database

- **SQLite** (lightweight, file-based database) — stores the mapping between short codes and original URLs.
- Alternatives: MySQL, PostgreSQL, MongoDB, or Redis for faster lookups.

Frontend

- **HTML + CSS** (via Flask templates) — provides a simple form for entering the long URL and displaying the short link.
- Optional: JavaScript for a smoother user experience.

Workflow:

1. **User Input**
    - The user enters a long URL into a form (e.g., a web page or API request).

2. **Validation**
    - The system checks if the input is a valid URL format (http:// or https://).
    - Optionally, it checks if the domain is safe (not malicious).

3. **Slug Generation**
    - The system creates a unique short code (slug), usually 5–8 random alphanumeric characters.
    - Example: 7bynSP

4. **Database Storage**
    - The slug and original URL are stored in a database.

5. **Short URL Creation**
    - The short URL is formed by combining the service's domain with the slug.

6. **User Click / Access**
    - A user visits the short URL in their browser.

7. **Database Lookup**
    - The system searches the database for the slug.
    - If found, it retrieves the corresponding original URL.

8. **Redirection**
    - The system sends an HTTP redirect response, taking the user to the original long URL.

Example:

Input (long URL):

`https://www.google.com/search?q=java`

Generated short URL:

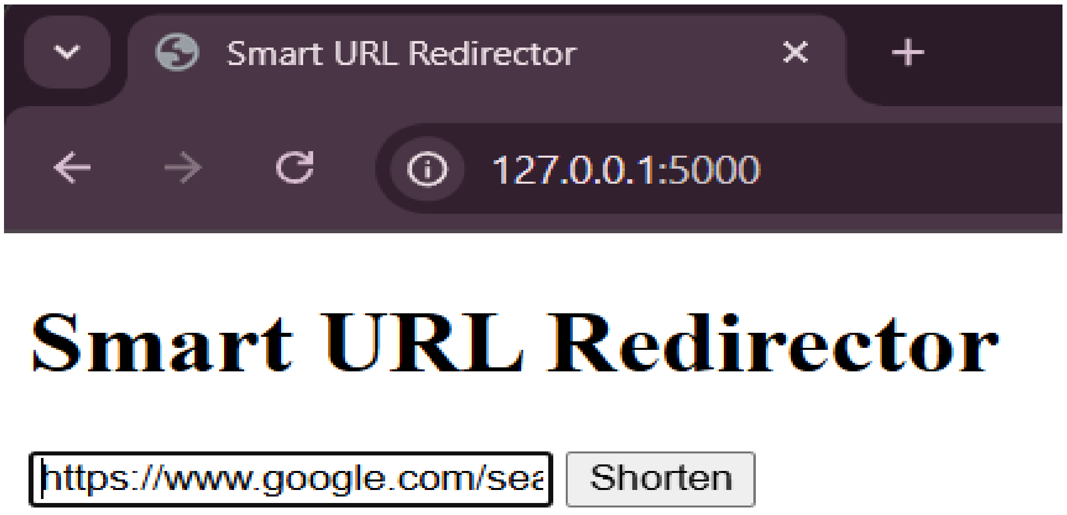
http://127.0.0.1:5000/xY98Za

Redirects to:

https://www.youtube.com

Screenshots:

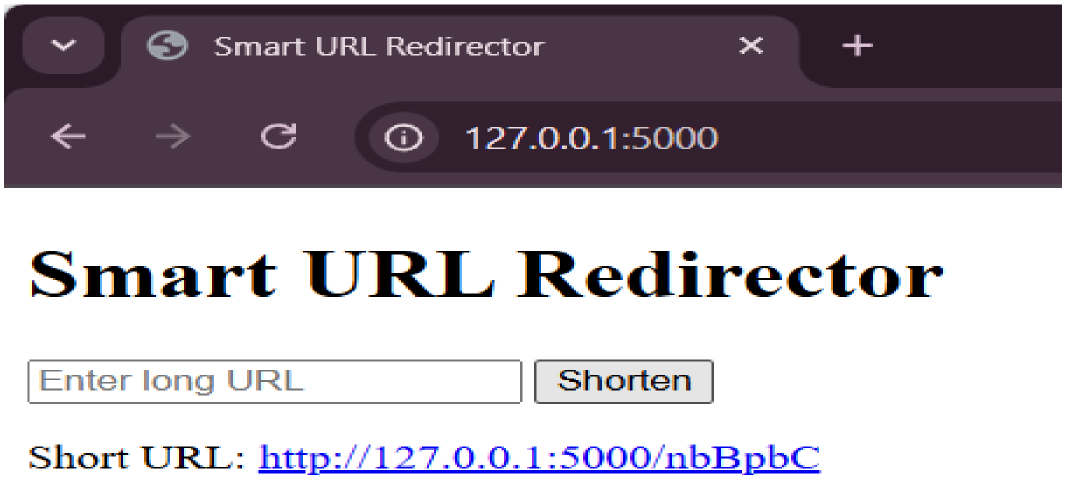
- **Input:**



Smart URL Redirector

https://www.google.com/sea

- **Output(Short URL):**

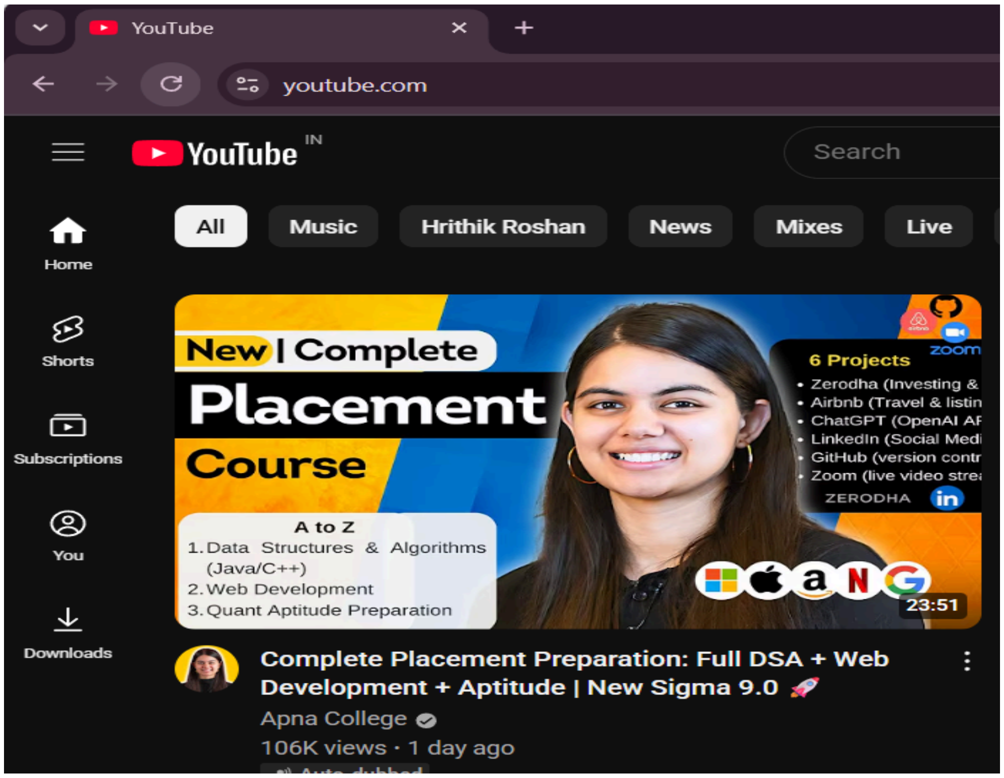


Smart URL Redirector

Enter long URL | Shorten

Short URL: http://127.0.0.1:5000/nbBpbC

- **Redirects to:**



Complete Placement Preparation: Full DSA + Web Development + Aptitude | New Sigma 9.0 🚀
Apna College
106K views · 1 day ago

Security Consideration:

1. **Validate User Input**
   - Ensure the submitted URL is in a valid format (http:// or https://).
   - Block malformed or non-URL strings.

2. **Prevent Malicious Links**
   - Check against a blacklist or Google Safe Browsing API to block phishing/malware sites.
   - Warn users if the destination looks suspicious.

3. **Avoid Duplicate Slugs**
   - Before inserting a new short code, check if it already exists in the database.

4. **Sanitize Database Inputs**
   - Use parameterized queries or ORM to prevent SQL Injection.

5. **HTTPS for Deployment**
   - Secure all traffic to prevent sniffing of URLs.

6. **Access Controls (Optional)**
   - If there's an admin dashboard, require authentication to manage links.

7. **Prevent Open Redirect Exploits**
   - Ensure the system only redirects to URLs stored in its database.

8. **Rate Limiting**
   - Prevent spam by limiting how many URLs a single user/IP can shorten in a given time.

Future Improvements:

1. **Base62 Encoding for Short Codes**
    - Instead of random strings, use sequential IDs encoded in Base62 (`0-9`, `A-Z`, `a-z`) for shorter and more predictable slugs.

2. **User Accounts & Dashboard**
    - Let users sign up and manage all their short links in one place.

3. **Click Analytics**
    - Track number of clicks, location, device type, and referrer for each link.

4. **Expiry Dates**
    - Allow links to automatically expire after a set time or number of clicks.

5. **Custom Short Codes**
    - Let users choose their own slug instead of random ones.

6. **QR Code Generation**
    - Automatically create a QR code for each shortened link.

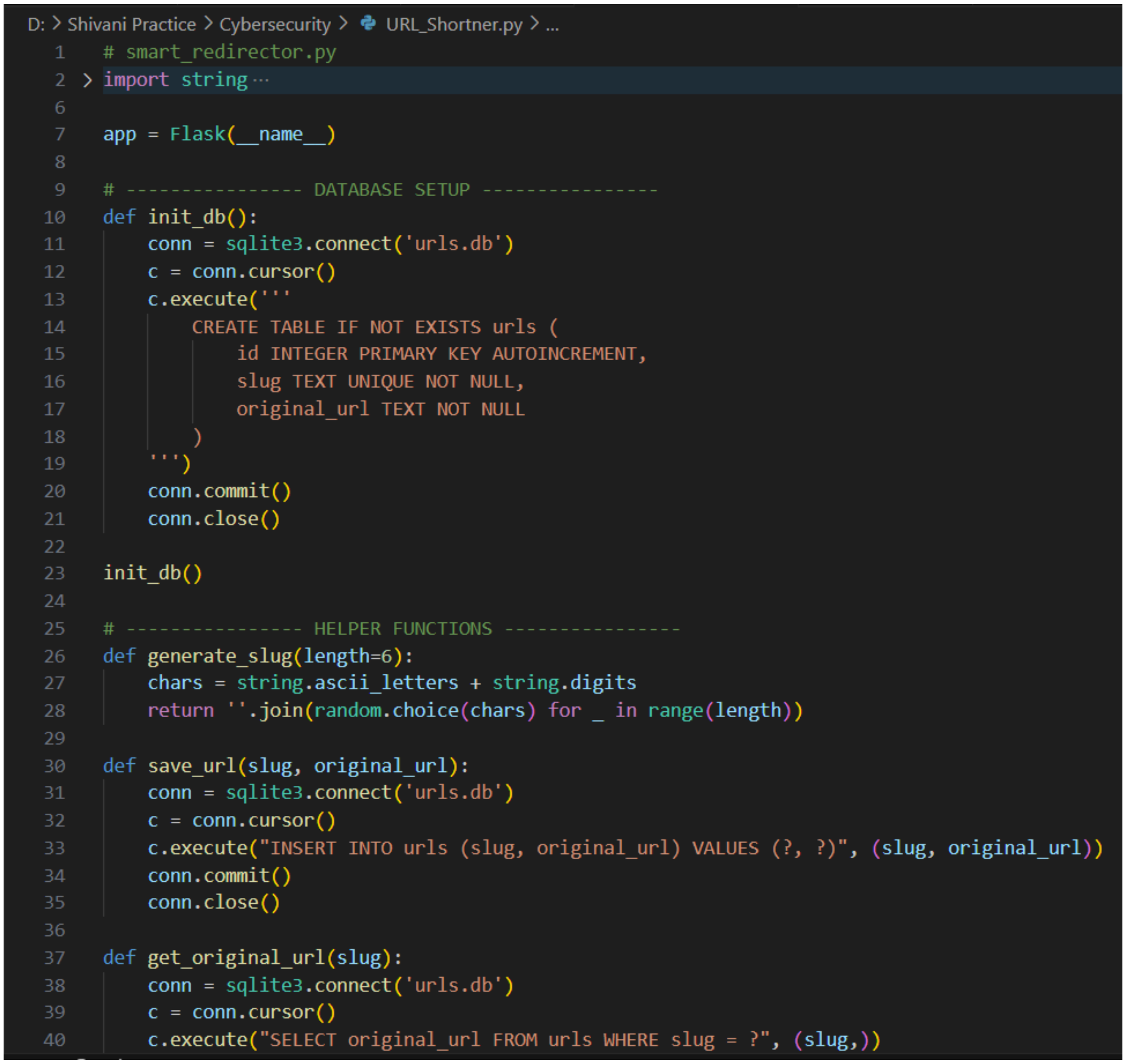
7. **Password Protection**
    - Require a password before redirecting to the original URL.

8. **Preview Page**
    - Show a preview of the destination before redirecting (to prevent phishing).

9. **Integration with APIs**
    - Provide an API so other applications can shorten links programmatically.

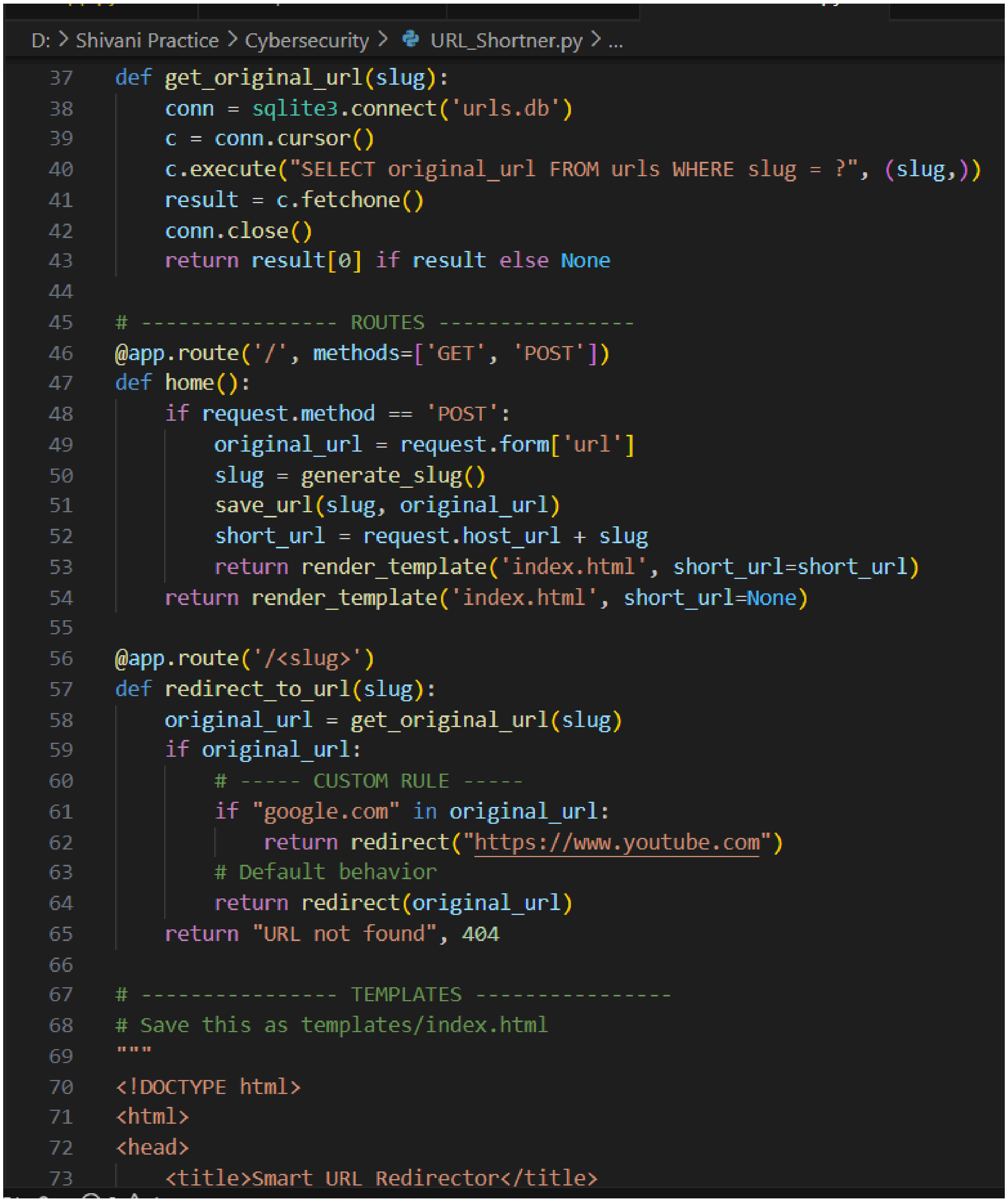
10. **Better UI/UX**
    - Responsive design, modern frontend framework, and mobile-friendly interface.

Code:



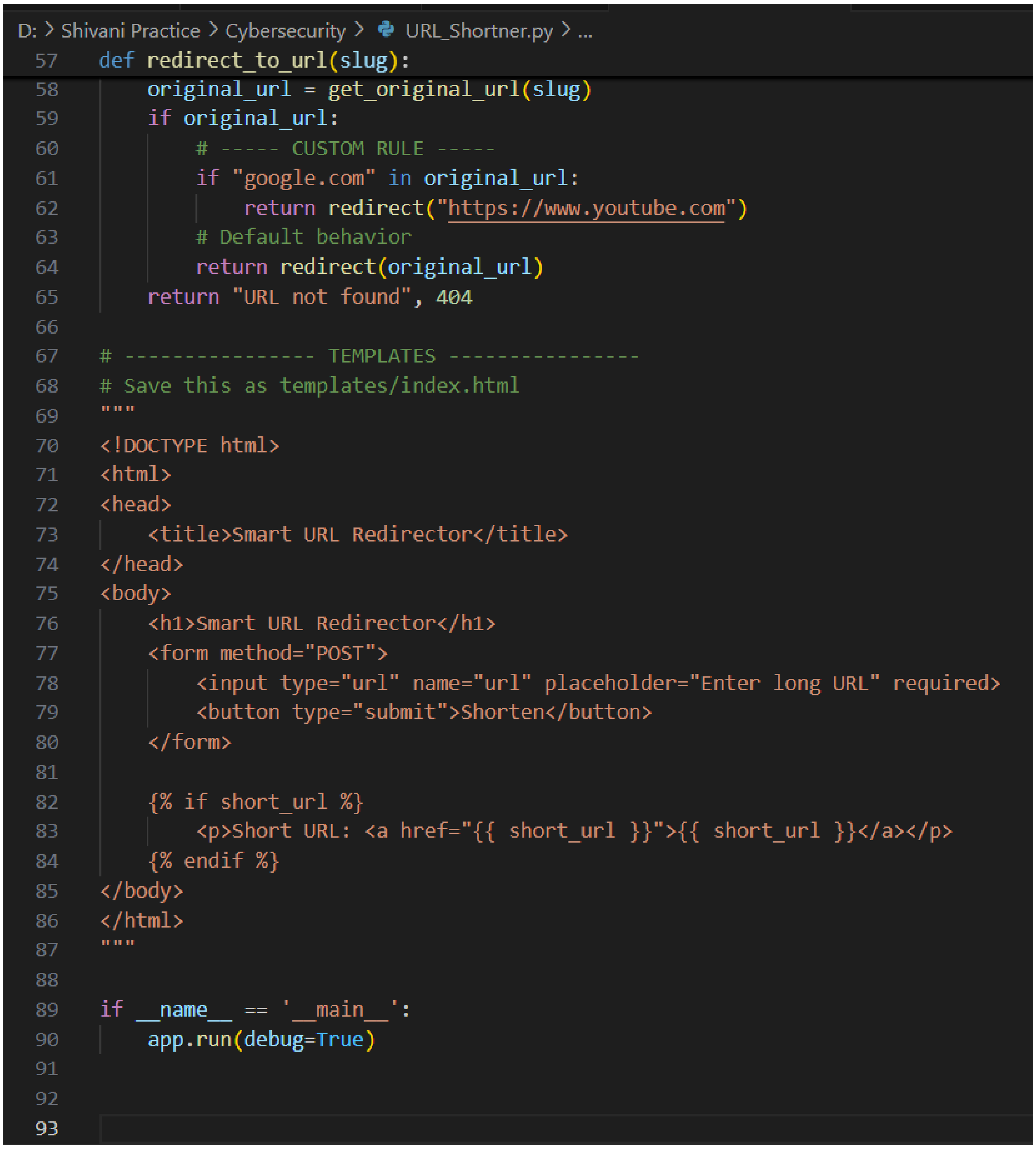
D: > Shivani Practice > Cybersecurity > URL_Shortner.py > ...

```python
# smart_redirector.py
import string ···

app = Flask(__name__)

# ---------------- DATABASE SETUP ----------------
def init_db():
    conn = sqlite3.connect('urls.db')
    c = conn.cursor()
    c.execute('''
        CREATE TABLE IF NOT EXISTS urls (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            slug TEXT UNIQUE NOT NULL,
            original_url TEXT NOT NULL
        )
    ''')
    conn.commit()
    conn.close()

init_db()

# ---------------- HELPER FUNCTIONS ----------------
def generate_slug(length=6):
    chars = string.ascii_letters + string.digits
    return ''.join(random.choice(chars) for _ in range(length))

def save_url(slug, original_url):
    conn = sqlite3.connect('urls.db')
    c = conn.cursor()
    c.execute("INSERT INTO urls (slug, original_url) VALUES (?, ?)", (slug, original_url))
    conn.commit()
    conn.close()

def get_original_url(slug):
    conn = sqlite3.connect('urls.db')
    c = conn.cursor()
    c.execute("SELECT original_url FROM urls WHERE slug = ?", (slug,))
```



D: > Shivani Practice > Cybersecurity > URL_Shortner.py > ...

```python
def get_original_url(slug):
    conn = sqlite3.connect('urls.db')
    c = conn.cursor()
    c.execute("SELECT original_url FROM urls WHERE slug = ?", (slug,))
    result = c.fetchone()
    conn.close()
    return result[0] if result else None

# ---------------- ROUTES ----------------
@app.route('/', methods=['GET', 'POST'])
def home():
    if request.method == 'POST':
        original_url = request.form['url']
        slug = generate_slug()
        save_url(slug, 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_url(slug):
    original_url = get_original_url(slug)
    if original_url:
        # ----- CUSTOM RULE -----
        if "google.com" in original_url:
            return redirect("https://www.youtube.com")
        # Default behavior
        return redirect(original_url)
    return "URL not found", 404

# ---------------- TEMPLATES ----------------
# Save this as templates/index.html
"""
<!DOCTYPE html>
<html>
<head>
    <title>Smart URL Redirector</title>
```



D: > Shivani Practice > Cybersecurity > URL_Shortner.py > ...

```
def redirect_to_url(slug):
    original_url = get_original_url(slug)
    if original_url:
        # ----- CUSTOM RULE -----
        if "google.com" in original_url:
            return redirect("https://www.youtube.com")
        # Default behavior
        return redirect(original_url)
    return "URL not found", 404

# ---------------- TEMPLATES ----------------
# Save this as templates/index.html
"""
<!DOCTYPE html>
<html>
<head>
    <title>Smart URL Redirector</title>
</head>
<body>
    <h1>Smart URL Redirector</h1>
    <form method="POST">
        <input type="url" name="url" placeholder="Enter 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>
"""

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


```