TASK: URL Shortner

Name:Samiksha Patil.

Intern ID:280

what is URL shortner ?

A **URL shortener** is a tool or service that takes a **long web address (URL)** and converts it into a **short, easy-to-share version**.

Why Use a URL Shortener?

1. **Easier to Share**
   Especially useful on social media, in messages, or on printed material where space is limited.
2. **Cleaner Appearance**
   Long URLs can look messy; short ones are neat and professional.
3. **Track Clicks (Analytics)**
   Some URL shorteners let you track how many times the link was clicked and from where.
4. **Mask Complex URLs**
   Useful when hiding complex or technical URLs from users.

How Does It Work ?

1. **User inputs** a long URL (like a full website link).
2. The system **generates a unique short code** (e.g., abc123).
3. It **stores a mapping** between the short code and the original URL.
4. When someone visits the short link (short.ly/abc123), the system:
   - Looks up the short code
   - Finds the matching long URL
   - **Redirects** the user to the original site

Real-World Examples

- **bit.ly**
- **tinyurl.com**
- **t.co** (used by Twitter)
- **goo.gl** (retired by Google)

Technologi Stack:

1. Frontend (User Interface)

- **Purpose:** Let users enter long URLs, see shortened ones, and manage links.
- **Technologies:**
  - **Languages:** HTML5, CSS3, JavaScript (or TypeScript)
  - **Frameworks/Libraries:** React.js / Vue.js / Angular
  - **UI Styling:** Tailwind CSS or Bootstrap

---

2. Backend (Core Logic)

- **Purpose:** Handle requests, generate short codes, redirect users, manage analytics.
- **Technologies:**
  - **Language:**
    - JavaScript (Node.js + Express.js) **or**
    - Python (Flask / Django) **or**
    - Go / Java (Spring Boot)
  - **Key Features Implemented:**
    - Short code generation (hashing or random string)
    - Mapping between short code and original URL
    - Analytics tracking (click counts, geolocation, timestamps)
    - REST API endpoints (e.g., POST /shorten, GET /:code)

---

3. Database

- **Purpose:** Store mappings of short code → long URL, plus analytics data.
- **Technologies:**
  - **Primary Data Store:**
    - Relational: PostgreSQL / MySQL
    - NoSQL: MongoDB / DynamoDB
  - **Caching (optional, for performance):** Redis (for quick lookups)

---

4. Infrastructure & Deployment

- **Purpose:** Host the app, ensure scalability, speed, and reliability.
- **Technologies:**

- **Hosting:** AWS (EC2, Lambda), Google Cloud, Azure, or Vercel/Netlify for frontend
- **Containerization:** Docker (optional)
- **Load Balancing & Scaling:** Nginx, Kubernetes (for large scale)
- **CI/CD:** GitHub Actions / Jenkins

---

5. Additional Tools

- **URL Hashing:** Base62 encoding, CRC32, or custom algorithms
- **Analytics:** Google Analytics or custom tracking service
- **Security:** HTTPS via SSL/TLS, rate limiting, input validation

---

Project Structure

```
url_shortener/
|
├── app.py          # Main Flask app
├── templates/
|   └── index.html     # HTML form and result display
├── url_data.db       # SQLite database (auto-created)
```

app.py – Flask Application

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

app = Flask(__name__)
DB_FILE = 'url_data.db'

# --- Helper: Generate Random Slug ---
def generate_slug(length=6):
```

```python
# --- Helper: Generate Random Slug
def generate_slug(length=6):
    characters = string.ascii_letters + string.digits
    return ''.join(random.choice(characters) for _ in range(length))

# --- Initialize DB ---
def init_db():
    if not os.path.exists(DB_FILE):
        conn = sqlite3.connect(DB_FILE)
        c = conn.cursor()
        c.execute('''
            CREATE TABLE urls (
                id INTEGER PRIMARY KEY AUTOINCREMENT,
```


```python
                id INTEGER PRIMARY KEY AUTOINCREMENT,
```


Copy

```python
                slug TEXT UNIQUE NOT NULL,
                original_url TEXT NOT NULL
            )
        ''')
        conn.commit()
        conn.close()

# --- Save Mapping ---
def save_url_mapping(slug, original_url):
    conn = sqlite3.connect(DB_FILE)
    c = conn.cursor()
    c.execute('INSERT INTO urls (slug, original_url) VALUES (?, ?)', (slug, original_url))
```

```python
    conn.commit()
    conn.close()

# --- Get Original URL ---
def get_original_url(slug):
    conn = sqlite3.connect(DB_FILE)
    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 ---
```

```python
@app.route('/', methods=['GET', 'POST'])
def index():
    if request.method == 'POST':
        original_url = request.form['long_url']
        slug = generate_slug()
        save_url_mapping(slug, original_url)
        short_url = request.host_url + slug
        return render_template('index.html', short_url=short_url)
    return render_template('index.html')

@app.route('/<slug>')
def redirect_to_original(slug):
    original_url = get_original_url(slug)
```

```python
def redirect_to_original(slug):
    original_url = get_original_url(slug)
    if original_url:
        return redirect(original_url)
    return 'URL not found', 404

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

2. `templates/index.html` – HTML Form

```html
<!DOCTYPE html>
<html>
<head>
    <title>URL Shortener</title>
</head>
<body>
    <h1>URL Shortener</h1>
    <form method="POST">
        <label for="long_url">Enter a long URL:</label><br>
```

```
        <label for="long_url">Enter a long URL:</label><br>
        <input type="url" name="long_url" required><br><br>
        <button type="submit">Shorten</button>
    </form>

    {% if short_url %}
        <p>Shortened URL: <a href="{{ short_url }}" target="_blank">{{ short_url }}</a></p>
    {% endif %}
</body>
</html>
```

How to Run:

1. **Install Flask if you haven't:**

   pip install flask

2. **Run the app**:

   python app.py

3. **Open browser**:

   Visit http://127.0.0.1:5000/ and try shortening a URL!

Future Improvements of URL Shorteners

1. Advanced Link Analytics

- **Real-time tracking**: More detailed and real-time data about who clicked the link, where they are, what device they used, etc.
- **User behavior insights**: Not just clicks—track time spent on the landing page, bounce rates, and conversions.
- **AI-powered insights**: Use machine learning to predict link performance or suggest better times/audiences for sharing.

2. Smart Redirection

- **Device-based routing**: Redirect users to different destinations based on device (e.g. App Store for iOS, Play Store for Android).
- **Location-based redirection**: Automatically route users to region-specific pages.
- **A/B testing support**: Route traffic randomly between different URLs to test performance.

3. Enhanced Security Features

- **Phishing and malware detection**: Integrate real-time scanners to block malicious links.
- **Link expiration with rules**: Auto-disable a link after a time, number of clicks, or location mismatch.

- **Encrypted or tokenized URLs**: Prevent manipulation or leaking of sensitive query parameters.

4. Customization & Branding

- **AI-generated custom aliases**: Suggest short but relevant names instead of random strings.
- **Dynamic branding**: Automatically match shortened links to your brand's style or campaign.
- **Rich preview support**: Customize how the link preview looks on social media (title, image, description).

5. Decentralization & Blockchain Integration

- **Tamper-proof records**: Use blockchain to store shortened URL mappings, ensuring transparency and security.
- **Tokenized link sharing**: Provide incentive-based sharing systems using crypto tokens.

6. Integration & Automation

- **No-code/low-code integration**: Easy embedding in apps, websites, and platforms.
- **Workflow automation**: Automatically generate short URLs for new blog posts, products, campaigns, etc., via tools like Zapier or APIs.
- **CRM & marketing tools**: Deep integration with email, ad tracking, and user engagement platforms.

7. Sustainability & Digital Hygiene

- **Link lifecycle management**: Encourage deletion or archiving of outdated links to avoid link rot.
- **Eco-friendly hosting**: URL shorteners could align with green hosting initiatives.

8. Privacy Enhancements

- **Consent-aware analytics**: Respect user privacy laws (GDPR, CCPA) and provide anonymized analytics.
- **Private short links**: Links that only work for selected users or within certain environments (e.g. internal teams).