Proof of Concept – URL Shortener

Objective

Create a web-based URL shortener that converts long URLs into compact, redirectable short links using custom slugs and backend logic.

Architecture Overview

Frontend:

- HTML form with a single input field for the long URL.
- "Shorten" button triggers a POST request to the backend.

Backend:

- **Framework**: Flask (Python) — lightweight, fast for prototyping.
- **Routing**:
  - `/shorten` (POST): Accepts a long URL and returns a shortened URL.
  - `/<slug>` (GET): Redirects to the original URL.
- **Slug Generation**:
  - Random 6-8 character string OR base62 encoding of auto-incremented ID.
- **Storage**:
  - SQLite (`urls.db`) with a table `urls(id, slug, original_url)`.

Security Considerations

- Sanitize input to avoid SSRF, open redirect, and XSS.
- Rate limiting on the `/shorten` endpoint to prevent abuse.
- Validate that the input URL is a proper, safe URL (e.g., using `validators.url` or regex).
- Add domain restriction (e.g., block internal IPs like 127.0.0.1 or AWS metadata IPs).
- Optionally: use `Referer` or `Origin` checks for CSRF protection.

Basic Workflow

1. **User Input**:
   - User enters: `https://example.com/super/long/path`
2. **Backend Logic**:
   - Generate slug: `XyZ123`
   - Store in DB: `{slug: 'XyZ123', url: 'https://example.com/super/long/path'}`
3. **Response**:
   - Return: `https://yourdomain.com/XyZ123`
4. **Redirection**:

- When `XyZ123` is visited → fetch from DB → redirect to long URL.

Example API Usage

POST `/shorten`

```
{
  "long_url": "https://example.com"
}
```

Response:

```
{
  "short_url": "http://localhost:5000/XyZ123"
}
```

GET `/XyZ123`

- Redirects to `https://example.com`.

DB Schema

ID	*Slug*	*Original_URL*
1	XyZ123	https://example.com
2	A1b2C3	https://www.google.com/

Tech Stack (Minimal)

- **Python** + **Flask**
- **SQLite** for lightweight storage
- **Jinja2** for minimal frontend templating (or just raw HTML)

Completion Criteria

- Functional shortening & redirection
- Unique slugs (collision handled)
- Frontend and backend integration
- Basic input sanitization
- Working SQLite integration