Proof of Concept – URL Shortener

Name: Purva Pawaskar

Intern Id:259

Objective

To design and implement a basic web-based URL shortener that converts long URLs into short, easy-to-share links. The system stores the mapping between the original and shortened URLs and redirects users to the original link when the short link is accessed.

Scope

This proof of concept demonstrates:

- Accepting a long URL from the user.
- Generating a unique short code (slug).
- Storing the mapping in a local SQLite database.
- Redirecting users from the short URL to the original URL.

Technology Stack

- **Backend**: Python (Flask framework)- Runs the website and logic.
- **Database**: SQLite -Remembers which short code goes with which long link.
- **Frontend**: Simple HTML form (via Flask templates)- Lets users enter their long link.
- **Algorithm**: Random alphanumeric slug generation-Creates the short code.

Workflow

1. **User Input**: The user enters a long URL into a web form.
2. **Slug Generation**: The system generates a random 6-character alphanumeric slug.
3. **Database Storage**: The slug and original URL are stored in the SQLite database.
4. **Short URL Creation**: A shortened link is created using the slug.
5. **Redirection**: When the short URL is accessed, the system retrieves the original URL and redirects the user.

Example

- **Input**:

  https://www.wikipedia.org/wiki/Python_(programming_language)

- **Output Short URL**:

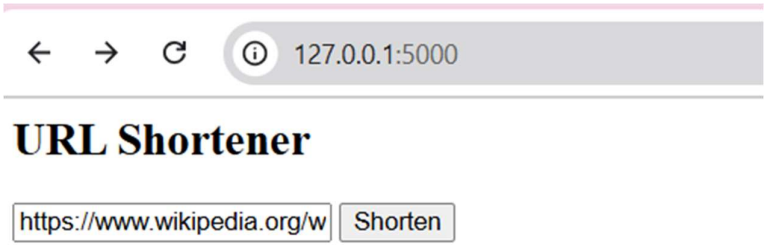
  http://127.0.0.1:5000/7bynSP

- **Redirects To**:

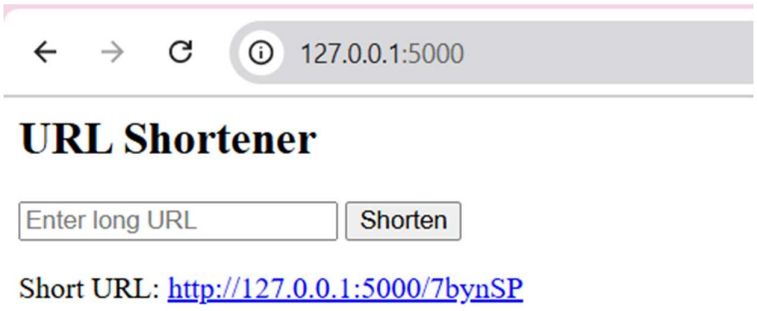
  https://en.wikipedia.org/wiki/Python_(programming_language)

Screenshots:

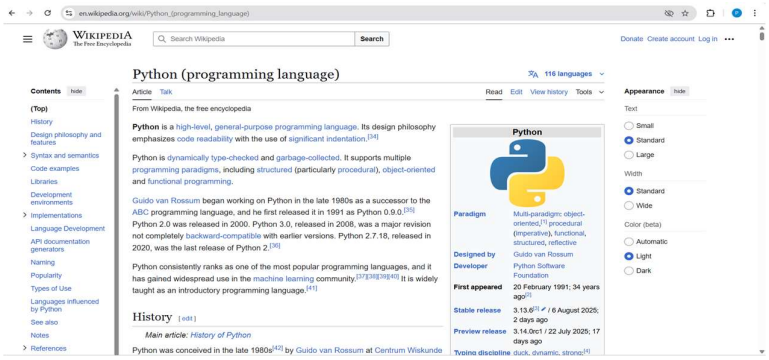
- **Input**:



- **Output Short URL**:



- **Redirects To**:



Security Considerations

- Validate user input to ensure it's a valid URL.
- Prevent duplicate slugs by checking before inserting.
- Sanitize database inputs to prevent SQL injection.
- Add HTTPS if deployed online.

  In short: **Check links → Block bad code → Avoid duplicates → Encrypt online → Check for harmful sites**.

Future Improvements

- Base62 encoding using sequential IDs instead of random slugs.
- User accounts for managing URLs.
- Analytics (click tracking).
- Expiry dates for links.

  In short: **Organized codes → User accounts → Click tracking → Expiry dates → Better design**.