URL SHORTENER WEB APPLICATION

(Advanced Project Report)

Name : Shaik Muzahid

1. Introduction

In today's digital world, sharing links has become a daily activity. Many websites generate very long and complex URLs, which are difficult to share, remember, and manage. Copying and pasting long URLs can also lead to errors, especially when sharing through messages, emails, or social media platforms.

A URL Shortener is a web-based application that converts a long URL into a short and easy-to-share link. When the short link is accessed, it redirects the user to the original long URL. Popular platforms like Bitly and TinyURL use this concept.

This project focuses on designing and developing a URL Shortener Web Application using Python and Flask. The application allows users to enter long URLs, generates short URLs, stores them in a database, and redirects users to the original URLs when accessed.

2. Problem Statement

Long URLs are inconvenient to share and manage. They are prone to errors during copying and can look unprofessional in communication. The problem is to build a web application that:

- Accepts a long URL from the user
- Generates a unique short URL
- Stores the mapping between long URL and short URL
- Redirects users from the short URL to the original URL

The solution should be simple, efficient, and easy to use.

3. Objectives of the Project

The main objectives of this project are:

1. To design a web application that shortens long URLs.
2. To store original URLs and their shortened versions in a database.
3. To implement URL redirection functionality.
4. To build a user-friendly interface.
5. To understand backend development using Flask.
6. To gain hands-on experience with databases and web application architecture.

4. Tools and Technologies Used

The following tools and technologies were used in this project:

Programming Language

Python: Used for backend logic and server-side processing.

Web Framework

Flask: A lightweight Python web framework used to build the application routes and logic.

Frontend Technologies

HTML: For structuring the web pages.

CSS: For styling and improving the appearance of the application.

Database

SQLite: Used to store original URLs and short codes.

Libraries

Flask-SQLAlchemy: For database management and ORM support.

5. System Architecture

The application follows a simple client-server architecture:

1. The user accesses the web application through a browser.
2. The user enters a long URL in the input form.
3. The Flask backend processes the request.
4. A unique short code is generated.
5. The long URL and short code are stored in the database.
6. The short URL is displayed to the user.
7. When the short URL is accessed, the application redirects the user to the original URL.

This architecture ensures smooth interaction between frontend, backend, and database.

6. Database Design

The database contains a single table with the following fields:

id: Unique identifier (Primary Key)

original_url: Stores the long URL

short_code: Stores the generated short code (Unique)

This simple design is sufficient for storing and retrieving URLs efficiently.

7. Implementation Details

URL Shortening Logic

- A random combination of letters and digits is generated as a short code.
- The short code length is fixed to ensure consistency.
- Each short code is unique to avoid conflicts.

Flask Routes

- **Home Route (/)**

Displays the input form and handles URL submission.

- **Redirect Route (/<short_code>)**

Fetches the original URL from the database and redirects the user.

Frontend Design

- A clean and simple interface is designed using HTML and CSS.
- The user can easily input a URL and receive the shortened link.

8. Challenges Faced

During the development of this project, the following challenges were encountered:

1. Generating unique short codes without duplication.
2. Understanding Flask routing and request handling.
3. Integrating the database with the Flask application.
4. Handling cases where a short URL does not exist.

These challenges helped improve problem-solving skills and understanding of backend development.

9. Results and Output

The final application successfully:

- Accepts long URLs from users.
- Generates short URLs instantly.
- Stores URLs in a database.
- Redirects users correctly from short URLs to original URLs.

The application runs smoothly on a local server and meets all project requirements.

10. Future Enhancements

The following features can be added in the future:

- User authentication and login system.
- Click count tracking for each short URL.
- URL expiration feature.
- Custom short URLs.
- Deployment on cloud platforms.

11. Conclusion

This project demonstrates the development of a complete URL Shortener Web Application using Flask and SQLite. It provided hands-on experience in backend development, database management, and web application design. The project helped strengthen understanding of real-world application development and improved practical coding skills.

Overall, the project successfully meets its objectives and serves as a strong foundation for building more advanced web applications in the future.