URL Shortener Web Application

Name : Shaik Muzahid

Final Project Explanation (Basic Version)

1. Introduction

In the modern digital era, web links play a crucial role in sharing information. However, many URLs are very long and complex, which makes them difficult to share, remember, and manage. Long URLs are inconvenient when sending links through emails, chat applications, or social media platforms. To overcome this problem, URL shortener applications are widely used.

A URL Shortener is a web-based application that converts long URLs into short, easy-to-share links. When a user accesses the shortened URL, they are automatically redirected to the original long URL. This project focuses on building a Basic URL Shortener Web Application using Python and Flask. The application provides a simple user interface where users can shorten URLs and view previously shortened links.

This project helps in understanding the fundamentals of web development, backend processing, database management, and frontend-backend interaction.

2. Objectives of the Project

The main objectives of this project are:

1. To develop a web application that shortens long URLs.
2. To provide a simple and user-friendly interface for users.
3. To store original URLs and their shortened versions in a database.
4. To allow users to view the history of previously shortened URLs.
5. To validate URLs before shortening them to avoid invalid entries.

3. Need for URL Shortener

In many real-life scenarios, users need to share links frequently. These links may be long, difficult to read, and prone to errors while copying or typing. Some common problems with long URLs include:

- Difficult to share via messages
- Increased chances of mistakes
- Poor readability
- Not user-friendly

A URL shortener solves these issues by generating a small and unique link for every long URL. It improves user experience and makes link sharing efficient and convenient. This project demonstrates how such a system can be implemented using Flask and SQLite.

4. Technologies Used

The following technologies are used in this project:

Frontend Technologies

- **HTML:** Used to create the structure of web pages.
- **CSS:** Used to style the web pages.
- **Basic JavaScript:** Used for copy-to-clipboard functionality.

Backend Technologies

- **Python:** Core programming language.
- **Flask:** Lightweight web framework used to build the backend logic.

Database

- **SQLite:** Lightweight relational database used to store URLs.

ORM

- **SQLAlchemy:** Used to interact with the database using Python objects instead of SQL queries.

5. Project Architecture

The project follows a simple client-server architecture:

1. The user interacts with the frontend through a browser.
2. The frontend sends requests to the Flask backend.
3. Flask processes the request and performs required operations.
4. Data is stored or retrieved from the SQLite database.
5. The response is sent back to the frontend.

This architecture ensures clear separation between the user interface, application logic, and data storage.

6. Frontend Design

The frontend of the application consists of two main pages:

6.1 Home Page

The Home page is the main interface of the application. It contains:

- A text input field to enter the long URL.
- A Shorten button to generate the short URL.
- A View History link to navigate to the history page.
- A section to display the shortened URL.
- A copy button to copy the shortened URL easily.

The layout is kept minimal and clean to improve usability.

6.2 History Page

The History page displays:

- All original URLs entered by users.
- Their corresponding shortened URLs.

This page helps users keep track of previously shortened links and demonstrates database storage functionality.

7. Backend Implementation

The backend is developed using Flask. Flask handles routing, form submission, URL processing, database operations, and redirection.

7.1 URL Shortening Logic

When a user submits a URL:

1. The backend validates whether the URL is correct.
2. A unique short code is generated using random characters.
3. The original URL and short code are saved in the database.
4. A shortened URL is created using the base URL and short code.

Example:

Original URL: https://www.google.com

Short URL: http://127.0.0.1:5000/aB3xK9

7.2 URL Redirection

When a shortened URL is accessed:

1. Flask extracts the short code from the URL.
2. It searches the database for the corresponding original URL.
3. The user is redirected to the original URL.

This ensures smooth and instant redirection.

8. URL Validation

URL validation is an important part of the project. It ensures that only valid URLs are shortened. The application checks:

- Whether the URL starts with http:// or https://
- Whether the URL format is correct

If an invalid URL is entered, an error message is displayed to the user. This improves data quality and prevents broken links.

9. Database Design

The project uses SQLite as the database. It is lightweight and suitable for small to medium applications.

URL Table Structure

Column Name		Description
Id	-	Unique identifier (Primary Key)
original_url	-	Stores the long URL
short_code	-	Stores the generated short code

Each row represents one shortened URL entry.

10. Project Workflow

The step-by-step workflow of the application is as follows:

1. User opens the Home page.
2. User enters a long URL in the input field.
3. User clicks the Shorten button.
4. The backend validates the URL.
5. A unique short code is generated.
6. URL data is stored in the database.
7. Shortened URL is displayed to the user.
8. User can copy the short URL.
9. User can visit the History page to view all URLs.

11. Advantages of the Project

- Easy to use and understand
- Improves link sharing
- Demonstrates Flask fundamentals
- Uses ORM for database operations
- Lightweight and fast
- Beginner-friendly implementation

12. Limitations of the Project

- No user authentication
- All users share the same URL history
- No click analytics
- No URL expiration feature

These limitations can be addressed in future versions.

13. Future Enhancements

The project can be enhanced by adding:

- User login and signup system
- User-specific URL history
- Click count analytics
- URL expiration dates

- Improved UI using Bootstrap
- Deployment on cloud platforms

14. Conclusion

The URL Shortener Web Application successfully demonstrates how long URLs can be converted into short, manageable links using Flask and SQLite. The project provides hands-on experience with web development concepts such as routing, database integration, URL validation, and frontend-backend interaction.

This project fulfills all the requirements of the Basic Version and serves as a strong foundation for implementing advanced features in the future.