Project Report

URL Shortener Web Application (Basic Version)

Dharanikota Teja Swaroop

1. Introduction

The internet has become an essential part of everyday life. People use web links to share information, documents, videos, articles, and learning resources. Many websites generate long and complex URLs that contain special characters, tracking information, and multiple parameters. These URLs are difficult to remember and inconvenient to share, especially on messaging platforms where character limits may apply.

Copying and pasting long URLs can also lead to errors. A single missing character or extra space can make a link invalid. This problem becomes more serious when users are sharing links professionally or academically. Therefore, a solution is required to simplify URL sharing.

A URL shortener is a web application that converts a long URL into a short and unique link. When a user opens the short link, the system redirects them automatically to the original web address. Popular platforms such as Bitly, TinyURL, and Rebrandly work using the same idea.

The purpose of this project is to design and implement a basic URL Shortener Web Application using Python Flask and SQLite database. The application enables users to generate short URLs, store them in a database, view previously generated URLs, and validate the correctness of input URLs.

This project also helped in understanding how frontend interfaces interact with backend logic and how data is managed efficiently in databases.

2. Objectives of the Project

The main objectives of this project include the following:

- To develop a web-based application that shortens long URLs into shorter versions.
- To store original URLs and their shortened URLs securely in a database.
- To display all previously shortened URLs on a history page for reference.
- To validate user input and prevent invalid URLs from being stored.
- To understand routing and redirection mechanisms in Flask.
- To gain practical exposure to database operations using SQLAlchemy ORM.
- To improve skills in full-stack web development.

3. Technologies Used

3.1 Frontend Technologies

HTML is used to structure the web pages. Input fields, buttons, tables, and headings are created using HTML tags. Bootstrap can optionally be used to improve appearance, responsiveness, and layout.

The frontend communicates with the backend through form submissions.

3.2 Backend Technologies

Flask is a lightweight Python web framework that allows developers to build web applications quickly. It provides routing, request handling, and template rendering features. Flask is suitable for beginners and small projects.

3.3 Database and ORM

SQLite is used as the database because it is simple, lightweight, and requires no separate installation. SQLAlchemy ORM allows database tables to be created and managed using Python classes instead of raw SQL queries.

4. System Workflow

The workflow of the URL Shortener application follows a simple sequence:

1. The user opens the web application in a browser.
2. The home page displays a text field for entering a URL.
3. The user enters a long URL and clicks the shorten button.
4. The backend validates whether the entered URL is valid.
5. If valid, a short code is generated.
6. The original URL and short code are saved in the database.
7. The shortened URL is displayed to the user.
8. The user may copy the shortened URL and share it.
9. When the shortened URL is opened, the system redirects to the original URL.
10. The history page shows all previously generated URLs.

5. Design and Architecture

The system follows a simple client-server architecture. The browser acts as the client and Flask acts as the server. The database stores persistent information.

The application contains three major components:

- User Interface Layer (HTML pages)

- Application Logic Layer (Flask backend)
- Data Storage Layer (SQLite database)

Each layer performs a specific role, making the system modular and easier to maintain.

6. Implementation Details

6.1 Application Setup

The Flask environment was initialized and required packages were installed using pip. Database configuration was completed using SQLAlchemy.

6.2 Database Model

A table named URL contains the following fields:

- ID: Unique identifier.
- Original URL: Stores the long URL.
- Short Code: Stores the generated code.

6.3 Short Code Generation

A random alphanumeric string of fixed length is generated using Python's random module. This ensures uniqueness.

6.4 URL Validation

The validators library checks whether the input string is a valid URL.

6.5 Routing and Redirection

Flask routes handle user requests and redirection.

6.6 Frontend Integration

Templates display results dynamically using Jinja.

7. Testing and Validation

Various test cases were executed including valid input, invalid input, redirection testing, and history page validation. Manual testing confirmed that the system behaves correctly under different scenarios.

8. Challenges and Solutions

During the development process, several challenges were encountered. Understanding ORM relationships required additional practice. Debugging routing issues required careful inspection of Flask logs. URL validation logic required research to ensure reliability.

9. Learning Outcomes

This project enhanced skills in Flask development, ORM usage, debugging techniques, web application architecture, and deployment preparation.

10. Future Enhancements

Future upgrades may include authentication systems, improved user interface design, analytics tracking, QR code generation, and cloud deployment.

11. Conclusion

The URL Shortener Web Application successfully meets all objectives and demonstrates a practical implementation of full-stack development concepts. The project strengthened technical understanding and confidence in building web-based solutions.