# URL SHORTENER WEB APPLICATION (BASIC)

## Project Report

## 1. Introduction

In today's digital world, URLs play a very important role in accessing information on the internet. Many websites generate long and complex URLs which are difficult to remember, share, or type manually. Sharing such long URLs through messages, emails, or social media platforms can be inconvenient and sometimes confusing. To overcome this issue, URL Shortener applications are widely used.

## 2. Objectives of the Project

The main objectives of this project are:

1. To create a web application that shortens long URLs into smaller and manageable URLs.
2. To allow users to store and view previously shortened URLs.
3. To implement URL validation to ensure that only valid URLs are accepted.
4. To understand how frontend and backend components communicate with each other.
5. To gain hands-on experience with Flask, database integration, and template rendering.

## 3. Need for a URL Shortener

Long URLs can be inconvenient for several reasons:

- They are difficult to remember.
- Copying and pasting long URLs is time-consuming.
- Long URLs look messy when shared in messages or social media.

- Typing long URLs manually increases the chance of errors.

## 4. Technologies Used

The project is divided into frontend and backend parts.

**Frontend Technologies**

- **HTML** – Used to create the structure of web pages.
- **CSS** – Used for styling and improving the visual appearance.
- **Bootstrap** – Used to make the UI responsive and visually appealing.

**Backend Technologies**

- **Python (Flask Framework)** – Used to handle routing, form submission, logic, and server-side operations.
- **Flask ORM (SQLAlchemy)** – Used to interact with the database.
- **SQLite Database** – Used to store original URLs and shortened URLs.

## 5. Project Architecture

The application follows a simple **MVC-like structure**:

- **Model**: Database tables using ORM to store URL data.
- **View**: HTML templates rendered using Jinja2.
- **Controller**: Flask routes that handle user requests and responses.

## 6.Frontend Description

The frontend of the application consists of two main pages: Home Page and History Page.

The Home Page allows users to enter a long URL that needs to be shortened. After clicking the “Shorten URL” button, the shortened link is displayed on the screen.

Users can easily copy the shortened URL using the copy button provided. A navigation link is also available to view the URL history.

The History Page displays all previously shortened URLs in a table format. It shows both the original URL and its corresponding shortened URL. Each link is clickable and opens in a new browser tab, helping users keep track of their past URLs.

## 7.Backend Implementation

The backend of the application is developed using the Flask framework. When a user submits a URL, the backend generates a unique short code and appends it to the base URL to create a shortened link. This mapping between the original URL and the short code is stored in the database.

Before shortening, the application validates the entered URL to ensure it is in the correct format and contains a valid protocol such as http or https. If the URL is invalid, an error message is displayed.

When a shortened URL is accessed, the backend checks the database for the corresponding original URL. If found, the user is redirected to the original website; otherwise, an error message is shown.

## 8. Project Workflow

1. The user enters a URL on the Home Page.
2. The backend validates the URL.
3. A shortened URL is generated.
4. The original URL and shortened URL are stored in the database.
5. The shortened URL is displayed to the user.
6. user can copy the URL or view previous URLs in the History Page.

## 9. Challenges Faced

Some challenges faced during development include:

- Validating URLs correctly.
- Ensuring unique short codes.
- Handling redirection smoothly.
- Designing a clean and responsive UI.
- Connecting frontend forms with backend logic.

These challenges helped in gaining a better understanding of Flask and full-stack development.

## 10. Conclusion

The URL Shortener Web Application successfully meets all the project objectives. It provides an easy-to-use interface for shortening URLs and viewing previous records. Through this project, I gained hands-on experience in Flask, frontend design, database handling, and full-stack application development.

## 11. Learning Outcome

By completing this project, I learned:

- Flask routing and template rendering.
- Database integration using ORM.
- Frontend-backend communication.
- URL validation techniques.

**Reported by:**
**Mittapally SaiKrishna**

**ID: IN125007511**