# URL SHORTENER WEB APPLICATION FOR BASIC USERS

**Project Report**

---

**Intern Name: Venkatesh Vijay Karnure**

**Internship Program: Data Science with GenAI**

**Organization: Innomatics Research Labs**

**Technology Stack:**

- Frontend: HTML, CSS, Bootstrap
- Backend: Flask (Python)
- ORM: SQLAlchemy
- Database: SQLite

---

## 1. Introduction

In today's digital era, sharing information through links has become a routine activity. However, long URLs are often inconvenient to share, difficult to remember, and prone to errors during copying and pasting. A URL Shortener is a practical solution that converts long URLs into short, easy-to-share links while ensuring proper redirection to the original webpage.

This project, **URL Shortener Web Application for Basic Users**, was developed as part of my **Data Science with GenAI internship**. The main objective of this project is to design and implement a full-stack web application that shortens URLs, stores them in a database, and allows users to access previously shortened URLs through a history page.

The project emphasizes backend development using Flask, database management using SQLAlchemy ORM, and frontend integration using HTML, CSS, and Bootstrap.

---

## 2. Project Objectives

The primary objectives of this project are:

1. To develop a web application that shortens long URLs.
2. To validate whether the user-entered URL is correct or not.
3. To store original and shortened URLs in a database.
4. To provide a history page displaying all stored URLs.
5. To redirect users from shortened URLs to the original URLs.
6. To design a clean and user-friendly interface using Bootstrap.
7. To understand real-world full-stack web development workflow.

---

## 3. Why a URL Shortener Is Needed

Long URLs can be difficult to manage, especially when sharing links via emails, messages, or social media platforms. URL shorteners solve this problem by:

- Improving readability
- Making links easier to share
- Reducing copying errors
- Providing a centralized history of links
- Enhancing user experience

This project demonstrates how a basic URL shortener can be implemented using Python-based technologies.

---

## 4. System Architecture

The application follows a **client-server architecture** and is divided into three main layers:

### 4.1 Frontend Layer

The frontend is developed using:

- **HTML** for structure
- **CSS** for basic styling
- **Bootstrap** for responsive and clean UI design

The frontend consists of two web pages:

- **Home Page**
- **History Page**

---

### 4.2 Backend Layer

The backend is built using **Flask**, a lightweight Python web framework. Flask handles:

- Routing and request handling
- URL validation
- Short URL generation
- Redirection logic
- Communication with the database

---

### 4.3 Database Layer

The application uses **SQLite** as the database and **SQLAlchemy ORM** for database operations. ORM allows interaction with the database using Python objects instead of raw SQL queries, making the code more readable and maintainable.

---

## 5. Frontend Description

### 5.1 Home Page

The Home Page allows users to:

- Enter a long URL
- Click on the **Shorten URL** button
- View the generated short URL
- Copy the shortened URL
- Navigate to the History Page

Bootstrap is used to ensure the page is responsive and visually clean.

---

### 5.2 History Page

The History Page displays:

- Original URLs
- Corresponding shortened URLs

This allows users to revisit previously shortened links easily.

---

## 6. Project Workflow

The application follows the workflow below:

1. The user enters a long URL on the Home Page.
2. The application validates the URL.
3. If valid, a unique short code is generated.
4. The original URL and short code are saved in the database.
5. The shortened URL is displayed to the user.
6. The user can copy the shortened URL.
7. Visiting the shortened URL redirects the user to the original URL.
8. The History Page displays all previously shortened URLs.

---

## 7. Database Design

The database consists of a single table with the following fields:

- **id** – Primary key
- **original_url** – Stores the original long URL
- **short_code** – Stores the generated short code
- **created_at** – Stores the timestamp of URL creation

Using SQLAlchemy ORM simplifies database operations such as insertion, querying, and deletion.

---

## 8. URL Validation

To ensure data correctness, the application validates URLs entered by users. This prevents invalid or malformed URLs from being stored in the database.

**Benefits of URL validation:**

- Improves data quality
- Prevents incorrect redirections
- Enhances user experience
- Avoids unnecessary database entries

---

## 9. Short URL Generation Logic

The short URL is generated using a random combination of alphanumeric characters. This method:

- Ensures uniqueness
- Keeps URLs short and readable
- Avoids predictable patterns

This approach is simple, efficient, and suitable for basic-level applications.

---

## 10. Features Implemented

- URL shortening functionality
- URL validation
- Database storage using SQLite
- ORM-based database interaction
- Redirect from short URL to original URL
- History page displaying saved URLs
- Bootstrap-based responsive UI

- Copy-to-clipboard functionality for shortened URLs

---

**11. Challenges Faced**

During development, the following challenges were encountered:

- Understanding ORM concepts and database relationships
- Validating URLs correctly
- Managing unique short codes
- Connecting frontend and backend smoothly
- Implementing redirection logic

---

**12. Solutions and Learnings**

The challenges were resolved by:

- Studying Flask routing and ORM concepts
- Using SQLAlchemy for clean database handling
- Applying proper validation logic
- Testing each feature step-by-step
- Keeping the code simple and modular

**Key Learnings:**

- Full-stack web development workflow
- Backend logic implementation
- Database management using ORM
- Debugging and problem-solving
- Writing clean and maintainable code

---

**13. Conclusion**

The **URL Shortener Web Application for Basic Users** successfully fulfills all project objectives. The application provides an efficient way to shorten URLs, store them in a database, and retrieve them later. This project significantly enhanced my understanding of Flask, SQLAlchemy ORM, database handling, and full-stack development.

Completing this project as part of my **Data Science with GenAI internship** helped bridge the gap between theoretical knowledge and practical implementation.

---

**14. Future Enhancements**

Possible future improvements include:

- User authentication
- Click analytics
- Custom short URLs
- URL expiration
- Deployment on cloud platforms