URL Shortener Web Application – Project Report

1. Introduction

The **URL Shortener Web Application** is a lightweight and user-friendly web application designed to convert long URLs into short, easy-to-share links. Long URLs are often inconvenient to share, remember, or display, and this application provides an efficient solution.

The application allows users to:

1. Enter a long URL through a web form
2. Generate a shortened URL instantly
3. Store both original and shortened URLs in a database
4. View a history of previously shortened URLs
5. Copy the shortened URL easily using a copy button

The project focuses on both **functionality and user experience**, ensuring input validation, duplicate prevention, and clear user feedback.

The application is developed using **Python** with **Flask** as the backend framework. Database operations are handled using **SQLAlchemy** and **SQLite**. The frontend uses **Bootstrap** for styling and **JavaScript** for interactive features.

2. Tools and Technologies Used

- **Python**
  Handles routing, logic, validation, and database interaction.
- **Flask**
  Used to build routes, manage requests, and render templates.
- **SQLAlchemy**
  Acts as an ORM to map Python classes to database tables.
- **SQLite**
  Stores original URLs and their corresponding short codes.

- **HTML & CSS**
  Provide structure and basic styling for the web pages.
- **Bootstrap**
  Ensures responsive and professional UI design.
- **JavaScript**
  Adds interactivity such as the copy-to-clipboard feature.

3. Application Architecture

The application follows a simple and clean architecture with three main routes:

1. Home Route (/)

- Accepts URL input via a POST request
- Validates the URL format
- Checks for duplicate URLs in the database
- Generates a new short code if the URL is not already present
- Displays flash messages for success or errors
- Shows the shortened URL with a copy button

2. Redirect Route (/<short_code>)

- Accepts a short code as a URL parameter
- Searches the database for the corresponding original URL
- Redirects the user to the original URL if found
- Returns a 404 error if the short code does not exist

3. History Route (/history)

- Displays all stored URLs
- Shows both original and shortened URLs
- Stores only unique URLs to keep the database clean

4. Database Design

The database is built using SQLite and managed through SQLAlchemy ORM.

Column Name	Data Type	Constraints	Description
id	Integer	Primary Key, Auto Increment	Unique identifier
original_url	String(500)	Not Null	Stores the original URL
short_url	String(10)	Unique, Not Null	Stores the generated short code

5. Backend Logic

The backend handles all core functionalities of the application.

Key Responsibilities:

- Validating user-entered URLs
- Generating random short codes
- Preventing duplicate URL entries
- Storing and retrieving data from the database
- Redirecting users to original URLs
- Displaying user feedback through flash messages

6. Error Handling and Validation

To ensure reliability and a smooth user experience, proper validation and error handling are implemented.

- Invalid URLs are rejected
- Empty inputs are not accepted
- Non-existent short codes return a 404 error

- Flash messages clearly inform users of errors and successful operations

7. Key Improvements and Features

- **Duplicate URL Prevention**
  Improves database efficiency and user experience.
- **Flash Messages**
  Provides real-time feedback for user actions.
- **Copy Button (JavaScript)**
  Enables one-click copying of shortened URLs.
- **Responsive Design (Bootstrap)**
  Ensures usability across devices.
- **Clean and Modular Code**
  Makes the project easy to maintain and extend.

8. Testing and Debugging

Manual testing was conducted to ensure correctness:

- Tested valid and invalid URLs
- Verified redirection functionality
- Confirmed duplicate URLs are not stored
- Checked responsiveness across screen sizes

Debugging helped fix issues related to form submission and database queries.

9. Learning Outcomes

This project helped in gaining practical understanding of:

- Flask-based web application structure
- Backend and frontend integration

- ORM-based database management
- Real-world features like validation and redirection
- Writing clean, readable, and maintainable code

10. Conclusion

The **URL Shortener Web Application** is a complete, beginner-friendly project demonstrating essential web development skills.

Highlights:

- Efficient backend using Python and Flask
- Clean database integration with SQLAlchemy and SQLite
- Responsive frontend using Bootstrap
- Interactive features like flash messages and copy button
- Real-world improvements such as duplicate prevention

This project serves as a strong foundation for advanced web development.