

URL Shortener Web Application – Project Report

1. Introduction

The **URL Shortener Web Application** is a lightweight web-based application designed to simplify long URLs into short, manageable links. This is particularly useful in situations where long URLs are difficult to share via messages, social media, or emails. The main objective of this project was to develop a **functional, user-friendly, and secure web application** that allows users to:

- Enter a long URL into a form.
- Generate a shortened URL that is easy to share.
- Keep track of all URLs created by a user.
- Copy shortened URLs quickly using a button.
- Ensure that each user sees only their own URLs after login.

The project was developed using **Python (Flask framework) for the backend, SQLite for database management, HTML/CSS for the frontend, and JavaScript** for interactive features such as the copy-to-clipboard functionality.

2. Tools and Technologies Used

Tool/Technology	Purpose
Python	Main programming language for backend logic.
Flask	Lightweight web framework to handle routing, templates, sessions, and form data.
SQLite	Database to store users, original URLs, and shortened URLs.
SQLAlchemy	Object Relational Mapper (ORM) to interact with the database in Python.
HTML	Structure of the web pages.
CSS	Styling for login, signup, and dashboard pages.
JavaScript	Client-side interactivity (copying short URLs).

Validators To ensure that entered URLs are valid.

Library

3. Approach to Solution

3.1 Understanding Requirements

The first step was to **understand the problem statement**: users need a way to shorten URLs and manage them easily. I outlined the key functionalities:

- User signup/login authentication.
- URL shortening with a unique short code.
- Avoid duplication: a URL should not be shortened multiple times by the same user.
- Display a history of all URLs created by the user.
- Enable copying short URLs with one click.

3.2 Designing the Database

I created **two tables** in SQLite:

1. User Table

- id – primary key.
- username – unique username.
- password – stored as plain text

2. URL Table

- id – primary key.
- original_url – the long URL input by the user.
- short_code – unique 6-character code.
- user_id – foreign key linking the URL to the user who created it.

3.3 Backend Implementation

Flask Routes:

1. Login Route (/)

- Handles user authentication.
- Uses Flask session to track logged-in users.
- Displays flash messages for invalid credentials.

2. Signup Route (/signup)

- Handles new user registration.
- Validates username length (5–9 characters) and uniqueness.
- Stores user in database and redirects to login page.

3. Dashboard Route (/dashboard)

- Protected route: redirects to login if user is not authenticated.
- Accepts a long URL via POST.
- Checks if URL is valid using the validators library.
- Generates a unique 6-character short code.
- Prevents duplication for the same user.
- Displays all URLs created by the logged-in user.
- Provides copy-to-clipboard functionality via JavaScript.

4. Redirect Route (/<short_code>)

- Maps short URLs to the original URL.
- Redirects automatically to the long URL.

5. Logout Route (/logout)

- Clears the session and redirects the user to login.

3.4 Frontend Implementation

Login Page (login.html)

- Contains a simple form for username and password.
- Displays error messages using Flask flash.
- Linked to a CSS file for clean styling.

Signup Page (signup.html)

- Similar form structure as login.
- Validation hints provided (5–9 characters for username).
- Flash messages confirm successful signup or errors.

Dashboard Page (dashboard.html)

- Form to enter a long URL.
- Displays the shortened URL after submission.
- Shows previous URLs with copy buttons for easy sharing.
- Flash messages are prominently displayed in red at the center.
- JavaScript handles copy-to-clipboard functionality.

CSS Design:

- Separate CSS for login/signup (style.css).
- Separate CSS for dashboard (dashboard.css) to prevent layout issues.
- Consistent styling for inputs, buttons, and flash messages.
- Responsive design ensures usability on different screens.

3.5 Key Challenges and Solutions

1. Ensuring Unique Short Codes

- Solution: Generated a random 6-character alphanumeric code.
- Future improvement: Add check for collisions.

2. Displaying User-Specific URLs

- Solution: Filtered URLs by user_id using SQLAlchemy queries.

3. Flash Messages Visibility

- Solution: Styled .flash class in CSS to show messages in red and centered.

4. Copy-to-Clipboard Functionality

- Solution: JavaScript function targeting specific input elements for copying.

3.6 Testing and Validation

Manual Testing Steps:

1. Signup as a new user → Check if user is added to the database.
2. Login → Verify session is created and dashboard is accessible.
3. Shorten multiple URLs → Confirm unique short codes are generated.
4. Attempt duplicate URL → Confirm flash message is displayed.
5. Copy buttons → Verify correct URL is copied to clipboard.
6. Logout → Verify session ends and login page is shown.

4. Conclusion

The **URL Shortener Web Application** was successfully implemented using Flask, SQLite, and JavaScript.