Assignment 12- Flask Blog

Kruthika Virupakshappa

Department of Applied Sciences, San Jose State University

DATA-230 Sec 21: Data Visualization

Venkata Duvvuri

November 11, 2025

Introduction

The purpose of this project is to develop a simple yet fully functional blog application using Flask, a lightweight web framework in Python. The application demonstrates the fundamental principles of web development including routing, database integration, and template rendering while keeping the structure minimal and easy to understand. This Flask Blog allows users to create, read, update, and delete (CRUD) blog posts stored in a SQLite database. It uses Flask-SQLAlchemy for database interactions for dynamic web pages. The user interface is designed to be clean and intuitive, enabling seamless navigation between posts, editing, and creation pages.

Tools and Technologies

- Flask - Python micro web framework used to build the backend
- Flask -SQLAlchemy- ORM for interacting with SQLite database
- SQLite - Lightweight file-based database used to store posts
- HTML - For dynamic templates and UI rendering
- CSS - To style the pages and provide a responsive layout
- VS Code/Terminal - Development environment and testing tool

The application supports full CRUD functionality:

- **Create:** Users can add new blog posts using a simple form with a title and content field.
- **Read:** Posts are displayed dynamically on the homepage and can be viewed individually.
- **Update:** Existing posts can be edited and updated directly from the browser.
- **Delete:** Posts can be removed permanently with a single click.

The Flask Blog application is made up of several interconnected web pages that together form a complete blogging system. Each page has a specific purpose and is dynamically rendered using Flask's Jinja2 templating engine, which allows the content to update in real time based on the data stored in the SQLite database.

Home Page

Home Page is the main landing page of the application. It displays all existing blog posts in reverse chronological order so that the most recent entries appear first. Each post shows the title, the date it was created, and a short excerpt from its content to give readers a quick preview. Users can click on a post title to open it and view its full content on a separate page. The home page also includes a "+ New Post" button at the top, allowing users to easily navigate to the Create Post page. This page is built using the index.html template and forms the central hub of the blog.

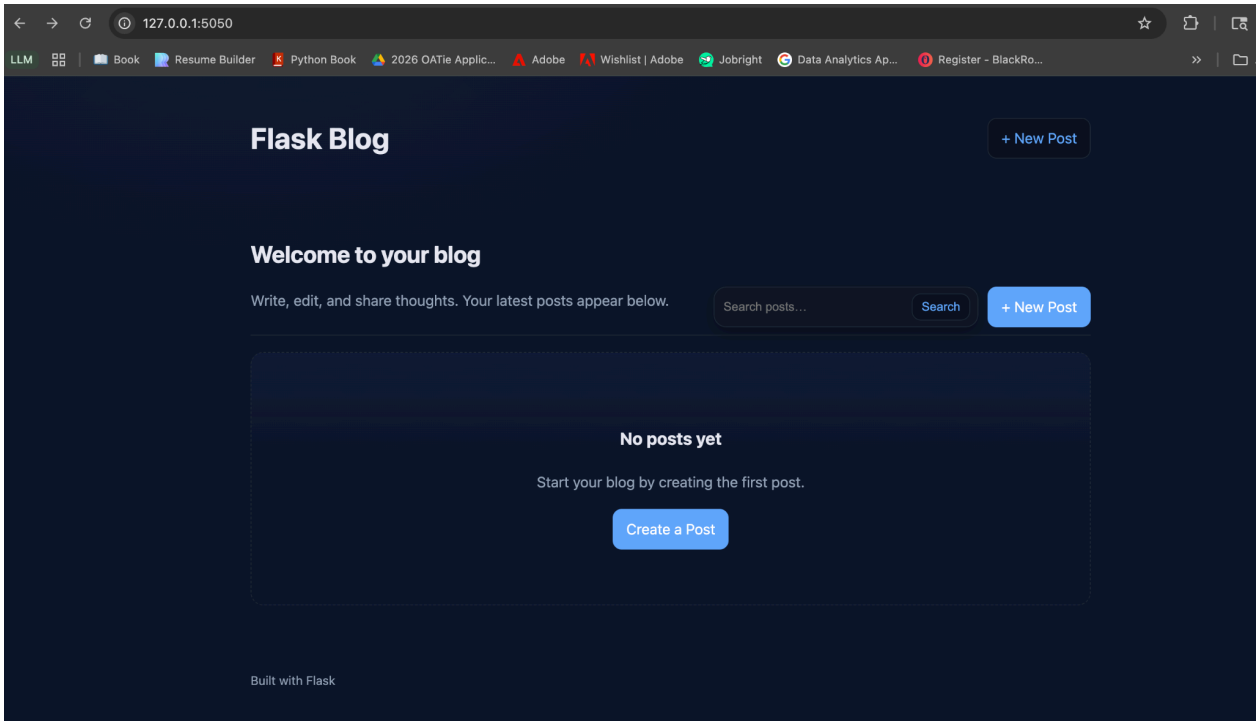

Figure 1. Home Page

Create Post Page

The Create Post page allows users to write and publish new blog entries. It contains a simple form with two fields, one for the post's title and another for the content. Once the form is submitted, the data is stored in the database, and the new post appears instantly on the home page. To make the user experience smoother, the application displays a flash message, such as "Post created successfully," confirming that the operation was completed. This page uses the form.html template, which is also reused later for editing posts.

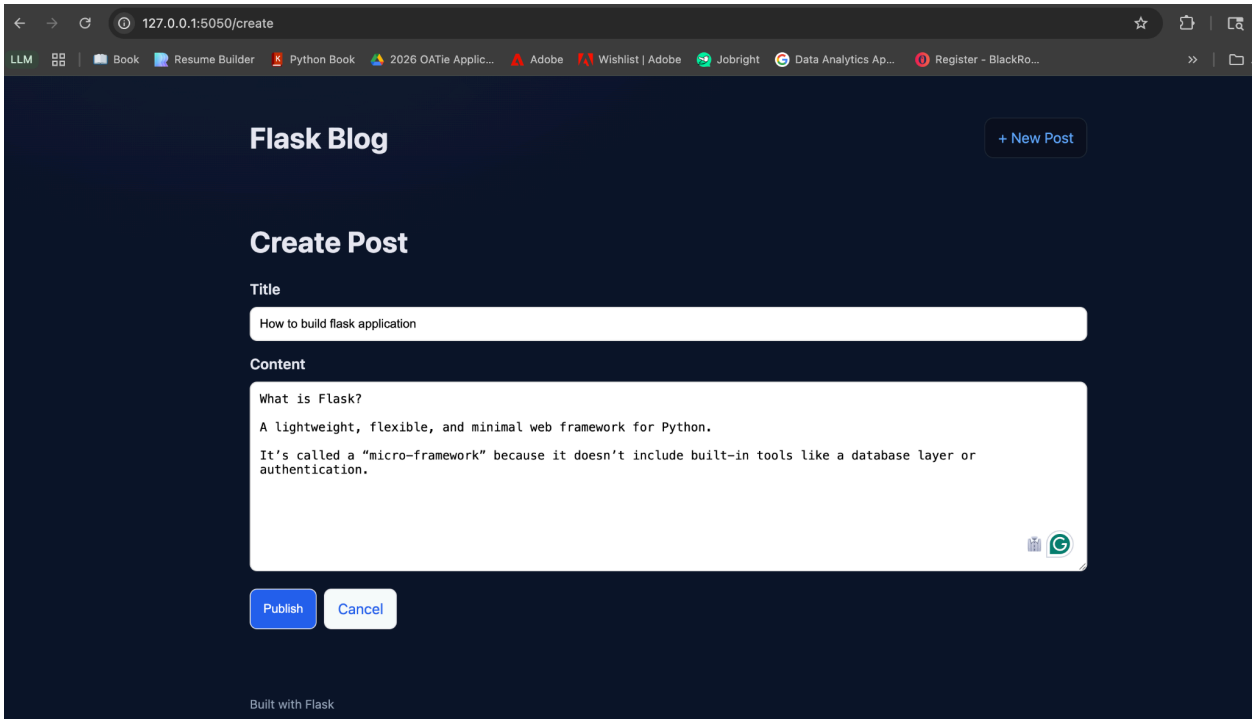

Figure 2. Creating post in Create Post Page

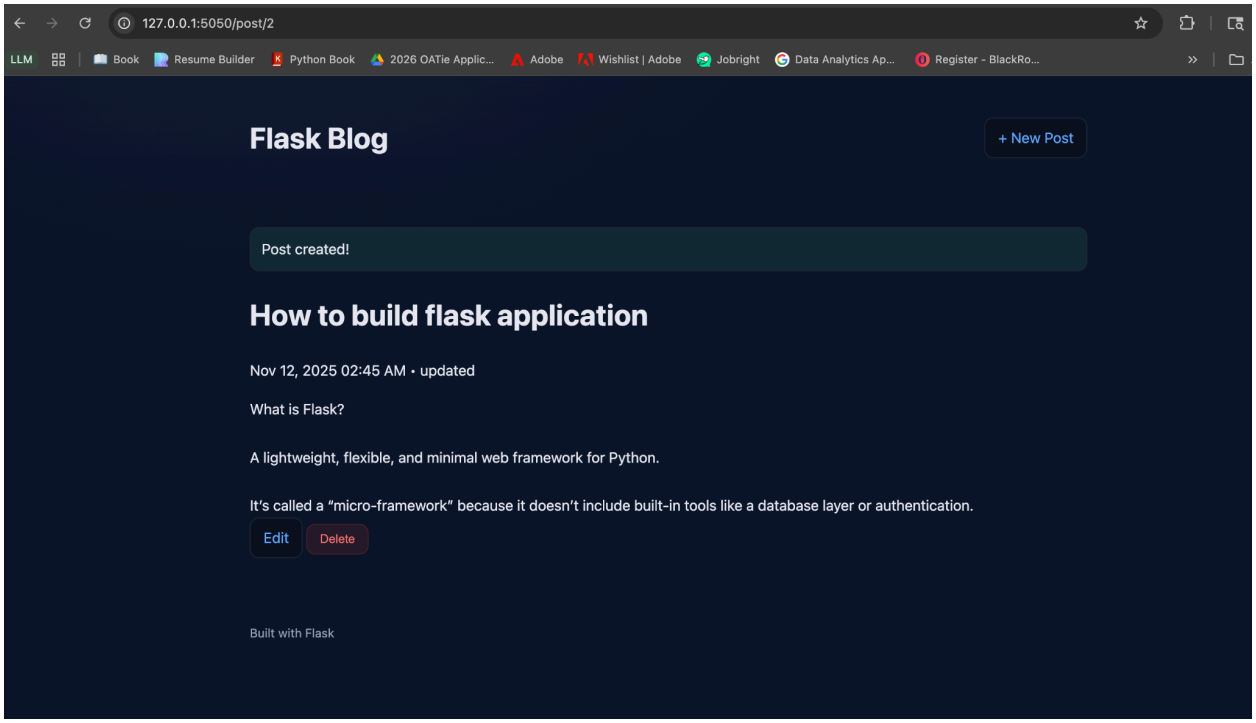

Figure 3. Saving the post created

The Post Detail Page

The Post Detail page displays the complete information about a single blog post. It shows the full title, publication date, and the entire content of the post in a clean layout. At the bottom of this page, there are two action buttons, one for editing the post and another for deleting it. When the user chooses to edit, they are redirected to the edit page where they can modify the post's title or content. If the delete button is clicked, the post is permanently removed from the database. This page uses the post_detail.html template.

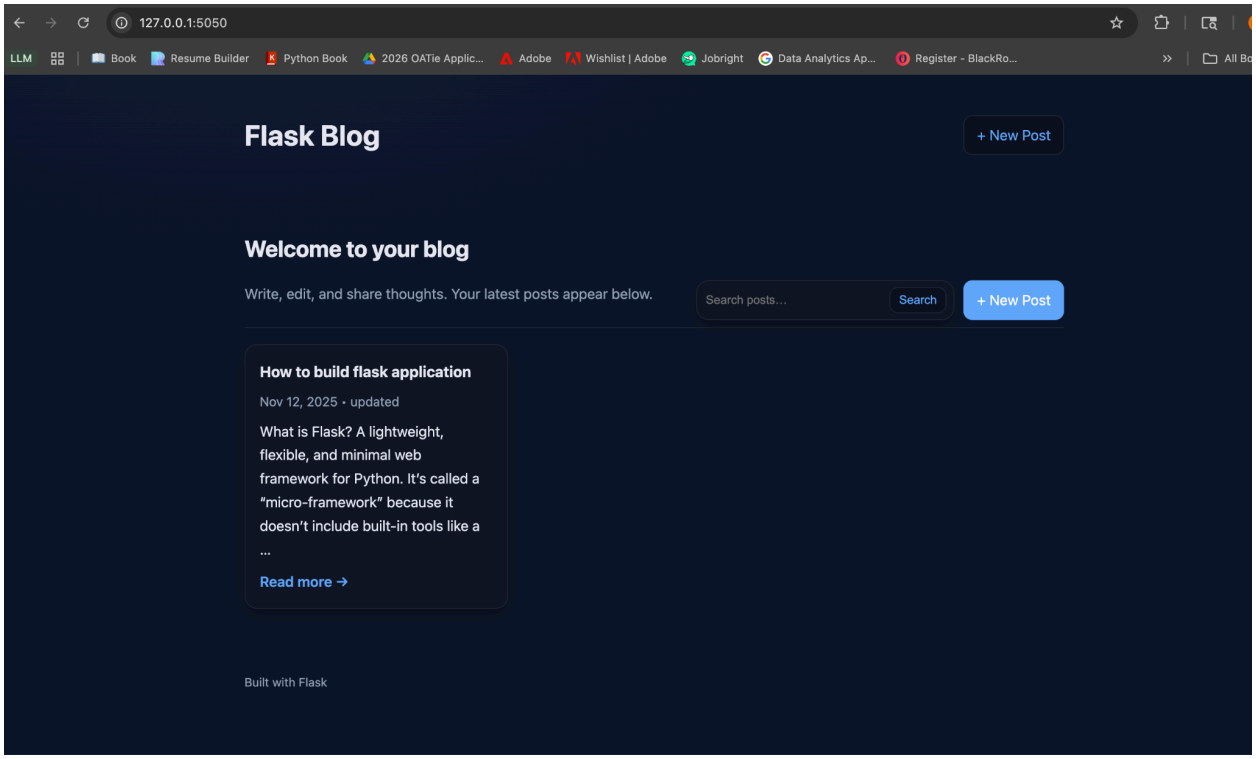

Figure 4. Post Detail Page

The Edit Post Page

The Edit Post Page allows users to make changes to an existing post. It loads the post's current title and content into the same form structure used for creating posts, making it easy for the user to make updates. After editing, when the form is submitted, the post is updated in the database, and the user is redirected back to the detailed post view. The edit page reuses the form.html template but functions in "edit mode," meaning the input fields are pre-filled with the existing data.

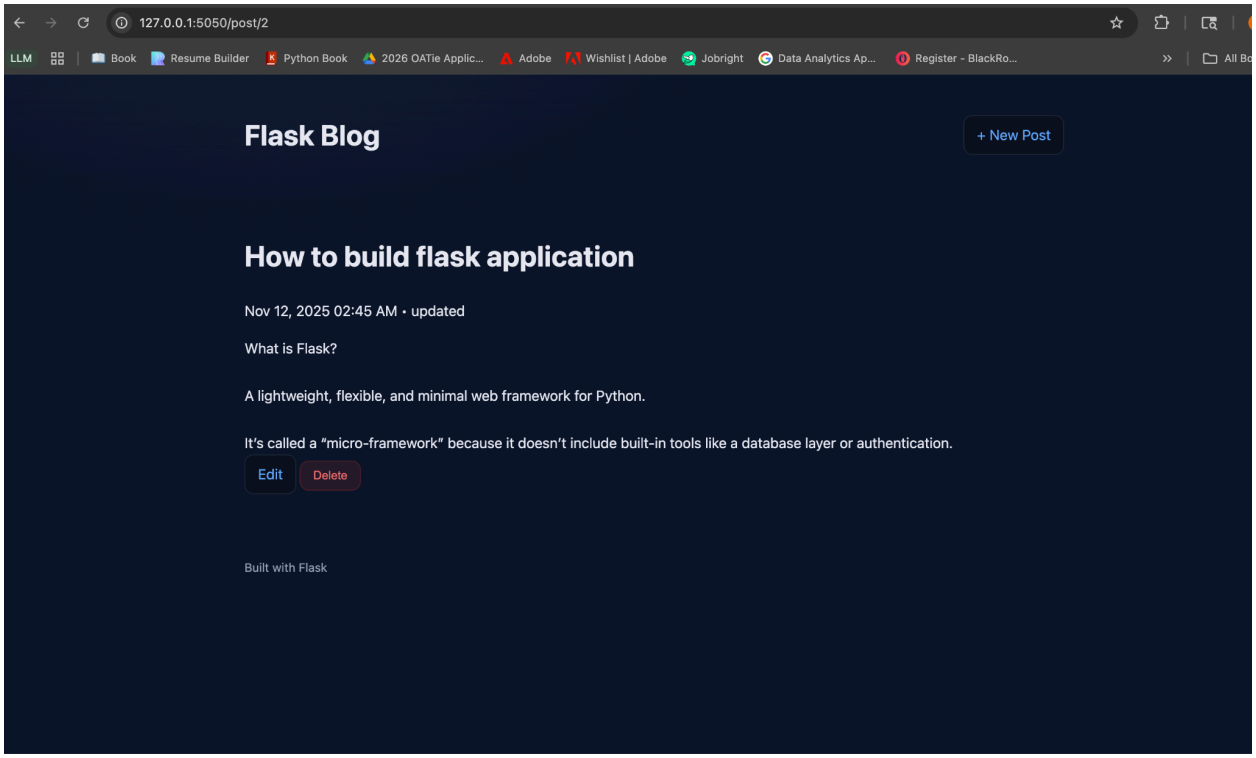

Figure 5. Edit and Delete options in Edit Post Page

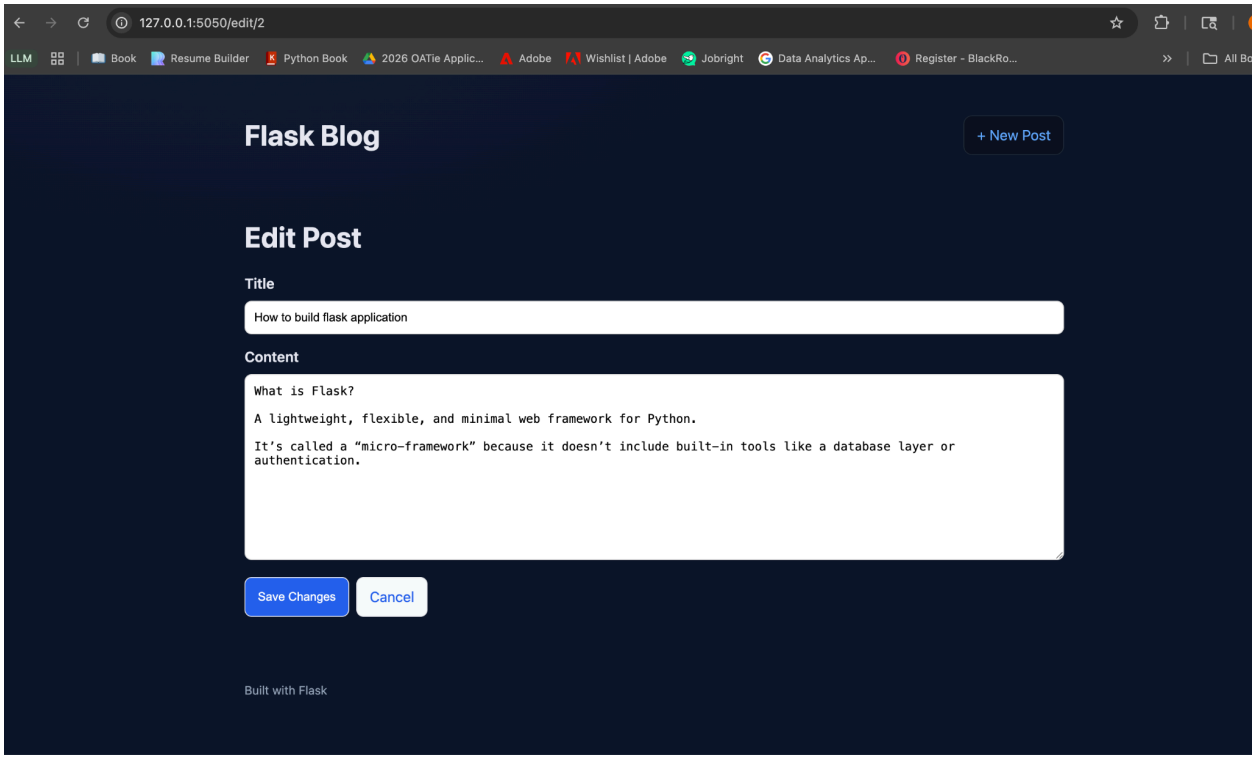

Figure 6. Changes are saved after edition

Delete Post Operation

The Delete Post operation does not have its own page. Instead, the delete functionality is triggered directly from the Post Detail page. When the user clicks the delete button, a confirmation prompt appears, and upon confirmation, a POST request is sent to the /delete/<id> route. The Flask application then removes the selected post from the database and redirects the user to the home page, where a flash message indicates that the post was successfully deleted.

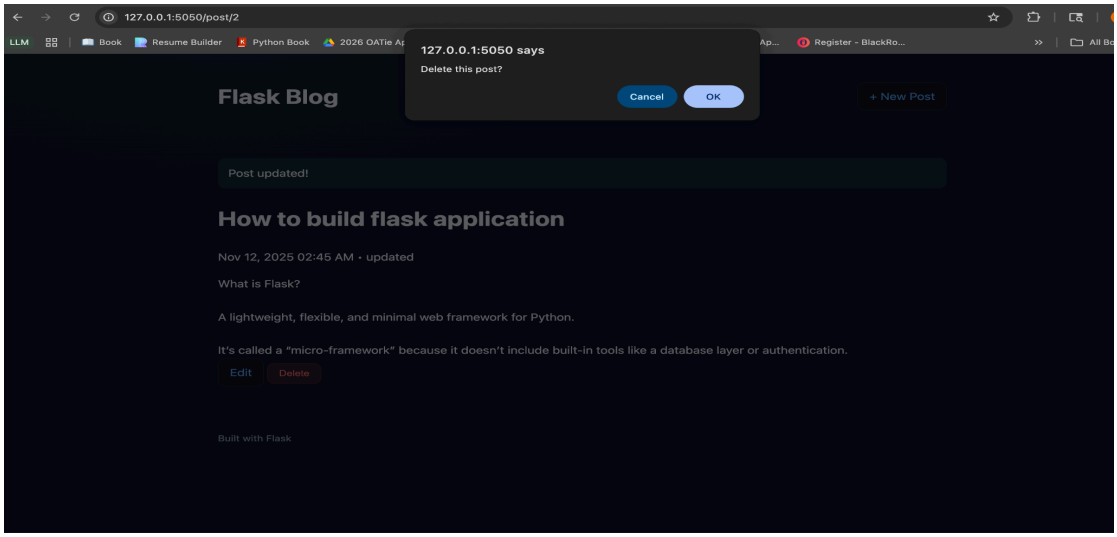

Figure 7. Delete confirmation dialogue box

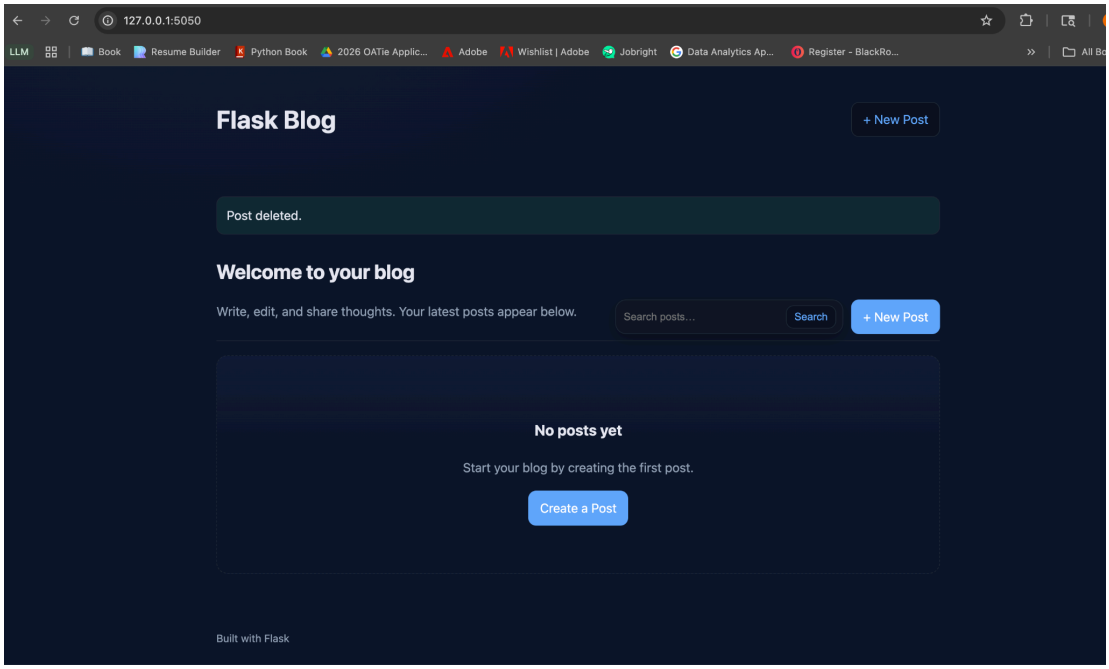

Figure 8. Post deletion screenshot

Conclusion

This project highlights the effectiveness of Flask for building small-to-medium-scale web applications that are both functional and easy to maintain. It provided practical experience with key web development concepts such as routing, templating, and ORM-based database handling and serves as a strong foundation for more advanced projects involving data analytics dashboards, user authentication, or data visualization web apps in the future.

References

Flask Documentation. *Welcome to Flask Documentation.* Retrieved from https://flask.palletsprojects.com/

Flask-SQLAlchemy Documentation. *Flask-SQLAlchemy 3.1 Documentation.* Retrieved from https://flask-sqlalchemy.palletsprojects.com/

Jinja2 Documentation. *Jinja2 Template Engine.* Retrieved from https://jinja.palletsprojects.com/

SQLite Documentation. *SQLite Official Documentation.* Retrieved from https://www.sqlite.org/docs.html