Issue 1: Restricted HTTP Route Access

The original application was configured to only accept "POST" requests. When a user first enters the website, the browser automatically sends a "GET" request to fetch the page content. Because the route didn't recognize GET, the server immediately returned a "405 Method Not Allowed" error.

- **My Approach:** I expanded the route's capabilities by including both "GET" and "POST" in the method list. This allows the script to handle the initial page load (GET) while still being able to process form submissions (POST).

Issue 2: Improper Form Data Retrieval

The code previously attempted to pull data using request.args.get(). This specific function is used to read "query parameters"—the text you see in a URL after a question mark. However, since the goal is to submit notes through a form, the data is stored in the request body, not the URL.

- **My Approach:** I swapped the retrieval logic to request.form.get(). This is the correct Flask method for accessing data submitted via an HTML form, ensuring the user's note is actually captured by the backend.

Issue 3: HTML Form Method Mismatch

In the original HTML file, the <form> tag was missing a method attribute. By default, browsers treat any form without a method as a "GET" form. This meant that every time a user tried to add a note, the text was being attached to the URL instead of being sent as a proper data packet to the server.

- **My Approach:** I explicitly added method="POST" to the form tag in the HTML template. This aligns the frontend submission with the backend logic, making the data transfer seamless and clean.

Issue 4: Logic Execution Errors

The original script ran the "append" logic every time the function was triggered. This meant that even if a user just refreshed the page without typing anything, a "None" value or an empty bullet point would be added to the notes list.

- **My Approach:** I introduced a conditional check using if request.method == "POST":. This creates a gatekeeper that ensures notes are only added when the user has actually submitted the form. I also added a check to make sure the note isn't an empty string before saving it.

Issue 5: Missing Input Validation

There was no protection against submitting empty notes, which would clutter the list with blank items.

- **My Approach:** I added a required attribute to the HTML input field. This provides immediate feedback to the user by preventing the form from being sent until they have actually typed something.