Content for Challenge 4: Implement the Blog Post Deletion

Requirements for Deletion

The blog post deletion feature allows users to remove a blog post with a confirmation step to prevent accidental deletions. Below are the detailed requirements:

- **Deletion Process**:
    - A "Delete" button must be visible on the blog post view or list item.
    - Clicking the "Delete" button opens a confirmation dialog.
    - The dialog must prompt the user with a message like "Are you sure you want to delete this post?"
    - It must offer two options: "Cancel" (closes the dialog, no action taken) and "Delete" (deletes the post and redirects to the blog list or another appropriate page).
- **Accessibility**:
    - The "Delete" button must be keyboard-navigable (e.g., focusable with the Tab key).
    - The confirmation dialog must be a modal that traps focus, restricting keyboard interaction to the dialog until dismissed.
    - Include ARIA attributes (e.g., `role="dialog"`, `aria-labelledby`, `aria-describedby`) to ensure screen reader compatibility.
    - Manage focus: move it to the dialog when opened and back to the "Delete" button when closed.
- **Responsiveness**:
    - The dialog must adapt to screen sizes:

        - **Desktop (1200px+)**: Centered, fixed-width dialog.
        - **Mobile (up to 768px)**: Near full-width dialog for readability and usability.
    - The "Delete" button must have a touch target of at least 44x44 pixels on mobile.

- **Interactivity**:

    - The dialog can be closed by clicking outside it or pressing "Escape."
    - The "Delete" button should show a loading state or be disabled during deletion to avoid duplicate submissions.

- **Styling**:

    - Follow the UI/UX designs below for consistency.
    - Use CSS modules to scope styles and avoid conflicts.
    - Ensure uniform typography, colors, and spacing.

- **Assumptions**:

    - Deletion logic (e.g., API calls) is handled externally via a callback.
    - Focus is on UI and interaction, not data persistence.

Component Structure

The deletion feature uses two React components: `DeleteButton` and `ConfirmationDialog`, separating the trigger from the confirmation for reusability and clarity.

DeleteButton Component

- **Purpose**: A button that initiates the deletion process by opening the confirmation dialog.
- **Props**:
    - `onClick`: Callback to open the dialog.
- **Behavior**:
    - Displays "Delete" text.
    - Triggers `onClick` when pressed.
- **Example Code**:

```
import React from 'react';

import styles from './DeleteButton.module.css';

const DeleteButton = ({ onClick }) => {

  return (
    <button className={styles.deleteButton} onClick={onClick}>
      Delete
    </button>
  );
};
export default DeleteButton;
```

ConfirmationDialog Component

- **Purpose**: A modal dialog to confirm the deletion action.
- **Props**:
    - `isOpen`: Boolean to show/hide the dialog.
    - `onClose`: Callback to close the dialog.
    - `onConfirm`: Callback to execute the deletion.
- **Behavior**:
    - Hidden when `isOpen` is false.
    - Shows a confirmation message and "Cancel"/"Delete" buttons when open.
    - "Cancel" or clicking outside calls `onClose`; "Delete" calls `onConfirm`.
    - Traps focus and closes on "Escape."
- **Example Code**:

```
import React, { useEffect, useRef } from 'react';
import styles from './ConfirmationDialog.module.css';
const ConfirmationDialog = ({ isOpen, onClose, onConfirm }) => {
  const dialogRef = useRef(null);
  useEffect(() => {
    if (isOpen) {
      dialogRef.current?.focus();
    }

  }, [isOpen]);

  if (!isOpen) return null;

  return (

    <div className={styles.overlay} onClick={onClose}>

      <div

        className={styles.dialog}

        role="dialog"

        aria-labelledby="dialog-title"

        aria-describedby="dialog-description"

        ref={dialogRef}

        tabIndex="-1"

        onClick={(e) => e.stopPropagation()}

      >

        <h2 id="dialog-title">Confirm Deletion</h2>

        <p id="dialog-description">Are you sure you want to delete this
post?</p>

        <div className={styles.buttons}>

          <button onClick={onClose}>Cancel</button>

          <button onClick={onConfirm}>Delete</button>

        </div>

      </div>
    </div>

  );

};

export default ConfirmationDialog;
```

Notes

- **Parent Component**: Controls dialog state and provides the `onConfirm` handler.
- **Files**: `DeleteButton.js`/`DeleteButton.module.css` and `ConfirmationDialog.js`/`ConfirmationDialog.module.css`.

UI/UX Designs

The designs ensure a consistent, user-friendly experience across devices.

Desktop View (1200px and above)

- **Delete Button**:
    - Background: Red (#FF0000), Text: White, Corners: 4px.
    - Padding: 10px 20px, Font: 16px, bold.
- **Confirmation Dialog**:
    - Centered, max-width 400px, white background (#FFFFFF), subtle shadow.
    - **Title**: `<h2>`, 24px, bold, #333333.
    - **Description**: `<p>`, 16px, #666666.
    - **Buttons**:
        - "Cancel": Gray (#CCCCCC), black text.
        - "Delete": Red (#FF0000), white text.
        - Both: 16px, 10px padding horizontally.
    - **Spacing**: 20px (title-description), 30px (description-buttons), 20px between buttons (side by side).

Mobile View (up to 768px)

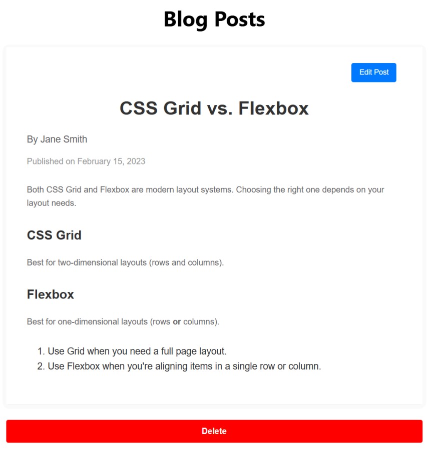
- **Delete Button**:
    - Same as desktop, full-width, min-height 44px.
- **Confirmation Dialog**:
    - 90% screen width, centered.
    - Same typography/colors as desktop.
    - Buttons stacked vertically, 10px gap.
    - **Spacing**: 15px (title-description), 20px (description-buttons).

Additional Notes

- **Font**: Sans-serif (e.g., Arial, Roboto).
- **Focus**: Blue outline (#007BFF) on focused buttons.
- **Overlay**: Semi-transparent (rgba(0, 0, 0, 0.5)).

Reference Screenshots

1. **Desktop view**



Blog Posts

Edit Post

CSS Grid vs. Flexbox

By Jane Smith

Published on February 15, 2023

Both CSS Grid and Flexbox are modern layout systems. Choosing the right one depends on your layout needs.

CSS Grid

Best for two-dimensional layouts (rows and columns).

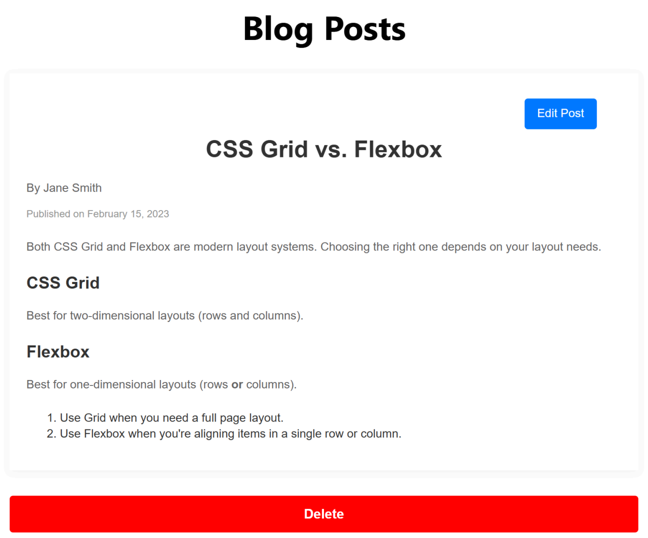
Flexbox

Best for one-dimensional layouts (rows **or** columns).

1. Use Grid when you need a full page layout.
2. Use Flexbox when you're aligning items in a single row or column.

Delete

2. **Tablet view**



Blog Posts

Edit Post

CSS Grid vs. Flexbox

By Jane Smith

Published on February 15, 2023

Both CSS Grid and Flexbox are modern layout systems. Choosing the right one depends on your layout needs.

CSS Grid

Best for two-dimensional layouts (rows and columns).

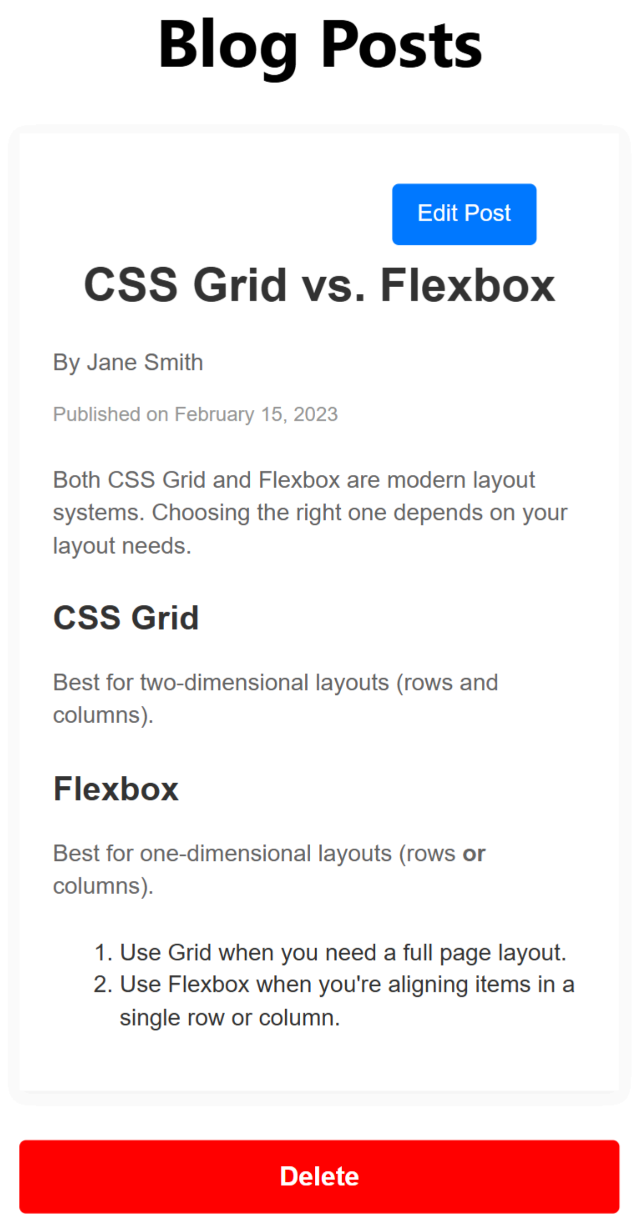
Flexbox

Best for one-dimensional layouts (rows **or** columns).

1. Use Grid when you need a full page layout.
2. Use Flexbox when you're aligning items in a single row or column.

Delete

3. **Mobile view**


Blog Posts

Edit Post

CSS Grid vs. Flexbox

By Jane Smith

Published on February 15, 2023

Both CSS Grid and Flexbox are modern layout systems. Choosing the right one depends on your layout needs.

CSS Grid

Best for two-dimensional layouts (rows and columns).

Flexbox

Best for one-dimensional layouts (rows **or** columns).

1. Use Grid when you need a full page layout.
2. Use Flexbox when you're aligning items in a single row or column.

Delete

This content provides everything needed to implement the blog post deletion feature, covering requirements, component structure, and UI/UX designs for accessibility and responsiveness. You can now proceed with developing the `DeleteButton` and `ConfirmationDialog` components accordingly!