Content for Challenge 2: Implement the Blog Post Viewing Feature

Requirements for Viewing Individual Blog Posts

The blog post viewing feature is designed to display the full details of a single blog post in a clear and user-friendly manner. The following requirements specify the expected functionality and behavior:

- **Content Display**:
    - Each blog post must include:
        - **Title**: The headline of the blog post, displayed prominently at the top.
        - **Content**: The full text of the blog post, which may include HTML elements like paragraphs, headings, and lists.
        - **Author**: The name of the post's author.
        - **Publication Date**: The date the post was published, formatted as "Month Day, Year" (e.g., "January 1, 2023").
    - All content must be fully visible without truncation.
- **Responsiveness**:
    - The layout must adapt to different screen sizes:
        - **Mobile (up to 768px)**: Single-column layout with smaller font sizes for readability.
        - **Tablet (769px to 1199px)**: Single-column layout with slightly larger fonts and padding.
        - **Desktop (1200px and above)**: Wider layout with increased padding and font sizes for a comfortable reading experience.
    - Content must be responsive and fit within their containers.

- **Data Handling**:
    - The component receives a blog post object with:
        - `title`: The post's title (string).
        - `content`: The full content (string, may include HTML).
        - `author`: The author's name (string).
        - `date`: The publication date (string in ISO format, e.g., "2023-01-01", or a Date object).
    - If the post data is unavailable, display: "Blog post not found."
- **Interactivity**:
    - Links within the content should be functional, opening external links in a new tab.
- **Accessibility**:
    - Use semantic HTML (e.g., `<h1>` for titles, `<p>` for text).
    - Ensure a logical reading order for screen readers.
- **Styling**:
    - Styles must follow the UI/UX designs provided below.
    - Use CSS modules to scope styles and avoid conflicts.
    - Maintain consistent typography, colors, and spacing.
- **Assumptions**:
    - Blog post data is provided by a parent component or router; no data fetching is required within the component.
    - Features like comments or sharing are out of scope unless specified.

Component Structure

The blog post viewing feature is implemented using a single React component: `BlogPostDetail`. This component handles rendering the full blog post details.

BlogPostDetail Component

- **Purpose**: Renders the title, content, author, and date of an individual blog post.
- **Props**:
    - `title` (string): The post's title.
    - `content` (string): The full content, potentially including HTML.

    - `author` (string): The author's name.
    - `date` (string/Date): The publication date.
- **Responsibilities**:
    - Display the title, author, and formatted date at the top.
    - Render the content, parsing HTML safely.
    - Show "Blog post not found" if data is missing.
    - Apply styles from `BlogPostDetail.module.css`.
- **Example Structure**:

```
import React from 'react';

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

const BlogPostDetail = ({ title, content, author, date }) => {

  if (!title || !content || !author || !date) {

    return <p>Blog post not found.</p>;

  }

  const formattedDate = new Date(date).toLocaleDateString('en-US', {

    month: 'long',

    day: 'numeric',

    year: 'numeric',

  });

  return (

    <div className={styles.blogPostDetail}>

      <h1 className={styles.title}>{title}</h1>

      <p className={styles.author}>By {author}</p>

      <p className={styles.date}>Published on {formattedDate}</p>

      <div className={styles.content} dangerouslySetInnerHTML={{ __html:
```

```
content }} />

    </div>

  );

};

export default BlogPostDetail;
```

Architecture Notes

- **Parent Component**: Passes the blog post data to `BlogPostDetail`. Data fetching occurs outside this component.
- **Dependencies**: Assumes React is available; `react-router-dom` may be used in the parent if navigation is involved.
- **File Structure**:
  - `BlogPostDetail.js`
  - `BlogPostDetail.module.css`

UI/UX Designs for Desktop and Mobile Views

The UI/UX designs outline the visual layout, typography, colors, and spacing for the blog post viewing feature across desktop and mobile devices.

Desktop View (1200px and above)

- **Layout**:
  - Single-column, centered layout.
  - Maximum content width: 800px.
  - Padding: 40px on both sides.
- **Typography**:
  - Title: `<h1>`, 36px, bold, #333333.
  - Author: `<p>`, 18px, regular, #666666, prefixed with "By".
  - Date: `<p>`, 16px, regular, #999999, prefixed with "Published on".
  - Content: `<div>`, 18px, line height 1.6, #333333.
- **Styling**:
  - Background: #FFFFFF with a subtle box shadow or border.
  - Links: Underlined, #007BFF, hover state #0056b3.
- **Spacing**:
  - 20px below title.
  - 10px below author and date.

    - 30px above content.

Mobile View (up to 768px)

- **Layout**:
    - Single-column, full-width layout.
    - Padding: 20px on both sides.
- **Typography**:
    - Title: `<h1>`, 28px, bold, #333333.
    - Author: `<p>`, 14px, regular, #666666.
    - Date: `<p>`, 12px, regular, #999999.
    - Content: `<div>`, 14px, line height 1.4, #333333.
- **Styling**:
    - Same colors as desktop.
- **Spacing**:
    - 15px below title.
    - 5px below author and date.
    - 20px above content.

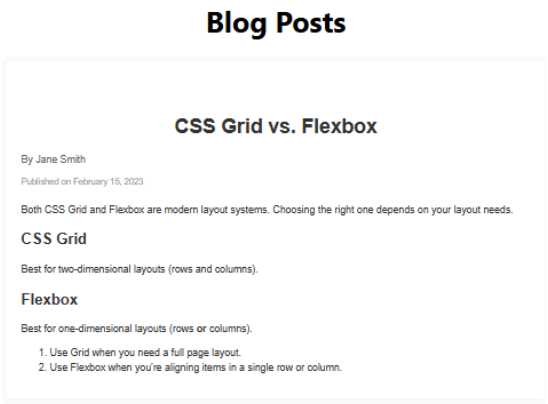
Visual Notes

- **Consistency**: Use a sans-serif font (e.g., Arial or Roboto) across all views.
- **Readability**: Maintain high contrast between text and background.

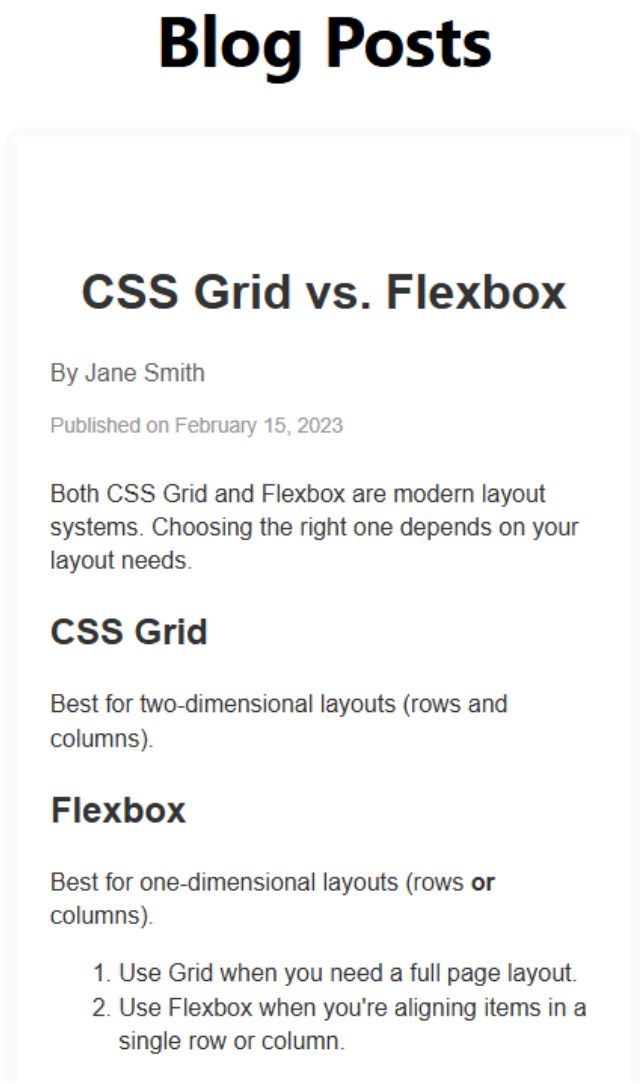
Reference Screenshots

Desktop view:

Tablet view:



Mobile View:



This content provides a complete foundation for implementing the blog post viewing feature. You can now use the requirements, component structure, and UI/UX designs to develop the `BlogPostDetail` component, ensuring it meets the responsive and accessibility standards outlined. Let me know if you need help with the coding process!