**Case Study: Personal Blog Application**

**Problem Statement:**

**Background**

MyBlog is a single-page application (SPA) designed to serve as a personal blog platform where users can read blog posts, view detailed content for individual posts, filter posts by category, and access a contact page. The application requires a robust client-side routing solution to provide seamless navigation without full page reloads, ensuring a smooth and engaging user experience. Security-related concerns, such as authentication or data protection, are not within the scope of this implementation.

**Objective**

Develop a React JS-based personal blog application called "MyBlog" using React Router (v6) to handle client-side routing. The application should enable users to navigate between different pages, access dynamic content based on URL parameters, maintain a consistent layout, and handle invalid routes gracefully.

**Requirements**

The application must meet the following routing-specific requirements:

1. **Navigation Routes**:
   * /: A Home page displaying a welcome message and a preview of featured blog posts.
   * /blog: A Blog page listing all available blog posts.
   * /blog/:postId: A dynamic route to display the details of a specific blog post based on its ID.
   * /category/:categoryName: A dynamic route to display blog posts filtered by a specific category (e.g., Tech, Lifestyle).
   * /contact: A Contact page with a simple form for users to submit inquiries or feedback.
   * \*: A catch-all route to handle 404 errors for invalid URLs.
2. **User Experience**:
   * Ensure seamless navigation between pages without full page reloads.
   * Maintain a consistent layout with a persistent header (including navigation links) and footer across all routes.
   * Provide intuitive navigation using links instead of traditional HTML anchor tags.
   * Display user-friendly error messages for invalid routes or non-existent posts/categories.
3. **Dynamic Routing**:
   * Implement dynamic routes to render specific blog post details using a URL parameter (postId).
   * Support category-based filtering of blog posts using a URL parameter (categoryName).
   * Handle cases where a post or category does not exist by displaying appropriate fallback messages.
4. **Technical Constraints**:
   * Use React JS for the front-end framework.
   * Use React Router (v6) for client-side routing.
   * Optional: Use Tailwind/Bootstrap CSS for styling to ensure a responsive and consistent design.

**Challenges**

* **Dynamic Route Handling**: Ensure robust handling of dynamic routes (/blog/:postId and /category/:categoryName) to prevent errors when invalid parameters are provided.
* **Content Filtering**: Implement efficient filtering of blog posts by category while maintaining performance.
* **Consistent UI**: Maintain a consistent header and footer across all routes while allowing the main content to change based on the route.
* **Error Handling**: Gracefully handle invalid routes or missing content with a user-friendly 404 page.