Take-Home Assessment: Blog Post Manager

Objective

Develop a Blog Post Manager application using Next.js that enables users to:

- View a list of blog posts.
- View details of a single blog post.
- Add a new blog post.
- Edit an existing blog post.
- Delete a blog post.

Technical Requirements

- Framework: Use Next.js 14+ (App Router preferred).
- Styling: Implement styles using Tailwind CSS and ShadCN UI.
- State Management: Utilize Redux Toolkit for state management.
- Data Fetching: Use React Query for API fetching and caching.
- Data: Pre-populate the application with at least 350 dummy blog posts.
- Form Validation: Implement form validation using React Hook Form and Zod.

Features to Implement

- **Homepage (/):**
  Display a paginated list of all blog posts, showing the title, author, and a short description.
  Implement pagination to efficiently handle the display of 350 posts.
- **View Single Blog Post (/post/{id}):**
  Display the full content of a selected post, including the title, author, and publication date.

- **Create New Blog Post (/new-post):**
  Provide a form to add a new blog post with fields for title, content, and author.
  Validate the form fields using React Hook Form and Zod.
  Handle form submission using React Query to manage the post request.
  Use a WYSIWYG editor for the content
- **Edit Blog Post (/edit/{id}):**
  Provide a form pre-populated with the existing post details, allowing users to edit and save changes.
- **Delete Blog Post:**
  Include a delete option for each post with a confirmation prompt before deletion.
  Use React Query to handle the deletion process and update the UI accordingly.

Data Handling

- Mock API: Set up a mock API using tools like JSON Server(https://jsonplaceholder.typicode.com/posts) to simulate backend functionality.
- Dummy Data: Generate and load 350 dummy blog posts into the mock API to simulate a realistic dataset.

Bonus Features (Optional)

- Search Functionality: Add a search bar to filter posts by title or content.
- Category Tags: Allow posts to have category tags and enable filtering by categories.

Submission Guidelines

- GitHub Repository: Submit the project as a GitHub repository with a clear and concise README.md file.
- README.md: Include instructions on how to set up and run the project locally, along with any other relevant information.

Evaluation Criteria

- Code Quality: Assess the organization, readability, and maintainability of the codebase.
- Functionality: Ensure all specified features are implemented correctly and work as intended.
- State Management: Evaluate the effective use of Redux Toolkit for managing application state.
- Data Fetching: Review the implementation of React Query for data fetching and caching strategies.
- Styling: Examine the use of Tailwind CSS and ShadCN UI for styling and responsiveness.
- Form Validation: Verify the implementation of form validation using React Hook Form and Zod.
- User Experience: Consider the overall user experience, including UI design and responsiveness.

This assessment is designed to evaluate your proficiency with Next.js, Tailwind CSS, ShadCN UI, Redux Toolkit, React Query, and form validation libraries. It also assesses your ability to handle a substantial dataset efficiently. Focus on code quality, best practices, and creating a seamless user experience.