dMACQ software Private Limited

Technical Round Interview Question

Role: Frontend Developer with react.js

Build a Next.js application with React Query to manage a mock Blog platform with authentication.

Requirements:

1) **Setup and Structure**
   i) Create a Next.js project with pages for login, registration, and a blog listing.
   ii) Use React Query for API interactions, including authentication and blog management.
2) **Authentication**
   i) Create a simple authentication flow with the following endpoints:
   ii) `POST /api/register`: Register a new user with `email` and `password`.
   iii) `POST /api/login`: Login and return a token.
   iv) `GET /api/auth-check`: Check if a user is authenticated (use a mock API).
   v) Store the token in cookies or localStorage and use it to authenticate API requests.
3) **Blog Management**
   i) Fetch blog posts using the following endpoints:
   ii) `GET /api/posts`: Fetch all posts (requires authentication).
   iii) `GET /api/posts/{id}`: Fetch a single post by ID (requires authentication).
4) **Features**:
   i) Paginated blog list (10 posts per page).
   ii) Search box to filter posts by title (client-side filtering).
   iii) Post detail view with a "like" button and toast notification.
   iv) Preload the next page's data when scrolling to the bottom (infinite scroll).
5) **Authentication Workflow**
6) **Login/Registration Pages**:
   i) Create forms for users to register and log in.
   ii) Use React Query for the API requests.
7) **Protected Routes**:
   i) Redirect unauthenticated users to the login page.
   ii) Allow authenticated users to access the blog and post details.

8) **React Query Usage**
   i) Use React Query to handle data fetching, caching, and mutations for blogs and authentication.
   ii) Implement query invalidation for authentication status.

9) **UI/UX**
   i) Use a CSS framework like MUI with Tailwind CSS for styling.
   ii) The application should be responsive and visually clean.

10) **Testing** (using Cyrpess):
   i) Write unit tests for components (e.g., blog list and blog detail).
   ii) Write integration tests for API calls.

11) ***Evaluation Criteria:***
   i) **Authentication Flow**: Secure and functional implementation of login, registration, and token-based authentication.
   ii) **Protected Routes**: Proper redirection and handling of authenticated/unauthenticated states.
   iii) **State Management**: Efficient use of React Query for fetching and managing state across the application.
   iv) **Error Handling**: Clear error messages for failed login/registration or API requests.
   v) **Pagination and Search**: Proper implementation of pagination and client-side search.
   vi) **UI/UX**: Clean, responsive, and user-friendly design.
   vii) **Performance**: Smooth scrolling, optimized re-renders, and caching.