Job Listing Platform (SPA)

Requirements

1. UI Layout: Design a clean, responsive layout with the following sections:

1. Header
   - Application name (e.g., “Job Finder”).
   - Search bar for jobs by Title or Company Name.
   - Navigation links.
2. Job List Page
   - Display job postings in a card/grid format with the following details:
   - Job Title
   - Company Name
   - Location
   - Salary Range (e.g., ₹60,609 - ₹85,340).
   - Employment Type (e.g., Full-Time, Part-Time, or Co-founder).
3. Job Details Page
   - When a user clicks on a job, navigate to the Job Details Page (/jobs/:id).
   - Show the following details:
   - Job Title
   - Company
   - Full Job Description
   - Qualifications (formatted as a list).
   - Number of Openings
   - Salary Range
   - Application Deadline
   - Contact Information
4. Footer
   - Simple footer (e.g., Copyright © 2024).

2. Features to Implement

1. Fetch Job Data
   - Use the API endpoint: https://jsonfakery.com/jobs
2. Search Functionality
   - Allow users to search jobs by Title or Company Name.
3. Filter Functionality

- Implement filtering for:
- Location
- Job Category
- Employment Type
- Add a toggle to display only Remote Jobs (is_remote_work).

4. Routing
   - Use React Router (React.js) or Next.js dynamic routes for navigation:
   - Home Page: /
   - Job Details Page: /jobs/:id
5. State Management
   - Use React Context API or libraries like Recoil/Redux to manage state for:
     a. Job data
     b. Filters and search terms
6. Responsive Design
   - Ensure the app works on mobile, tablet, and desktop devices.

3. Tech Stack

1. Framework: React.js (Vite/CRA) or Next.js
2. Routing: React Router / Next.js
3. State Management: Context API, Recoil, or Redux
4. Data Fetching: Fetch API or Axios
5. UI Library (optional): Tailwind CSS (Shadcn or Radix can be used)
6. Datagrid: AGGrid

4. Evaluation Criteria

1. Code Quality
   - Code readability, structure, and use of modern practices.
2. UI/UX
   - Clean and responsive design.
3. Functionality
   - Working search, filters, and job details page.

4. State Management

    - Proper use of state for dynamic data.

5. Error Handling

    - Handle edge cases (e.g., no jobs found or API failure).