React Frontend Assignment: Doctor's Directory with Search and Filtering

Objective:

The goal of this assignment is to assess the candidate's ability to build a functional and well-structured React application. The candidate will create a Doctor's Directory that allows users to view, search, and filter a list of doctors based on specialty and location.

The candidate is expected to focus on:
- Attention to detail (UI consistency, form validation, error handling, etc.)
- Code structure and component reusability
- State management and clean API integration
- Responsiveness and cross-browser compatibility

Assignment Brief:

App Requirements:

You are required to create a web application that displays a list of doctors and allows users to:
1. View all doctors in a list format.
2. Search doctors by name.
3. Filter doctors by specialty and location.

Functionalities to Implement:

1. **Doctor List Page:**
   - Display a list of doctors fetched from a mock API or static JSON file.
   - Each doctor card should show:
     - Doctor's name
     - Specialty
     - Location (City, State)
     - Rating (1 to 5 stars)
   - Make the page responsive for different screen sizes (desktop, tablet).

2. **Search Bar:**
   - Implement a search input that allows users to search doctors by name.
   - The search should dynamically filter the doctor list based on the entered text.

3. **Filters:**
   - Implement two dropdown filters:
     - Specialty: Filter doctors by their medical specialty (e.g., Cardiology, Dermatology).
     - Location: Filter doctors by their location (City or State).
   - Filters should work in combination with the search bar, so users can search and filter at the same time.

4. **Doctor Detail Modal:**
   - When a doctor card is clicked, open a modal showing more detailed information about the doctor:
   - Name, specialty, location, phone number, and email.
   - The modal should have a close button to dismiss it.

5. **Loading & Error States:**
   - Show a loading spinner while the data is being fetched.
   - Implement basic error handling to show a message if the data fetch fails.

Bonus (Optional):

- **Pagination:**
  - If there are more than 10 doctors, implement pagination to navigate through the list.

- **Form Validation:**
  - Implement a form for adding new doctors (just front-end, no backend required) with form validation.

Design Considerations:

- **Responsiveness:** Ensure the layout works well on both mobile and desktop devices.
- **Consistency**: Maintain a clean and consistent UI design (you can use any CSS framework or write custom CSS).
- **Performance:** Optimize your components, especially when rendering a large list of doctors.
- **Code Quality:** Follow best practices for code organization, naming conventions, and component structure.

Sample Doctor Data (JSON):

```
Unset
[
  {
    "id": 1,
    "name": "Dr. John Doe",
    "specialty": "Cardiology",
    "location": "New York, NY",
    "rating": 4.8,
    "phone": "(123) 456-7890",
    "email": "johndoe@example.com"
  },
  {
    "id": 2,
    "name": "Dr. Jane Smith",
    "specialty": "Dermatology",
    "location": "Los Angeles, CA",
    "rating": 4.7,
    "phone": "(987) 654-3210",
    "email": "janesmith@example.com"
  }
]
```

Deliverables:

1. A GitHub repository containing:
   - All the code, organized with clear folder structure and comments.
   - A **README.md** file with setup instructions and details of your approach.
   - Add them as collaborators to the GitHub repository: jain.sankeet2210@gmail.com, letsomics@gmail.com and shekharp77.

2. Live demo (optional): If possible, deploy the application on a platform like Vercel or Netlify for easier testing.

Evaluation Criteria:

1. Attention to Detail:
- Are the UI elements consistent (buttons, cards, modals)?
- Are search and filter functionalities working as expected?
- Does the app handle loading and error states gracefully?

2. Component Structure:
- Are React components well-organized and reusable?
- Is the code modular and maintainable?

3. State Management:
- How well is the state managed between the search, filter, and doctor list?
- Is the state logic clean and easy to understand?

4. Responsiveness:
- Does the application work well on different screen sizes (desktop, tablet, mobile)?

5. Code Quality:
- Is the code clean and easy to follow?
- Are proper naming conventions followed?
- Is the application free of bugs and logical errors?

6. Bonus Points:
- Pagination, Dark Mode, and Form Validation will be considered for bonus points if implemented well.

Turnaround Time:

- **The expected completion time for this assignment is 2 days.**