NestJS + ReactJS with Database Integration

Full Stack Interview Task:

Part 1: Framework Mastery

Objective: Create a Book Management System where users can view a list of books and add new books to the list. Data persistence should be achieved by using a database of your choice (e.g., PostgreSQL, MongoDB).

Backend (NestJS):

1. Set up a NestJS project and integrate it with a database using TypeORM, Prisma, or another ORM/ODM of your choice.
2. Design a database schema for books. Each book should have:
   - a unique ID
   - title
   - author
   - published date
   - a brief description (optional)
3. Create a migration to initialize the database schema.
4. Implement an API endpoint `GET /books` that fetches the list of books from the database.
5. Create an API endpoint `POST /books` that accepts book data and saves it to the database.

Frontend (ReactJS):

1. Set up a React project.
2. Create a main page that displays a list of books fetched from the NestJS backend.
3. Implement a form to add a new book. When submitted, it should make a POST request to the backend and refresh the book list.

Part 2: Problem Solving & Algorithmic Skills

Objective: Introduce features for searching books by title and filtering them based on the year of publication.

Backend (NestJS):

1. Modify the `GET /books` endpoint to accept query parameters for title search (`title`) and year filter (`year`).
2. Implement a search algorithm to filter books by title in the database. It should support partial matches.
3. Implement a filter in the database query to return books published in the specified year.

Frontend (ReactJS):

1. Add a search bar to search books by title. As users type, the list of books should dynamically update to show only the matching titles.
2. Implement a dropdown or input field to filter books by year. Once a year is selected or input, only books from that year should be displayed.

Bonus (Optional):

Backend (NestJS):

1. Add pagination support to the `GET /books` endpoint.
2. Implement an API endpoint `DELETE /books/:id` to allow book removal.

Frontend (ReactJS):

1. Implement pagination controls on the frontend to navigate through large lists of books.
2. Add a delete button next to each book. When clicked, it should remove the book and refresh the list.

Evaluation Criteria:

- **Correctness**: All specified endpoints and features should work as described.
- **Database Design**: The schema should be well-designed and normalized where appropriate. Indexing should be considered for faster search.
- **Code Quality**: The submitted code should be clean, organized, and follow best practices.
- **Error Handling**: Proper error handling for both frontend and backend. Consider scenarios like database connection failures, incomplete form submissions, etc.

- **Responsiveness**: The React frontend should be reasonably responsive and handle dynamic updates smoothly.

Submission:

You are expected to provide:

1. A GitHub repository link containing the full-stack project.
2. Instructions on how to set up, run, migrate the database, and test the project locally.

You are expected to adhere to the time restrictions of the task. Late submissions will be penalized.