**Challenges Faced and Resolutions**

1. **Duplicate Validation**
   * **Challenge:** Ensuring no duplicate contacts (based on email or first name + last name) were added to the database.
   * **Resolution:** Added server-side validation in the backend to check for existing records before creating or updating a contact. Also implemented error handling in the frontend to display meaningful alerts to the user.
2. **Pagination and Sorting**
   * **Challenge:** Displaying a large number of contacts in a manageable way with sorting and pagination.
   * **Resolution:** Utilized Material-UI's TablePagination and TableSortLabel components for a seamless user experience.
3. **Frontend-Backend Synchronization**
   * **Challenge:** Keeping the frontend in sync with backend operations (add, edit, delete) without needing a full page refresh.
   * **Resolution:** Used axios for API requests and updated the local state (useState) dynamically to reflect backend changes.
4. **Error Handling for User Feedback**
   * **Challenge:** Providing clear and actionable feedback to users when errors occurred, such as duplicate entries or server issues.
   * **Resolution:** Integrated error handling in the frontend to catch backend validation errors and display alerts with the specific issue.

**Chosen Database: MongoDB**

MongoDB, a NoSQL document-based database, was chosen for this project due to its flexibility, scalability, and suitability for handling dynamic data structures.

**Why MongoDB Fits This Project:**

1. **Document-Oriented Structure**:
   * MongoDB stores data in JSON-like documents, which makes it easy to map the contact data model (e.g., first name, last name, email, etc.) directly to database documents. This eliminates the need for complex schema design.
2. **Flexibility**:
   * The dynamic schema of MongoDB allows for easy addition or modification of fields (e.g., adding new contact fields in the future), which makes it ideal for evolving applications like this contact management system.
3. **Scalability**:
   * MongoDB is designed to handle large datasets and scale horizontally, which makes it a great choice if the project grows to handle a large number of contacts.
4. **Ease of Integration**:
   * MongoDB integrates seamlessly with Node.js and the Mongoose library, simplifying database operations such as CRUD (Create, Read, Update, Delete).
5. **Cloud-Based Solution**:
   * Using MongoDB Atlas, a fully managed cloud database, simplifies setup, management, and maintenance. It also provides features like automated backups and global availability.

### ****Conclusion****

MongoDB’s flexibility, scalability, and ease of use made it the ideal choice for this project, enabling rapid development and the ability to handle potential future requirements.