Challenges Faced During the Assignment and Solutions

1. Integrating Frontend and Backend

- **Challenge:** Ensuring smooth communication between the ReactJS frontend and the Node.js backend while handling asynchronous requests and responses. There were instances of CORS (Cross-Origin Resource Sharing) errors when the frontend tried to fetch data from the backend running on a different port.
- **Solution:** Implemented the cors middleware in the backend to allow cross-origin requests:

  ```
  const cors = require('cors');

  app.use(cors());
  ```

This resolved the CORS issues and enabled seamless API communication.

---

2. Data Validation and Error Handling

- **Challenge:** Validating input data on both the frontend and backend while preventing invalid or duplicate entries. For example, preventing duplicate email entries in the database.
- **Solution:**
  - Added form validation using React's state and Material-UI's validation helpers (e.g., error and helperText).
  - Implemented server-side validation using Mongoose's unique field and error handling in the API:

    ```
    contactSchema = new mongoose.Schema({

      email: { type: String, required: true, unique: true },

    });
    ```

---

3. Implementing Sorting and Pagination

- **Challenge:** Sorting and paginating large datasets in the frontend while ensuring the table remains responsive and performant.
- **Solution:** Used Material-UI's Table component with built-in support for sorting and pagination. For backend support, added query parameters (page and limit) in the GET /contacts API:

  ```
  app.get('/contacts', async (req, res) => {

    const { page = 1, limit = 10 } = req.query;

    const contacts = await Contact.find()

      .skip((page - 1) * limit)

      .limit(Number(limit));
  ```

```
  res.json(contacts);
});
```

---

4. Form State Management

- **Challenge:** Managing form state while implementing features like pre-filling form fields for editing and clearing the form after submission.
- **Solution:** Used React hooks (useState and useEffect) to manage form state dynamically. Conditional logic was added to determine whether the form was being used for adding or editing a contact.

---

5. Database Connection and Deployment

- **Challenge:** Configuring MongoDB to work seamlessly across local development and deployment environments.
- **Solution:** Used environment variables to configure the MongoDB URI, ensuring flexibility for different setups:

  ```
  const MONGO_URI = process.env.MONGO_URI || 'mongodb://localhost:27017/contactsDB';
  mongoose.connect(MONGO_URI, { useNewUrlParser: true, useUnifiedTopology: true });
  ```

  Tested the database connection thoroughly before moving to API development.

---

6. Styling Consistency

- **Challenge:** Maintaining a consistent UI design throughout the app while ensuring responsiveness across different devices.
- **Solution:** Leveraged Material-UI’s built-in styling system, themes, and layout components (e.g., Grid, Box). This standardized the design and reduced the need for custom CSS.

These challenges were great learning opportunities, helping me understand practical implementation issues and equipping me with strategies to tackle them in future projects.