# Assessment

**Module-13**:React – Applying Redux

**Case Overview**
**Scenario:**
You are tasked with building a scalable React + Redux application to manage Hotel Booking **Feedback** after a visit. The system must integrate with a REST API using Redux Toolkit, allow form submissions, manage records with search and filter features, and support create, update, and delete operations.

**Core Functionality**

| **Feature** | **Description** |
|---|---|
| Responsive Web Form | • **Fields**: Full Name,Phone No, Hotel Feedback Comment, Hotel Ratting.<br>• Controlled inputs using useState.<br>• **Validation rules:** Name>=3 chars, Phone=10 digits, Comment Length valid, Ratting validation. Inline Error messages for invalid fields. |
| Feedback Submission (API) | • Use createSlice for crud. On valid submission, send POST request to API. Update Redux store and UI instantly. |
| View Feedback | • Fetch data from api and display in a table with columns:Name,Phone,Comment,Ratting. Include search filters by Name. Show “No records found” if empty. |
| Delete Feedback | • Add the Delete button for each record. On click,confirm action, send DELETE request to api, remove from UI immediately. |

**Key Competencies Tested**

- React Form Handling & Controlled Components
- JavaScript Input Validation
- Redux Toolkit (createSlice, createAsyncThunk, useDispatch, useSelector)

- API Integration (GET, POST, PUT, DELETE)
  Search & Filter Logic
- Real-time UI Updates via Redux Store
- Context API for UI state
- RTK Query for advanced API handling

**Reflective Thinking Questions**

- How would you structure the Redux store for separating book data from UI state (e.g., modals)?
- When should infinite scroll be preferred over pagination in such systems?
- How can RTK Query simplify repeated API calls in this app?
- How would you secure access — e.g., making book deletion available only to admin users?