**ReactJS Exercises Handbook**

**Module 1: React Basics Refresher**

**🔹 Exercise 1: Create a ShoppingCart Component**

**Objective**: Create a ShoppingCart component that accepts a list of products via props and displays them.

**Instructions**:

* Each product has id, name, price, and quantity.
* Render each product inside a <table> or list.

**Hint**:

* Use map() to iterate over products.
* Use key={product.id} in child components.

**🔹 Exercise 2: Add Button to Add Item**

**Objective**: Add an "Add Item" button next to each product.

**Instructions**:

* When clicked, increase the quantity of the product in the cart.

**Hint**:

* Use useState to maintain the cart state.
* Use onClick handlers and update the array immutably.

**Module 2: Forms and State Management**

**🔹 Exercise 3: Add Product Form**

**Objective**: Create a form to add a new product to the cart.

**Instructions**:

* Include fields: Product Name, Price, Quantity
* Validate fields for empty/invalid input before submission

**Hint**:

* Use controlled inputs with useState.
* Prevent form submission with e.preventDefault().
* Use simple if conditions for validation.

**🔹 Exercise 4: Custom Validation Logic**

**Objective**: Implement custom validation messages below input fields.

**Instructions**:

* Show errors only when the user has interacted with the form.

**Hint**:

* Track "touched" fields using an object like { nameTouched: true }.
* Display messages conditionally based on field state and value.

**🔹 Exercise 5: Reusable Input Component**

**Objective**: Create a reusable InputField component with label, input, and error display.

**Hint**:

* Accept label, name, value, onChange, and error as props.

**Module 3: Intermediate Forms + Validation**

**🔹 Exercise 6: Formik Integration (Optional)**

**Objective**: Use Formik to manage form state and validation schema.

**Hint**:

* Start with useFormik() hook and initialValues.
* Use validate or Yup for validation rules.

**🔹 Exercise 7: Cart Summary and Total Price**

**Objective**: Display a live total price calculation below the cart list.

**Hint**:

* Use reduce() on the cart items to calculate the total.

**🔹 Exercise 8: Remove Product from Cart**

**Objective**: Add a "Remove" button to delete a product from the cart.

**Hint**:

* Use filter() to remove the item based on id.

**Module 4: React & Spring Boot Integration**

**🔹 Exercise 9: Setup Spring Boot API**

**Objective**: Create Spring Boot endpoints for:

* GET /cart – get all cart items
* POST /cart – add item
* DELETE /cart/{id} – delete item

**Hint**:

* Create ShoppingCartController, ShoppingCartService, and ShoppingCart entity.
* Use in-memory H2 database.

**🔹 Exercise 10: Fetch Cart Data from Spring Boot**

**Objective**: Fetch cart data using fetch or axios and display it in your React component.

**Hint**:

* Use useEffect() to make API calls when the component mounts.
* Store data in useState.

**🔹 Exercise 11: Add Product to Spring Cart**

**Objective**: Submit product form and POST data to Spring Boot API.

**Hint**:

* Use axios.post with appropriate headers.
* Reset form on successful submission.

**🔹 Exercise 12: Delete Product from Spring Cart**

**Objective**: Delete a cart item via Spring Boot DELETE /cart/{id}.

**Hint**:

* On delete button click, call axios.delete with item id.
* Update local state to reflect deletion without full page reload.

**🔹 Exercise 13: Show Loader and Error States**

**Objective**: Display a loader while fetching and show error message if request fails.

**Hint**:

* Use useState for loading and error.
* Conditionally render content based on those states.