

React Candidate Test Task

Objective

You are required to build a React-based CRUD interface that manages Roles and Permissions, the implementation should follow the provided APIs and replicate the required functionality.

Requirements

- **Role Name Handling:**

Create an input field for entering a Role Name. On entering a role name, check if the role already exists using GET <https://api.test.helpinglab.com/api/role/fetchtestrole>.

If it exists, enter Modify Mode (pre-select checkboxes for permissions where isAllowed = true). If not, enter Add Mode to create a new role.

- **Module Selection:**

Create a multi-select dropdown that fetches modules from API (GET <https://api.test.helpinglab.com/api/role/fetchtestrole>). The dropdown should only display moduleName. When one or more modules are selected, display their submodules in a table with permissions.

- **Permissions Table:**

Each permission should have a checkbox. “**Checking**” adds the permissionID to a payload array and “**Unchecking**” removes it.

- **Form Submission:**

- For adding new role → POST
<https://api.test.helpinglab.com/api/Role/addtestrole>
- For modifying existing roles → PUT
<https://api.test.helpinglab.com/api/Role/modifytestrole>

Submit Payload Example:

```
{ "name": "role name", "permissions": [ 1, 2 // array of selected permission IDs ] }
```

The image shows two screenshots of a REST client interface. The top screenshot is for a GET request to `/api/Role/fetchtestrole`. It shows a 'Parameters' tab with a 'roleName' query parameter input field. The bottom screenshot is for a POST request to `/api/Role/addtestrole`. It shows a 'Request body' tab with a JSON payload: `{ "name": "string", "permissions": [] }`. The 'application/json' content type is selected. Both screenshots have an 'Execute' button at the bottom.

PUT

/api/Role/modifytestrole/{roleName}

Cancel

Parameters

Name

Description

roleName

string (path)

roleName

Request body

application/json

```
{
  "name": "string",
  "permissions": [
    {}
  ]
}
```

Execute

Reference UI (as shown below):

Enter Role Name

Enter Role Name

Select Modules

Sale X

X | v

Modules	Sub Modules	Permissions
Sale	Customers	<input type="checkbox"/> Add Customers <input type="checkbox"/> Update Customers <input type="checkbox"/> Delete Customers <input type="checkbox"/> Manage Customers <input type="checkbox"/> View Customers
	Proforma	<input type="checkbox"/> Add Proforma <input type="checkbox"/> Update Proforma <input type="checkbox"/> Delete Proforma <input type="checkbox"/> Manage Proforma <input type="checkbox"/> View Proforma
	Sample Orders	<input type="checkbox"/> Add Sample Orders <input type="checkbox"/> Update Sample Orders <input type="checkbox"/> Delete Sample Orders <input type="checkbox"/> Manage Sample Orders <input type="checkbox"/> View Sample Orders
	Cargo	<input type="checkbox"/> Add Cargo <input type="checkbox"/> Update Cargo <input type="checkbox"/> Delete Cargo <input type="checkbox"/> Manage Cargo <input type="checkbox"/> View Cargo
	Leads	<input type="checkbox"/> Add Leads <input type="checkbox"/> Update Leads <input type="checkbox"/> Delete Leads <input type="checkbox"/> Manage Leads <input type="checkbox"/> View Leads
	Quote	<input type="checkbox"/> Add Quote <input type="checkbox"/> Update Quote <input type="checkbox"/> Delete Quote <input type="checkbox"/> Manage Quote <input type="checkbox"/> View Quote
	Sale Order	<input type="checkbox"/> Add Sale Order <input type="checkbox"/> Update Sale Order <input type="checkbox"/> Delete Sale Order <input type="checkbox"/> Manage Sale Order <input type="checkbox"/> View Sale Order

Key Deliverables:

- React App with: Role Input Field (Add/Modify logic), Multi-select Dropdown, Dynamic Submodule Table, and Submit button with API integration.
- Reusable Components: Dropdown, Permissions Table, Form handling logic.
- State Management: Using React hooks or context.
- Error Handling: Display validation errors (e.g., Role name required) and handle API errors gracefully.

Evaluation Criteria:

- Correct functionality (CRUD works as described).
- Clean, reusable, and modular React components.
- Proper API integration and error handling.
- UI/UX clarity and usability.
- Code readability and best practices (hooks, props, separation of concerns)