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.

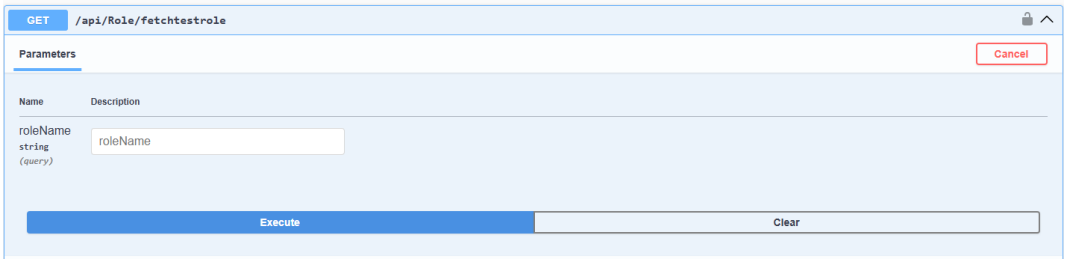
- **Permissions Table:**

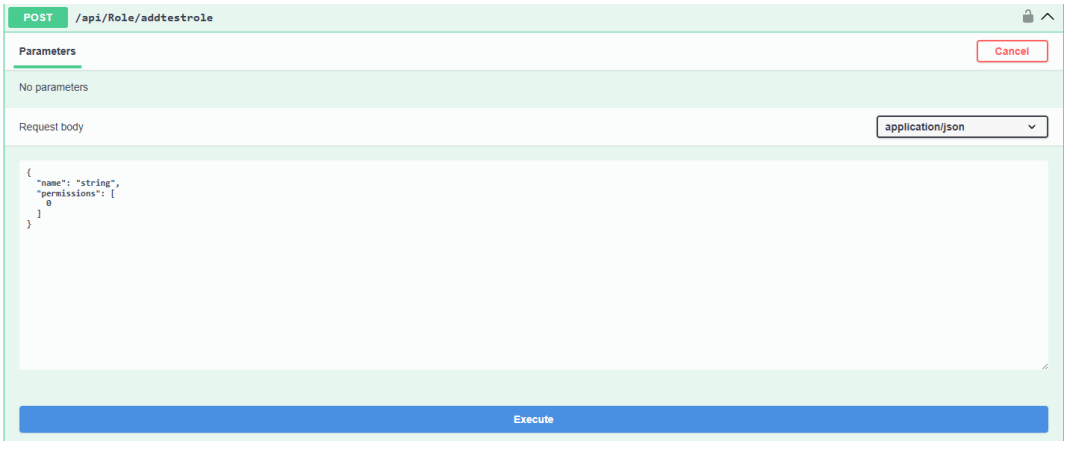
Each permission should have a checkbox. "**Checking**" adds the permissionID to a payload array and "**Unchecking"** removes it.

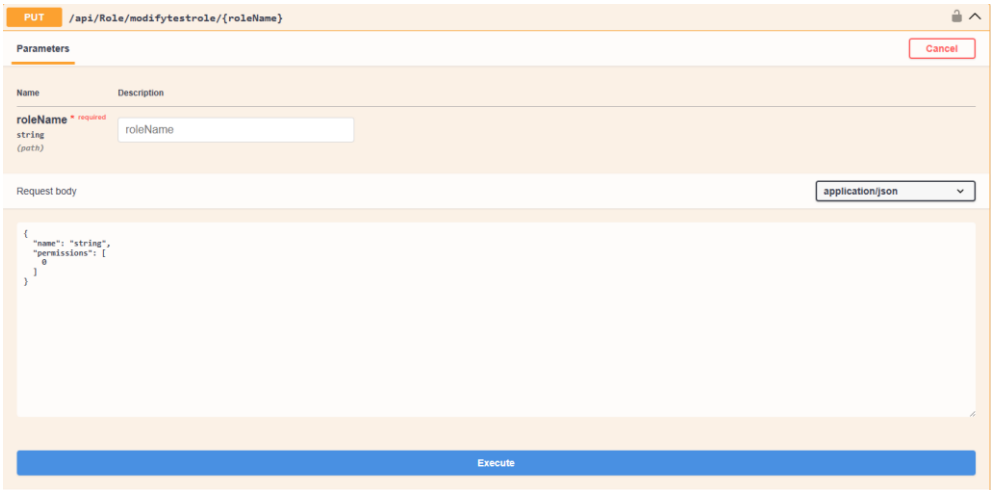
- **Form Submission:**
  - For adding new role → POST
    https://api.test.helpinglab.com/api/Role/addtestrole
  - For modifying existing roles → PUT
    https://api.test.helpinglab.com/api/Role/modifytestrole

Submit Payload Example:

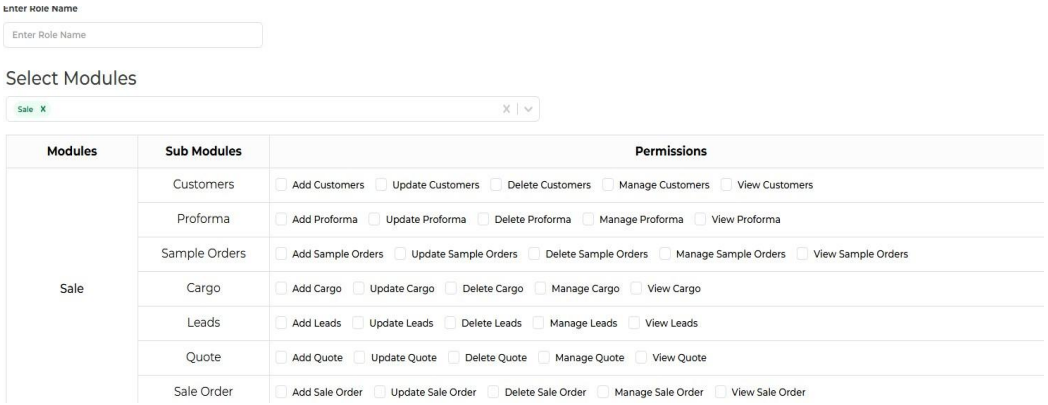
```
{ "name": "role name", "permissions": [ 1, 2 // array of selected permission IDs ] }
```







Reference UI (as shown below):



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)