### **React Assignment: Floor Management Application**

**Objective:** Develop a "Floor Management" application for managing tables in a restaurant. The application should allow users to drag and drop tables, configure table details, and save the room layout. The focus of the assignment is on implementing the core functionalities and ensuring a smooth and user-friendly experience.

**Requirements:**

### **Functional Requirements:**

#### **Table Management:**

1. Users should be able to drag and drop tables onto the floor plan with precise positioning.
2. Users should be able to select a table to view and edit its details (e.g., Table Name, Min Covers, Max Covers).
3. Users should be able to set the height and width for a selected or dragged table (optional).
4. Users should be able to rotate, duplicate, or delete tables using the provided icons.

#### **Table Details:**

1. Users should be able to set the table name, minimum covers, and maximum covers.
2. Users should be able to toggle the table's online status (Active/Inactive).

#### **Room Management:**

1. Users should be able to add new rooms.
2. Users should be able to save the room layout.

#### **Advanced Settings:**

1. Implement a placeholder for advanced settings that can be expanded in the future.

#### **Statistics:**

1. Display the total number of tables, main covers, max covers, and online capacity at the bottom of the UI.

### **Technical Requirements:**

1. **React:** Use React for the frontend framework.
2. **State Management:** Use state management libraries like Redux or Context API for managing the state of the application.
3. **UI/UX:** Use a UI library such as Material-UI, Ant Design, or Tailwind CSS for styling.
4. **Drag and Drop:** Implement precise drag-and-drop functionality using a library like react-beautiful-dnd, react-dnd, or any other suitable option.
5. **Form Handling:** Use form handling libraries like Formik or react-hook-form for managing table details.
6. **Persistence:** Use local storage or a mock backend to save the room layout.

### **Bonus Points:**

1. Implement unit tests using Jest and React Testing Library. (Optional)
2. Implement end-to-end tests using Cypress. (Optional)
3. Use TypeScript for type safety.
4. Integrate a real backend using Node.js/Express and a database of your choice (e.g., MongoDB, PostgreSQL). (Optional)

### **Submission:**

1. Create a GitHub repository and push your code.
2. Provide a README file with instructions on how to run the application.
3. Ensure the code is well-documented and follows best practices.
4. Include a section in the README to validate the use of chosen libraries and tools.
5. Describe the mechanism used for precise drag-and-drop positioning and justify the approach.

### **Evaluation Criteria:**

1. Code quality and organization.
2. Implementation of the required functionalities.
3. User experience and UI design.
4. Use of modern React practices and state management.
5. Testing coverage.
6. Bonus implementations (if any).

