### **Angular Task to Assess Senior-Level Candidate Skills**

#### **Objective**

Build a **Task Management System** using Angular that includes advanced features and demonstrates a solid understanding of Angular concepts, best practices, and problem-solving skills.

### **Requirements**

#### **Functional Requirements**

1. **Dashboard View**
   1. Display a list of tasks in a responsive table.
   2. Columns: Task Name, Description, Priority, Status, Assigned To, Due Date, Actions (View/Edit/Delete).
2. **Add/Edit Task**
   1. Implement a form with fields:
      1. Task Name (required, max 50 characters).
      2. Description (optional, max 200 characters).
      3. Priority (Dropdown: Low, Medium, High).
      4. Status (Dropdown: Pending, In Progress, Completed).
      5. Assigned To (Dropdown populated with dummy users).
      6. Due Date (Date picker, required, cannot be a past date).
   2. The form should validate inputs and display errors where needed.
3. **View Task**
   1. Show detailed task information in a read-only format.
4. **Delete Task**
   1. Prompt a confirmation modal before deletion.
5. **Search and Filter**
   1. Search tasks by name or description.
   2. Filter tasks by priority, status, or assigned user.
6. **Task Status Update**
   1. Implement drag-and-drop functionality to update the task status (e.g., move tasks between columns: Pending → In Progress → Completed).

#### **Technical Requirements**

1. **Components**
   1. Use a modular approach with separate feature modules.
   2. At least 4 components:
      1. TaskListComponent (to display tasks).
      2. TaskFormComponent (to add/edit tasks).
      3. TaskDetailsComponent (to view details).
      4. HeaderComponent (for navigation).
2. **State Management**
   1. Use **NgRx** or **BehaviorSubject** for managing the tasks' state.
   2. Implement actions and reducers/effects for tasks (if using NgRx).
3. **Reusable Services**
   1. Create services for:
      1. Task management (TaskService): CRUD operations.
      2. User management (UserService): Fetch dummy user data.
4. **Routing**
   1. Implement lazy-loaded routes for different views:
      1. /tasks → List of tasks.
      2. /tasks/add → Add task form.
      3. /tasks/:id/edit → Edit task form.
      4. /tasks/:id → View task details.
5. **Unit Tests**
   1. Write unit tests for at least two components and one service using Jasmine/Karma.
6. **Responsive Design**
   1. Use **Angular Material** or a CSS framework (e.g., Bootstrap) to ensure a responsive layout.
7. **Error Handling**
   1. Handle errors for all CRUD operations with meaningful feedback to the user.
8. **Code Quality**
   1. Follow Angular best practices:
      1. Use proper lifecycle hooks.
      2. Ensure separation of concerns.
      3. Use TypeScript features effectively (e.g., interfaces, type checking).

### **Stretch Goals (Optional)**

* **Task History:** Track task changes and show a history log.
* **Pagination:** Implement pagination for the task list.
* **Dark Mode:** Add a theme toggle (light/dark mode).

### **Evaluation Criteria**

1. **Code Quality**:
   1. Clean, modular, and reusable code.
   2. Proper folder structure and adherence to Angular best practices.
2. **Problem-Solving**:
   1. Logical implementation of requirements.
   2. Correct use of state management and services.
3. **UI/UX**:
   1. Responsive and user-friendly interface.
4. **Testing**:
   1. Coverage and quality of unit tests.
5. **Creativity**:
   1. How the candidate goes beyond the requirements to make the app robust and feature-rich.

### **Submission Guidelines**

* The candidate should submit the project as a GitHub repository or a ZIP file.
* Include a README.md with:
  + Instructions for setup and running the project.
  + Details of implemented features.
  + Any assumptions or limitations.