Objective

Build a Task Tracker App using Angular 8+ to demonstrate proficiency in Angular development, including advanced features like pipes, directives, and public API integration, and adherence to good programming practices. The app allows users to manage tasks with complex forms, view archived tasks, and includes a challenge to implement a dynamic prioritization system.

Requirements

Functional Requirements

1. Task Management:
    - Users can add a new task with the following fields:
        - Title: String, required.
        - Description: String, optional.
        - Status: Enum ('To Do', 'In Progress', 'Done').
        - Due Date: Date, optional.
        - Category: Enum ('Work', 'Personal', 'Urgent', 'Other').
        - Tags: Array of strings, optional (selected from API-fetched user names or custom user input, e.g., ['Leanne Graham', 'Ervin Howell']).
    - Users can view a list of all tasks displaying title, status, due date, category, and tags.
    - Users can edit an existing task to update any of its fields.
    - Users can delete a task.
    - Users can archive completed tasks (move to 'Done' status and mark as archived).
    - Users can restore archived tasks to their original status.
2. Routing:
    - Create three routes:
        - /tasks: Displays the task list (non-archived tasks).

        - /task/:id: Displays a single task’s details with an edit form.
        - /archive: Displays archived tasks with an option to restore them.
    - Include a navigation bar to switch between routes.
3. Data Storage:
    - Store tasks in-memory using an Angular service (no backend required).
    - Add an archived boolean field to the task model to track archived status.
4. UI Styling:
    - Use Angular Material for a clean, responsive UI.
    - Ensure the app is mobile-responsive and user-friendly.
5. Public API Integration:
    - Use the JSONPlaceholder API (https://jsonplaceholder.typicode.com/users) to fetch a list of predefined tags.
        - Extract the name field from each user object to use as a tag option in the Tags field (e.g., “Leanne Graham”).
        - Limit to the first 10 users to avoid overwhelming the form.
    - Allow users to select tags from the API-fetched names or enter custom tags.
    - Handle API errors gracefully:
        - Display a fallback message (e.g., “Unable to load tags, please enter custom tags”) if the API request fails.
        - Allow form submission with custom tags even if the API fails.
    - Use Angular’s HttpClient module to make the API request.
    - Cache the API response in the service to avoid repeated requests.

Technical Requirements

- Use Angular 8 or higher.
- Organize code into components (e.g., TaskListComponent, TaskDetailComponent, ArchiveComponent), services (e.g., TaskService), and models (e.g., Task interface).
- Use TypeScript interfaces for type safety.

- Implement reactive forms for task creation and editing.
- Follow Angular style guide for naming conventions and folder structure.
- Write clean, modular, and well-commented code.

Form Validation Guide

Implement the following validations using Angular reactive forms and display user-friendly error messages (e.g., using Angular Material's mat-error):

1. Title:
    - Required.
    - Minimum length: 3 characters.
    - Maximum length: 50 characters.
    - Error messages:
        - Required: “Title is required.”
        - Min length: “Title must be at least 3 characters.”
        - Max length: “Title cannot exceed 50 characters.”
2. Description:
    - Optional.
    - Maximum length: 200 characters.
    - Error message:
        - Max length: “Description cannot exceed 200 characters.”
3. Status:
    - Required.
    - Must be one of: ‘To Do’, ‘In Progress’, ‘Done’.
    - Error message:
        - Required: “Status is required.”
4. Due Date:
    - Optional.

- Must be a valid date (use Angular Material date picker).
- Must not be in the past (relative to today’s date).
- Error messages:
  - Invalid date: “Please select a valid date.”
  - Past date: “Due date cannot be in the past.”

5. Category:
   - Required.
   - Must be one of: ‘Work’, ‘Personal’, ‘Urgent’, ‘Other’.
   - Error message:
     - Required: “Category is required.”
6. Tags:
   - Optional.
   - Maximum 5 tags (from API-fetched names or custom input).
   - Each tag: Minimum 2 characters, maximum 20 characters.
   - Error messages:
     - Max tags: “Cannot add more than 5 tags.”
     - Tag min length: “Each tag must be at least 2 characters.”
     - Tag max length: “Each tag cannot exceed 20 characters.”

- Use custom validators for due date (no past dates) and tags (array length and string constraints).
- Disable the submit button until the form is valid.
- Show validation errors only after the user interacts with a field (e.g., on blur or submit).

Pipes Requirements

1. Custom Due Date Pipe:
   - Create a pipe named dueDateFormat that formats the task’s due date:

        - If the due date is today or in the future, display in MM/DD/YYYY format.
        - If the due date is in the past, display as “Overdue (MM/DD/YYYY)”.
        - If no due date, display “No Due Date”.
    - Apply this pipe in the task list and task detail views.
2. Description Truncation Pipe:
    - Use a built-in pipe (e.g., slice) or create a custom pipe named truncate to limit task descriptions to 50 characters in the task list view, appending “...” if truncated.
    - Ensure the full description is visible in the task detail view.

Directives Requirements

1. Custom Overdue Highlight Directive:
    - Create a directive named appOverdueHighlight that applies a red background or border to task list items with a due date in the past.
    - Apply this directive in the /tasks and /archive routes.
2. Disable Completed Task Fields - OPTIONAL:
    - Use a built-in directive (e.g., ngIf or ngClass) or create a custom directive named appDisableCompleted to disable form fields (except the status field) in the /task/:id edit form when the task status is ‘Done’.
    - Ensure users can still change the status to unarchive or edit if restored.

Challenge: Dynamic Task Priority System - OPTIONAL

- Add a Priority field to each task (enum: ‘Low’, ‘Medium’, ‘High’).
- Allow users to set or update a task’s priority when adding/editing.
- Implement a feature to sort the task list dynamically based on priority (High > Medium > Low) when a “Sort by Priority” button is clicked.
- Ensure the sort persists until the user refreshes the page or adds a new task.
- Bonus: Allow users to toggle between sorting by priority and sorting by creation date.

Submission Guidelines

- Provide a GitHub repository with the complete project.
- Include a README.md with
- Ensure the app runs without errors.
- Submit the repository link by [insert deadline].

Tips

- Start with a minimal setup using ng new task-tracker and add Angular Material via ng add @angular/material.
- Use ng generate to create components, services, pipes, and directives.
- Implement custom validators, pipes, and directives in separate files for reusability.
- Use Angular's HttpClientModule and handle API errors with RxJS operators like catchError.
- Cache API-fetched names in the service using a simple array or BehaviorSubject.
- For the Tags field, consider using Angular Material's autocomplete or multi-select to display API-fetched names.
- Keep the UI simple but polished to focus on functionality, validation, and Angular features.

Good luck, and we look forward to reviewing your work!