**Problem Statement: TaskFlow – A Type-Safe Task Management Dashboard**

**Overview**

Build a **task management dashboard** using TypeScript that allows users to create, update, delete, and filter tasks. The application should support multiple task categories, priority levels, and status transitions. It must be built with a strong emphasis on **type safety**, **modular architecture**, and **API contract validation**.

**Core Requirements**

**1. Task Model**

Define a type-safe Task interface with the following fields:

* id: string
* title: string
* description?: string
* status: "todo" | "in-progress" | "done"
* priority: "low" | "medium" | "high"
* category: string
* createdAt: Date
* updatedAt: Date

Use **TypeScript enums** or **discriminated unions** where appropriate.

**2. Task State Management**

Implement a reducer-based state management system (custom or using Redux Toolkit) with the following actions:

* ADD\_TASK
* UPDATE\_TASK
* DELETE\_TASK
* FILTER\_TASKS\_BY\_CATEGORY
* CHANGE\_TASK\_STATUS

Ensure all actions and state transitions are **type-safe** using discriminated unions and narrowing.

**3. API Client Layer**

Create a mock API client using fetch or Axios with the following endpoints:

* GET /tasks
* POST /tasks
* PUT /tasks/:id
* DELETE /tasks/:id

Use **Zod** or a similar schema validator to enforce contract testing between client and server. All API responses should be validated before use.

**4. Selectors**

Write type-safe selectors to:

* Get all tasks
* Get tasks by category
* Get tasks by status
* Get high-priority tasks

Use narrowing and type guards to ensure correct usage.

**5. Event Handling**

Implement DOM event handlers for:

* Form submission (create/update task)
* Button clicks (delete, change status)
* Dropdown selection (filter by category)

Ensure all event types are correctly annotated (e.g., MouseEvent, SubmitEvent) and DOM elements are narrowed using instanceof or as.