Job Task: Smart Task & Workflow Management System (Mini SaaS)

🎯 Goal

Build a **workflow-driven task management system** where tasks move through dynamic stages, support role-based access, a real–time–ready architecture, and analytics.

Think: **Trello + simple automation + analytics (mini Jira)**.

🧩 Core Requirements

1️⃣ Tech Stack (Mandatory)

- **Frontend:** Next.js (App Router preferred), TypeScript
- **State Management:** Redux Toolkit + RTK Query
- **Backend:** Express.js + TypeScript
- **Database:** MongoDB + Mongoose
- **Auth:** JWT (access + refresh preferred)

🔐2️⃣ Authentication & Roles

- Roles:
  - `Admin`
  - `Manager`
  - `Member`
- Features:
  - Login / Register
  - Protected routes (both frontend & backend)
  - Role-based permissions:
    - Admin: full access
    - Manager: manage tasks & workflows
    - Member: only assigned tasks

🧠3️⃣ Dynamic Workflow Engine (Key Innovation)

Instead of fixed statuses, workflows are **configurable**.

Workflow Example:

Backlog → In Progress → Code Review → QA → Done

- Admin/Manager can:
  - Create workflows
  - Reorder stages
  - Assign workflow to projects
- Tasks must **follow workflow order** (cannot jump stages illegally)

📋4️⃣ Task Management

Each task includes:

- Title
- Description (markdown supported)
- Priority (Low, Medium, High)
- Current Stage
- Assigned Users
- Due Date
- Activity Log (auto-tracked)

Rules:

- Task stage change must:
  - Validate workflow order
  - Be logged in activity history
- Members can only update their assigned tasks

⚙️5️⃣ Automation Rule (Medium+ Complexity)

Create **1 simple automation**:

> "When task reaches `Done`, automatically set completedAt and notify assigned users"

(No real email needed – just log/store notification)

📊6️⃣ Analytics Dashboard

Build a dashboard showing:

- Tasks per stage
- Overdue tasks count
- Avg completion time per workflow
- Tasks completed per user

Backend aggregation using **MongoDB pipelines** is REQUIRED.

🔁7️⃣ RTK Query Usage (Must-Have)

- Proper API slice design
- Tag-based cache invalidation
- Optimistic updates for task stage change
- Error handling & loading states

🧪8️⃣ Code Quality Expectations

- Type-safe APIs (DTOs / interfaces)
- Clean folder structure
- Meaningful commit messages
- Environment-based configs
- README with setup + architecture explanation

⏱️ Time Expectation

3 days

⭐ Bonus (Optional – Strong Signal)

- Drag & drop task stage movement
- Server-side pagination
- Unit tests for workflow validation logic