SDE Assignment - Voice-Enabled Task Tracker

Overview

Build a task tracking application inspired by tools like Linear, with a unique voice input feature that allows users to create tasks by speaking naturally. The application should intelligently parse spoken input to extract task details such as title, description, due date, priority, and status.

This assignment evaluates your ability to build a functional full-stack application, work with modern APIs, and leverage AI tools effectively.

The Problem Statement

Task management tools require users to manually fill in multiple fields when creating a task. This can be tedious and interrupts workflow. Your goal is to build a task tracker where users can simply speak their task, and the system intelligently extracts and populates the relevant fields.

Example Voice Input:

> "Create a high priority task to review the pull request for the authentication module by tomorrow evening"

Expected Parsed Output:

- **Title:** Review the pull request for the authentication module
- **Priority:** High
- **Due Date:** [Tomorrow's date], 6:00 PM
- **Status:** To Do (default)

Functional Requirements

Task Management (Core Features)

1. **Create Task**
    - Manual creation with fields: Title, Description, Status, Priority, Due Date

- Voice-based creation (see Voice Input section below)

2. **View Tasks**
   - Display tasks in a board view (Kanban-style with columns: To Do, In Progress, Done) AND a list view
   - Show task details: title, priority indicator, due date, and status
3. **Update Task**
   - Edit any task field
   - Move tasks between statuses (drag-and-drop in kanban-style view)
4. **Delete Task**
   - Remove tasks with confirmation
5. **Filter and Search**
   - Filter tasks by status, priority, or due date
   - Search tasks by title or description

Voice Input Feature (Key Differentiator)

This is the core feature that differentiates this assignment. Implement it thoughtfully.

1. **Speech-to-Text Capture**
   - Provide a microphone button to start/stop voice recording
   - Convert speech to text using any third-party service
2. **Intelligent Parsing**
   - Extract structured data from natural language input
   - Fields to extract:
     - **Title:** The main task description
     - **Due Date:** Parse relative dates ("tomorrow", "next Monday", "in 3 days") and absolute dates ("15th January", "Jan 20")
     - **Priority:** Identify keywords like "urgent", "high priority", "low priority", "critical"
     - **Status:** Default to "To Do" unless specified otherwise

- Handle variations in phrasing (e.g., "due by Friday" vs "by Friday" vs "before Friday")

3. **Review Before Save**
   - After parsing, show the extracted fields to the user for review and editing before saving
   - Allow users to correct any incorrectly parsed fields
4. **Graceful Handling**
   - If certain fields cannot be extracted, leave them empty or use sensible defaults
   - Show the raw transcript alongside parsed fields so users understand what was captured

---

Technical Requirements

Frontend

- **Framework:** React
- **Styling:** Your choice
- **State Management:** Redux (if required)
- **Voice Input:** Your choice of third-party API

Backend

- **Framework:** Node.js with Express
- **Database:** Your choice
- **API Design:** RESTful APIs

General

- Clean, modular, and readable code
- Proper error handling on both frontend and backend
- Input validation for all API endpoints
- Responsive design (should work on desktop; mobile responsiveness is a bonus)

Sample User Flows

Flow 1: Manual Task Creation

1. User clicks "Add Task" button
2. A form appears with fields for title, description, status, priority, and due date
3. User fills in the fields and clicks "Save"
4. Task appears in the appropriate column/list

Flow 2: Voice Task Creation

1. User clicks the microphone icon
2. User speaks: "Remind me to send the project proposal to the client by next Wednesday, it's high priority"
3. Application stops recording (automatically after silence or manually via button)
4. A preview modal shows:
    - **Transcript:** "Remind me to send the project proposal to the client by next Wednesday, it's high priority"
    - **Parsed Title:** Send the project proposal to the client
    - **Parsed Due Date:** [Next Wednesday's date]
    - **Parsed Priority:** High
    - **Status:** To Do
5. User reviews, optionally edits, and clicks "Create Task"
6. Task is saved and appears on the board

Flow 3: Task Update

1. User clicks on an existing task
2. Task details open in an edit view or modal
3. User modifies fields and saves
4. Task updates in real-time on the board

What We're Looking For

- **Functionality:** All core features work as described.
- **Code Quality:** Code is organized, readable, and follows consistent conventions. Components and functions are modular and reusable.
- **API Design:** RESTful endpoints are logically structured. Proper HTTP methods, status codes, and error responses.
- **User Experience:** The application is intuitive to use. AI features has clear user feedback
- **Problem Solving:** Sensible decisions when faced with ambiguity. Assumptions are reasonable and documented.

Out of Scope (Not Required)

- User authentication and multi-user support
- Real-time collaboration or WebSocket updates
- Mobile application
- Offline support
- Task assignments to team members
- Project or workspace management

Focus on building a solid single-user experience with excellent voice input functionality.

Deliverables

GitHub Repository

- Public repo with clear structure (e.g. `/frontend`, `/backend`).
- `.env.example` listing all required environment variables (no secrets).

README (Mandatory)

Include at least:

1. Project Setup

    a. Prerequisites (Node version, DB, API keys).
    b. Install steps (frontend & backend).
    c. How to configure email sending/receiving.
    d. How to run everything locally.
    e. Any seed data or initial scripts.
2. Tech Stack
    a. Frontend, backend, DB, AI provider, email solution, and key libraries.
3. API Documentation
    a. Main endpoints: method + path, request body/params, example success & error responses.
4. Decisions & Assumptions
    a. Key design decisions (models, flows, scoring, etc.).
    b. Assumptions you made (about emails, formats, limitations, etc.).
5. AI Tools Usage
    a. Which AI tools you used while building (Copilot, ChatGPT, Claude, Cursor, etc.).
    b. What they helped with (boilerplate, debugging, design, parsing ideas, etc.).
    c. Any notable prompts/approaches.
    d. What you learned or changed because of these tools.

Demo Video (Mandatory)

Create a 5-10 minute screen recording that includes:

1. Quick walkthrough of the application
2. Voice input demonstration: Show at least 3 different voice inputs with varying complexity
3. Brief code walkthrough

Upload to: Loom or Google Drive (with public access)

Submission

- GitHub repo link
- Demo video link
- Any additional notes (known limitations, what you’d do next, etc.)