Take-Home Assignment — InsightBoard AI Dashboard

Scenario

You have been hired by **InsightBoard AI**, a fictional productivity SaaS, to build a smart dashboard application. The goal is to help users:

- Submit meeting transcripts
- Receive AI-generated action items
- Track, manage, and visualize progress

This assignment will help us assess your skills in full-stack development, AI integration, and your ability to deliver a clean, functional, and deployable product within a short timeline.

Recommended Tech Stack

- **Frontend:**
  - **Framework:** Next.js (preferred) or React
  - **Language:** TypeScript
  - **UI Library:** Material UI, Chakra UI, or Tailwind + Shadcn
- **Backend:**
  - **Language:** TypeScript (Bun + hono or Express) **or** Python (FastAPI or Flask)
  - **LLM Integration:** OpenAI, Anthropic, Gemini, Claude, or any hosted LLM API
  - **Database (Level 2+):** PostgreSQL (Prisma/Supabase), MongoDB (Mongoose), or similar
- **Charts:** Recharts, Chart.js, or similar
- **Hosting & Deployment:** Vercel, Netlify, Render (Level 1)
  - AWS Lambda / ECS / Cloud Run / GCP (Level 3)

Assignment Structure

The assignment is designed in **three progressive levels**.

- **Level 1 is mandatory for all candidates.**
- Levels 2 and 3 are optional, but completing them will allow you to demonstrate additional skills.

Level 1 — Core Features (Required)

You are expected to build and deploy a fully functional, hosted application that includes:

1. **Transcript Submission**
    - Provide a form with a multi-line text area for users to submit meeting transcripts.
2. **AI-Powered Action Item Generation**
    - The backend should call a Large Language Model (LLM) API (e.g., OpenAI, Anthropic, Gemini) to generate actionable tasks based on the submitted transcript.
3. **Task Interaction**
    - Display action items in a list.
    - Allow users to mark tasks as complete or delete them.
    - Update the UI and chart accordingly when the status changes.
4. **Progress Visualization**
    - Include a pie chart to display the percentage of completed vs pending tasks.
5. **Modern UI**
    - Use a component or UI library (e.g., Material UI, Chakra UI, Shadcn with Tailwind) to deliver a polished, responsive interface.
6. **Hosting**
    - The app must be deployed and accessible via a public URL (e.g., Vercel, Netlify, Render).
7. **Documentation**
    - Include a clear `README.md` with:
        - Tech stack and LLM API choices
        - Setup instructions (local and hosted)
        - The hosted app link
        - **The level you completed**

Level 2 — Enhancements (Bonus)

You may extend the app with:

1. **Filter and Sort Functionality**
    - Enable filtering by status, priority, or keyword.
    - Allow sorting by creation date, priority, or completion status.
2. **AI-Powered Prioritization**
    - Modify your AI prompt so the model assigns a priority (High, Medium, Low) to each action item.
    - Display priority in the UI and charts.
    - (Optional) Allow users to manually edit priority.
3. **Bar Chart Visualization**
    - Add a bar chart that shows the count of tasks by priority level.
4. **Database Integration**
    - Store action items in a real database (PostgreSQL, Supabase, MongoDB, etc.).
    - Example fields: `id`, `text`, `status`, `priority`, `tags`, `createdAt`.

Level 3 — Advanced Features (Bonus)

Further improvements can include:

1. **Cloud Deployment**
    - Deploy the backend on AWS, GCP, or similar.
    - Document the infrastructure setup clearly in your README.
2. **Authentication**
    - Add basic authentication (mocked or real) to protect routes.
3. **Testing**
    - Include minimal unit or integration tests (backend and/or frontend).
4. **AI Auto-Tagging**
    - Extend your AI prompt to assign team tags (e.g., `@Marketing`, `@Tech`) to action items.
    - (Optional) Include sentiment analysis for submitted transcripts.

Time Limit

You must submit your solution **within 24 hours** of receiving the assignment.

Submission Checklist

Your submission must include:

- GitHub repository link
- Live hosted app link
- A `README.md` with:
  - The level you completed (1, 2, or 3)
  - The LLM API used
  - The tech and infra stack
  - Clear setup instructions for local run
  - Link to the hosted deployment

What We're Evaluating

- Code clarity, modularity, and readability
- Thoughtful and practical AI integration
- Quality of UI and user experience
- Hosting and deployment
- Use of modern tools, frameworks, and best practices
- (Bonus) Sensible database and infrastructure decisions

Important Guidelines

- Do **not** commit API keys or secrets. Use environment variables (`.env.local` or equivalent).
- A hosted app is **mandatory** — we must be able to access and test it live.
- Focus on **Level 1 first** — bonus levels are optional, but encouraged.
- The app should work from a fresh clone, with minimal setup as documented.