FortiSys Tech Test: The AI-Native Engineer Challenge

This test is designed to evaluate your fundamental software engineering skills and your ability to leverage AI as a knowledgeable co-pilot. We will not be grading on the speed of your completion, but on the quality of your code, your ability to critically assess AI-generated output, and your process for solving real-world engineering problems.

Problem Statement

Your task is to build a minimal, fullstack application that simulates a real-time worker vitals dashboard. The dashboard's purpose is to help us monitor worker vitals, providing a core foundation for our life-saving safety systems. The solution must be built using the **latest versions** of NestJS and Next.js. We encourage you to use your preferred AI code assistant to complete the task, but a simple "copy and paste" approach will not be sufficient.

The application should handle a high-volume, continuous stream of data, simulating hundreds of vital records from workers in the field.

Core Technical Requirements

1. **Frontend (Next.js)**
    - Use the **latest Next.js framework**.
    - Create a single page that acts as the dashboard.
    - The dashboard should display a form with inputs for heartRate and temperature, and a "Send Data" button.
    - This form must submit the data to your NestJS backend.
    - The dashboard should also display a list of the last 10 vital sign records for a hardcoded workerId. This initial data must be fetched using a method suitable for optimal performance.
2. **Backend (NestJS)**
    - Create an API endpoint that receives the vital sign data submitted from the frontend.
    - This endpoint must perform basic **data validation** (e.g., check for missing or invalid values for heart rate and temperature).
    - The endpoint should save the data to a **PostgreSQL** database.
3. **Database (PostgreSQL)**
    - Set up a single table with an appropriate schema to store the vital sign records. This should include columns for workerId (string), heartRate (integer), temperature (float), and timestamp (ISO string).

The AI-Native Engineer Challenge

A successful candidate will use their foundational knowledge to guide the AI and correct its

output to address real-world engineering problems. You will be evaluated on your ability to:

- Identify and correct code that does not follow the latest framework conventions.
- Ensure the solution is performant and can handle a large volume of data efficiently.
- Build a secure system that protects against common vulnerabilities.

Submission & Evaluation

Your submission should include the following:

1. **Your Complete Code:** A single, complete project that is **deployable with no critical build errors**. Minor linting or TypeScript errors are acceptable, but demonstrating an effort to reduce them is highly recommended.
2. **A Markdown Document (README.md):** This document is the most critical part of the test. It should include the following sections:
    - **Prompts Used:** List all the key prompts you used to get the AI to generate the code.
    - **The Engineering Process:** Describe what issues you identified in the AI-generated code and how you resolved them. Explain your thought process for making these fixes.

This tech test is designed to measure your ability to move beyond simple code generation and act as a strategic partner to your AI assistant. Good luck!