Employee Management System ： Developer Interview Task (2025 Edition)

Overview

Build a simplified Employee Management System that showcases clean architecture, layered design, basic validation, structured logging and ships as a Docker container.

Tech stack & setup

Area	Requirement
IDE	Visual Studio 2022 (17.10+) or Visual Studio Code
Runtime	**.NET 8.0 (**ASP.NET Core, an open-source web development framework | .NET **Core MVC)**
Web server	Kestrel, IIS or other docker supported server
Client UI	ASP.NET MVC views (.cshtml, Razor). Free to use any client framework or native javascript
Container	Multi-stage Dockerfile (Linux), exposes 8080

Core features

Employee CRUD

- List employees in a table with paging/sorting.
- Add, Edit, Delete (with confirmation).

Property	Rule
Id	Auto-increment (int)
FirstName, LastName	Required
Email	Required, valid e-mail format
HireDate	Cannot be in the future
Salary	Must be > 0
DepartmentId	Must reference an existing Department

When adding/editing, department is selected from a dropdown (can be static values from a file, fetch it as you would from a DB async server call).

Department CRUD

- List / Add / Edit / Delete (with confirmation).
- Prevent deletion of a department that still has employees.

Basic dashboard

- Total employee count.
- Employees grouped by department (chart *or* table).
- Recent hires (last 30 days).
- Small search box to filter employees by **name** or **department**.

Validation & error handling

- Apply server-side validation.
- Global exception middleware or filter:
    - Log exceptions to `Logs/log-YYYY-MM-DD.json` or in a “Logs” table in the DB.
    - Return user-friendly error page.

Data & storage

Use JSON file on any SQL/NoSQL DB.

- If you choose JSON:
  - Store files in `App_Data` (inside the container volume).
  - Lock file access to prevent corruption.
  - Handle IO exceptions gracefully.
- If you choose SQL:
  - Create a migration for all DB tables.
  - Make sure to explain how to load your DB (DB user, run commands, etc.)

Deliverable

- Runs locally via **Docker** (preferred).
- Clear **README.md** with:
  - Prerequisites
  - Build/run instructions (for running the Docker, and all other necessary runs/builds like DB, tests etc.)
  - Architecture rationale.
- **Bonus:** Create GitHub Actions workflow compiling & running tests on push.

Submission

- Provide a **GitHub repository** (public or invited reviewer) *or* a ZIP file (keep download size reasonable).
- Ensure the repository history is clean (no IDE cruft or secrets).