Unilink
Excellence Through Innovation

🧩 Technical Exercise — Task Manager API

Overview

Your challenge is to build a small **REST API** in **C#** that allows users to manage simple "tasks".
A task should include some basic information such as a title, description, due date, and whether it's completed.

The goal isn't to build a full production system, but to demonstrate how you approach designing and implementing a small but complete piece of software.

Requirements

- Create a RESTful API that can **create, read, update, and delete** tasks.
- You may use any .NET technologies or packages you prefer (for example, **Entity Framework**, **Dapper**, **Minimal APIs**, or a simple in-memory store).
- The API should be easy to run — a simple command-line start or a Dockerfile is ideal.
- Beyond that, **how you structure it is up to you**. We're interested in seeing:
    - How you design your models and endpoints.
    - How you organise your code and handle data flow.
    - Any patterns, conventions, or tooling you choose to apply.
- Please publish your solution in a publicly accessible repository (for example, GitHub or GitLab) and share the link with us.

What We're Looking For

We'll be looking for:

- **Clarity and cleanliness** of code.
- **Good design decisions**, with appropriate separation of concerns.
- **Readability** — how easily someone else can follow your intent.
- **Initiative** — for example, if you choose to include validation, tests, documentation, or logging, we'll consider how and why you applied them.

This is intentionally open-ended. Do as much or as little as you feel best demonstrates your skills within **a couple of hours**.

Deliverables

- A runnable project (for example, a .NET 8 Web API).
- A brief README.md explaining:
    - How to build and run the solution.
    - Any design decisions or trade-offs you made.
    - Optional: examples of API requests/responses.

Bonus Ideas (Optional)

If you have time and want to go further, you might:

- Add filtering or sorting (e.g. “show only completed tasks”).
- Include simple OpenAPI documentation.
- Use a lightweight database such as SQLite or an in-memory provider.
- Add a small set of unit tests or validation logic.

These are not required — they’re just opportunities to show your personal touch.