# Take-home assignment: PartsTracker

*(C# version)*

### Context

Mercedes-Benz's global factories rely on accurate, near-real-time visibility of the parts that flow through each production line. Your task is to build a thin vertical slice of a *PartsTracker* platform.

### Time estimate

Plan for 4 **hours**. Polished, working core features beat half-finished extras.

## Requirements

### Task 1: Project setup and core API

1. **Create an ASP.NET Core 8 project** (Web API template).
2. Model a **`Part`** entity:
    - `PartNumber` (string, PK)
    - `Description` (string)
    - `QuantityOnHand` (int)
    - `LocationCode` (string)
    - `LastStockTake` (DateTime)
3. Implement full **CRUD** endpoints under `/api/parts`.
4. **Persist to PostgreSQL** via EF Core. Provide a Dockerised Postgres instance in `docker-compose.yml`.
5. **Seed** at least three sample parts.
6. Add a **`/health`** endpoint returning service and DB status.

### Task 2: Validation and tests

1. Add **validation** (e.g., `QuantityOnHand >= 0`, required fields).
2. Write **unit + minimal integration tests** with xUnit/NUnit covering happy & edge cases.

### Task 3: Front-end

1. Build a **single-page app in React *or* Angular** (your choice):
    - list parts, show details, edit/create/delete.
    - display server-side validation errors clearly.
2. Use **Fetch/axios/http-client**; no state-management lib needed.

### Task 4: Cloud and Devops essentials

1. Add a **Dockerfile** that runs the API.
2. Provide minimal **Infrastructure-as-Code**:
    - **Terraform** (preferred) or **Bicep/CloudFormation** file that provisions:
        - an AWS RDS PostgreSQL instance (or Azure PostgreSQL),
        - an ECS Fargate task (or Azure Container Apps) running the container.
    - Keep it *plan-only*, no actual deploy is required.
3. Create a single-job **GitHub Actions** workflow that:
    - restores, builds, tests, and packages the Docker image,
    - (optional) runs `terraform fmt && terraform validate`.

### Task 5: Architecture and trade-offs

In your demo video, spend 5 minutes walking us through:

- your **high-level design** & why it's cloud-friendly,
- key **trade-offs** (e.g., why Postgres vs. Mongo, ECS vs. AKS),
- how you'd scale, secure, monitor, and evolve this slice,
- how you would **mentor a junior** implementing new endpoints.

## Deliverables

**1. Demo video (5 - 10 minutes)**

- Record a video (Loom is recommended) explaining your solution, including your thought process, key design decisions, and trade-offs.

**2. A GitHub repository** with your solution, including:

- `README.md`: Setup instructions, architecture diagram, and rationale
- `SOLUTION.md`: Architecture overview, trade-offs, security/monitoring notes, and cost strategies
- All source code

### Keep in mind

- Prioritise the core API, a single happy-path workflow, and clear documentation before polishing extras.
- Feel free to stub or mock anything that would take you beyond the 4-hour mark, just note your intent in SOLUTION.md.