Energy Management System – Programming Task

Design and implement a **RESTful Energy Management API** that allows standard users to view the sites they have access to and allows technicians to manage devices and their metrics.

1. Time Guidance

This exercise is scoped to ***~2–3 hours*** of focused work. **Do not over-engineer; polish what you have time for and document trade-offs.**
If you have a public repository that showcases your coding style—particularly in Python—feel free to share the GitHub link along with your completed task.

Good luck – we look forward to reading your code! 🚀

2. Problem Statement

You are building a minimal back-end for an energy management system. The **core domain object** is a **Site**. Each Site has several **Devices** (batteries, inverters, PV panels, wind turbines …), and every Device reports at least one **metric** (e.g. battery state-of-charge, PV power, wind speed).

Your goal is to expose a set of HTTP endpoints that satisfy the functional requirements below and to provide automated tests that prove the behaviour.

3. Functional Requirements

#	Requirement
R1	*Site catalogue* – Standard users can list and inspect the Sites they are authorised to see.
R2	*Device CRUD* – Technicians can **create** / **update** / **delete** devices under any Site they are assigned to; standard users have **read-only** access.
R3	*Metric feed* – Every Device exposes at least one metric; the API must return the **latest value** together with metadata (timestamp,

	unit).
R4	*Metric subscription* – A user can create a **subscription** that references an arbitrary set of metrics (across multiple sites and devices).
R5	*Time-series endpoint* – The system returns a **mocked list of numbers** that represents the subscribed metric history for the requested time window.
R6	*Persistence* – All data are stored in a relational database (SQLite, PostgreSQL or similar).
R7	*Tests* – Provide unit/integration tests that cover the behaviour you consider critical.

4. Optional Enhancements (nice-to-have)

1. **Authentication & Authorisation** (e.g. JWT, session cookies).
2. **Server-side rendered HTML** pages for manually exploring Sites/Devices.
3. **Dockerisation** – 1-click setup via `docker compose up`.

Feel free to implement any (or none) of these if you have spare time; they will be considered a bonus.

5. Non-Functional Expectations

- ***Clean code & project structure*** – favour readability over micro-optimisations.
- ***Documentation*** – briefly explain design decisions and how to run the project.
- ***Automated tests*** – green on `pytest` / `npm test` / etc.
- ***API description*** – OpenAPI/Swagger or equivalent is appreciated.

6. Deliverables

1. ***Source code*** and this `README.md` (update if you deviate from the spec).
2. ***Test suite*** demonstrating the behaviour.
3. *(Optional)* `docker-compose.yml`, migration scripts, seed data.

Submit a link to a public Git repository (GitHub / GitLab / Bitbucket) or a zipped archive.