1) Start from the goal, not the code

Goal: Model Human factors (training, shift length, fatigue, staffing) so they change how fast alarms are handled and how much downtime accrues.

That implies two effects:

- A site-level effect that scales reaction/repair times.
- A per-equipment effect that runs acknowledge/repair timers whenever an alarm occurs.

2) Split responsibilities (ECS-style)

ECS rule: state lives in components; behavior in systems.

- Site/state (applies to everyone): “HumanFactors” -> stores training, fatigues, shift, staff, and outputs a computed multiplier “reaction_time_mult”.
- Pre-asset/state (only on alarmable things): “AlarmResponse” -> stores whether a response is active, ack/repair timers, and the target delays.
- Org merics/state (executive KPIs): “SiteKPI -> counts alarms, tracks downtime seconds.

Why three components?

Different lifetimes & scope:

- “HumanFactors: singleton (one per site)
- AlarmResponse: Per entity that can alarm.
- SiteKPI: singleton collector for dashboard.

3) Decide data flow (dependencies -> tick order)

List what each stage needs and produces:

- AlarmSystem: needs plant physics already computed; sets “Alarmable.hi/lo/latched.
- HumanFactorsSystem: reads “HumanFactors” inputs; produces “reaction_time_mult”.
- ResponseSystem: reads “Alarmable” + “reaction_time_mult”; advances “AlarmResponse” (ack/repair timers), may clear alarm; updates “SiteKPI” (downtime, counts).

The Order

-> Hydraulics/Thermal -> AlarmSystem -> HumanFactorSystems -> ResponseSystem -> Analytics/Render

Rationale:

- You can’t evaluate alarms until physics ran.
- You can’t scale response times until human factors multipler is known.
- You can’t udate KPIs until alarms/response were processed.

4) Choose where to instantiate

You need at least:

- One equipment entity that already has “Alarmable” -> add “AlarmResponse” to it.
- One site entity -> add “HumanFactors” and “SiteKPI” to it.

Why site entity not globals?

ECS works best when everything is data in the registry – easy to save / serialize, easy to replace later with a JSON loader or multiple files.

5) Pick stable defaults (determinism)

Simulations should run even with sparse configs. So:

- Give safe defaults in components (e.g., training 0.7, shift 8.0h).
- Pick demo-friendly response times (shorter ack/repair) so you can see it work immediately.
- Clamp multipliers to sane bounds (e.g., 0.5 … 2.0) to avoid runaway timers.

6) Keep coupling low (testability & reuse)

- HumanFactorsSystem: only computs a multiplier. It doesn't poke timers.
- ResponseSystem: only consumes the multipler + alarms and updates "AlarmResponse" / "SiteKPI". This separation make each system easy to test and swap (e.g., try a different fatigue formula).

7) UI/KPI tap points (observability)

Executives need numbers, so we expose read-only values in the UI:

- Tank level (you already had), pumps running (quick health check).
- SiteKPI: downtime minutes, active alarms.
- HumanFactos: reaction-time multipler (so sliders later will "feel" immediate). These are probes – they don't change sim state, they just show it.

8) Patter you can resuse for any new feature

Every new feature follows the same recipe:

1) Responsibility: what does it change each tick?
2) Inputs/outputs: which components does it read/write?
3) Components: add state you must persist.
4) System: put behaviour after its inputs and before its consumers.
5) Instantiate: attach components in scenario/loader.
6) Probe: add KPIs/plots so you can tune it.

9) Why those specific names/fields?

- HumanFactos: fields match the executive levers they'll want on sliders: training, fatigue, shift length, staff. The multiplier is a clean interface for others.
- AlarmResponse mirrors how plants work: acknowledge -> repair -> clear. Keeping both target times and live timers lets you model scaling and partial progress.
- SiteKPI: isolates rollups (alarms raised/active, downtime) so analytics stays simple and serializable.

10) Edge cases accounted for

- Trainsient alarms: if condition clears quickly before ack, we allow auto-clear (prevents sticky responses on blips).
- No site factors present: default multiplier 1.0 (system still works).
- Multiple alarmable assets: view iterates all entities; KPIs aggregate naturally.

That's the reasoning chain.