GearGuard: The Ultimate Maintenance Tracker

1. Module Overview

Objective: Develop a maintenance management system that allows a company to track its assets (machines, vehicles, computers) and manage maintenance requests for those assets.
Core Philosophy: The module must seamlessly connect **Equipment** (what is broken), **Teams** (who fix it), and **Requests** (the work to be done).

---

2. Key Functional Areas

A. Equipment

The system serves as a central database for all company assets. Participants must create a robust "Equipment" record that tracks ownership and technical details.

- **Equipment Tracking(use the search or group by for tracking the request):**
  - **By Department:** (e.g., A CNC Machine belongs to the "Production" department).
  - **By Employee:** (e.g., A Laptop belongs to "Person name").
- **Responsibility:** Each equipment must have a dedicated **Maintenance Team** and a technician is assigned to it by default.
- **Key fields:**
  - Equipment Name & Serial Number.
  - Purchase Date & Warranty Information.
  - **Location:** Where is this machine physically located?

B. Maintenance Team

The system must support multiple specialized teams.

- **Team Name:** Ability to define teams (e.g., *Mechanics, Electricians, IT Support*).
- **Team Member Name:** Link specific users (Technicians) to these teams.
- **Workflow Logic:** When a request is created for a specific team, only team members should pick it up.

C. Maintenance Request

This is the transactional part of the module. It handles the lifecycle of a repair job.

- **Request Types:**
    - **Corrective:** Unplanned repair (Breakdown).
    - **Preventive:** Planned maintenance (Routine Checkup).
- **Key fields:**
    - **Subject:** What is wrong? (e.g., "Leaking Oil").
    - **Equipment:** Which machine is affected?
    - **Scheduled Date:** When should the work happen?
    - **Duration:** How long did the repair take?

---

3. The Functional Workflow

Participants must implement the following business logic to make the module "alive."

Flow 1: The Breakdown

1. **Request:** Any user can create a request.
2. **Auto-Fill Logic:** When the user selects an **Equipment** (e.g., "Printer 01"):
    - The system should automatically fetch the **Equipment category** and **Maintenance Team** from the equipment record and fill them into the request.
3. **Request state:** The request starts in the **New** stage.
4. **Assignment:** A manager or technician assigns themselves to the ticket.
5. **Execution:** The stage moves to **In Progress**.
6. **Completion:** The technician records the **Hours Spent** (Duration) and moves the stage to **Repaired**.

Flow 2: The Routine Checkup

1. **Scheduling:** A manager creates a request with the type **Preventive**.
2. **Date Setting:** The user sets a **Scheduled Date** (e.g., Next Monday).
3. **Visibility:** This request must appear on the **Calendar View** on the specific date so the technician knows they have a job to do.

---

4. User Interface & Views Requirements

To provide a good User Experience (UX), the following views are required:

1. The Maintenance Kanban Board

The primary workspace for technicians.

- **Group By:** Stages (New | In Progress | Repaired | Scrap).
- **Drag & Drop:** Users must be able to drag a card from "New" to "In Progress."
- **Visual Indicators:**
    - **Technician:** Show the avatar of the assigned user.
    - **Status Color:** Display a red strip or text if the request is **Overdue**.

2. The Calendar View

- Display all **Preventive** maintenance requests.
- Allow users to click a date to schedule a new maintenance request.

3. The Pivot/Graph Report (Optional/Advanced)

- A report showing the **Number of Requests** per **Team** or per **Equipment Category**.

---

5. Required Automation & Smart Features

These features distinguish a basic form from a smart "Odoo-like" module.

- **Smart Buttons:**
    - On the **Equipment Form**, add a button labeled **"Maintenance"**.
    - **Function:** Clicking this button opens a list of all requests related *only* to that specific machine.
    - **Badge:** The button should display the count of open requests.
- **Scrap Logic:**
    - If a request is moved to the **Scrap** stage, the system should logically indicate that the equipment is no longer usable (e.g., log a note or set a flag).

Mockup link - https://link.excalidraw.com/l/65VNwvy7c4X/5y5Qt87q1Qp