Blade Management Tool – Full Stack Case Study

**Objective**

Build a functional blade management platform that helps wind farm operators track and maintain blade health across multiple sites and turbines.

**Input Dataset**

You will be provided with 4 Excel/CSV files containing:

1. **Sites**: Site ID, Name, and Location (includes duplicate and inconsistent values).
2. **Turbines**: Turbine ID, Site ID, Model (includes inconsistencies and duplicated IDs).
3. **Blades**: Blade ID, Turbine ID, Type, Length (5,000 records with missing or inconsistent data).
4. **Maintenance History**: Maintenance logs from the last 3 years, with different statuses, issues, and assigned technicians.

**Backend Tasks**

**1. Database Design & Normalization**

* Design a **relational database schema** in PostgreSQL or MySQL.
* Use appropriate keys and constraints to enforce data integrity.
* Normalize the data to eliminate redundancy (3NF preferred).

**2. Data Ingestion & Cleaning**

* Write a script (Python/Node.js) to ingest the CSV data into your database.
* Perform the following data cleaning tasks:
  + Trim whitespace from text fields (e.g., "East Side " → "East Side").
  + Convert inconsistent casing (e.g., s002 → S002).
  + Handle and log missing or duplicate entries gracefully.

**3. RESTful API Development**

Implement the following endpoints using Flask, FastAPI, Django REST Framework, or Node.js (Express):

**🔍 Read Endpoints**

|  |  |  |
| --- | --- | --- |
| **Endpoint** | **Method** | **Description** |
| /sites | GET | List all sites |
| /turbines | GET | List all turbines |
| /blades | GET | List all blades |
| /maintenance | GET | List all maintenance records |

**🔗 Relationship Endpoints**

|  |  |  |
| --- | --- | --- |
| **Endpoint** | **Method** | **Description** |
| /sites/{site\_id}/turbines | GET | List turbines for a specific site |
| /turbines/{turbine\_id}/blades | GET | List blades for a specific turbine |
| /blades/{blade\_id}/maintenance | GET | Get maintenance records for a blade |

**✏️ Write/Update Endpoints**

|  |  |  |
| --- | --- | --- |
| **Endpoint** | **Method** | **Description** |
| /blades | POST | Add a new blade |
| /blades/{blade\_id} | PUT | Update a blade’s type or length |
| /maintenance | POST | Log a new maintenance event |
| /maintenance/{maintenance\_id} | PUT | Update maintenance status, issue, or technician |

**Validation Logic**

* Ensure turbine exists before adding a blade.
* Ensure blade exists before adding maintenance.
* Handle foreign key constraints gracefully.

**Frontend Tasks (React)**

**1. Dashboard & Navigation**

* Create a clean, professional user interface.
* Use a navigation sidebar or tabbed layout.

**2. View Screens**

|  |  |
| --- | --- |
| **Feature** | **Description** |
| Sites View | List of all sites. Click a site to view its turbines. |
| Turbines View | Show all turbines in a selected site. |
| Blades View | Show all blades per turbine with their type and length. |
| Maintenance View | List of all maintenance records for a blade. |

**3. Edit Forms**

* Editable forms to:
  + Change blade type and length.
  + Update maintenance status, issue found, and technician.
* Forms should be validated on the frontend before submitting.

**4. Search & Filters**

* Filter blades by:
  + Turbine ID
  + Site ID
  + Repair status (Pending, Completed, etc.)
* Optional: global search bar.

**Bonus (Optional)**

**1. Maintenance Dashboard**

Create a simple dashboard with:

* Total blades maintained this year
* Number of maintenance cases by issue type (bar chart)
* Pie chart of repair status distribution

**2. Recurring Issue Flagging**

Highlight or tag blades that have:

* 3+ maintenance records within the past year
* Repeated issues of the same type

**Expected Deliverables**

* ✅ Codebase (GitHub repo or zip file)
* ✅ README with:
  + Setup instructions
  + Tech stack used
  + Assumptions made
* ✅ Postman Collection (for API testing)