Database Design Document

Project Title: AI-Powered Predictive Maintenance for Industrial Equipment
Track: AI & Data Science
Database: MongDB
Backend: NodeJS
Data Source: **Binary Classification of Machine Failures**
Date: October 2025

1. Introduction

The purpose of this database design is to establish a structured and scalable data management foundation for the AI Powered Predictive Maintenance for Industrial Equipment project. The database serves as the central repository for all operational and analytical data used in predicting equipment failures through machine learning models.

In this project, real or publicly available industrial datasets such as those provided by Kaggle · Playground Prediction Competition are used to capture the behavioral characteristics of machines under varying operational conditions. Each data entry represents a snapshot of an equipment's state, including key parameters such as air temperature, process temperature, rotational speed, torque, and tool wear. These readings, combined with binary failure indicators and detailed failure modes (e.g., tool-wear, power,etc.), allow for the development of predictive models capable of forecasting equipment degradation and potential failure events.

To efficiently manage this information, the database is implemented using MongoDB. MongoDB's structure enables rapid insertion of streaming IoT data, dynamic schema evolution for new sensor attributes, and seamless integration with the Node.js backend and analytical workflows. This design ensures that both real time and historical data are available for model training, validation, and deployment in a predictive maintenance setting.

2. Database Requirements

The predictive maintenance system urge specific requirements on the underlying database due to the nature of industrial data. The database must:

- **Support time-series data ingestion** from multiple machines, with the ability to manage thousands of readings per minute during real-time monitoring.
- **Accommodate diverse data attributes** including environmental (air temperature) and process-specific (rotational speed, torque) parameters, while maintaining flexibility for additional features in future datasets.
- **Integrate seamlessly with the AI pipeline** allowing data scientists to query, filter, and extract structured data for model training and evaluation.
- **Ensure data consistency and traceability** particularly between operational records and their corresponding prediction outputs.
- **Facilitate efficient data retrieval** for visualization dashboards and failure trend analysis without compromising system performance.
- **Provide scalability and reliability** enabling the system to expand to additional equipment, datasets, or industrial environments as needed.

These requirements justify the use of MongoDB as the primary database engine, as it offers a document-based architecture well-suited to semi-structured industrial datasets, and can scale horizontally to meet increasing data demands.

3. Conceptual Design

The conceptual design defines the primary data entities and their relationships within the predictive maintenance ecosystem. Given the structure of the dataset, the database centers around three core collections **Equipment**, **OperationRecord**, and **PredictionLog** with **FailureRecord** collection for historical tracking.

1. **Equipment** represents each industrial machine or production unit. It contains identifying details such as product ID, machine type, and operational status.
2. **OperationRecord** serves as the central repository for all recorded observations, capturing each machine's operational parameters at specific time intervals. Each document stores values for air temperature, process temperature, rotational speed, torque, tool wear, and multiple failure indicators. This collection replaces the need for separate sensor tables by consolidating all sensor related variables into a single unified schema.
3. **PredictionLog** records the outputs of the AI model, including predicted probability of failure, prediction timestamps, and model versioning information. These entries enable continuous monitoring of model accuracy and system reliability over time.
4. **FailureRecord** is used to log verified machine failure events and their corresponding types, supporting root cause analysis and model validation.

The relationships between collections are organized as follows:

- One **Equipment** document can have many **OperationRecord** documents (1 to many).
- One **Equipment** document can have many **PredictionLog** entries (1 to many).
- One **Equipment** document can optionally reference several **FailureRecord** entries (1 to many).

This structure balances simplicity and scalability, ensuring that real-time operational data and predictive analytics remain tightly connected while maintaining a clear separation of responsibilities between entities.

4. Logical Design

The logical design defines MongoDB collections, their fields, data types, and relationships.

Equipment

● Basic information about machines.

Field Name	Data Type	Description
_id	**ObjectId**	**Unique identifier for each equipment**
product_id	**String**	**ID of the specific production unit or equipment**
type	**String**	**Type/category of product or machine**
status	**String**	**Current operational status**

OperationRecord

● Each record = one observation from the dataset

Field Name	Data Type	Description
_id	**ObjectId**	**Unique record ID**
product_id	**ObjectId (Ref → Equipment)**	**Links to the machine**
air_temperature	**Double**	**Air temperature [K]**

process_temperature	Double	Process temperature [K]
rotational_speed	Double	Speed [rpm]
torque	Double	Torque [Nm]
tool_wear	Double	Tool wear [min]
machine_failure	Boolean	1 = failure occurred
failure_types	Object	{TWF: 0, HDF: 1, PWF: 0, OSF: 0, RNF: 0}

FailureRecord

● Explicitly tracking actual failures

Field Name	Data Type	Description
_id	ObjectId	Unique failure ID
failure_time	DateTime	When failure occurred
failure_type	String	“TWF ” , ”HDF” , Etc..

PredictionLog

● Model predictions will be stored here

Field Name	Data Type	Description
_id	ObjectId	Unique prediction ID
predicted_failure_probability	Double	Model output
predicted_label	String	“Likely to fail” / “Normal”
model_version	String	Version of ML model
actual_outcome	String	Observed outcome for evaluation

5. Example MongoDB Documents

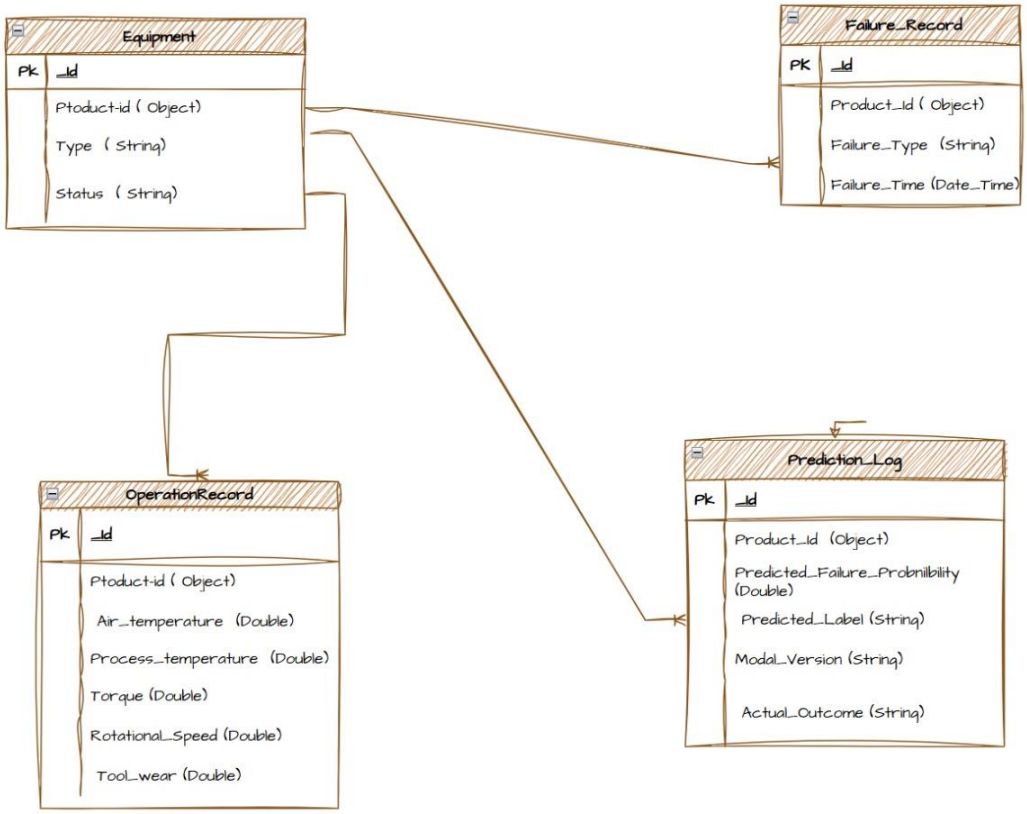
Equipment

{'_id': "ObjectId('64a5f...')", 'product_id': ' ObjectId('M45004')', 'type': 'Centrifugal Pump', 'status': 'Active'}

PredictionLog

{'_id': "ObjectId('64a5a...')", 'product_id': "ObjectId('M45004.')", 'predicted_failure_probability': 0.82, 'predicted_label': 'Will Fail Soon', 'model_version': 'v1.3.0', 'actual_outcome': 'Pending'}

6. ERD (Entity Relation Diagram)



7. Conclusion

The database design outlined in this document provides a robust and adaptable foundation for predictive maintenance using artificial intelligence. By organizing operational data into coherent collections and linking it directly to equipment identifiers and prediction outputs, the design enables an integrated view of each machine's lifecycle from normal operation to failure prediction and eventual maintenance.

The use of MongoDB offers significant advantages in terms of scalability, flexibility, and compatibility with modern data pipelines. It allows the system to evolve alongside the growing complexity of industrial datasets, supporting both real time monitoring and long-term analytical tasks. This design not only facilitates efficient data handling and model integration but also enhances the overall reliability of predictive insights.

Ultimately, this database design contributes to reducing unplanned downtime, optimizing maintenance schedules, and extending equipment lifespan, therefore demonstrating the business value of the AI driven predictive maintenance systems.