Classes and attributes for the Maintenance Inspection Forms Application

This document presents the foundational class structure identified during the domain modelling phase of the project. Its purpose is to provide a clear, technology-independent representation of the core entities (“conceptual classes”) involved in the digital maintenance inspection process, along with the essential information each must contain.

The conceptual classes were identified through:

* Analysis of existing paper-based inspection forms
* Review of operational workflows
* Stakeholder discussions with engineers and management

This analysis resulted in a generalised structure that supports a wide range of inspection types, from facilities checks such as *emergency lighting* or *fire door inspections* to machine safety checks such as *emergency stop tests*. The structure emphasises clarity, reusability, and scalability across inspection contexts.

At this stage, only essential attributes for each class are recorded. These attributes will later be transformed into data fields within an Entity–Relationship Diagram (ERD), which will then form the structural basis for the database. Methods, behaviours, and technology-specific details are intentionally excluded to avoid premature design decisions. This approach aligns with TM354 guidance, which recommends separating domain logic from implementation logic to ensure a more adaptable and maintainable system design.

The resulting classes and attributes form a critical bridge between the problem domain and the subsequent phases of requirements engineering, structural modelling (via UML), and implementation. They will also guide validation activities to ensure alignment with stakeholder needs before database or user interface development begins.

# Conceptual Classes and Attributes (No Methods)

## User

Represents an individual interacting with the system, either as an engineer performing

inspections or a manager reviewing outcomes. Each user is automatically identified

on login, enabling audit trails and role-based access control throughout the application.

**Attributes:**

* userID
* username
* fullName
* role
* email

## Inspection

Serves as the base class for all inspections. It captures generic details such as inspection date, inspector identity, and the category of inspection. All specific inspections (e.g., facility or machine-based) inherit from this. Having both categories: facility and machine safety under one common inspection class allows more generic and dynamic way of storing information.

**Attributes:**

* inspectionID
* inspectionDate
* inspectionCategory (Facilities/MachineSafety)
* inspectedBy -> User
* zone -> Zone
* site -> Site
* itemType (e.g Emergency Lighting, Die-Cut)
* itemName (e.g. E-lights (Emergency Lighting name), Bobst (Die-Cut machine name))

## Subcheck

This class represents the subcomponents of a single inspection, like "Test E-stop," "Inspect guards," or "Check light fitting”. So, in other words, it represents what is being checked across any inspection.

**Attributes:**

* itemID: unique ID for the individual inspection item (sub-check)
* itemName: short label (e.g. “E-Stop”, “Door”)
* itemDescription: optional full description
* valueType: defines expected input type (e.g., "boolean", "number", "status")
* passCriteria: String – logic to evaluate pass/fail (e.g. “true”, “>=50”, “GREEN”)
* statusType: String (optional) – for status-based categories e.g. (“RED”, “AMBER”, “GREEN”)

The statusType will tell the system what kind of value is expected for this specific sub-check.

* Example values:
  + "Boolean" → expects Pass/Fail or True/False
  + "Numeric" → expects numbers (e.g., meter readings)
  + "Status" → expects categorical values like RED/AMBER/GREEN

## InspectionResult

Stores the actual outcome for a given item during an inspection. Includes values like Pass/Fail, timestamp, optional comments, and attached evidence (e.g., photos).

**Attributes:**

* resultID: unique result ID
* inspectionID -> Inspection (Foreign Key to Inspection)
* itemID -> InspectionSubCheck (Foreign Key to InspectionSubCheck)
* status: Boolean (e.g., actual outcome: Pass/ Fail)
* comment: optional inspector comment
* photoAttachment: String (file path or URL) - optional photo link
* checkTime: timestamp of check (optional for audit readiness, multiple checks tracking, and analytics)

The status attribute records the actual observed value during an inspection.

* For a numeric sub-check → it holds a number (e.g., 12.3)
* For a RED/AMBER/GREEN sub-check → it holds a string
* For a boolean → it could hold "Pass" or "Fail" — or just true/false

## Site

Represents the buildings or sites being inspected (e.g. Unit 6, Unit 22). Useful for organising inspections by physical location or reporting at the site level.

**Attributes:**

* siteID: unique ID
* buildingNumber (e.g. “U20”, “U22”, “U6”)
* siteName (Packaging, Conductive, Protective)

## Zone

Sub-location within a unit. This helps narrow down item locations for inspections (e.g., Rear, Front, Right Side, etc.).

**Attributes:**

* zoneID: unique zone ID
* zoneName: Zone label
* zoneDescription: additional zone description
* siteID: Foreign Key to Site

## InspectionParameter

Describes how an item should be inspected—such as the measurement method, value type, and the expected result. It defines rules used to validate readings.

**Attributes:**

* parameterID: unique ID
* itemType: What type of item it applies to
* description: (e.g., “Check for visible damage”)
* inspectionMethod: Manual/Visual/Reading
* valueType: Boolean, Number, Text

## Reading

Captures quantitative or categorical data during inspections (e.g., meter readings, temperature checks). Readings are tied to items and validated using parameters.

**Attributes:**

* readingID: Unique ID
* itemID: Foreign Key to item
* inspectionID: FK to inspection
* parameterID: FK to InspectionParameter
* value: Recorded data
* unit: unit of measurement (e.g. "°C", "kWh")

## Attachment

Supports additional files such as photos for visual evidence for any item check or document files as a means of additional information. Attached photos could be used for audit or fault investigation.

**Attributes:**

* attachedID: Unique ID
* fileData:
* filePath: Storage reference
* caption: Optional note
* resultID: FK to related InspectionResult