Classes and attributes for the Maintenance Inspection System Application

# This document presents the foundational class structure developed during the domain modelling phase of the project. It supports the early analysis of the system by identifying the main entities (or "conceptual classes") involved in the digital maintenance inspection process and defining their attributes. These classes serve as a precursor to the domain structural model and provide a clear, non-technology-specified view of the information that the system must manage.

# The conceptual classes and attributes have been derived from a detailed review of existing paper-based inspection forms, operational workflows, and discussions with stakeholders. The goal was to create a generalised structure that could support multiple inspection types—ranging from emergency lighting and fire doors to pest control and machine safety checks—while maintaining clarity and reusability.

# Each class includes only essential attributes (data fields), avoiding premature design decisions such as methods or interfaces. This approach is consistent with TM354 guidance, which recommends separating domain logic from implementation logic to ensure a more flexible and robust system design. The classes and their attributes also form the basis for the domain structural model, where relationships and associations between classes are formally defined using UML notation.

# This appendix provides a critical bridge between the problem domain and subsequent stages of system design, requirements engineering, and implementation.

**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 (system login)

fullName (engineer name)

role (Engineer, Manager, Admin)

email

**Inspection (superclass)**

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.

**Attributes:**

inspectionID

inspectionDate

inspectionCategory (Facilities/MachineSafety)

inspectedBy -> User

site -> Site (association rather than a string)

zone -> Zone (association)

**FacilitiesInspection (inherits from Inspection)**

This represents an individual routine facilities inspection – like checking emergency lights, fire doors, ladders, pest control, or water meters.  
It stores general information about the inspection, such as:

* Where it happened (e.g. unit, zone)
* What type of facility check it was (e.g. lighting, fire safety)
* It helps the system organise and display the right checklist for that type of check.

**Attributes:**

facilityCheckID: Unique ID for the facilities check

locationDetails: Zone or physical area description

checkType: (Lighting, Fire Door, Meter Reading, etc)

**MachineSafetyInspection (inherits from Inspection)**

Used when inspecting the safety features of specific machines. It helps define which machine is being checked and what type it is, so the correct safety checklist can be dynamically loaded.

**Attributes:**

machineSafetyCheckID: unique machine check identifier

machineType (e.g. Print/Die-Cut, Bespoke Finishing, Finishing)

machineName (e.g. Rotary, LMC, Bobst, Coater, Supernova, etc.)

**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:**

subCheckID: unique ID for the individual inspection item (sub-check)

subCheckName: short label (e.g. “E-Stop”, “Door”)

description: 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

fileAttachment: 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 visual evidence for any item check. Photos are linked to a result and may be used for audit or fault investigation.

**Attributes:**

attachmentID: Unique ID

attachmentData: Binary or base64 image content

filePath: Storage reference

caption: Optional note

resultID: FK to related InspectionResult

Initial class diagram:  
A diagram of a computer program

AI-generated content may be incorrect.

Following this class and attributes document, further analysis was carried out and proposed multiplicities were added to the conceptual class diagram. Below are the proposed multiplicities for associations between the classes and a short explanation behind the decision.

**Proposed Multiplicities:**

**User – Inspection**

* One user (engineer or manager) can create many inspections, but each inspection is created by one user.  
  User (1) ----- (\*) Inspection

**Inspection – Subcheck**

* Each inspection consists of many subchecks, but each subcheck is linked to exactly one inspection.  
  Inspection (1) ----- (\*) Subcheck

**Subcheck – InspectionResult**

* Each subcheck produces exactly one inspection result per inspection, but a result is tied to only one subcheck.  
  Subcheck (1) ----- (1) InspectionResult  
  (if a subcheck can be repeated in different time intervals in a single inspection the multiplicity could also be 0..\*)

**InspectionResult – Attachment**

* One inspection result can have zero or more attachments, but each attachment belongs to exactly one result.  
  InspectionResult (1) ----- (0..\*) Attachment

**Inspection – Site**

* Each inspection takes place at exactly one site, but each site can have many inspections over time**.**  
  Site (1) ---- (\*) Inspection

**Site – Zone**

* Each site consists of one or more zones, but each zone belongs to exactly one site.  
  Site (1) ---- (1..\*) Zone

**Inspection – Zone**

* Each inspection occurs in exactly one zone, but each zone can have many inspections.  
  Zone (1) -- (\*) Inspection

**Inspection – InspectionResult**

* Each inspection can have only one result, but one InspectionResult can be assigned to many inspections.

Inspection (1) ---- (\*) InspectionResult