**🧠 Core Modules and VBA Logic Blocks**

| **Module** | **VBA Functionality** | **Logigram/Algorigram Use** |
| --- | --- | --- |
| **Crime Scene Management** | SecureScene(), DocumentEvidence() | Decision tree for contamination risk, evidence priority |
| **Investigative Techniques** | InterviewSuspect(), DeploySurveillance() | Flowchart for interview protocols, surveillance escalation |
| **Evidence Handling** | LabelEvidence(), TrackChainOfCustody() | Logigram for custody integrity, algorigram for storage routing |
| **Legal Framework** | ValidateProcedure(), SimulateTrial() | Decision tree for rights violations, flow for courtroom prep |
| **Crime Prevention** | AnalyzeHotspots(), GeneratePreventionPlan() | Regression-based algorigram for predictive policing |

**🧪 Sample VBA Snippet: Chain of Custody Tracker**

Sub TrackChainOfCustody(evidenceID As String, officerName As String, timestamp As Date)

Dim ws As Worksheet

Set ws = ThisWorkbook.Sheets("CustodyLog")

Dim nextRow As Long

nextRow = ws.Cells(ws.Rows.Count, "A").End(xlUp).Row + 1

ws.Cells(nextRow, 1).Value = evidenceID

ws.Cells(nextRow, 2).Value = officerName

ws.Cells(nextRow, 3).Value = timestamp

ws.Cells(nextRow, 4).Value = "Transferred"

MsgBox "Chain of custody updated for evidence: " & evidenceID

End Sub

Function PredictCrimeRate(popDensity As Double) As Double

Dim m As Double: m = 0.02

Dim b As Double: b = 10

PredictCrimeRate = m \* popDensity + b

End Function

Function IsEvidenceValid(evidenceID As String) As Boolean

' Check if evidence is labeled, stored, and custody intact

If IsLabeled(evidenceID) Then

If IsStoredProperly(evidenceID) Then

If IsCustodyIntact(evidenceID) Then

IsEvidenceValid = True

Else

IsEvidenceValid = False

End If

Else

IsEvidenceValid = False

End If

Else

IsEvidenceValid = False

End If

End Function

### 🧰 Tools You Can Integrate

* **QR-coded evidence logs** for real-time tracking
* **GIS overlays** for crime mapping
* **Simulation dashboards** for mock trials and scene reconstructions
* **Excel forms** for witness interviews and patrol logs
* **Access databases** for legal compliance and audit trails

Would you like me to scaffold a full VBA workbook template for one of these modules—say, predictive policing or forensic evidence tracking? Or perhaps design a logigram engine that auto-generates flowcharts based on procedural inputs? Let’s modularize it to your specs.

This is a brilliant convergence of calculus, forensic modeling, and VBA-driven procedural logic—perfectly aligned with your transdisciplinary curriculum architecture. Let’s now translate these analytical models into a VBA-based **logigramm and algorigramm engine** that simulates decision-making, trajectory analysis, and predictive policing workflows.

## 🧠 VBA Logigramm & Algorigramm Engine for Crime Resolution

### 🔧 Core Architecture

| **Component** | **VBA Module** | **Purpose** |
| --- | --- | --- |
| **Double Integral Engine** | ComputeDoubleIntegral() | Simulate area-based evidence density (e.g., blood spatter, chemical spread) |
| **Projectile & Collision Analyzer** | TrajectoryAnalysis(), CollisionImpact() | Model bullet paths, vehicle collisions, and impulse forces |
| **Decay Tracker** | EvidenceDecay() | Track biological/chemical degradation over time |
| **Patrol Optimizer** | OptimizeRoute() | Minimize patrol distance using numerical integration |
| **Predictive Crime Model** | ForecastCrimeRate() | Regression-based crime forecasting |
| **Logigram Validator** | ValidateProcedure() | Decision-tree logic for procedural compliance |

### 📌 1. Double Integral Simulation (Area Density)

Function ComputeDoubleIntegral(radius As Double) As Double

' Approximates ∫∫\_R (x^2 + y^2) dx dy over a circular region using polar coordinates

Dim theta As Double, r As Double

Dim dTheta As Double: dTheta = 0.01

Dim dR As Double: dR = 0.01

Dim sum As Double: sum = 0

For theta = 0 To 2 \* WorksheetFunction.Pi() Step dTheta

For r = 0 To radius Step dR

sum = sum + (r ^ 2) \* r \* dR \* dTheta

Next r

Next theta

ComputeDoubleIntegral = sum

End Function

🧪 Result for radius = 2 → 8π ≈ 25.13

### 🚗 2. Collision Scenario Analyzer

Function Velocity(t As Double) As Double

Velocity = 10 \* t + 2

End Function

Function Acceleration(t As Double) As Double

Acceleration = 10

End Function

acceleration.

### 🧬 3. Evidence Decay Model

Function EvidenceDecay(C0 As Double, lambda As Double, t As Double) As Double

EvidenceDecay = C0 \* Exp(-lambda \* t)

End Function

Function PatrolDistance() As Double

Dim x As Double, dx As Double: dx = 0.01

Dim sum As Double: sum = 0

For x = 0 To 1 Step dx

sum = sum + Sqr(1 + 4 \* x ^ 2) \* dx

Next x

PatrolDistance = sum

End Function

🔫 5. Ballistic Range Calculator

Function BallisticRange(v0 As Double, thetaDeg As Double) As Double

Const g As Double = 9.8

Dim thetaRad As Double

thetaRad = thetaDeg \* WorksheetFunction.Pi() / 180

BallisticRange = (v0 ^ 2 \* Sin(2 \* thetaRad)) / g

End Function

📊 6. Predictive Crime Regression

Function ForecastCrimeRate(m As Double, b As Double, days As Double) As Double

ForecastCrimeRate = m \* days + b

End Function

unction ValidateProcedure(evidenceLabeled As Boolean, storedProperly As Boolean, custodyIntact As Boolean) As Boolean

ValidateProcedure = evidenceLabeled And storedProperly And custodyIntact

End Function

## 🧭 Deployment Strategy

* Integrate these modules into a **multi-sheet Excel workbook** with:
  + **Dashboard** for input/output
  + **Logigram sheet** for procedural flow
  + **Algorigram sheet** for simulation paths
* Use **QR-coded evidence logs** and **Access-linked audit trails** for real-time validation
* Embed **graphical outputs** using chart objects for trajectory and decay visualization

Would you like me to scaffold a full workbook structure or simulate a specific scenario—say, a ballistic incident with decay tracking and courtroom validation? We can modularize it into a certifiable training tool.

## Workbook structure and named ranges

Create these sheets and fields exactly; names are used by the code.

### Sheets

* **Dashboard**
  + **Inputs:** Progress actions, buttons
  + **Outputs:** Status, next module, warnings
* **Modules**
  + Columns: A: ModuleID, B: ModuleName, C: Sequence, D: Required (TRUE/FALSE)
* **Progress**
  + Columns: A: DateTime, B: User, C: ModuleID, D: EventType, E: Score, F: Duration
* **Quizzes**
  + Columns: A: ModuleID, B: QuizName, C: MaxScore, D: PassingScore
* **Procurement**
  + Columns: A: Field, B: Value (Department, ProjectCode, EstimatedCostPerDay, FundingSource, Resources)
* **Quotation**
  + Columns: A: Field, B: Value (FullName, Company, Contact, VAT\_BEE, Delivery)
* **Intake**
  + Columns: A: Field, B: Value (Name, Institution, Contact, Reason)
* **Workshops**
  + Columns: A: ModuleName, B: Type, C: Date, D: Facilitator, E: Room, F: Notes
* **CareerMap**
  + Columns: A: Position, B: Requirements, C: TimeFrame, D: Mentoring
* **RAndD**
  + Columns: A: Topic, B: FocusArea, C: Output, D: Status
* **Config**
  + Columns: A: Key, B: Value (e.g., CurrentUser, PassingPolicy)

### Named ranges

* **CurrentUser** (Config!B1)
* **EstimatedCostPerDay** (Procurement!B where Field="EstimatedCostPerDay")
* **PassingPolicy** (Config!B2)

## Logigram rules and algorigram flows

* **Course order (logigram):** You must complete modules in the strict sequence 1→6. A module can only unlock if all prior Required modules have EventType="Completed" in Progress.
* **Quiz gating (logigram):** If a module has quizzes, completion requires an average score ≥ policy threshold from Config!PassingPolicy.
* **Workflow orchestration (algorigram):**
  + On “Complete Module”: validate sequencing → log event → recompute status → update Dashboard.
  + On “Record Quiz”: validate module exists → log score/time → recompute module readiness.
  + On “Generate Portfolio”: pull Procurement, Quotation, Intake, Workshops, CareerMap → compose printable summary.

## Core VBA modules

Option Explicit

Public Enum EventTypeEnum

evt\_Started = 1

evt\_Quiz = 2

evt\_Completed = 3

End Enum

Function GetWs(name As String) As Worksheet

Set GetWs = ThisWorkbook.Worksheets(name)

End Function

Function NowStamp() As String

NowStamp = Format(Now, "yyyy-mm-dd hh:nn:ss")

End Function

Function GetConfig(key As String, Optional defaultValue As String = "") As String

Dim ws As Worksheet: Set ws = GetWs("Config")

Dim lastRow As Long: lastRow = ws.Cells(ws.Rows.Count, "A").End(xlUp).Row

Dim i As Long

For i = 1 To lastRow

If ws.Cells(i, 1).Value = key Then

GetConfig = CStr(ws.Cells(i, 2).Value)

Exit Function

End If

Next i

GetConfig = defaultValue

End Function

### 2) Course logigram: sequencing and status

vba

Function IsModuleUnlocked(moduleID As Variant) As Boolean

Dim wsM As Worksheet: Set wsM = GetWs("Modules")

Dim seq As Long, i As Long

seq = Application.WorksheetFunction.Index(wsM.Range("C:C"), \_

Application.WorksheetFunction.Match(moduleID, wsM.Range("A:A"), 0))

If seq <= 1 Then IsModuleUnlocked = True: Exit Function

For i = 1 To seq - 1

Dim priorID As Variant

priorID = Application.WorksheetFunction.Index(wsM.Range("A:A"), \_

Application.WorksheetFunction.Match(i, wsM.Range("C:C"), 0))

If IsModuleRequired(priorID) Then

If Not IsModuleCompleted(priorID) Then

IsModuleUnlocked = False

Exit Function

End If

End If

Next i

IsModuleUnlocked = True

End Function

Function IsModuleRequired(moduleID As Variant) As Boolean

Dim wsM As Worksheet: Set wsM = GetWs("Modules")

IsModuleRequired = CBool(Application.WorksheetFunction.Index(wsM.Range("D:D"), \_

Application.WorksheetFunction.Match(moduleID, wsM.Range("A:A"), 0)))

End Function

Function IsModuleCompleted(moduleID As Variant) As Boolean

Dim wsP As Worksheet: Set wsP = GetWs("Progress")

Dim lastRow As Long: lastRow = wsP.Cells(wsP.Rows.Count, "A").End(xlUp).Row

Dim i As Long

For i = lastRow To 2 Step -1

If wsP.Cells(i, 3).Value = moduleID And wsP.Cells(i, 4).Value = "Completed" Then

IsModuleCompleted = True

Exit Function

End If

Next i

IsModuleCompleted = False

End Function

Sub CompleteModule(moduleID As Variant)

If Not IsModuleUnlocked(moduleID) Then

MsgBox "Module " & moduleID & " is locked. Complete prior modules first.", vbExclamation

Exit Sub

End If

If Not MeetsQuizPolicy(moduleID) Then

MsgBox "Quiz policy not met for module " & moduleID & ".", vbExclamation

Exit Sub

End If

LogProgress moduleID, evt\_Completed, 0, 0

UpdateDashboard

MsgBox "Module " & moduleID & " marked as completed."

End Sub

Function MeetsQuizPolicy(moduleID As Variant) As Boolean

Dim wsQ As Worksheet: Set wsQ = GetWs("Quizzes")

Dim lastRow As Long: lastRow = wsQ.Cells(wsQ.Rows.Count, "A").End(xlUp).Row

Dim total As Double, countQ As Long, i As Long, avgScore As Double

For i = 2 To lastRow

If wsQ.Cells(i, 1).Value = moduleID Then

Dim qName As String: qName = wsQ.Cells(i, 2).Value

Dim maxS As Double: maxS = wsQ.Cells(i, 3).Value

Dim score As Double: score = GetLatestQuizScore(moduleID, qName)

If maxS > 0 Then

total = total + (score / maxS) \* 100

countQ = countQ + 1

End If

End If

Next i

If countQ = 0 Then MeetsQuizPolicy = True: Exit Function

avgScore = total / countQ

Dim policy As Double: policy = CDbl(Val(GetConfig("PassingPolicy", "50")))

MeetsQuizPolicy = (avgScore >= policy)

End Function

Function GetLatestQuizScore(moduleID As Variant, quizName As String) As Double

Dim wsP As Worksheet: Set wsP = GetWs("Progress")

Dim lastRow As Long: lastRow = wsP.Cells(wsP.Rows.Count, "A").End(xlUp).Row

Dim i As Long

For i = lastRow To 2 Step -1

If wsP.Cells(i, 3).Value = moduleID And wsP.Cells(i, 4).Value = "Quiz:" & quizName Then

GetLatestQuizScore = CDbl(wsP.Cells(i, 5).Value)

Exit Function

End If

Next i

GetLatestQuizScore = 0

End Function

Sub LogProgress(moduleID As Variant, evt As EventTypeEnum, score As Double, durationSec As Long)

Dim ws As Worksheet: Set ws = GetWs("Progress")

Dim r As Long: r = ws.Cells(ws.Rows.Count, "A").End(xlUp).Row + 1

ws.Cells(r, 1).Value = NowStamp()

ws.Cells(r, 2).Value = GetConfig("CurrentUser", "Learner")

ws.Cells(r, 3).Value = moduleID

Select Case evt

Case evt\_Started: ws.Cells(r, 4).Value = "Started"

Case evt\_Quiz: ws.Cells(r, 4).Value = "Quiz:" & ActiveQuizName()

Case evt\_Completed: ws.Cells(r, 4).Value = "Completed"

End Select

ws.Cells(r, 5).Value = score

ws.Cells(r, 6).Value = durationSec

End Sub

Function ActiveQuizName() As String

' Optionally pull from a cell on Dashboard

ActiveQuizName = GetWs("Dashboard").Range("B5").Value

If Len(ActiveQuizName) = 0 Then ActiveQuizName = "Introduction to AI"

End Function

Sub RecordQuizAttempt(moduleID As Variant, quizName As String, score As Double, durationSec As Long)

Dim ws As Worksheet: Set ws = GetWs("Progress")

Dim r As Long: r = ws.Cells(ws.Rows.Count, "A").End(xlUp).Row + 1

ws.Cells(r, 1).Value = NowStamp()

ws.Cells(r, 2).Value = GetConfig("CurrentUser", "Learner")

ws.Cells(r, 3).Value = moduleID

ws.Cells(r, 4).Value = "Quiz:" & quizName

ws.Cells(r, 5).Value = score

ws.Cells(r, 6).Value = durationSec

UpdateDashboard

End Sub

Sub UpdateDashboard()

Dim wsD As Worksheet: Set wsD = GetWs("Dashboard")

Dim wsM As Worksheet: Set wsM = GetWs("Modules")

Dim lastRow As Long: lastRow = wsM.Cells(wsM.Rows.Count, "A").End(xlUp).Row

Dim i As Long, nextMod As Variant: nextMod = ""

For i = 2 To lastRow

Dim mid As Variant: mid = wsM.Cells(i, 1).Value

If Not IsModuleCompleted(mid) Then

If IsModuleUnlocked(mid) Then

nextMod = mid

Exit For

End If

End If

Next i

wsD.Range("B2").Value = IIf(nextMod = "", "All modules completed", "Next module: " & nextMod)

wsD.Range("B3").Value = "User: " & GetConfig("CurrentUser", "Learner")

wsD.Range("B4").Value = "Policy: " & GetConfig("PassingPolicy", "50") & "%"

End Sub

5) Procurement and quotation validators

Function ValidateProcurement() As Boolean

Dim ws As Worksheet: Set ws = GetWs("Procurement")

Dim dept As String, estCost As Variant, fund As String, res As String

dept = GetField(ws, "Department")

estCost = GetField(ws, "EstimatedCostPerDay")

fund = GetField(ws, "FundingSource")

res = GetField(ws, "Resources")

If Len(dept) = 0 Or Len(fund) = 0 Or Len(res) = 0 Then

MsgBox "Missing procurement fields (Department/Funding/Resources).", vbExclamation

ValidateProcurement = False: Exit Function

End If

If Not IsNumeric(estCost) Or CDbl(estCost) <= 0 Then

MsgBox "Estimated cost per day must be a positive number (e.g., R385,000/day).", vbExclamation

ValidateProcurement = False: Exit Function

End If

ValidateProcurement = True

End Function

Function GetField(ws As Worksheet, fieldName As String) As String

Dim lastRow As Long: lastRow = ws.Cells(ws.Rows.Count, "A").End(xlUp).Row

Dim i As Long

For i = 1 To lastRow

If ws.Cells(i, 1).Value = fieldName Then

GetField = CStr(ws.Cells(i, 2).Value)

Exit Function

End If

Next i

GetField = ""

End Function

6) Portfolio generator (single-click export)

Sub GeneratePortfolioSummary()

If Not ValidateProcurement Then Exit Sub

Dim wsD As Worksheet: Set wsD = GetWs("Dashboard")

Dim tmp As Worksheet

On Error Resume Next: Application.DisplayAlerts = False

ThisWorkbook.Worksheets("Portfolio").Delete

Application.DisplayAlerts = True: On Error GoTo 0

Set tmp = ThisWorkbook.Worksheets.Add

tmp.Name = "Portfolio"

Dim r As Long: r = 1

tmp.Cells(r, 1).Value = "Portfolio Summary": r = r + 2

r = CopySection(tmp, r, "Course Status", Array( \_

"User", GetConfig("CurrentUser", "Learner"), \_

"Status", wsD.Range("B2").Value, \_

"Policy", GetConfig("PassingPolicy", "50") & "%"))

r = r + 1

r = CopyKeyValues(tmp, r, "Procurement", GetWs("Procurement"))

r = r + 1

r = CopyKeyValues(tmp, r, "Quotation", GetWs("Quotation"))

r = r + 1

r = CopyTable(tmp, r, "Workshops", GetWs("Workshops"))

r = r + 1

r = CopyTable(tmp, r, "Career Mapping", GetWs("CareerMap"))

r = r + 1

r = CopyTable(tmp, r, "R&D Topics", GetWs("RAndD"))

tmp.Columns.AutoFit

Dim f As String

f = ThisWorkbook.Path & "\Portfolio\_" & Format(Now, "yyyy-mm-dd\_hhnn") & ".pdf"

On Error Resume Next

tmp.ExportAsFixedFormat Type:=xlTypePDF, Filename:=f

On Error GoTo 0

MsgBox "Portfolio generated: " & f, vbInformation

End Sub

Function CopySection(ws As Worksheet, startRow As Long, title As String, kvArr As Variant) As Long

ws.Cells(startRow, 1).Value = title

Dim i As Long, r As Long: r = startRow + 1

For i = LBound(kvArr) To UBound(kvArr) Step 2

ws.Cells(r, 1).Value = kvArr(i)

ws.Cells(r, 2).Value = kvArr(i + 1)

r = r + 1

Next i

CopySection = r

End Function

Function CopyKeyValues(ws As Worksheet, startRow As Long, title As String, src As Worksheet) As Long

ws.Cells(startRow, 1).Value = title

Dim r As Long: r = startRow + 1

Dim lastRow As Long: lastRow = src.Cells(src.Rows.Count, "A").End(xlUp).Row

Dim i As Long

For i = 1 To lastRow

ws.Cells(r, 1).Value = src.Cells(i, 1).Value

ws.Cells(r, 2).Value = src.Cells(i, 2).Value

r = r + 1

Next i

CopyKeyValues = r

End Function

Function CopyTable(ws As Worksheet, startRow As Long, title As String, src As Worksheet) As Long

ws.Cells(startRow, 1).Value = title

Dim r As Long: r = startRow + 1

Dim rng As Range

Set rng = src.Range("A1").CurrentRegion

rng.Copy Destination:=ws.Cells(r, 1)

CopyTable = r + rng.Rows.Count + 1

End Function

## Buttons and quick wiring

* **Dashboard buttons:**
  + **Start Module:** Assign macro StartModulePrompt (below).
  + **Record Quiz:** Assign RecordQuizPrompt.
  + **Complete Module:** Assign CompleteModulePrompt.
  + **Generate Portfolio:** Assign GeneratePortfolioSummary.

vba

Sub StartModulePrompt()

Dim mid As Variant: mid = InputBox("Enter ModuleID to start (e.g., 1):")

If Len(mid) = 0 Then Exit Sub

LogProgress mid, evt\_Started, 0, 0

UpdateDashboard

End Sub

Sub RecordQuizPrompt()

Dim mid As Variant: mid = InputBox("ModuleID:")

If Len(mid) = 0 Then Exit Sub

Dim q As String: q = InputBox("Quiz name:")

Dim s As Double: s = Val(InputBox("Score:"))

Dim d As Long: d = CLng(Val(InputBox("Duration (sec):")))

RecordQuizAttempt mid, q, s, d

End Sub

Sub CompleteModulePrompt()

Dim mid As Variant: mid = InputBox("ModuleID to complete:")

If Len(mid) = 0 Then Exit Sub

CompleteModule mid

End Sub

## ow to use and tailor

* **Initialize modules list:**
  + Populate Modules with:
    - 1 AI Basics
    - 2 Generative AI
    - 3 Internet Search and Beyond
    - 4 Responsible AI
    - 5 What is Microsoft Copilot
    - 6 AI For All
  + Set Required = TRUE for all.
* **Set policy and user:**
  + Config:
    - CurrentUser = “Tshingombe Fiston Tshitadi”
    - PassingPolicy = 50 (or your desired threshold)
* **Record your existing quiz result:**
  + Use Record Quiz button for “Introduction to AI”, score 3, max defined on Quizzes, duration 39 seconds.
* **Generate your submission pack:**
  + Fill Procurement, Quotation, Intake, Workshops, CareerMap, RAndD.
  + Click Generate Portfolio to export a single PDF for DHET/SAQA/Expo.

## Optional extensions

* **Evidence-based audit trail:** Add a “Hash” column in Progress and compute SHA-256 for tamper-evident logs.
* **UserForms:** Replace InputBox calls with forms for Intake, Quotation, and Procurement to improve data quality.
* **Validation rules:** Data Validation lists for Departments (Education, Career Center, ICT) and SETA sectors (MERSETA, TETA, HASTA, SASSETA).
* **KPI tiles on Dashboard:** Completed count, next module, average quiz score, procurement readiness flag.
* **Exports:** Add CSV and DOCX exports for institutions requiring alternate formats.

If you want, I can package this into a ready-to-import .bas modules set and a sample workbook skeleton with the sheets prewired and a one-click Setup

## Workbook schema and setup

Create these sheets with the specified columns. Names are referenced by the VBA code.

* **Dashboard**
  + **Inputs:** Buttons, current session status
  + **Outputs:** Next step, safety status, calibration status, data quality flags
* **Assets**
  + Columns: A: AssetID, B: Category, C: Model, D: SerialNo, E: Location, F: Status
* **Sessions**
  + Columns: A: SessionID, B: Trainee, C: Module, D: StartTime, E: EndTime, F: Facilitator, G: Status
* **Safety**
  + Columns: A: ChecklistItem, B: Required, C: Completed, D: Notes
* **Calibration**
  + Columns: A: SensorID, B: Type, C: Date, D: ReadingKnown, E: ForceKnownN, F: Scale, G: Offset
* **Measurements**
  + Columns: A: SessionID, B: TestType, C: X\_Pos\_m, D: LoadType, E: LoadValue, F: DynoLeft\_N, G: DynoRight\_N, H: Dial1\_mm, I: Dial2\_mm, J: Temp\_C
* **Analysis**
  + Columns: A: SessionID, B: Computation, C: Param1, D: Param2, E: Param3, F: Result
* **Procurement**
  + Columns: A: Field, B: Value (Department, ProjectCode, EstimatedCostPerDay, FundingSource, Resources)
* **Config**
  + Columns: A: Key, B: Value (CurrentUser, PassingPolicy, E\_Modulus\_Pa, Beam\_Length\_m, Beam\_Width\_m, Beam\_Height\_m, Gravity)

Named ranges (Config!B cell next to key):

* **CurrentUser**, **E\_Modulus\_Pa**, **Beam\_Length\_m**, **Beam\_Width\_m**, **Beam\_Height\_m**, **Gravity**

## Safety and readiness logigram

* **Rule 1 (assets ready):** All required assets for the module must be Status="Available".
* **Rule 2 (safety):** All Safety items with Required=TRUE must have Completed=TRUE before Start.
* **Rule 3 (calibration):** All sensors in use must have non-empty Scale/Offset from same-day calibration.
* **Rule 4 (data sanity):** Dynamometer reactions must statically balance applied loads within tolerance.
* Option Explicit
* Function GetWs(name As String) As Worksheet
* Set GetWs = ThisWorkbook.Worksheets(name)
* End Function
* Function Cfg(key As String, Optional defVal As Variant) As Variant
* Dim ws As Worksheet: Set ws = GetWs("Config")
* Dim r As Range: Set r = ws.Columns(1).Find(what:=key, LookIn:=xlValues, lookat:=xlWhole)
* If r Is Nothing Then
* Cfg = defVal
* Else
* Cfg = r.Offset(0, 1).Value
* If IsEmpty(Cfg) Then Cfg = defVal
* End If
* End Function
* Function SafetyReady() As Boolean
* Dim ws As Worksheet: Set ws = GetWs("Safety")
* Dim last As Long: last = ws.Cells(ws.Rows.Count, "A").End(xlUp).Row
* Dim i As Long
* For i = 2 To last
* If CBool(ws.Cells(i, 2).Value) = True Then
* If CBool(ws.Cells(i, 3).Value) = False Then
* SafetyReady = False: Exit Function
* End If
* End If
* Next i
* SafetyReady = True
* End Function
* Function CalibrationReady(sensorType As String) As Boolean
* Dim ws As Worksheet: Set ws = GetWs("Calibration")
* Dim last As Long: last = ws.Cells(ws.Rows.Count, "A").End(xlUp).Row
* Dim today As Date: today = Date
* Dim ok As Boolean: ok = False
* Dim i As Long
* For i = 2 To last
* If LCase(ws.Cells(i, 2).Value) = LCase(sensorType) Then
* If ws.Cells(i, 6).Value <> "" And ws.Cells(i, 7).Value <> "" Then
* If CDate(ws.Cells(i, 3).Value) = today Then ok = True
* End If
* End If
* Next i
* CalibrationReady = ok
* End Function
* Function AssetsReady(moduleName As String) As Boolean
* Dim ws As Worksheet: Set ws = GetWs("Assets")
* Dim last As Long: last = ws.Cells(ws.Rows.Count, "A").End(xlUp).Row
* Dim need As Long, have As Long, i As Long
* For i = 2 To last
* If InStr(1, LCase(moduleName), LCase(ws.Cells(i, 2).Value), vbTextCompare) > 0 Then
* need = need + 1
* If LCase(ws.Cells(i, 6).Value) = "available" Then have = have + 1
* End If
* Next i
* AssetsReady = (need > 0 And have = need)
* End Function
* Function SessionStartAllowed(moduleName As String) As Boolean
* If Not AssetsReady(moduleName) Then
* MsgBox "Assets not ready for module: " & moduleName, vbExclamation
* SessionStartAllowed = False: Exit Function
* End If
* If Not SafetyReady() Then
* MsgBox "Safety checklist incomplete.", vbExclamation
* SessionStartAllowed = False: Exit Function
* End If
* If Not CalibrationReady("dynamometer") Then
* MsgBox "Dynamometer calibration missing today.", vbExclamation
* SessionStartAllowed = False: Exit Function
* End If
* If Not CalibrationReady("dial") Then
* MsgBox "Dial indicator calibration missing today.", vbExclamation
* SessionStartAllowed = False: Exit Function
* End If
* SessionStartAllowed = True
* End Function

## Mechanics algorigram: beams, frames, calibration, and validation

These functions support DL ST033-type labs: reactions from loads, shear/moment, deflection, sensor calibration, and static balance checks.

' --- Geometry and material helpers ---

Function BeamInertiaRect(b As Double, h As Double) As Double

BeamInertiaRect = b \* h ^ 3 / 12#

End Function

' --- Calibration: linear sensor y = Scale \* x + Offset ---

Sub CalibrateSensor(sensorID As String, sensorType As String, readingKnown As Double, forceKnownN As Double)

Dim ws As Worksheet: Set ws = GetWs("Calibration")

Dim last As Long: last = ws.Cells(ws.Rows.Count, "A").End(xlUp).Row + 1

Dim scale As Double, offset As Double

' For simplicity: assume two-point method stored elsewhere; here we compute one-step scale if offset ~ 0

' Extend by storing previous point to compute full linear fit.

scale = IIf(readingKnown <> 0, forceKnownN / readingKnown, 0)

offset = 0

ws.Cells(last, 1).Value = sensorID

ws.Cells(last, 2).Value = sensorType

ws.Cells(last, 3).Value = Date

ws.Cells(last, 4).Value = readingKnown

ws.Cells(last, 5).Value = forceKnownN

ws.Cells(last, 6).Value = scale

ws.Cells(last, 7).Value = offset

MsgBox "Calibrated " & sensorType & " [" & sensorID & "] scale=" & Format(scale, "0.0000")

End Sub

Function ApplyCalibration(sensorID As String, rawReading As Double) As Double

Dim ws As Worksheet: Set ws = GetWs("Calibration")

Dim last As Long: last = ws.Cells(ws.Rows.Count, "A").End(xlUp).Row

Dim i As Long

For i = last To 2 Step -1

If ws.Cells(i, 1).Value = sensorID Then

ApplyCalibration = ws.Cells(i, 6).Value \* rawReading + ws.Cells(i, 7).Value

Exit Function

End If

Next i

ApplyCalibration = rawReading ' fallback

End Function

' --- Statics: simply supported beam, point load P at position a (from left), span L ---

Sub Reactions\_PointLoad(L As Double, a As Double, P As Double, ByRef Rleft As Double, ByRef Rright As Double)

' SumMoments@Left: Rright\*L = P\*a => Rright = P\*a/L ; Rleft = P - Rright

Rright = P \* a / L

Rleft = P - Rright

End Sub

' --- Statics: uniformly distributed load w (N/m) over entire span L ---

Sub Reactions\_UDL(L As Double, w As Double, ByRef Rleft As Double, ByRef Rright As Double)

' Resultant = wL at midspan => equal reactions for full-length uniform load

Rleft = w \* L / 2#

Rright = w \* L / 2#

End Sub

' --- Shear/Moment arrays (discrete for plotting or post-processing) ---

Sub ShearMoment\_PointLoad(L As Double, a As Double, P As Double, stepX As Double, outWs As Worksheet, startRow As Long)

Dim Rl As Double, Rr As Double

Reactions\_PointLoad L, a, P, Rl, Rr

Dim x As Double, V As Double, M As Double, r As Long: r = startRow

For x = 0 To L Step stepX

If x < a Then

V = Rl

M = Rl \* x

Else

V = Rl - P

M = Rl \* x - P \* (x - a)

End If

outWs.Cells(r, 1).Value = x

outWs.Cells(r, 2).Value = V

outWs.Cells(r, 3).Value = M

r = r + 1

Next x

End Sub

' --- Euler-Bernoulli deflection at position x for point load at a ---

Function Deflection\_PointLoad(E As Double, I As Double, L As Double, a As Double, P As Double, x As Double) As Double

' Closed-form for simply supported beam:

' For x <= a: y = (P\*b\*x/(6\*L\*E\*I))\*(L^2 - b^2 - x^2), with b = L - a

' For x >= a: y = (P\*b/(6\*L\*E\*I))\*((L/x)\*(L^2 - b^2) - (x^3)/x) simplified below

Dim b As Double: b = L - a

If x <= a Then

Deflection\_PointLoad = (P \* b \* x / (6# \* L \* E \* I)) \* (L ^ 2 - b ^ 2 - x ^ 2)

Else

Deflection\_PointLoad = (P \* b / (6# \* L \* E \* I)) \* (L ^ 2 - b ^ 2 - x ^ 2) \* (L - x)

' Note: For brevity we use a compact symmetrical form adequate for lab comparisons.

End If

End Function

' --- Uniform load maximum deflection at midspan (simply supported) ---

Function DeflectionMax\_UDL(E As Double, I As Double, L As Double, w As Double) As Double

' y\_max = (5 w L^4) / (384 E I)

DeflectionMax\_UDL = (5# \* w \* L ^ 4) / (384# \* E \* I)

End Function

' --- Sensor fusion check: static balance tolerance ---

Function StaticBalanceOK(P\_total As Double, Rleft As Double, Rright As Double, Optional tolPct As Double = 2) As Boolean

Dim sumR As Double: sumR = Rleft + Rright

If P\_total = 0 Then StaticBalanceOK = False: Exit Function

StaticBalanceOK = (Abs(sumR - P\_total) / P\_total \* 100# <= tolPct)

End Function

Quick math references for learners:

* Shear/moment are derived from equilibrium. For a point load, reactions are RL=P(1−a/L)R\_L = P(1 - a/L), RR=P(a/L)R\_R = P(a/L).
* Uniform load deflection maximum: ymax⁡=5wL4384EIy\_{\max} = \frac{5 w L^4}{384 E I}.
* Deflection functions above are adequate for training comparisons; you can extend to multiple loads via superposition.

## Session orchestration, measurement logging, and reporting

This flow drives a full lab: start → record → analyze → validate → export.

' --- Start a lab session ---

Sub StartSession()

Dim moduleName As String: moduleName = "Beams and Frames"

If Not SessionStartAllowed(moduleName) Then Exit Sub

Dim ws As Worksheet: Set ws = GetWs("Sessions")

Dim newID As String: newID = "S" & Format(Now, "yymmddhhnnss")

Dim r As Long: r = ws.Cells(ws.Rows.Count, "A").End(xlUp).Row + 1

ws.Cells(r, 1).Value = newID

ws.Cells(r, 2).Value = Cfg("CurrentUser", "Trainee")

ws.Cells(r, 3).Value = moduleName

ws.Cells(r, 4).Value = Now

ws.Cells(r, 6).Value = "Facilitator"

ws.Cells(r, 7).Value = "In Progress"

GetWs("Dashboard").Range("B2").Value = "Active Session: " & newID

MsgBox "Session started: " & newID, vbInformation

End Sub

' --- Record a beam test measurement row ---

Sub RecordBeamMeasurement()

Dim ws As Worksheet: Set ws = GetWs("Measurements")

Dim sid As String: sid = GetWs("Dashboard").Range("B2").Value

If InStr(1, sid, "Active Session: ") = 0 Then

MsgBox "No active session. StartSession first.", vbExclamation: Exit Sub

End If

sid = Replace(sid, "Active Session: ", "")

Dim L As Double: L = CDbl(Cfg("Beam\_Length\_m", 1.2))

Dim b As Double: b = CDbl(Cfg("Beam\_Width\_m", 0.03))

Dim h As Double: h = CDbl(Cfg("Beam\_Height\_m", 0.006))

Dim E As Double: E = CDbl(Cfg("E\_Modulus\_Pa", 2E11))

Dim a As Double: a = Val(InputBox("Load position a (m) from left, 0 to L:", "Beam"))

Dim P As Double: P = Val(InputBox("Point load P (N):", "Beam"))

Dim dynoL\_raw As Double: dynoL\_raw = Val(InputBox("Dynamometer LEFT raw:", "Sensors"))

Dim dynoR\_raw As Double: dynoR\_raw = Val(InputBox("Dynamometer RIGHT raw:", "Sensors"))

Dim dial1\_mm As Double: dial1\_mm = Val(InputBox("Dial indicator 1 reading (mm):", "Sensors"))

Dim dynoL\_N As Double: dynoL\_N = ApplyCalibration("DYNO\_L", dynoL\_raw)

Dim dynoR\_N As Double: dynoR\_N = ApplyCalibration("DYNO\_R", dynoR\_raw)

Dim r As Long: r = ws.Cells(ws.Rows.Count, "A").End(xlUp).Row + 1

ws.Cells(r, 1).Value = sid

ws.Cells(r, 2).Value = "PointLoad"

ws.Cells(r, 3).Value = a

ws.Cells(r, 4).Value = "P"

ws.Cells(r, 5).Value = P

ws.Cells(r, 6).Value = dynoL\_N

ws.Cells(r, 7).Value = dynoR\_N

ws.Cells(r, 8).Value = dial1\_mm

ws.Cells(r, 10).Value = Cfg("Lab\_Temperature\_C", 22)

' Analysis and validation

Dim Rl As Double, Rr As Double

Reactions\_PointLoad L, a, P, Rl, Rr

Dim ok As Boolean: ok = StaticBalanceOK(P, dynoL\_N, dynoR\_N, 5)

GetWs("Dashboard").Range("B3").Value = IIf(ok, "Static balance OK", "Check load/reaction mismatch")

' Deflection prediction at midspan

Dim I As Double: I = BeamInertiaRect(b, h)

Dim y\_mid As Double: y\_mid = Deflection\_PointLoad(E, I, L, a, P, L / 2#)

Dim wa As Worksheet: Set wa = GetWs("Analysis")

Dim ra As Long: ra = wa.Cells(wa.Rows.Count, "A").End(xlUp).Row + 1

wa.Cells(ra, 1).Value = sid

wa.Cells(ra, 2).Value = "Predicted midspan deflection (m)"

wa.Cells(ra, 3).Value = L

wa.Cells(ra, 4).Value = a

wa.Cells(ra, 5).Value = P

wa.Cells(ra, 6).Value = y\_mid

MsgBox "Measurement logged. Predicted midspan deflection (m): " & Format(y\_mid, "0.000000")

End Sub

' --- End session and generate summary ---

Sub EndSessionAndReport()

Dim ws As Worksheet: Set ws = GetWs("Sessions")

Dim sid As String: sid = GetWs("Dashboard").Range("B2").Value

If InStr(1, sid, "Active Session: ") = 0 Then

MsgBox "No active session.", vbExclamation: Exit Sub

End If

sid = Replace(sid, "Active Session: ", "")

Dim r As Range: Set r = ws.Columns(1).Find(sid, LookIn:=xlValues, lookat:=xlWhole)

If r Is Nothing Then

MsgBox "Session ID not found.", vbCritical: Exit Sub

End If

r.Offset(0, 5).Value = Now

r.Offset(0, 6).Value = "Completed"

GenerateSessionReport sid

GetWs("Dashboard").Range("B2").Value = ""

GetWs("Dashboard").Range("B3").Value = ""

MsgBox "Session closed and report generated."

End Sub

Sub GenerateSessionReport(sessionID As String)

Dim wb As Workbook: Set wb = ThisWorkbook

Dim wsM As Worksheet: Set wsM = GetWs("Measurements")

Dim wsA As Worksheet: Set wsA = GetWs("Analysis")

Dim wsS As Worksheet: Set wsS = GetWs("Sessions")

On Error Resume Next: Application.DisplayAlerts = False

wb.Worksheets("Report").Delete

Application.DisplayAlerts = True: On Error GoTo 0

Dim wr As Worksheet: Set wr = wb.Worksheets.Add

wr.Name = "Report"

Dim row As Long: row = 1

wr.Cells(row, 1).Value = "Mechanical Lab Session Report": row = row + 2

row = WriteKVs(wr, row, "Session Meta", Array( \_

"SessionID", sessionID, \_

"Trainee", GetValue(wsS, sessionID, "B"), \_

"Module", GetValue(wsS, sessionID, "C"), \_

"Start", GetValue(wsS, sessionID, "D"), \_

"End", GetValue(wsS, sessionID, "E"), \_

"Status", GetValue(wsS, sessionID, "G")))

row = row + 1

row = CopyRowsForSession(wr, row, "Measurements", wsM, sessionID)

row = row + 1

row = CopyRowsForSession(wr, row, "Analysis", wsA, sessionID)

wr.Columns.AutoFit

Dim f As String: f = wb.Path & "\Session\_" & sessionID & ".pdf"

On Error Resume Next

wr.ExportAsFixedFormat xlTypePDF, f

On Error GoTo 0

End Sub

Function WriteKVs(ws As Worksheet, startRow As Long, title As String, kv As Variant) As Long

ws.Cells(startRow, 1).Value = title

Dim r As Long: r = startRow + 1

Dim i As Long

For i = LBound(kv) To UBound(kv) Step 2

ws.Cells(r, 1).Value = kv(i)

ws.Cells(r, 2).Value = kv(i + 1)

r = r + 1

Next i

WriteKVs = r

End Function

Function GetValue(ws As Worksheet, sessionID As String, colLetter As String) As Variant

Dim r As Range: Set r = ws.Columns(1).Find(sessionID, LookIn:=xlValues, lookat:=xlWhole)

If r Is Nothing Then GetValue = "": Exit Function

GetValue = ws.Cells(r.Row, Range(colLetter & "1").Column).Value

End Function

Function CopyRowsForSession(dst As Worksheet, startRow As Long, title As String, src As Worksheet, sessionID As String) As Long

Dim r As Long: r = startRow

dst.Cells(r, 1).Value = title: r = r + 1

Dim rng As Range: Set rng = src.Range("A1").CurrentRegion

Dim i As Long

For i = 2 To rng.Rows.Count

If rng.Cells(i, 1).Value = sessionID Then

rng.Rows(1).Copy dst.Cells(r, 1) ' header once

rng.Rows(i).Copy dst.Cells(r + 1, 1)

r = r + 2

End If

Next i

CopyRowsForSession = r

End Function

## Buttons and quick wiring

Add buttons on Dashboard and assign:

* **Start Session:** StartSession
* **Record Measurement:** RecordBeamMeasurement
* **End & Report:** EndSessionAndReport
* **Calibrate Sensor:** CalibrateSensorPrompt
* Sub CalibrateSensorPrompt()
* Dim id As String: id = InputBox("Sensor ID (e.g., DYNO\_L):")
* If Len(id) = 0 Then Exit Sub
* Dim typ As String: typ = InputBox("Type (dynamometer/dial):", "Type", "dynamometer")
* Dim r As Double: r = Val(InputBox("Known reading (raw units):"))
* Dim F As Double: F = Val(InputBox("Known force (N) or displacement (mm->N use fixture):"))
* CalibrateSensor id, typ, r, F
* End Sub

## Extending to electronic/audiometer labs and procurement

* **Electronic/audiometer modules:**
  + **Add Measurements columns:** Frequency\_Hz, Level\_dB, Output\_Vpp, Thresholds.
  + **Add Calibration type:** “audio” with Scale in dB per volt.
  + **Add Analysis:** hearing curve plotting (store session computations in Analysis).
* **Procurement integration:**
  + Use the procurement sheet to auto-validate session resource costs.
  + Add a cost roll-up in Report from EstimatedCostPerDay and session duration.

If you want, I can add a multi-load superposition engine (arbitrary loads), shear-moment charting, and an audiometry measurement form so trainees can capture frequency–threshold curves with auto-fit and competency scoring.

# VBA logigram and algorigram for DL NGL lab integration (BASE, STUDENT, CIMSIM, IoT, DATA, CYBER)

Below is a modular Excel VBA engine that turns your DL NGL ecosystem into auditable, simulation-ready workflows. It enforces infrastructure readiness (logigram), orchestrates learning scenarios (algorigram), and captures evidence for portfolios and accreditation.

## Workbook schema

Create sheets exactly as named. Columns are referenced by code.

* StationRegistry
  + A: StationID, B: Role (Teacher/Student), C: Hostname, D: IP, E: Connectivity (LAN/WiFi), F: Status (Online/Offline), G: DL\_WORKSPACE (Yes/No)
* ModuleCatalog
  + A: ModuleID, B: Name (CIMSIM/IoT/DATA/CYBER), C: RequiredAssets (comma list), D: PrereqModules (comma list), E: Enabled (TRUE/FALSE)
* DeviceRegistry
  + A: DeviceID, B: Type (PLC/DevIoT/Sensor/Actuator), C: Model, D: PortMap, E: Protocols (MQTT/Modbus), F: AssignedStation, G: Status
* ScenarioBook
  + A: ScenarioID, B: ModuleID, C: Name, D: Objective, E: Steps (CSV), F: PassCriteria
* Events
  + A: Timestamp, B: User, C: ScenarioID, D: EventType, E: Payload1, F: Payload2, G: Notes
* Measurements
  + A: ScenarioID, B: Metric, C: Value, D: Unit, E: SourceDevice, F: Timestamp
* Config
  + A: Key, B: Value (CurrentUser, MinStudents, RequireDL\_WORKSPACE, MQTT\_Topic\_OnOff, SafetyPolicy, EvidenceDir)
* Safety
  + A: ChecklistItem, B: Required (TRUE/FALSE), C: Completed (TRUE/FALSE), D: Notes
* Portfolio
  + Generated by macro (no manual columns)

Named ranges (Config!B next to key):

* CurrentUser, MinStudents, RequireDL\_WORKSPACE, MQTT\_Topic\_OnOff, SafetyPolicy, EvidenceDir

## Logigram rules

* Infrastructure readiness:
  + Teacher station Online and DL\_WORKSPACE = Yes.
  + Count(Students Online) ≥ MinStudents.
  + Required assets for the selected module are Status = Available/Online.
* Safety gating:
  + All Required items in Safety are Completed = TRUE.
* Module prerequisites:
  + All PrereqModules are Enabled and previously run (in Events as Completed).
* Scenario approval:
  + Scenario steps are executable with current Devices and Protocols.

## Algorigram flows

* StartScenario:
  + Validate infrastructure → Validate safety → Check module prereqs → Lock resources → Log Started.
* RunStep:
  + Execute step dispatcher (CIMSIM | IoT | DATA | CYBER) → Capture measurement(s) → Log checkpoint.
* EvaluateScenario:
  + Compare measurements against PassCriteria → Log Completed/Failed → Export evidence (PDF/CSV).

## Core VBA

### Utilities and config

vba

Option Explicit

Function WS(name As String) As Worksheet

Set WS = ThisWorkbook.Worksheets(name)

End Function

Function Cfg(key As String, Optional defVal As Variant = "") As Variant

Dim r As Range

Set r = WS("Config").Columns(1).Find(what:=key, LookIn:=xlValues, lookat:=xlWhole)

If r Is Nothing Then Cfg = defVal Else Cfg = IIf(IsEmpty(r.Offset(0, 1).Value), defVal, r.Offset(0, 1).Value)

End Function

Function NowStamp() As String

NowStamp = Format(Now, "yyyy-mm-dd hh:nn:ss")

End Function

Sub LogEvent(scenarioID As String, evt As String, Optional p1 As String = "", Optional p2 As String = "", Optional note As String = "")

Dim ws As Worksheet: Set ws = WS("Events")

Dim r As Long: r = ws.Cells(ws.Rows.Count, 1).End(xlUp).Row + 1

ws.Cells(r, 1).Value = NowStamp()

ws.Cells(r, 2).Value = Cfg("CurrentUser", "Learner")

ws.Cells(r, 3).Value = scenarioID

ws.Cells(r, 4).Value = evt

ws.Cells(r, 5).Value = p1

ws.Cells(r, 6).Value = p2

ws.Cells(r, 7).Value = note

End Sub

Sub RecordMetric(scenarioID As String, metric As String, val As Double, unitStr As String, src As String)

Dim ws As Worksheet: Set ws = WS("Measurements")

Dim r As Long: r = ws.Cells(ws.Rows.Count, 1).End(xlUp).Row + 1

ws.Cells(r, 1).Value = scenarioID

ws.Cells(r, 2).Value = metric

ws.Cells(r, 3).Value = val

ws.Cells(r, 4).Value = unitStr

ws.Cells(r, 5).Value = src

ws.Cells(r, 6).Value = NowStamp()

End Sub

### Readiness logigram

vba

Function TeacherReady() As Boolean

Dim r As Range, f As Range, ok As Boolean: ok = False

With WS("StationRegistry")

Set f = .Range("A1").CurrentRegion

End With

Dim i As Long

For i = 2 To f.Rows.Count

If LCase(f.Cells(i, 2).Value) = "teacher" Then

If LCase(f.Cells(i, 6).Value) = "online" Then

If CBool(Cfg("RequireDL\_WORKSPACE", True)) Then

If LCase(f.Cells(i, 7).Value) = "yes" Then ok = True

Else

ok = True

End If

End If

End If

Next i

TeacherReady = ok

End Function

Function StudentsReady() As Boolean

Dim f As Range, i As Long, cnt As Long

Set f = WS("StationRegistry").Range("A1").CurrentRegion

For i = 2 To f.Rows.Count

If LCase(f.Cells(i, 2).Value) = "student" And LCase(f.Cells(i, 6).Value) = "online" Then cnt = cnt + 1

Next i

StudentsReady = (cnt >= CLng(Cfg("MinStudents", 1)))

End Function

Function SafetyReady() As Boolean

Dim f As Range, i As Long

Set f = WS("Safety").Range("A1").CurrentRegion

For i = 2 To f.Rows.Count

If CBool(f.Cells(i, 2).Value) = True And CBool(f.Cells(i, 3).Value) = False Then

SafetyReady = False: Exit Function

End If

Next i

SafetyReady = True

End Function

Function AssetsForModuleReady(moduleID As String) As Boolean

Dim mc As Worksheet: Set mc = WS("ModuleCatalog")

Dim dr As Worksheet: Set dr = WS("DeviceRegistry")

Dim req As String

req = GetModuleField(moduleID, "RequiredAssets")

If Len(Trim(req)) = 0 Then AssetsForModuleReady = True: Exit Function

Dim arr() As String: arr = Split(req, ",")

Dim i As Long

For i = LBound(arr) To UBound(arr)

If DeviceStatus(Trim(arr(i))) = False Then

AssetsForModuleReady = False: Exit Function

End If

Next i

AssetsForModuleReady = True

End Function

Function DeviceStatus(deviceID As String) As Boolean

Dim r As Range

Set r = WS("DeviceRegistry").Columns(1).Find(deviceID, LookIn:=xlValues, lookat:=xlWhole)

If r Is Nothing Then DeviceStatus = False: Exit Function

DeviceStatus = (LCase(r.Offset(0, 6).Value) = "online" Or LCase(r.Offset(0, 6).Value) = "available")

End Function

Function GetModuleField(moduleID As String, fieldName As String) As String

Dim mc As Worksheet: Set mc = WS("ModuleCatalog")

Dim r As Range: Set r = mc.Columns(1).Find(moduleID, LookIn:=xlValues, lookat:=xlWhole)

If r Is Nothing Then GetModuleField = "": Exit Function

Select Case LCase(fieldName)

Case "requiredassets": GetModuleField = CStr(r.Offset(0, 2).Value)

Case "prereqmodules": GetModuleField = CStr(r.Offset(0, 3).Value)

Case "enabled": GetModuleField = CStr(r.Offset(0, 4).Value)

Case "name": GetModuleField = CStr(r.Offset(0, 1).Value)

Case Else: GetModuleField = ""

End Select

End Function

Function ModulePrereqsMet(moduleID As String) As Boolean

Dim pre As String: pre = GetModuleField(moduleID, "PrereqModules")

If Len(Trim(pre)) = 0 Then ModulePrereqsMet = True: Exit Function

Dim a() As String: a = Split(pre, ",")

Dim i As Long

For i = LBound(a) To UBound(a)

If Not HasModuleEvent(Trim(a(i)), "Completed") Then

ModulePrereqsMet = False: Exit Function

End If

Next i

ModulePrereqsMet = True

End Function

Function HasModuleEvent(moduleID As String, evt As String) As Boolean

Dim ews As Worksheet: Set ews = WS("Events")

Dim last As Long: last = ews.Cells(ews.Rows.Count, 1).End(xlUp).Row

Dim i As Long

For i = 2 To last

If ews.Cells(i, 4).Value = evt And ews.Cells(i, 5).Value = moduleID Then

HasModuleEvent = True: Exit Function

End If

Next i

HasModuleEvent = False

End Function

### Scenario lifecycle

vba Function StartScenario(scenarioID As String) As Boolean

Dim srow As Range

Set srow = WS("ScenarioBook").Columns(1).Find(scenarioID, LookIn:=xlValues, lookat:=xlWhole)

If srow Is Nothing Then MsgBox "Scenario not found.", vbExclamation: Exit Function

Dim moduleID As String: moduleID = srow.Offset(0, 1).Value

If Not TeacherReady() Then MsgBox "Teacher station not ready.", vbExclamation: Exit Function

If Not StudentsReady() Then MsgBox "Insufficient student stations.", vbExclamation: Exit Function

If Not SafetyReady() Then MsgBox "Safety checklist incomplete.", vbExclamation: Exit Function

If Not AssetsForModuleReady(moduleID) Then MsgBox "Required assets unavailable.", vbExclamation: Exit Function

If Not ModulePrereqsMet(moduleID) Then MsgBox "Module prerequisites not met.", vbExclamation: Exit Function

' Lock devices (simple status change to "InUse")

Call LockModuleAssets(moduleID, True)

LogEvent scenarioID, "Started", moduleID, "", "Scenario initiated"

StartScenario = True

End Function

Sub LockModuleAssets(moduleID As String, lockOn As Boolean)

Dim req As String: req = GetModuleField(moduleID, "RequiredAssets")

If Len(Trim(req)) = 0 Then Exit Sub

Dim arr() As String: arr = Split(req, ",")

Dim i As Long, r As Range

For i = LBound(arr) To UBound(arr)

Set r = WS("DeviceRegistry").Columns(1).Find(Trim(arr(i)), LookIn:=xlValues, lookat:=xlWhole)

If Not r Is Nothing Then

r.Offset(0, 6).Value = IIf(lockOn, "InUse", "Online")

End If

Next i

End Sub

## Scenario step dispatchers

### CIMSIM conveyor: start/stop, sensor events, sort logic

Sub CIMSIM\_RunStep(scenarioID As String, stepName As String, Optional param As String = "")

Select Case LCase(stepName)

Case "motor\_start"

LogEvent scenarioID, "Action", "CIMSIM", "MotorStart", "DC motor 12V enabled"

Case "motor\_stop"

LogEvent scenarioID, "Action", "CIMSIM", "MotorStop", "DC motor disabled"

Case "read\_ir"

' Simulated detection: param can be "present"/"absent"

LogEvent scenarioID, "Sensor", "IR", param, "Object " & param

Case "read\_rgb"

' param e.g., "R/G/B"

LogEvent scenarioID, "Sensor", "RGB", param, "Color sensed"

Case "sort\_defect"

LogEvent scenarioID, "Control", "Actuator", "Diverter", "Defect diverted"

Case "plc\_status"

' Simulate PLC I/O scan

LogEvent scenarioID, "PLC", "Scan", "OK", "Inputs/Outputs nominal"

Case Else

LogEvent scenarioID, "Warning", "UnknownStep", stepName, "No-op"

End Select

End Sub

IoT MQTT switch-to-lamp simulation (on one workstation) Sub IoT\_RunStep(scenarioID As String, stepName As String, Optional payload As String = "")

Dim topic As String: topic = Cfg("MQTT\_Topic\_OnOff", "OnOff")

Static lampState As String

Select Case LCase(stepName)

Case "publish\_switch"

' payload "ON"/"OFF"

LogEvent scenarioID, "MQTT-PUB", topic, payload, "Switch state published"

Case "subscribe\_lamp"

lampState = payload

LogEvent scenarioID, "MQTT-SUB", topic, lampState, "Lamp updated"

RecordMetric scenarioID, "LampState", IIf(lampState = "ON", 1, 0), "state", "DevIoT"

Case "heartbeat"

LogEvent scenarioID, "DevIoT", "Heartbeat", "OK", "Device alive"

Case Else

LogEvent scenarioID, "Warning", "UnknownStep", stepName, "No-op"

End Select

End Sub

### DATA (Spark-like) learning outcomes — rubric and placeholder metrics

Sub DATA\_RunStep(scenarioID As String, stepName As String, Optional param As String = "")

Select Case LCase(stepName)

Case "load\_dataset"

LogEvent scenarioID, "Data", "Load", param, "Dataset loaded"

Case "fit\_model"

' param e.g., "Regression/Clustering"

LogEvent scenarioID, "ML", "Model", param, "Model fitted"

RecordMetric scenarioID, "Accuracy", 0.82, "ratio", "MLlib-Sim"

Case "evaluate"

RecordMetric scenarioID, "AUC", 0.75, "ratio", "MLlib-Sim"

LogEvent scenarioID, "Eval", "Metrics", "AUC=0.75", "Evaluation complete"

Case Else

LogEvent scenarioID, "Warning", "UnknownStep", stepName, ""

End Select

End Sub

### CYBER — safe, controlled, in-lab simulations only

vba

Sub CYBER\_RunStep(scenarioID As String, stepName As String, Optional param As String = "")

Select Case LCase(stepName)

Case "arp\_demo"

LogEvent scenarioID, "NetSim", "ARP\_Table", "Updated", "Isolated lab demo"

Case "vpn\_config"

LogEvent scenarioID, "Security", "VPN", "Configured", "Tunneled segment in lab"

Case "firewall\_rules"

LogEvent scenarioID, "Security", "Firewall", "Applied", "Ruleset enforced"

Case Else

LogEvent scenarioID, "Warning", "UnknownStep", stepName, ""

End Select

End Sub

## Scenario runner and evaluation

vba

Sub RunScenarioPrompt()

Dim sid As String: sid = InputBox("Enter ScenarioID:")

If Len(sid) = 0 Then Exit Sub

If Not StartScenario(sid) Then Exit Sub

' Fetch steps as CSV from ScenarioBook

Dim r As Range: Set r = WS("ScenarioBook").Columns(1).Find(sid, LookIn:=xlValues, lookat:=xlWhole)

Dim moduleID As String: moduleID = r.Offset(0, 1).Value

Dim stepsCSV As String: stepsCSV = CStr(r.Offset(0, 4).Value)

Dim steps() As String: steps = Split(stepsCSV, ",")

Dim i As Long

For i = LBound(steps) To UBound(steps)

Call DispatchStep(sid, moduleID, Trim(steps(i)))

Next i

EvaluateScenario sid

LockModuleAssets moduleID, False

End Sub

Sub DispatchStep(scenarioID As String, moduleID As String, stepToken As String)

Dim parts() As String: parts = Split(stepToken, ":")

Dim stepName As String: stepName = parts(0)

Dim param As String: If UBound(parts) >= 1 Then param = parts(1) Else param = ""

Select Case UCase(moduleID)

Case "CIMSIM": CIMSIM\_RunStep scenarioID, stepName, param

Case "IOT": IoT\_RunStep scenarioID, stepName, param

Case "DATA": DATA\_RunStep scenarioID, stepName, param

Case "CYBER": CYBER\_RunStep scenarioID, stepName, param

Case Else: LogEvent scenarioID, "Warning", "UnknownModule", moduleID, ""

End Select

End Sub

Sub EvaluateScenario(scenarioID As String)

' Generic pass criteria parser: e.g., "LampState==1;Accuracy>=0.8"

Dim r As Range: Set r = WS("ScenarioBook").Columns(1).Find(scenarioID, LookIn:=xlValues, lookat:=xlWhole)

Dim criteria As String: criteria = CStr(r.Offset(0, 5).Value)

Dim tokens() As String: tokens = Split(criteria, ";")

Dim passAll As Boolean: passAll = True

Dim i As Long

For i = LBound(tokens) To UBound(tokens)

If Len(Trim(tokens(i))) > 0 Then

If Not CriterionMet(scenarioID, Trim(tokens(i))) Then passAll = False

End If

Next i

LogEvent scenarioID, IIf(passAll, "Completed", "Failed"), "", "", "Evaluation " & IIf(passAll, "passed", "failed")

GenerateScenarioReport scenarioID

End Sub

Function CriterionMet(scenarioID As String, expr As String) As Boolean

' Supports forms like Metric>=value or Metric==value

Dim op As String

If InStr(expr, ">=") > 0 Then op = ">=" \_

ElseIf InStr(expr, "<=") > 0 Then op = "<=" \_

ElseIf InStr(expr, "==") > 0 Then op = "==" \_

ElseIf InStr(expr, ">") > 0 Then op = ">" \_

ElseIf InStr(expr, "<") > 0 Then op = "<" Else op = ""

If op = "" Then CriterionMet = False: Exit Function

Dim parts() As String: parts = Split(expr, op)

Dim metric As String: metric = Trim(parts(0))

Dim target As Double: target = CDbl(Val(Trim(parts(1))))

Dim val As Double: val = LatestMetric(scenarioID, metric)

Select Case op

Case ">=": CriterionMet = (val >= target)

Case "<=": CriterionMet = (val <= target)

Case "==": CriterionMet = (Abs(val - target) < 0.0001)

Case ">": CriterionMet = (val > target)

Case "<": CriterionMet = (val < target)

Case Else: CriterionMet = False

End Select

End Function

Function LatestMetric(scenarioID As String, metric As String) As Double

Dim ws As Worksheet: Set ws = WS("Measurements")

Dim last As Long: last = ws.Cells(ws.Rows.Count, 1).End(xlUp).Row

Dim i As Long

For i = last To 2 Step -1

If ws.Cells(i, 1).Value = scenarioID And ws.Cells(i, 2).Value = metric Then

LatestMetric = CDbl(ws.Cells(i, 3).Value): Exit Function

End If

Next i

LatestMetric = 0

End Function

One-click evidence report ub GenerateScenarioReport(scenarioID As String)

On Error Resume Next: Application.DisplayAlerts = False

ThisWorkbook.Worksheets("Portfolio").Delete

Application.DisplayAlerts = True: On Error GoTo 0

Dim wr As Worksheet: Set wr = ThisWorkbook.Worksheets.Add

wr.Name = "Portfolio"

Dim sb As Worksheet: Set sb = WS("ScenarioBook")

Dim r As Range: Set r = sb.Columns(1).Find(scenarioID, LookIn:=xlValues, lookat:=xlWhole)

Dim row As Long: row = 1

wr.Cells(row, 1).Value = "DL NGL Scenario Evidence": row = row + 2

wr.Cells(row, 1).Value = "ScenarioID": wr.Cells(row, 2).Value = scenarioID: row = row + 1

wr.Cells(row, 1).Value = "Module": wr.Cells(row, 2).Value = r.Offset(0, 1).Value: row = row + 1

wr.Cells(row, 1).Value = "Name": wr.Cells(row, 2).Value = r.Offset(0, 2).Value: row = row + 1

wr.Cells(row, 1).Value = "Objective": wr.Cells(row, 2).Value = r.Offset(0, 3).Value: row = row + 2

row = CopySectionTable(wr, row, "Events", WS("Events"), 3, scenarioID)

row = CopySectionTable(wr, row, "Measurements", WS("Measurements"), 1, scenarioID)

wr.Columns.AutoFit

Dim outDir As String: outDir = Cfg("EvidenceDir", ThisWorkbook.Path)

Dim f As String: f = outDir & "\Evidence\_" & scenarioID & ".pdf"

On Error Resume Next

wr.ExportAsFixedFormat xlTypePDF, f

On Error GoTo 0

End Sub

Function CopySectionTable(dst As Worksheet, startRow As Long, title As String, src As Worksheet, matchCol As Integer, key As String) As Long

dst.Cells(startRow, 1).Value = title

Dim r As Long: r = startRow + 1

Dim rng As Range: Set rng = src.Range("A1").CurrentRegion

Dim i As Long, wroteHeader As Boolean

For i = 2 To rng.Rows.Count

If CStr(rng.Cells(i, matchCol).Value) = key Then

If Not wroteHeader Then

rng.Rows(1).Copy dst.Cells(r, 1): r = r + 1

wroteHeader = True

End If

rng.Rows(i).Copy dst.Cells(r, 1): r = r + 1

End If

Next i

CopySectionTable = r + 1

End Function

## How to populate and ru

* Config
  + CurrentUser = Tshingombe Fiston Tshitadi
  + MinStudents = 1
  + RequireDL\_WORKSPACE = TRUE
  + MQTT\_Topic\_OnOff = OnOff
  + EvidenceDir = C:\Evidence (or your path)
* StationRegistry
  + One Teacher row: Status=Online, DL\_WORKSPACE=Yes
  + One or more Student rows: Status=Online
* DeviceRegistry
  + PLC\_1 (PLC, …, Status=Online)
  + DevIoT\_1 (DevIoT, Protocols=MQTT, AssignedStation=Student1, Status=Online)
  + Sensors/Actuators as needed (IR\_1, RGB\_1, Lamp\_1)
* ModuleCatalog
  + CIMSIM: RequiredAssets = PLC\_1,IR\_1,RGB\_1
  + IOT: RequiredAssets = DevIoT\_1,Lamp\_1
  + DATA: RequiredAssets = (blank)
  + CYBER: RequiredAssets = (blank)
  + Set Enabled=TRUE
* ScenarioBook (examples)
  + SCN\_CIM\_01 | CIMSIM | Conveyor Sort | Detect and divert colored object | motor\_start,read\_ir:present,read\_rgb:R,sort\_defect,motor\_stop | (no strict criteria)
  + SCN\_IOT\_01 | IOT | Switch-Lamp via MQTT | Switch controls remote lamp | publish\_switch:ON,subscribe\_lamp:ON | LampState==1
  + SCN\_DATA\_01 | DATA | Fit Model | Train and evaluate | load\_dataset:Energy,fit\_model:Regression,evaluate | Accuracy>=0.8;AUC>=0.7

Run with RunScenarioPrompt, enter ScenarioID (e.g., SCN\_IOT\_01). The engine runs steps, evaluates pass criteria, and exports a Portfolio PDF with Events and Measurements.

## Extensions you can add next

* DL WORKSPACE handshake: add a ping cell per station to confirm IDE availability.
* PLC structured I/O: a sheet for I/O maps and ladder-logic state snapshots appended to Events.
* SCADA view: a simple dashboard that turns lamp state, conveyor status, and sensor flags into shapes/colors.
* QCTO/SAQA mapping: add a sheet OutcomesMap linking ScenarioID → Specific Outcomes → Evidence rows, and append it to the Portfolio export.
* Assessment rubrics: add a Rubrics sheet (Criteria, Weight, EvidenceQuery) and compute a competency score per scenario.

If you want, I can package this into ready-to-import .bas modules and a starter workbook with sample rows prefilled for CIMSIM and IoT.

# VBA logigram and algorigram for multi-domain DL labs (IoT, ignition DM91, water hammer, HVAC, chemical, renewable)

You’re building a serious, audit-ready lab spine. Below is a modular Excel VBA engine that enforces readiness (logigram), runs domain scenarios (algorigram), captures evidence, and exports a single portfolio artifact. It covers IoT (MQTT), ignition system DM91, water hammer (hydraulics), HVAC air treatment, chemical processes, and renewable energy (PV/wind/hybrid).

## Workbook schema

Create sheets exactly as named; columns are referenced by code.

* StationRegistry
  + A: StationID, B: Role (Teacher/Student), C: Hostname, D: IP, E: Link (LAN/WiFi), F: Status (Online/Offline), G: WorkspaceOK (Yes/No)
* DeviceRegistry
  + A: DeviceID, B: Domain (IoT/DM91/HYD/HVAC/CHEM/REN), C: Type, D: Model, E: Protocols, F: AssignedStation, G: Status (Online/InUse/Available)
* ScenarioBook
  + A: ScenarioID, B: Domain, C: Name, D: Objective, E: StepsCSV, F: PassCriteria, G: RequiredDevicesCSV, H: SafetyChecklistCSV
* Safety
  + A: Item, B: Required (TRUE/FALSE), C: Completed (TRUE/FALSE), D: Notes
* Measurements
  + A: ScenarioID, B: Metric, C: Value, D: Unit, E: Source, F: Timestamp
* Events
  + A: Timestamp, B: User, C: ScenarioID, D: EventType, E: K1, F: K2, G: Notes
* Config
  + A: Key, B: Value
  + Keys: CurrentUser, MinStudents, RequireWorkspace, EvidenceDir, MQTT\_Topic, DM91\_SparkThreshold\_kV, WaterHammer\_MaxBar, HVAC\_TempSet\_C, PV\_STC\_W, Wind\_Rated\_W
* Portfolio
  + Generated automatically (no manual columns)

Tips:

* StepsCSV uses tokens like domain\_step:parameter, e.g., “iot\_publish:ON, iot\_subscribe:ON”.
* PassCriteria uses semicolon-separated expressions, e.g., “LampState==1;PeakPressureBar<=8”.

## Logigram rules

* Infrastructure readiness:
  + Teacher station Online and (if required) WorkspaceOK=Yes.
  + Students Online ≥ MinStudents.
* Safety gate:
  + Every Safety item listed in ScenarioBook.H (if Required) must be Completed.
* Device availability:
  + All ScenarioBook.G devices found in DeviceRegistry and Status=Online/Available.
* Domain-specific prechecks:
  + IoT: MQTT topic configured.
  + DM91: Lab power OK, panel interlocks ready.
  + Hydraulics: Bench present, compressed air available device flag.
  + HVAC: Sensor calibration present (temp/RH/anemometer).
  + Chemical: Reactor sensors online (T, flow, cond).
  + Renewable: PV/wind emulators online or outdoor flag set.

## Core utilities and orchestration

vba

Option Explicit

Function WS(name As String) As Worksheet: Set WS = ThisWorkbook.Worksheets(name): End Function

Function Cfg(key As String, Optional defVal As Variant = "") As Variant

Dim r As Range: Set r = WS("Config").Columns(1).Find(key, , xlValues, xlWhole)

If r Is Nothing Then Cfg = defVal Else Cfg = IIf(IsEmpty(r.Offset(0, 1)), defVal, r.Offset(0, 1))

End Function

Function NowStamp() As String: NowStamp = Format(Now, "yyyy-mm-dd hh:nn:ss"): End Function

Sub LogEvent(scn As String, evt As String, Optional k1 As String = "", Optional k2 As String = "", Optional note As String = "")

Dim ws As Worksheet: Set ws = WS("Events")

Dim r As Long: r = ws.Cells(ws.Rows.Count, 1).End(xlUp).Row + 1

ws.Cells(r, 1) = NowStamp(): ws.Cells(r, 2) = Cfg("CurrentUser", "Learner")

ws.Cells(r, 3) = scn: ws.Cells(r, 4) = evt: ws.Cells(r, 5) = k1: ws.Cells(r, 6) = k2: ws.Cells(r, 7) = note

End Sub

Sub RecordMetric(scn As String, metric As String, val As Double, unitStr As String, src As String)

Dim ws As Worksheet: Set ws = WS("Measurements")

Dim r As Long: r = ws.Cells(ws.Rows.Count, 1).End(xlUp).Row + 1

ws.Cells(r, 1) = scn: ws.Cells(r, 2) = metric: ws.Cells(r, 3) = val

ws.Cells(r, 4) = unitStr: ws.Cells(r, 5) = src: ws.Cells(r, 6) = NowStamp()

End Sub

### Readiness checks and locks

vba

Function TeacherReady() As Boolean

Dim f As Range: Set f = WS("StationRegistry").Range("A1").CurrentRegion

Dim i As Long, ok As Boolean

For i = 2 To f.Rows.Count

If LCase(f.Cells(i, 2)) = "teacher" And LCase(f.Cells(i, 6)) = "online" Then

If CBool(Cfg("RequireWorkspace", True)) Then

If LCase(f.Cells(i, 7)) = "yes" Then ok = True

Else

ok = True

End If

End If

Next i

TeacherReady = ok

End Function

Function StudentsReady() As Boolean

Dim f As Range: Set f = WS("StationRegistry").Range("A1").CurrentRegion

Dim i As Long, c As Long, need As Long: need = CLng(Cfg("MinStudents", 1))

For i = 2 To f.Rows.Count

If LCase(f.Cells(i, 2)) = "student" And LCase(f.Cells(i, 6)) = "online" Then c = c + 1

Next i

StudentsReady = (c >= need)

End Function

Function SafetyReady(listCSV As String) As Boolean

If Len(Trim(listCSV)) = 0 Then SafetyReady = True: Exit Function

Dim a() As String: a = Split(listCSV, ",")

Dim i As Long, item As String, r As Range

For i = LBound(a) To UBound(a)

item = Trim(a(i))

Set r = WS("Safety").Columns(1).Find(item, , xlValues, xlWhole)

If r Is Nothing Then SafetyReady = False: Exit Function

If CBool(r.Offset(0, 1)) = True And CBool(r.Offset(0, 2)) = False Then SafetyReady = False: Exit Function

Next i

SafetyReady = True

End Function

Function DevicesReady(reqCSV As String) As Boolean

If Len(Trim(reqCSV)) = 0 Then DevicesReady = True: Exit Function

Dim a() As String: a = Split(reqCSV, ",")

Dim i As Long, r As Range

For i = LBound(a) To UBound(a)

Set r = WS("DeviceRegistry").Columns(1).Find(Trim(a(i)), , xlValues, xlWhole)

If r Is Nothing Then DevicesReady = False: Exit Function

If LCase(r.Offset(0, 6)) <> "online" And LCase(r.Offset(0, 6)) <> "available" Then DevicesReady = False: Exit Function

Next i

DevicesReady = True

End Function

Sub LockDevices(reqCSV As String, lockOn As Boolean)

If Len(Trim(reqCSV)) = 0 Then Exit Sub

Dim a() As String: a = Split(reqCSV, ",")

Dim i As Long, r As Range

For i = LBound(a) To UBound(a)

Set r = WS("DeviceRegistry").Columns(1).Find(Trim(a(i)), , xlValues, xlWhole)

If Not r Is Nothing Then r.Offset(0, 6) = IIf(lockOn, "InUse", "Online")

Next i

End Sub

### Scenario runner and evaluator

Function StartScenario(scn As String) As Boolean

Dim sb As Worksheet: Set sb = WS("ScenarioBook")

Dim s As Range: Set s = sb.Columns(1).Find(scn, , xlValues, xlWhole)

If s Is Nothing Then MsgBox "Scenario not found", vbExclamation: Exit Function

If Not TeacherReady() Then MsgBox "Teacher station not ready", vbExclamation: Exit Function

If Not StudentsReady() Then MsgBox "Insufficient student stations", vbExclamation: Exit Function

If Not SafetyReady(CStr(s.Offset(0, 7).Value)) Then MsgBox "Safety checklist incomplete", vbExclamation: Exit Function

If Not DevicesReady(CStr(s.Offset(0, 6).Value)) Then MsgBox "Devices unavailable", vbExclamation: Exit Function

LockDevices CStr(s.Offset(0, 6).Value), True

LogEvent scn, "Started", CStr(s.Offset(0, 1).Value), "", "Scenario initiated"

StartScenario = True

End Function

Sub RunScenarioPrompt()

Dim scn As String: scn = InputBox("ScenarioID to run:")

If Len(scn) = 0 Then Exit Sub

If Not StartScenario(scn) Then Exit Sub

Dim s As Range: Set s = WS("ScenarioBook").Columns(1).Find(scn, , xlValues, xlWhole)

Dim domain As String: domain = UCase(CStr(s.Offset(0, 1).Value))

Dim steps() As String: steps = Split(CStr(s.Offset(0, 4).Value), ",")

Dim i As Long

For i = LBound(steps) To UBound(steps)

DispatchStep scn, domain, Trim(steps(i))

Next i

EvaluateScenario scn

LockDevices CStr(s.Offset(0, 6).Value), False

End Sub

Sub EvaluateScenario(scn As String)

Dim s As Range: Set s = WS("ScenarioBook").Columns(1).Find(scn, , xlValues, xlWhole)

Dim crit As String: crit = CStr(s.Offset(0, 5).Value)

Dim tokens() As String: tokens = Split(crit, ";")

Dim ok As Boolean: ok = True

Dim i As Long

For i = LBound(tokens) To UBound(tokens)

If Len(Trim(tokens(i))) > 0 Then If Not CriterionMet(scn, Trim(tokens(i))) Then ok = False

Next i

LogEvent scn, IIf(ok, "Completed", "Failed"), "", "", IIf(ok, "Pass", "Fail")

GenerateScenarioReport scn

End Sub

Function CriterionMet(scn As String, expr As String) As Boolean

Dim op As String

If InStr(expr, ">=") > 0 Then op = ">=" ElseIf InStr(expr, "<=") > 0 Then op = "<=" \_

ElseIf InStr(expr, "==") > 0 Then op = "==" ElseIf InStr(expr, ">") > 0 Then op = ">" \_

ElseIf InStr(expr, "<") > 0 Then op = "<"

If Len(op) = 0 Then CriterionMet = False: Exit Function

Dim parts() As String: parts = Split(expr, op)

Dim metric As String: metric = Trim(parts(0))

Dim target As Double: target = CDbl(Val(Trim(parts(1))))

Dim val As Double: val = LatestMetric(scn, metric)

Select Case op

Case ">=": CriterionMet = (val >= target)

Case "<=": CriterionMet = (val <= target)

Case "==": CriterionMet = (Abs(val - target) < 0.0001)

Case ">": CriterionMet = (val > target)

Case "<": CriterionMet = (val < target)

Case Else: CriterionMet = False

End Select

End Function

Function LatestMetric(scn As String, metric As String) As Double

Dim ws As Worksheet: Set ws = WS("Measurements")

Dim last As Long: last = ws.Cells(ws.Rows.Count, 1).End(xlUp).Row

Dim i As Long

For i = last To 2 Step -1

If ws.Cells(i, 1) = scn And ws.Cells(i, 2) = metric Then LatestMetric = CDbl(ws.Cells(i, 3)): Exit Function

Next i

LatestMetric = 0

End Function

## Domain step dispatchers

### IoT (MQTT, sensors, actuators)

Sub DispatchStep(scn As String, domain As String, token As String)

Dim stepName As String, param As String, p()

p = Split(token, ":"): stepName = LCase(p(0)): If UBound(p) >= 1 Then param = p(1) Else param = ""

Select Case domain

Case "IOT": IOT\_Step scn, stepName, param

Case "DM91": DM91\_Step scn, stepName, param

Case "HYD": HYD\_Step scn, stepName, param

Case "HVAC": HVAC\_Step scn, stepName, param

Case "CHEM": CHEM\_Step scn, stepName, param

Case "REN": REN\_Step scn, stepName, param

Case Else: LogEvent scn, "Warn", "UnknownDomain", domain, token

End Select

End Sub

Sub IOT\_Step(scn As String, stepName As String, param As String)

Dim topic As String: topic = Cfg("MQTT\_Topic", "OnOff")

Static lampState As String

Select Case stepName

Case "publish": LogEvent scn, "MQTT-PUB", topic, param, "Switch state"

Case "subscribe": lampState = param: LogEvent scn, "MQTT-SUB", topic, lampState, "Lamp update"

RecordMetric scn, "LampState", IIf(lampState = "ON", 1, 0), "state", "DevIoT"

Case "sensor\_temp": RecordMetric scn, "TempC", Val(param), "C", "PT100"

Case "sensor\_hr": RecordMetric scn, "HeartRate", Val(param), "bpm", "HR"

Case "actuator\_pwm": LogEvent scn, "Actuator", "PWM", param, "Motor drive"

Case Else: LogEvent scn, "IOT-Unknown", stepName, param, ""

End Select

End Sub

### DM91 ignition system panel (faults, signals)

Sub DM91\_Step(scn As String, stepName As String, param As String)

' param examples: system=Hall/magnetic/optical/COP; fault=OpenCoil/Misfire/SensorLoss

Select Case stepName

Case "select\_system": LogEvent scn, "DM91", "System", param, "Ignition topology selected"

Case "inject\_fault": LogEvent scn, "DM91", "FaultSet", param, "Fault injected"

Case "clear\_fault": LogEvent scn, "DM91", "FaultClear", param, "Fault cleared"

Case "measure\_spark"

Dim kv As Double: kv = Val(param)

RecordMetric scn, "Spark\_kV", kv, "kV", "Scope"

RecordMetric scn, "Spark\_OK", IIf(kv >= CDbl(Cfg("DM91\_SparkThreshold\_kV", 12)), 1, 0), "bool", "Derived"

Case "measure\_rpm": RecordMetric scn, "EngineRPM", Val(param), "rpm", "Tach"

Case Else: LogEvent scn, "DM91-Unknown", stepName, param, ""

End Select

End Sub

### Water hammer trainer (hydraulics)

Sub HYD\_Step(scn As String, stepName As String, param As String)

' param may hold numeric values or tags like 'fast\_close'

Select Case stepName

Case "set\_valve": LogEvent scn, "HYD", "Valve", param, "Position set"

Case "pulse\_close": LogEvent scn, "HYD", "Valve", "FastClose", "Transient initiated"

Case "measure\_p\_peekbar"

Dim pb As Double: pb = Val(param)

RecordMetric scn, "PeakPressureBar", pb, "bar", "Transducer"

RecordMetric scn, "WH\_Pass", IIf(pb <= CDbl(Cfg("WaterHammer\_MaxBar", 8)), 1, 0), "bool", "Derived"

Case "measure\_celerity": RecordMetric scn, "Celerity\_ms", Val(param), "m/s", "Derived"

Case "surge\_tank"

RecordMetric scn, "SurgeDecayTau\_s", Val(param), "s", "Fit"

Case Else: LogEvent scn, "HYD-Unknown", stepName, param, ""

End Select

End Sub

### HVAC air treatment (cool/heat/humidify/flow)

Sub HVAC\_Step(scn As String, stepName As String, param As String)

Select Case stepName

Case "set\_temp": LogEvent scn, "HVAC", "SetpointC", param, "Controller set"

Case "measure\_temp": RecordMetric scn, "AirTempC", Val(param), "C", "Sensor"

Case "measure\_rh": RecordMetric scn, "RHpct", Val(param), "%", "Sensor"

Case "measure\_flow": RecordMetric scn, "Airflow\_m3h", Val(param), "m3/h", "Anemometer"

Case "coil\_state": LogEvent scn, "HVAC", "Coil", param, "Cooling/Heating/Humidifying"

Case "efficiency"

' param e.g., COP=3.1

Dim v As Double: v = Val(Split(param, "=")(1))

RecordMetric scn, "COP", v, "ratio", "Computed"

Case Else: LogEvent scn, "HVAC-Unknown", stepName, param, ""

End Select

End Sub

### Chemical process (reactor)

vba

Sub CHEM\_Step(scn As String, stepName As String, param As String)

Select Case stepName

Case "set\_flow": LogEvent scn, "CHEM", "FlowSet\_Lpm", param, "Feed set"

Case "measure\_temp": RecordMetric scn, "ReactorTempC", Val(param), "C", "PT100"

Case "measure\_cond": RecordMetric scn, "Conductivity\_mScm", Val(param), "mS/cm", "Probe"

Case "convert\_yield": RecordMetric scn, "Yield\_pct", Val(param), "%", "Analysis"

Case "pid\_tune": LogEvent scn, "CHEM", "PID", param, "Controller tuned"

Case Else: LogEvent scn, "CHEM-Unknown", stepName, param, ""

End Select

End Sub

### Renewable energy (PV/wind/hybrid)

vba

Sub REN\_Step(scn As String, stepName As String, param As String)

Select Case stepName

Case "pv\_set\_irr": LogEvent scn, "REN", "Irradiance\_Wm2", param, "Emulator set"

Case "pv\_measure"

Dim p As Double: p = Val(param)

RecordMetric scn, "PV\_Power\_W", p, "W", "PV-Emu"

RecordMetric scn, "PV\_Ratio", p / CDbl(Cfg("PV\_STC\_W", 200)), "ratio", "Derived"

Case "wind\_set\_speed": LogEvent scn, "REN", "WindSpeed\_ms", param, "Tunnel set"

Case "wind\_measure"

Dim w As Double: w = Val(param)

RecordMetric scn, "Wind\_Power\_W", w, "W", "WT-Emu"

RecordMetric scn, "Wind\_Ratio", w / CDbl(Cfg("Wind\_Rated\_W", 300)), "ratio", "Derived"

Case "hybrid\_soc": RecordMetric scn, "Battery\_SOC\_pct", Val(param), "%", "BMS"

Case "grid\_sync": LogEvent scn, "REN", "GridSync", param, "Inverter sync"

Case Else: LogEvent scn, "REN-Unknown", stepName, param, ""

End Select

End Sub

## Evidence report and quick-start

### One-click portfolio export

vba

Sub GenerateScenarioReport(scn As String)

On Error Resume Next: Application.DisplayAlerts = False

ThisWorkbook.Worksheets("Portfolio").Delete

Application.DisplayAlerts = True: On Error GoTo 0

Dim wr As Worksheet: Set wr = ThisWorkbook.Worksheets.Add

wr.Name = "Portfolio"

Dim s As Range: Set s = WS("ScenarioBook").Columns(1).Find(scn, , xlValues, xlWhole)

Dim r As Long: r = 1

wr.Cells(r, 1) = "Scenario Evidence": r = r + 2

wr.Cells(r, 1) = "ScenarioID": wr.Cells(r, 2) = scn: r = r + 1

wr.Cells(r, 1) = "Domain": wr.Cells(r, 2) = s.Offset(0, 1): r = r + 1

wr.Cells(r, 1) = "Name": wr.Cells(r, 2) = s.Offset(0, 2): r = r + 1

wr.Cells(r, 1) = "Objective": wr.Cells(r, 2) = s.Offset(0, 3): r = r + 2

r = CopySection(wr, r, "Events", WS("Events"), 3, scn)

r = CopySection(wr, r, "Measurements", WS("Measurements"), 1, scn)

wr.Columns.AutoFit

Dim outDir As String: outDir = Cfg("EvidenceDir", ThisWorkbook.Path)

Dim f As String: f = outDir & "\Evidence\_" & scn & ".pdf"

On Error Resume Next: wr.ExportAsFixedFormat xlTypePDF, f: On Error GoTo 0

MsgBox "Portfolio generated: " & f, vbInformation

End Sub

Function CopySection(dst As Worksheet, startRow As Long, title As String, src As Worksheet, matchCol As Integer, key As String) As Long

dst.Cells(startRow, 1) = title

Dim r As Long: r = startRow + 1, rng As Range, i As Long, header As Boolean

Set rng = src.Range("A1").CurrentRegion

For i = 2 To rng.Rows.Count

If CStr(rng.Cells(i, matchCol)) = key Then

If Not header Then rng.Rows(1).Copy dst.Cells(r, 1): r = r + 1: header = True

rng.Rows(i).Copy dst.Cells(r, 1): r = r + 1

End If

Next i

CopySection = r + 1

End Function

### Populate and run

### Populate and run

* Config:
  + CurrentUser = Tshingombe Fiston Tshitadi
  + MinStudents = 1
  + RequireWorkspace = TRUE
  + EvidenceDir = C:\Evidence
  + MQTT\_Topic = OnOff
  + DM91\_SparkThreshold\_kV = 12
  + WaterHammer\_MaxBar = 8
  + HVAC\_TempSet\_C = 22
  + PV\_STC\_W = 200, Wind\_Rated\_W = 300
* ScenarioBook examples:
  + SCN\_IOT\_01 | IOT | Switch-to-Lamp | Control lamp via MQTT | iot\_publish:ON, iot\_subscribe:ON | LampState==1 | DevIoT\_1,Lamp\_1 | PPE,EmergencyStop
  + SCN\_DM91\_01 | DM91 | Misfire diagnosis | Detect low spark | select\_system:Hall, inject\_fault:Misfire, measure\_spark:9.5 | Spark\_OK==1 | DM91\_Panel,Scope
  + SCN\_HYD\_01 | HYD | Water hammer safe op | Limit peak | pulse\_close:fast, measure\_p\_peekbar:6.8 | PeakPressureBar<=8 | Transducer\_A,Bench,AirSupply
  + SCN\_HVAC\_01 | HVAC | Setpoint tracking | Comfort band | set\_temp:22, coil\_state:cool, measure\_temp:22.3, measure\_rh:48 | AirTempC>=21;AirTempC<=23 | TempSensor,RHSensor
  + SCN\_REN\_01 | REN | PV curve point | Ratio check | pv\_set\_irr:800, pv\_measure:165 | PV\_Ratio>=0.7 | PV\_Emu
* Run:
  + Press RunScenarioPrompt and enter ScenarioID.
  + The engine validates readiness (logigram), executes steps (algorigram), evaluates criteria, and exports a Portfolio PDF.

If you want, I can add:

* Rubrics and SAQA/QCTO outcome mapping for each ScenarioID.
* Sensor calibration sheets and drift checks per domain.
* Charts on Portfolio (pressure surge, HVAC step response, PV IV points) for visual evidence.

## Workbook schema

Create sheets exactly as named; columns are referenced by the code.

* TraineeProfile
  + A: Field, B: Value (Name, ID, Trade Level, Workplace, Assessor)
* Modules
  + A: ModuleID, B: ModuleName, C: Sequence, D: Required (TRUE/FALSE), E: Domain (ELEC/ROAD/FIRE/AID), F: Enabled (TRUE/FALSE)
* Safety
  + A: ChecklistItem, B: Required (TRUE/FALSE), C: Completed (TRUE/FALSE), D: Domain, E: Notes
* Exercises
  + A: ExerciseID, B: ModuleID, C: Name, D: Objective, E: StepsCSV, F: PassCriteria, G: RequiredToolsCSV, H: RiskTagsCSV
* Tools
  + A: ToolID, B: Name, C: Status (Available/InUse/Out), D: CalDueDate, E: Domain
* Events
  + A: Timestamp, B: User, C: ExerciseID, D: EventType, E: K1, F: K2, G: Notes
* Measurements
  + A: ExerciseID, B: Metric, C: Value, D: Unit, E: Source, F: Timestamp
* Config
  + A: Key, B: Value
  + Keys: CurrentUser, MinPPE, EvidenceDir, Elec\_IsolationTimeout\_s, Fire\_MAX\_Risk, FirstAid\_TrainerPresent (TRUE/FALSE)
* Portfolio
  + Generated automatically

Named ranges (Config!B next to key):

* CurrentUser, MinPPE, EvidenceDir, Elec\_IsolationTimeout\_s, Fire\_MAX\_Risk, FirstAid\_TrainerPresent

## Logigram rules

* **Infrastructure:** Module must be Enabled; all prior Required modules by Sequence completed.
* **Safety:** Required Safety items for the module’s Domain must be Completed before start.
* **Tools:** All RequiredToolsCSV items must be Status=Available and calibration date valid (if applicable).
* **Domain-specific:**
  + **ELEC:** Verified isolation/LOTO and zero-voltage check within Elec\_IsolationTimeout\_s.
  + **FIRE:** Risk tag sum must be ≤ Fire\_MAX\_Risk (or mitigations documented).
  + **AID:** FirstAid\_TrainerPresent must be TRUE for CPR simulation logging.

## Algorigram flows

* **StartExercise:** Validate module order → safety → tools → domain prechecks → lock tools → log Started.
* **RunStep:** Dispatch to domain handlers (ELEC/ROAD/FIRE/AID) → record metrics → add evidence events.
* **EvaluateExercise:** Check PassCriteria expressions against latest metrics → log Completed/Failed → export Portfolio.

## Core VBA utilities

vba

Option Explicit

Function WS(name As String) As Worksheet: Set WS = ThisWorkbook.Worksheets(name): End Function

Function Cfg(key As String, Optional defVal As Variant = "") As Variant

Dim r As Range: Set r = WS("Config").Columns(1).Find(key, , xlValues, xlWhole)

If r Is Nothing Then Cfg = defVal Else Cfg = IIf(IsEmpty(r.Offset(0, 1)), defVal, r.Offset(0, 1))

End Function

Function NowStamp() As String: NowStamp = Format(Now, "yyyy-mm-dd hh:nn:ss"): End Function

Sub LogEvent(exID As String, evt As String, Optional k1 As String = "", Optional k2 As String = "", Optional note As String = "")

Dim w As Worksheet: Set w = WS("Events")

Dim r As Long: r = w.Cells(w.Rows.Count, 1).End(xlUp).Row + 1

w.Cells(r, 1) = NowStamp(): w.Cells(r, 2) = Cfg("CurrentUser", "Trainee")

w.Cells(r, 3) = exID: w.Cells(r, 4) = evt: w.Cells(r, 5) = k1: w.Cells(r, 6) = k2: w.Cells(r, 7) = note

End Sub

Sub RecordMetric(exID As String, metric As String, val As Double, unitStr As String, src As String)

Dim w As Worksheet: Set w = WS("Measurements")

Dim r As Long: r = w.Cells(w.Rows.Count, 1).End(xlUp).Row + 1

w.Cells(r, 1) = exID: w.Cells(r, 2) = metric: w.Cells(r, 3) = val

w.Cells(r, 4) = unitStr: w.Cells(r, 5) = src: w.Cells(r, 6) = NowStamp()

End Sub

## Module order and readiness logigram

vba

Function ModuleEnabled(modID As String) As Boolean

Dim r As Range: Set r = WS("Modules").Columns(1).Find(modID, , xlValues, xlWhole)

If r Is Nothing Then ModuleEnabled = False Else ModuleEnabled = CBool(r.Offset(0, 5).Value)

End Function

Function ModuleDomain(modID As String) As String

Dim r As Range: Set r = WS("Modules").Columns(1).Find(modID, , xlValues, xlWhole)

If r Is Nothing Then ModuleDomain = "" Else ModuleDomain = CStr(r.Offset(0, 4).Value)

End Function

Function SequenceOf(modID As String) As Long

Dim r As Range: Set r = WS("Modules").Columns(1).Find(modID, , xlValues, xlWhole)

If r Is Nothing Then SequenceOf = 9999 Else SequenceOf = CLng(r.Offset(0, 2).Value)

End Function

Function IsModuleCompleted(modID As String) As Boolean

Dim e As Worksheet: Set e = WS("Events")

Dim last As Long: last = e.Cells(e.Rows.Count, 1).End(xlUp).Row

Dim i As Long

For i = last To 2 Step -1

If e.Cells(i, 4).Value = "ModuleCompleted" And e.Cells(i, 5).Value = modID Then IsModuleCompleted = True: Exit Function

Next i

IsModuleCompleted = False

End Function

Function PriorRequiredCompleted(modID As String) As Boolean

Dim m As Worksheet: Set m = WS("Modules")

Dim seq As Long: seq = SequenceOf(modID)

Dim last As Long: last = m.Cells(m.Rows.Count, 1).End(xlUp).Row

Dim i As Long

For i = 2 To last

If CBool(m.Cells(i, 3).Value) = True Then

If m.Cells(i, 3).Offset(0, -1).Value <> "" Then

If CLng(m.Cells(i, 3).Offset(0, -1).Value) < seq Then

If Not IsModuleCompleted(CStr(m.Cells(i, 1).Value)) Then PriorRequiredCompleted = False: Exit Function

End If

End If

End If

Next i

PriorRequiredCompleted = True

End Function

Function SafetyReady(domain As String, listCSV As String) As Boolean

If Len(Trim(listCSV)) = 0 Then SafetyReady = True: Exit Function

Dim a() As String: a = Split(listCSV, ",")

Dim i As Long, item As String, r As Range

For i = LBound(a) To UBound(a)

item = Trim(a(i))

Set r = WS("Safety").Columns(1).Find(item, , xlValues, xlWhole)

If r Is Nothing Then SafetyReady = False: Exit Function

If LCase(CStr(r.Offset(0, 3).Value)) <> LCase(domain) Then SafetyReady = False: Exit Function

If CBool(r.Offset(0, 1).Value) = True And CBool(r.Offset(0, 2).Value) = False Then SafetyReady = False: Exit Function

Next i

SafetyReady = True

End Function

Function ToolsReady(reqCSV As String) As Boolean

If Len(Trim(reqCSV)) = 0 Then ToolsReady = True: Exit Function

Dim a() As String: a = Split(reqCSV, ",")

Dim i As Long, r As Range

For i = LBound(a) To UBound(a)

Set r = WS("Tools").Columns(1).Find(Trim(a(i)), , xlValues, xlWhole)

If r Is Nothing Then ToolsReady = False: Exit Function

If LCase(r.Offset(0, 2).Value) <> "available" Then ToolsReady = False: Exit Function

If Not IsEmpty(r.Offset(0, 3).Value) Then

If Date > CDate(r.Offset(0, 3).Value) Then ToolsReady = False: Exit Function

End If

Next i

ToolsReady = True

End Function

Sub LockTools(reqCSV As String, lockOn As Boolean)

If Len(Trim(reqCSV)) = 0 Then Exit Sub

Dim a() As String: a = Split(reqCSV, ",")

Dim i As Long, r As Range

For i = LBound(a) To UBound(a)

Set r = WS("Tools").Columns(1).Find(Trim(a(i)), , xlValues, xlWhole)

If Not r Is Nothing Then r.Offset(0, 2).Value = IIf(lockOn, "InUse", "Available")

Next i

End Sub

## Scenario lifecycle

vba

Function StartExercise(exID As String) As Boolean

Dim ex As Range: Set ex = WS("Exercises").Columns(1).Find(exID, , xlValues, xlWhole)

If ex Is Nothing Then MsgBox "Exercise not found", vbExclamation: Exit Function

Dim modID As String: modID = CStr(ex.Offset(0, 1).Value)

Dim domain As String: domain = ModuleDomain(modID)

Dim steps As String: steps = CStr(ex.Offset(0, 4).Value)

Dim tools As String: tools = CStr(ex.Offset(0, 6).Value)

Dim safetyList As String: safetyList = CStr(ex.Offset(0, 7).Value)

If Not ModuleEnabled(modID) Then MsgBox "Module disabled", vbExclamation: Exit Function

If Not PriorRequiredCompleted(modID) Then MsgBox "Complete prior required modules", vbExclamation: Exit Function

If Not SafetyReady(domain, safetyList) Then MsgBox "Safety checklist incomplete", vbExclamation: Exit Function

If Not ToolsReady(tools) Then MsgBox "Tools unavailable or calibration expired", vbExclamation: Exit Function

If Not DomainPrecheck(domain) Then MsgBox "Domain precheck failed", vbExclamation: Exit Function

LockTools tools, True

LogEvent exID, "Started", modID, domain, "Exercise initiated"

StartExercise = True

End Function

Function DomainPrecheck(domain As String) As Boolean

Select Case UCase(domain)

Case "ELEC": DomainPrecheck = ElecPrecheck()

Case "FIRE": DomainPrecheck = FirePrecheck()

Case "AID": DomainPrecheck = AidPrecheck()

Case Else: DomainPrecheck = True

End Select

End Function

Function ElecPrecheck() As Boolean

' Example: ensure isolation/LOTO verified and ZVV done within timeout window

Dim timeout As Long: timeout = CLng(Cfg("Elec\_IsolationTimeout\_s", 300))

' In practice, you may store a timestamp in Events when "ZVV\_Pass" logged.

ElecPrecheck = True ' Keep permissive; enforce via first step gating below.

End Function

Function FirePrecheck() As Boolean

' Could verify extinguisher presence via Tools table

FirePrecheck = True

End Function

Function AidPrecheck() As Boolean

AidPrecheck = CBool(Cfg("FirstAid\_TrainerPresent", False))

End Function

Sub RunExercisePrompt()

Dim exID As String: exID = InputBox("Enter ExerciseID:")

If Len(exID) = 0 Then Exit Sub

If Not StartExercise(exID) Then Exit Sub

Dim ex As Range: Set ex = WS("Exercises").Columns(1).Find(exID, , xlValues, xlWhole)

Dim domain As String: domain = UCase(ModuleDomain(CStr(ex.Offset(0, 1).Value)))

Dim steps() As String: steps = Split(CStr(ex.Offset(0, 4).Value), ",")

Dim i As Long

For i = LBound(steps) To UBound(steps)

DispatchStep exID, domain, Trim(steps(i))

Next i

EvaluateExercise exID

LockTools CStr(ex.Offset(0, 6).Value), False

End Sub

Sub EvaluateExercise(exID As String)

Dim ex As Range: Set ex = WS("Exercises").Columns(1).Find(exID, , xlValues, xlWhole)

Dim crit As String: crit = CStr(ex.Offset(0, 5).Value)

Dim tokens() As String: tokens = Split(crit, ";")

Dim ok As Boolean: ok = True

Dim i As Long

For i = LBound(tokens) To UBound(tokens)

If Len(Trim(tokens(i))) > 0 Then If Not CriterionMet(exID, Trim(tokens(i))) Then ok = False

Next i

LogEvent exID, IIf(ok, "Completed", "Failed"), "", "", IIf(ok, "Pass", "Fail")

GeneratePortfolio exID

End Sub

Function CriterionMet(exID As String, expr As String) As Boolean

Dim op As String

If InStr(expr, ">=") > 0 Then op = ">=" ElseIf InStr(expr, "<=") > 0 Then op = "<=" \_

ElseIf InStr(expr, "==") > 0 Then op = "==" ElseIf InStr(expr, ">") > 0 Then op = ">" \_

ElseIf InStr(expr, "<") > 0 Then op = "<"

If Len(op) = 0 Then CriterionMet = False: Exit Function

Dim parts() As String: parts = Split(expr, op)

Dim metric As String: metric = Trim(parts(0))

Dim target As Double: target = CDbl(Val(Trim(parts(1))))

Dim val As Double: val = LatestMetric(exID, metric)

Select Case op

Case ">=": CriterionMet = (val >= target)

Case "<=": CriterionMet = (val <= target)

Case "==": CriterionMet = (Abs(val - target) < 0.0001)

Case ">": CriterionMet = (val > target)

Case "<": CriterionMet = (val < target)

Case Else: CriterionMet = False

End Select

End Function

Function LatestMetric(exID As String, metric As String) As Double

Dim w As Worksheet: Set w = WS("Measurements")

Dim last As Long: last = w.Cells(w.Rows.Count, 1).End(xlUp).Row

Dim i As Long

For i = last To 2 Step -1

If w.Cells(i, 1) = exID And w.Cells(i, 2) = metric Then LatestMetric = CDbl(w.Cells(i, 3)): Exit Function

Next i

LatestMetric = 0

End Function

## Domain step dispatchers

Sub DispatchStep(exID As String, domain As String, token As String)

Dim parts() As String: parts = Split(token, ":")

Dim stepName As String: stepName = LCase(parts(0))

Dim param As String: If UBound(parts) >= 1 Then param = parts(1) Else param = ""

Select Case domain

Case "ELEC": ELEC\_Step exID, stepName, param

Case "ROAD": ROAD\_Step exID, stepName, param

Case "FIRE": FIRE\_Step exID, stepName, param

Case "AID": AID\_Step exID, stepName, param

Case Else: LogEvent exID, "Warn", "UnknownDomain", domain, token

End Select

End Sub

### Electrical installation/repair (generators, relays)

Sub ELEC\_Step(exID As String, stepName As String, param As String)

Select Case stepName

Case "loto\_apply": LogEvent exID, "ELEC", "LOTO", "Applied", "Isolation/Lockout applied"

Case "zvv\_test"

' param is measured voltage, expect near 0 V

Dim v As Double: v = Val(param)

RecordMetric exID, "ZVV\_V", v, "V", "DMM"

LogEvent exID, "ELEC", "ZVV", IIf(v < 1, "Pass", "Fail"), ""

Case "relay\_test"

' param e.g., "pickup=18.5,drop=7.2"

Dim pk As Double: pk = Val(Split(Split(param, ",")(0), "=")(1))

Dim dr As Double: dr = Val(Split(Split(param, ",")(1), "=")(1))

RecordMetric exID, "Relay\_Pickup\_V", pk, "V", "Bench"

RecordMetric exID, "Relay\_Drop\_V", dr, "V", "Bench"

Case "generator\_inspect": LogEvent exID, "ELEC", "Inspection", param, "Visual/Mechanical check"

Case "insulation"

' param in MΩ

RecordMetric exID, "IR\_MOhm", Val(param), "MΩ", "Megger"

Case "functional\_run"

' param e.g., "V=231,I=8.2,F=49.9"

Dim vln As Double: vln = Val(Split(Split(param, ",")(0), "=")(1))

Dim iA As Double: iA = Val(Split(Split(param, ",")(1), "=")(1))

Dim f As Double: f = Val(Split(Split(param, ",")(2), "=")(1))

RecordMetric exID, "Volt\_V", vln, "V", "Meter"

RecordMetric exID, "Curr\_A", iA, "A", "Clamp"

RecordMetric exID, "Freq\_Hz", f, "Hz", "Meter"

Case Else: LogEvent exID, "ELEC-Unknown", stepName, param, ""

End Select

End Sub

### Road safety and traffic signals

Sub ROAD\_Step(exID As String, stepName As String, param As String)

Select Case stepName

Case "signal\_meaning"

' param e.g., "Red=Stop"

LogEvent exID, "ROAD", "Signal", param, "Interpretation logged"

Case "sign\_identify"

' param e.g., "Triangular=Warning"

LogEvent exID, "ROAD", "Sign", param, "Category recognized"

Case "marking\_rule"

LogEvent exID, "ROAD", "Marking", param, "Rule recalled"

Case "quiz\_score"

RecordMetric exID, "RoadQuizScore", Val(param), "pct", "Quiz"

Case Else: LogEvent exID, "ROAD-Unknown", stepName, param, ""

End Select

End Sub

### Fire safety and extinguishers

Sub FIRE\_Step(exID As String, stepName As String, param As String)

Select Case stepName

Case "class\_identify"

' param e.g., "ClassB=Flammable liquids"

LogEvent exID, "FIRE", "Class", param, ""

Case "ext\_match"

' param e.g., "CO2=Electrical"

LogEvent exID, "FIRE", "ExtinguisherMatch", param, ""

Case "pass\_demo"

LogEvent exID, "FIRE", "PASS", "Performed", "Pull-Aim-Squeeze-Sweep"

Case "risk\_score"

' param numeric cumulative risk (mitigated)

Dim r As Double: r = Val(param)

RecordMetric exID, "FireRiskScore", r, "score", "Assessor"

RecordMetric exID, "FireRiskOK", IIf(r <= CDbl(Cfg("Fire\_MAX\_Risk", 5)), 1, 0), "bool", "Derived"

Case Else: LogEvent exID, "FIRE-Unknown", stepName, param, ""

End Select

End Sub

### First aid and CPR (simulated)

ub AID\_Step(exID As String, stepName As String, param As String)

If Not CBool(Cfg("FirstAid\_TrainerPresent", False)) Then

LogEvent exID, "AID", "Trainer", "Absent", "CPR practice gated"

Exit Sub

End If

Select Case stepName

Case "abc\_check": LogEvent exID, "AID", "ABC", param, "Airway-Breathing-Circulation"

Case "cpr\_cycles"

' param numeric cycles performed

RecordMetric exID, "CPR\_Cycles", Val(param), "cycles", "Instructor"

Case "bleeding\_control"

LogEvent exID, "AID", "Hemostasis", param, "Direct pressure applied"

Case "report\_emergency"

' param e.g., "108=Ambulance"

LogEvent exID, "AID", "Report", param, "Call simulated"

Case Else: LogEvent exID, "AID-Unknown", stepName, param, ""

End Select

End Sub

## Portfolio export

vba

Sub GeneratePortfolio(exID As String)

On Error Resume Next: Application.DisplayAlerts = False

ThisWorkbook.Worksheets("Portfolio").Delete

Application.DisplayAlerts = True: On Error GoTo 0

Dim wr As Worksheet: Set wr = ThisWorkbook.Worksheets.Add

wr.Name = "Portfolio"

Dim ex As Range: Set ex = WS("Exercises").Columns(1).Find(exID, , xlValues, xlWhole)

Dim modID As String: modID = CStr(ex.Offset(0, 1).Value)

Dim domain As String: domain = ModuleDomain(modID)

Dim r As Long: r = 1

wr.Cells(r, 1) = "Electrician Training Evidence": r = r + 2

wr.Cells(r, 1) = "Trainee": wr.Cells(r, 2) = CStr(WS("TraineeProfile").Range("B1").Value): r = r + 1

wr.Cells(r, 1) = "ExerciseID": wr.Cells(r, 2) = exID: r = r + 1

wr.Cells(r, 1) = "Module": wr.Cells(r, 2) = modID: r = r + 1

wr.Cells(r, 1) = "Domain": wr.Cells(r, 2) = domain: r = r + 2

r = CopySection(wr, r, "Events", WS("Events"), 3, exID)

r = CopySection(wr, r, "Measurements", WS("Measurements"), 1, exID)

wr.Columns.AutoFit

Dim outDir As String: outDir = Cfg("EvidenceDir", ThisWorkbook.Path)

Dim f As String: f = outDir & "\Portfolio\_" & exID & ".pdf"

On Error Resume Next: wr.ExportAsFixedFormat xlTypePDF, f: On Error GoTo 0

MsgBox "Portfolio generated: " & f, vbInformation

End Sub

Function CopySection(dst As Worksheet, startRow As Long, title As String, src As Worksheet, matchCol As Integer, key As String) As Long

dst.Cells(startRow, 1) = title

Dim r As Long: r = startRow + 1

Dim rng As Range: Set rng = src.Range("A1").CurrentRegion

Dim i As Long, header As Boolean

For i = 2 To rng.Rows.Count

If CStr(rng.Cells(i, matchCol)) = key Then

If Not header Then rng.Rows(1).Copy dst.Cells(r, 1): r = r + 1: header = True

rng.Rows(i).Copy dst.Cells(r, 1): r = r + 1

End If

Next i

CopySection = r + 1

End Function

Sub GeneratePortfolio(exID As String)

On Error Resume Next: Application.DisplayAlerts = False

ThisWorkbook.Worksheets("Portfolio").Delete

Application.DisplayAlerts = True: On Error GoTo 0

Dim wr As Worksheet: Set wr = ThisWorkbook.Worksheets.Add

wr.Name = "Portfolio"

Dim ex As Range: Set ex = WS("Exercises").Columns(1).Find(exID, , xlValues, xlWhole)

Dim modID As String: modID = CStr(ex.Offset(0, 1).Value)

Dim domain As String: domain = ModuleDomain(modID)

Dim r As Long: r = 1

wr.Cells(r, 1) = "Electrician Training Evidence": r = r + 2

wr.Cells(r, 1) = "Trainee": wr.Cells(r, 2) = CStr(WS("TraineeProfile").Range("B1").Value): r = r + 1

wr.Cells(r, 1) = "ExerciseID": wr.Cells(r, 2) = exID: r = r + 1

wr.Cells(r, 1) = "Module": wr.Cells(r, 2) = modID: r = r + 1

wr.Cells(r, 1) = "Domain": wr.Cells(r, 2) = domain: r = r + 2

r = CopySection(wr, r, "Events", WS("Events"), 3, exID)

r = CopySection(wr, r, "Measurements", WS("Measurements"), 1, exID)

wr.Columns.AutoFit

Dim outDir As String: outDir = Cfg("EvidenceDir", ThisWorkbook.Path)

Dim f As String: f = outDir & "\Portfolio\_" & exID & ".pdf"

On Error Resume Next: wr.ExportAsFixedFormat xlTypePDF, f: On Error GoTo 0

MsgBox "Portfolio generated: " & f, vbInformation

End Sub

Function CopySection(dst As Worksheet, startRow As Long, title As String, src As Worksheet, matchCol As Integer, key As String) As Long

dst.Cells(startRow, 1) = title

Dim r As Long: r = startRow + 1

Dim rng As Range: Set rng = src.Range("A1").CurrentRegion

Dim i As Long, header As Boolean

For i = 2 To rng.Rows.Count

If CStr(rng.Cells(i, matchCol)) = key Then

If Not header Then rng.Rows(1).Copy dst.Cells(r, 1): r = r + 1: header = True

rng.Rows(i).Copy dst.Cells(r, 1): r = r + 1

End If

Next i

CopySection = r + 1

End Function

## Quick population and examples

* Config:
  + CurrentUser = Tshingombe Fiston Tshitadi
  + MinPPE = 3
  + EvidenceDir = C:\Evidence
  + Elec\_IsolationTimeout\_s = 300
  + Fire\_MAX\_Risk = 5
  + FirstAid\_TrainerPresent = TRUE
* Modules:
  + 1 | Electrical Safety & LOTO | Seq 1 | TRUE | ELEC | TRUE
  + 2 | Generators & Relays | Seq 2 | TRUE | ELEC | TRUE
  + 3 | Road Safety & Signals | Seq 3 | FALSE | ROAD | TRUE
  + 4 | Fire Safety | Seq 4 | TRUE | FIRE | TRUE
  + 5 | First Aid & CPR | Seq 5 | TRUE | AID | TRUE
* Tools:
  + T\_DMM, T\_Megger, T\_RelayBench, T\_CO2\_Ext, T\_DryPowder, T\_FirstAidManikin (all Available; set CalDueDate accordingly)
* Exercises:
  + EX\_ELEC\_01 | 1 | LOTO + ZVV | Isolate and verify zero voltage | loto\_apply, zvv\_test:0.3 | ZVV\_V<=1 | T\_DMM | PPE,InsulatedGloves
  + EX\_ELEC\_02 | 2 | Relay test | Verify pickup/drop | relay\_test:pickup=18.5,drop=7.2 | Relay\_Pickup\_V>=18;Relay\_Drop\_V<=9 | T\_RelayBench,T\_DMM |
  + EX\_ROAD\_01 | 3 | Signals quiz | Identify signals | signal\_meaning:Red=Stop, sign\_identify:Triangular=Warning, quiz\_score:85 | RoadQuizScore>=70 | | HiVis
  + EX\_FIRE\_01 | 4 | Extinguisher match | Select correct agent | class\_identify:ClassB=Flammable liquids, ext\_match:CO2=Electrical, pass\_demo, risk\_score:3 | FireRiskOK==1 | T\_CO2\_Ext,T\_DryPowder | PPE
  + EX\_AID\_01 | 5 | ABC + CPR | Simulated rescue | abc\_check:Clear, cpr\_cycles:5, report\_emergency:108=Ambulance | CPR\_Cycles>=3 | T\_FirstAidManikin | FirstAidKit

Run with RunExercisePrompt and enter an ExerciseID (e.g., EX\_ELEC\_01). The engine validates safety and tools (logigram), executes steps (algorigram), evaluates criteria, and exports a Portfolio PDF.

## Want me to add a visual dashboard?

* Buttons: Start Exercise, Run Exercise, Generate Portfolio, Mark Module Completed.
* KPI tiles: Safety gate, Tools status, Latest score, Pass/Fail.
* If you share your preferred columns or existing workbook, I’ll tailor the macros to your exact schema and SAQA/NSQF evidence mapping.

# VBA logigram and algorigram for 5S, tool standards, lifting, cutting, and marking

Below is a modular Excel VBA engine that enforces readiness (logigram) and runs practical sessions (algorigram) across five domains:

* 5S workplace organization
* Tool standards and care (BIS/NEC-aligned)
* Lifting and moving equipment
* Cutting (hacksaw) and blade selection
* Marking and measurement (rules, punches, calipers)

It captures evidence, evaluates pass criteria, and exports an auditable portfolio.

## Workbook schema

Create the following sheets and columns exactly; code references names and positions.

* Modules
  + A: ModuleID, B: Name, C: Sequence, D: Required (TRUE/FALSE), E: Domain (FIVES/TOOLS/LIFT/CUT/MARK), F: Enabled (TRUE/FALSE)
* Exercises
  + A: ExerciseID, B: ModuleID, C: Name, D: Objective, E: StepsCSV, F: PassCriteria, G: RequiredItemsCSV, H: SafetyChecklistCSV
* Safety
  + A: Item, B: Required (TRUE/FALSE), C: Completed (TRUE/FALSE), D: Domain, E: Notes
* Inventory
  + A: ItemID, B: Category (Tool/PPE/Fixture), C: StandardCode (e.g., BIS 3650), D: Status (Available/InUse/Out), E: CalDueDate, F: Domain
* Standards
  + A: Code, B: Title, C: Domain, D: Notes (e.g., BIS 3650 Combination Pliers)
* Measurements
  + A: ExerciseID, B: Metric, C: Value, D: Unit, E: Source, F: Timestamp
* Events
  + A: Timestamp, B: User, C: ExerciseID, D: EventType, E: K1, F: K2, G: Notes
* Config
  + A: Key, B: Value
  + Keys: CurrentUser, EvidenceDir, MinPPE, Lifting\_MaxMass\_kg, Cut\_MaxKerf\_mm, Mark\_MaxError\_mm, 5S\_MinScore, Tools\_MinScore
* Portfolio
  + Generated by macro

## Logigram rules

* **Module gate:** Module Enabled = TRUE; all prior Required modules by Sequence are completed.
* **Safety gate:** All Required items in SafetyChecklistCSV for the module’s Domain are Completed = TRUE.
* **Inventory gate:** All RequiredItemsCSV are Status = Available; calibration not expired when applicable.
* **Domain prechecks:**
  + **FIVES:** Minimum PPE present; 5S audit will compute score target.
  + **TOOLS:** Tool standard codes must be known in Standards table; tool care checks recorded.
  + **LIFT:** Mass ≤ Lifting\_MaxMass\_kg; path clear; proper devices selected.
  + **CUT:** Blade type/pitch aligned to material and thickness; kerf and edge quality recorded.
  + **MARK:** Marking media and instrument fit-for-purpose; accuracy tolerance Mark\_MaxError\_mm.

## Core utilities

Option Explicit

Function WS(name As String) As Worksheet

Set WS = ThisWorkbook.Worksheets(name)

End Function

Function Cfg(key As String, Optional defVal As Variant = "") As Variant

Dim r As Range: Set r = WS("Config").Columns(1).Find(key, , xlValues, xlWhole)

If r Is Nothing Then Cfg = defVal Else Cfg = IIf(IsEmpty(r.Offset(0, 1)), defVal, r.Offset(0, 1))

End Function

Function NowStamp() As String

NowStamp = Format(Now, "yyyy-mm-dd hh:nn:ss")

End Function

Sub LogEvent(exID As String, evt As String, Optional k1 As String = "", Optional k2 As String = "", Optional note As String = "")

Dim w As Worksheet: Set w = WS("Events")

Dim r As Long: r = w.Cells(w.Rows.Count, 1).End(xlUp).Row + 1

w.Cells(r, 1) = NowStamp(): w.Cells(r, 2) = Cfg("CurrentUser", "Trainee")

w.Cells(r, 3) = exID: w.Cells(r, 4) = evt: w.Cells(r, 5) = k1: w.Cells(r, 6) = k2: w.Cells(r, 7) = note

End Sub

Sub RecordMetric(exID As String, metric As String, val As Double, unitStr As String, src As String)

Dim w As Worksheet: Set w = WS("Measurements")

Dim r As Long: r = w.Cells(w.Rows.Count, 1).End(xlUp).Row + 1

w.Cells(r, 1) = exID: w.Cells(r, 2) = metric: w.Cells(r, 3) = val

w.Cells(r, 4) = unitStr: w.Cells(r, 5) = src: w.Cells(r, 6) = NowStamp()

End Sub

## Readiness checks and module order

Function ModuleEnabled(modID As String) As Boolean

Dim r As Range: Set r = WS("Modules").Columns(1).Find(modID, , xlValues, xlWhole)

ModuleEnabled = Not r Is Nothing And CBool(r.Offset(0, 5).Value)

End Function

Function ModuleDomain(modID As String) As String

Dim r As Range: Set r = WS("Modules").Columns(1).Find(modID, , xlValues, xlWhole)

If r Is Nothing Then ModuleDomain = "" Else ModuleDomain = CStr(r.Offset(0, 4).Value)

End If: End Function

Function SequenceOf(modID As String) As Long

Dim r As Range: Set r = WS("Modules").Columns(1).Find(modID, , xlValues, xlWhole)

If r Is Nothing Then SequenceOf = 9999 Else SequenceOf = CLng(r.Offset(0, 2).Value)

End If: End Function

Function IsModuleCompleted(modID As String) As Boolean

Dim w As Worksheet: Set w = WS("Events")

Dim i As Long, last As Long: last = w.Cells(w.Rows.Count, 1).End(xlUp).Row

For i = last To 2 Step -1

If w.Cells(i, 4).Value = "ModuleCompleted" And w.Cells(i, 5).Value = modID Then IsModuleCompleted = True: Exit Function

Next i

IsModuleCompleted = False

End Function

Function PriorRequiredCompleted(modID As String) As Boolean

Dim m As Worksheet: Set m = WS("Modules")

Dim seq As Long: seq = SequenceOf(modID)

Dim i As Long, last As Long: last = m.Cells(m.Rows.Count, 1).End(xlUp).Row

For i = 2 To last

If CBool(m.Cells(i, 4).Value) = True Then

If CLng(m.Cells(i, 3).Value) < seq Then

If Not IsModuleCompleted(CStr(m.Cells(i, 1).Value)) Then PriorRequiredCompleted = False: Exit Function

End If

End If

Next i

PriorRequiredCompleted = True

End Function

Function SafetyReady(domain As String, listCSV As String) As Boolean

If Len(Trim(listCSV)) = 0 Then SafetyReady = True: Exit Function

Dim a() As String: a = Split(listCSV, ",")

Dim i As Long, r As Range, item As String

For i = LBound(a) To UBound(a)

item = Trim(a(i))

Set r = WS("Safety").Columns(1).Find(item, , xlValues, xlWhole)

If r Is Nothing Then SafetyReady = False: Exit Function

If LCase(CStr(r.Offset(0, 3).Value)) <> LCase(domain) Then SafetyReady = False: Exit Function

If CBool(r.Offset(0, 1).Value) And Not CBool(r.Offset(0, 2).Value) Then SafetyReady = False: Exit Function

Next i

SafetyReady = True

End Function

Function InventoryReady(reqCSV As String) As Boolean

If Len(Trim(reqCSV)) = 0 Then InventoryReady = True: Exit Function

Dim a() As String: a = Split(reqCSV, ",")

Dim i As Long, r As Range

For i = LBound(a) To UBound(a)

Set r = WS("Inventory").Columns(1).Find(Trim(a(i)), , xlValues, xlWhole)

If r Is Nothing Then InventoryReady = False: Exit Function

If LCase(r.Offset(0, 3).Value) <> "available" Then InventoryReady = False: Exit Function

If Not IsEmpty(r.Offset(0, 4).Value) Then

If Date > CDate(r.Offset(0, 4).Value) Then InventoryReady = False: Exit Function

End If

Next i

InventoryReady = True

End Function

Sub LockInventory(reqCSV As String, lockOn As Boolean)

If Len(Trim(reqCSV)) = 0 Then Exit Sub

Dim a() As String: a = Split(reqCSV, ",")

Dim i As Long, r As Range

For i = LBound(a) To UBound(a)

Set r = WS("Inventory").Columns(1).Find(Trim(a(i)), , xlValues, xlWhole)

If Not r Is Nothing Then r.Offset(0, 3).Value = IIf(lockOn, "InUse", "Available")

Next i

End Sub

## Scenario lifecycle

vba

Function StartExercise(exID As String) As Boolean

Dim ex As Range: Set ex = WS("Exercises").Columns(1).Find(exID, , xlValues, xlWhole)

If ex Is Nothing Then MsgBox "Exercise not found", vbExclamation: Exit Function

Dim modID As String: modID = CStr(ex.Offset(0, 1).Value)

Dim domain As String: domain = ModuleDomain(modID)

Dim tools As String: tools = CStr(ex.Offset(0, 6).Value)

Dim safetyList As String: safetyList = CStr(ex.Offset(0, 7).Value)

If Not ModuleEnabled(modID) Then MsgBox "Module disabled", vbExclamation: Exit Function

If Not PriorRequiredCompleted(modID) Then MsgBox "Complete prior required modules", vbExclamation: Exit Function

If Not SafetyReady(domain, safetyList) Then MsgBox "Safety checklist incomplete", vbExclamation: Exit Function

If Not InventoryReady(tools) Then MsgBox "Required items unavailable or expired", vbExclamation: Exit Function

If Not DomainPrecheck(domain) Then MsgBox "Domain precheck failed", vbExclamation: Exit Function

LockInventory tools, True

LogEvent exID, "Started", modID, domain, "Exercise initiated"

StartExercise = True

End Function

Function DomainPrecheck(domain As String) As Boolean

Select Case UCase(domain)

Case "FIVES": DomainPrecheck = True

Case "TOOLS": DomainPrecheck = True

Case "LIFT": DomainPrecheck = True

Case "CUT": DomainPrecheck = True

Case "MARK": DomainPrecheck = True

Case Else: DomainPrecheck = True

End Select

End Function

Sub RunExercisePrompt()

Dim exID As String: exID = InputBox("Enter ExerciseID:")

If Len(exID) = 0 Then Exit Sub

If Not StartExercise(exID) Then Exit Sub

Dim ex As Range: Set ex = WS("Exercises").Columns(1).Find(exID, , xlValues, xlWhole)

Dim domain As String: domain = UCase(ModuleDomain(CStr(ex.Offset(0, 1).Value)))

Dim steps() As String: steps = Split(CStr(ex.Offset(0, 4).Value), ",")

Dim i As Long

For i = LBound(steps) To UBound(steps)

DispatchStep exID, domain, Trim(steps(i))

Next i

EvaluateExercise exID

LockInventory CStr(ex.Offset(0, 6).Value), False

End Sub

Sub EvaluateExercise(exID As String)

Dim ex As Range: Set ex = WS("Exercises").Columns(1).Find(exID, , xlValues, xlWhole)

Dim crit As String: crit = CStr(ex.Offset(0, 5).Value)

Dim tokens() As String: tokens = Split(crit, ";")

Dim ok As Boolean: ok = True

Dim i As Long

For i = LBound(tokens) To UBound(tokens)

If Len(Trim(tokens(i))) > 0 Then If Not CriterionMet(exID, Trim(tokens(i))) Then ok = False

Next i

LogEvent exID, IIf(ok, "Completed", "Failed"), "", "", IIf(ok, "Pass", "Fail")

GeneratePortfolio exID

End Sub

Function CriterionMet(exID As String, expr As String) As Boolean

Dim op As String

If InStr(expr, ">=") > 0 Then op = ">=" ElseIf InStr(expr, "<=") > 0 Then op = "<=" \_

ElseIf InStr(expr, "==") > 0 Then op = "==" ElseIf InStr(expr, ">") > 0 Then op = ">" \_

ElseIf InStr(expr, "<") > 0 Then op = "<"

If Len(op) = 0 Then CriterionMet = False: Exit Function

Dim parts() As String: parts = Split(expr, op)

Dim metric As String: metric = Trim(parts(0))

Dim target As Double: target = CDbl(Val(Trim(parts(1))))

Dim val As Double: val = LatestMetric(exID, metric)

Select Case op

Case ">=": CriterionMet = (val >= target)

Case "<=": CriterionMet = (val <= target)

Case "==": CriterionMet = (Abs(val - target) < 0.0001)

Case ">": CriterionMet = (val > target)

Case "<": CriterionMet = (val < target)

Case Else: CriterionMet = False

End Select

End Function

Function LatestMetric(exID As String, metric As String) As Double

Dim w As Worksheet: Set w = WS("Measurements")

Dim i As Long, last As Long: last = w.Cells(w.Rows.Count, 1).End(xlUp).Row

For i = last To 2 Step -1

If w.Cells(i, 1) = exID And w.Cells(i, 2) = metric Then LatestMetric = CDbl(w.Cells(i, 3)): Exit Function

Next i

LatestMetric = 0

End Function

## Domain step dispatchers

Sub DispatchStep(exID As String, domain As String, token As String)

Dim parts() As String: parts = Split(token, ":")

Dim stepName As String: stepName = LCase(parts(0))

Dim param As String: If UBound(parts) >= 1 Then param = parts(1) Else param = ""

Select Case domain

Case "FIVES": FIVES\_Step exID, stepName, param

Case "TOOLS": TOOLS\_Step exID, stepName, param

Case "LIFT": LIFT\_Step exID, stepName, param

Case "CUT": CUT\_Step exID, stepName, param

Case "MARK": MARK\_Step exID, stepName, param

Case Else: LogEvent exID, "Warn", "UnknownDomain", domain, token

End Select

End Sub

### 5S workplace organization

Sub FIVES\_Step(exID As String, stepName As String, param As String)

Select Case stepName

Case "seiri" ' sort count of removed items

RecordMetric exID, "5S\_SortRemoved", Val(param), "items", "Audit"

LogEvent exID, "5S", "Seiri", param, "Unnecessary removed"

Case "seiton" ' average retrieval time before/after (seconds)

RecordMetric exID, "5S\_SeitonTime\_s", Val(param), "s", "Stopwatch"

LogEvent exID, "5S", "Seiton", param, "Arrangement timed"

Case "seiso" ' cleanliness score 0-5

RecordMetric exID, "5S\_SeisoScore", Val(param), "score", "Audit"

Case "seiketsu" ' standard docs created

RecordMetric exID, "5S\_StandardsCount", Val(param), "docs", "SOP"

Case "shitsuke" ' audit sustain score 0-5

RecordMetric exID, "5S\_SustainScore", Val(param), "score", "Audit"

Case "total\_score"

RecordMetric exID, "5S\_TotalScore", Val(param), "score", "Computed"

Case Else

LogEvent exID, "5S", "Unknown", stepName, param

End Select

End Sub

Sub TOOLS\_Step(exID As String, stepName As String, param As String)

Select Case stepName

Case "verify\_bis" ' param e.g., "BIS 3650"

Dim r As Range: Set r = WS("Standards").Columns(1).Find(param, , xlValues, xlWhole)

LogEvent exID, "TOOLS", "BIS\_Check", param, IIf(r Is Nothing, "Unknown", "Recognized"))

RecordMetric exID, "Tools\_BIS\_Recognized", IIf(r Is Nothing, 0, 1), "bool", "Standards"

Case "inspect\_tool" ' param e.g., "CombinationPliers=OK"

LogEvent exID, "TOOLS", "Inspection", param, "Condition logged"

Case "care\_task" ' param e.g., "LubricateHinge=Done"

LogEvent exID, "TOOLS", "Care", param, "Maintenance"

Case "selection\_quiz" ' numeric %

RecordMetric exID, "Tools\_QuizScore", Val(param), "pct", "Quiz"

Case "neon\_tester\_use" ' param numeric within rated voltage? 0/1

RecordMetric exID, "Tools\_NeonUseOK", Val(param), "bool", "Assessor"

Case Else

LogEvent exID, "TOOLS", "Unknown", stepName, param

End Select

End Sub

### Lifting and moving equipment

Sub LIFT\_Step(exID As String, stepName As String, param As String)

Select Case stepName

Case "assess\_mass" ' kg

Dim mkg As Double: mkg = Val(param)

RecordMetric exID, "Lift\_Mass\_kg", mkg, "kg", "Scale"

RecordMetric exID, "Lift\_MassOK", IIf(mkg <= CDbl(Cfg("Lifting\_MaxMass\_kg", 50)), 1, 0), "bool", "Derived"

Case "device\_select" ' e.g., "Slings/Winch/Rollers"

LogEvent exID, "LIFT", "Device", param, "Selected"

Case "path\_clear" ' 0/1

RecordMetric exID, "Lift\_PathClear", Val(param), "bool", "Assessor")

Case "center\_gravity" ' 0/1 correctly centered

RecordMetric exID, "Lift\_CG\_OK", Val(param), "bool", "Assessor"

Case "corner\_roll" ' 0/1 executed per SOP

RecordMetric exID, "Lift\_CornerRoll\_OK", Val(param), "bool", "Assessor"

Case Else

LogEvent exID, "LIFT", "Unknown", stepName, param

End Select

End Sub

### Cutting (hacksaw) and blade selection

Sub CUT\_Step(exID As String, stepName As String, param As String)

Select Case stepName

Case "blade\_type" ' All-hard/Flexible

LogEvent exID, "CUT", "BladeType", param, "Selected"

Case "pitch\_tpi" ' teeth per 25 mm

RecordMetric exID, "Cut\_TeethPer25mm", Val(param), "t/25mm", "Spec"

Case "tooth\_set" ' Staggered/Wave

LogEvent exID, "CUT", "ToothSet", param, "Pattern"

Case "kerf\_mm" ' measure kerf width

Dim k As Double: k = Val(param)

RecordMetric exID, "Cut\_Kerf\_mm", k, "mm", "Gauge"

RecordMetric exID, "Cut\_KerfOK", IIf(k <= CDbl(Cfg("Cut\_MaxKerf\_mm", 1.2)), 1, 0), "bool", "Derived"

Case "edge\_quality" ' 0-5

RecordMetric exID, "Cut\_EdgeQuality", Val(param), "score", "Assessor"

Case Else

LogEvent exID, "CUT", "Unknown", stepName, param

End Select

End Sub

### Marking and measurement

vba

Sub MARK\_Step(exID As String, stepName As String, param As String)

Select Case stepName

Case "media" ' Whitewash/CuSO4/Lacquer/PrussianBlue

LogEvent exID, "MARK", "Media", param, "Selected"

Case "rule\_size" ' 150/300/600

LogEvent exID, "MARK", "RuleSize", param, "Engineer rule"

Case "punch\_type" ' Center/Prick

LogEvent exID, "MARK", "Punch", param, "Selected"

Case "caliper\_type" ' Inside/Outside/Vernier

LogEvent exID, "MARK", "Caliper", param, "Selected"

Case "mark\_error\_mm" ' absolute marking error

Dim e As Double: e = Val(param)

RecordMetric exID, "Mark\_Error\_mm", e, "mm", "Micrometer"

RecordMetric exID, "Mark\_TolOK", IIf(e <= CDbl(Cfg("Mark\_MaxError\_mm", 0.5)), 1, 0), "bool", "Derived"

Case Else

LogEvent exID, "MARK", "Unknown", stepName, param

End Select

End Sub

Portfolio export Sub GeneratePortfolio(exID As String)

On Error Resume Next: Application.DisplayAlerts = False

ThisWorkbook.Worksheets("Portfolio").Delete

Application.DisplayAlerts = True: On Error GoTo 0

Dim wr As Worksheet: Set wr = ThisWorkbook.Worksheets.Add

wr.Name = "Portfolio"

Dim ex As Range: Set ex = WS("Exercises").Columns(1).Find(exID, , xlValues, xlWhole)

Dim modID As String: modID = CStr(ex.Offset(0, 1).Value)

Dim domain As String: domain = ModuleDomain(modID)

Dim r As Long: r = 1

wr.Cells(r, 1) = "Electrician Practical Evidence": r = r + 2

wr.Cells(r, 1) = "ExerciseID": wr.Cells(r, 2) = exID: r = r + 1

wr.Cells(r, 1) = "Module": wr.Cells(r, 2) = modID: r = r + 1

wr.Cells(r, 1) = "Domain": wr.Cells(r, 2) = domain: r = r + 1

wr.Cells(r, 1) = "Objective": wr.Cells(r, 2) = CStr(ex.Offset(0, 3).Value): r = r + 2

r = CopySection(wr, r, "Events", WS("Events"), 3, exID)

r = CopySection(wr, r, "Measurements", WS("Measurements"), 1, exID)

wr.Columns.AutoFit

Dim outDir As String: outDir = Cfg("EvidenceDir", ThisWorkbook.Path)

Dim f As String: f = outDir & "\Portfolio\_" & exID & ".pdf"

On Error Resume Next: wr.ExportAsFixedFormat xlTypePDF, f: On Error GoTo 0

MsgBox "Portfolio generated: " & f, vbInformation

End Sub

Function CopySection(dst As Worksheet, startRow As Long, title As String, src As Worksheet, matchCol As Integer, key As String) As Long

dst.Cells(startRow, 1) = title

Dim r As Long: r = startRow + 1

Dim rng As Range: Set rng = src.Range("A1").CurrentRegion

Dim i As Long, header As Boolean

For i = 2 To rng.Rows.Count

If CStr(rng.Cells(i, matchCol)) = key Then

If Not header Then rng.Rows(1).Copy dst.Cells(r, 1): r = r + 1: header = True

rng.Rows(i).Copy dst.Cells(r, 1): r = r + 1

End If

Next i

CopySection = r + 1

End Function

## Quick population examples

* Config:
  + **CurrentUser:** Tshingombe Fiston Tshitadi
  + **EvidenceDir:** C:\Evidence
  + **MinPPE:** 3
  + **Lifting\_MaxMass\_kg:** 50
  + **Cut\_MaxKerf\_mm:** 1.2
  + **Mark\_MaxError\_mm:** 0.5
  + **5S\_MinScore:** 70
  + **Tools\_MinScore:** 70
* Modules:
  + 1 | 5S Workplace Audit | 1 | TRUE | FIVES | TRUE
  + 2 | Tool Standards & Care | 2 | TRUE | TOOLS | TRUE
  + 3 | Lifting & Handling | 3 | TRUE | LIFT | TRUE
  + 4 | Hacksaw Cutting | 4 | TRUE | CUT | TRUE
  + 5 | Marking & Measurement | 5 | TRUE | MARK | TRUE
* Standards (examples):
  + BIS 3650 | Combination Pliers | TOOLS | Cutting, twisting, gripping
  + BIS 844 | Screwdrivers | TOOLS | Tightening/loosening screws
  + BIS 5579 | Neon Tester | TOOLS | Checking live voltage
* Inventory (examples):
  + T\_PLIERS | Tool | BIS 3650 | Available | 2026-12-31 | TOOLS
  + T\_SCREW | Tool | BIS 844 | Available | 2027-01-01 | TOOLS
  + PPE\_GLOVES | PPE | — | Available | | FIVES
  + L\_WINCH | Fixture | — | Available | | LIFT
  + CUT\_SAW | Tool | — | Available | 2026-05-01 | CUT
  + MARK\_RULE300 | Tool | — | Available | 2028-01-01 | MARK
* Safety (examples):
  + PPE | TRUE | TRUE | FIVES | Gloves, goggles, boots
  + ClearPath | TRUE | TRUE | LIFT | Barriers placed
  + FireExt Nearby | TRUE | TRUE | CUT | CO2 or DP extinguisher
  + EyeProtection | TRUE | TRUE | MARK | Safety glasses
* Exercises:
  + EX\_5S\_01 | 1 | 5S Quick Kaizen | Improve bench area | seiri:12,seiton:38,seiso:4,seiketsu:3,shitsuke:4,total\_score:76 | 5S\_TotalScore>=70 | PPE\_GLOVES | PPE
  + EX\_TOOL\_01 | 2 | Tool Standards Check | Verify BIS + care | verify\_bis:BIS 3650,inspect\_tool:CombinationPliers=OK,care\_task:LubricateHinge=Done,selection\_quiz:85,neon\_tester\_use:1 | Tools\_BIS\_Recognized==1;Tools\_QuizScore>=70 | T\_PLIERS,T\_SCREW | PPE
  + EX\_LIFT\_01 | 3 | Move Transformer | Safe relocation | assess\_mass:42,device\_select:Winch,path\_clear:1,center\_gravity:1,corner\_roll:1 | Lift\_MassOK==1;Lift\_PathClear==1;Lift\_CG\_OK==1 | L\_WINCH,PPE\_GLOVES | ClearPath
  + EX\_CUT\_01 | 4 | Cut Conduit | Clean cut | blade\_type:Flexible,pitch\_tpi:24,tooth\_set:Wave,kerf\_mm:1.0,edge\_quality:4 | Cut\_KerfOK==1;Cut\_EdgeQuality>=3 | CUT\_SAW | FireExt Nearby
  + EX\_MARK\_01 | 5 | Mark Panel | Accurate layout | media:PrussianBlue,rule\_size:300,punch\_type:Center,caliper\_type:Vernier,mark\_error\_mm:0.3 | Mark\_TolOK==1 | MARK\_RULE300 | EyeProtection

## Optional dashboard buttons

* **Run Exercise:** assign to RunExercisePrompt.
* **Generate Portfolio:** assign to GeneratePortfolioPrompt wrapper if you add one.
* **Mark Module Completed:** log a “ModuleCompleted” event when an exercise passes.

If you want, I can add a compact dashboard with tiles for safety gate, inventory status, and pass/fail, plus charts for 5S scores, kerf distributions, and marking accuracy trends.

# VBA logigram and algorigram for tool standards, NEC, lifting, cutting, and marking

This engine enforces readiness (logigram) and runs practical sessions (algorigram) across:

* 5S/tool organization and BIS compliance
* Measuring and cutting tools (hacksaw, drills, tapes)
* Lifting and moving equipment
* Marking and measurement (rules, punches, calipers)
* Standards and codes (BIS/ISO and NEC Part 1)

It logs events, records metrics, evaluates pass criteria, and exports an auditable portfolio.

## Workbook schema

Create these sheets and columns exactly (the code relies on them).

* Modules
  + A: ModuleID, B: Name, C: Sequence, D: Required (TRUE/FALSE), E: Domain (TOOLS/LIFT/CUT/MARK/CODE), F: Enabled (TRUE/FALSE)
* Exercises
  + A: ExerciseID, B: ModuleID, C: Name, D: Objective, E: StepsCSV, F: PassCriteria, G: RequiredItemsCSV, H: SafetyChecklistCSV
* Safety
  + A: Item, B: Required (TRUE/FALSE), C: Completed (TRUE/FALSE), D: Domain, E: Notes
* Inventory
  + A: ItemID, B: Category (Tool/PPE/Fixture), C: StandardCode (e.g., BIS 2029), D: Status (Available/InUse/Out), E: CalDueDate, F: Domain
* Standards
  + A: Code (e.g., BIS 2029), B: Title, C: Domain, D: Notes
* NEC
  + A: Section (1–20), B: Title, C: Focus
* Events
  + A: Timestamp, B: User, C: ExerciseID, D: EventType, E: K1, F: K2, G: Notes
* Measurements
  + A: ExerciseID, B: Metric, C: Value, D: Unit, E: Source, F: Timestamp
* Config
  + A: Key, B: Value
  + Keys: CurrentUser, EvidenceDir, Lifting\_MaxMass\_kg, Cut\_MaxKerf\_mm, Mark\_MaxError\_mm, Tools\_MinScore
* Portfolio
  + Generated by macro

## Core utilities and logigram gates

vba

Option Explicit

Function WS(name As String) As Worksheet: Set WS = ThisWorkbook.Worksheets(name): End Function

Function Cfg(key As String, Optional defVal As Variant = "") As Variant

Dim r As Range: Set r = WS("Config").Columns(1).Find(key, , xlValues, xlWhole)

If r Is Nothing Then Cfg = defVal Else Cfg = IIf(IsEmpty(r.Offset(0, 1)), defVal, r.Offset(0, 1))

End Function

Function NowStamp() As String: NowStamp = Format(Now, "yyyy-mm-dd hh:nn:ss"): End Function

Sub LogEvent(exID As String, evt As String, Optional k1 As String = "", Optional k2 As String = "", Optional note As String = "")

Dim w As Worksheet: Set w = WS("Events")

Dim r As Long: r = w.Cells(w.Rows.Count, 1).End(xlUp).Row + 1

w.Cells(r, 1) = NowStamp(): w.Cells(r, 2) = Cfg("CurrentUser", "Trainee")

w.Cells(r, 3) = exID: w.Cells(r, 4) = evt: w.Cells(r, 5) = k1: w.Cells(r, 6) = k2: w.Cells(r, 7) = note

End Sub

Sub RecordMetric(exID As String, metric As String, val As Double, unitStr As String, src As String)

Dim w As Worksheet: Set w = WS("Measurements")

Dim r As Long: r = w.Cells(w.Rows.Count, 1).End(xlUp).Row + 1

w.Cells(r, 1) = exID: w.Cells(r, 2) = metric: w.Cells(r, 3) = val

w.Cells(r, 4) = unitStr: w.Cells(r, 5) = src: w.Cells(r, 6) = NowStamp()

End Sub

Function ModuleEnabled(modID As String) As Boolean

Dim r As Range: Set r = WS("Modules").Columns(1).Find(modID, , xlValues, xlWhole)

ModuleEnabled = Not r Is Nothing And CBool(r.Offset(0, 5).Value)

End Function

Function ModuleDomain(modID As String) As String

Dim r As Range: Set r = WS("Modules").Columns(1).Find(modID, , xlValues, xlWhole)

If r Is Nothing Then ModuleDomain = "" Else ModuleDomain = CStr(r.Offset(0, 4).Value)

End Function

Function SequenceOf(modID As String) As Long

Dim r As Range: Set r = WS("Modules").Columns(1).Find(modID, , xlValues, xlWhole)

If r Is Nothing Then SequenceOf = 9999 Else SequenceOf = CLng(r.Offset(0, 2).Value)

End Function

Function IsModuleCompleted(modID As String) As Boolean

Dim e As Worksheet: Set e = WS("Events")

Dim i As Long, last As Long: last = e.Cells(e.Rows.Count, 1).End(xlUp).Row

For i = last To 2 Step -1

If e.Cells(i, 4).Value = "ModuleCompleted" And e.Cells(i, 5).Value = modID Then IsModuleCompleted = True: Exit Function

Next i

IsModuleCompleted = False

End Function

Function PriorRequiredCompleted(modID As String) As Boolean

Dim m As Worksheet: Set m = WS("Modules")

Dim seq As Long: seq = SequenceOf(modID)

Dim i As Long, last As Long: last = m.Cells(m.Rows.Count, 1).End(xlUp).Row

For i = 2 To last

If CBool(m.Cells(i, 4).Value) Then

If CLng(m.Cells(i, 3).Value) < seq Then

If Not IsModuleCompleted(CStr(m.Cells(i, 1).Value)) Then PriorRequiredCompleted = False: Exit Function

End If

End If

Next i

PriorRequiredCompleted = True

End Function

Function SafetyReady(domain As String, listCSV As String) As Boolean

If Len(Trim(listCSV)) = 0 Then SafetyReady = True: Exit Function

Dim a() As String: a = Split(listCSV, ",")

Dim i As Long, r As Range, item As String

For i = LBound(a) To UBound(a)

item = Trim(a(i))

Set r = WS("Safety").Columns(1).Find(item, , xlValues, xlWhole)

If r Is Nothing Then SafetyReady = False: Exit Function

If LCase(CStr(r.Offset(0, 3).Value)) <> LCase(domain) Then SafetyReady = False: Exit Function

If CBool(r.Offset(0, 1).Value) And Not CBool(r.Offset(0, 2).Value) Then SafetyReady = False: Exit Function

Next i

SafetyReady = True

End Function

Function InventoryReady(reqCSV As String) As Boolean

If Len(Trim(reqCSV)) = 0 Then InventoryReady = True: Exit Function

Dim a() As String: a = Split(reqCSV, ",")

Dim i As Long, r As Range

For i = LBound(a) To UBound(a)

Set r = WS("Inventory").Columns(1).Find(Trim(a(i)), , xlValues, xlWhole)

If r Is Nothing Then InventoryReady = False: Exit Function

If LCase(r.Offset(0, 3).Value) <> "available" Then InventoryReady = False: Exit Function

If Not IsEmpty(r.Offset(0, 4).Value) Then

If Date > CDate(r.Offset(0, 4).Value) Then InventoryReady = False: Exit Function

End If

Next i

InventoryReady = True

End Function

Sub LockInventory(reqCSV As String, lockOn As Boolean)

If Len(Trim(reqCSV)) = 0 Then Exit Sub

Dim a() As String: a = Split(reqCSV, ",")

Dim i As Long, r As Range

For i = LBound(a) To UBound(a)

Set r = WS("Inventory").Columns(1).Find(Trim(a(i)), , xlValues, xlWhole)

If Not r Is Nothing Then r.Offset(0, 3).Value = IIf(lockOn, "InUse", "Available")

Next i

End Sub

## Domain helpers and dispatchers

' -------- Standards & BIS / NEC helpers --------

Function BISKnown(code As String) As Boolean

Dim r As Range: Set r = WS("Standards").Columns(1).Find(code, , xlValues, xlWhole)

BISKnown = Not r Is Nothing

End Function

Function NECSectionKnown(sec As Long) As Boolean

Dim r As Range: Set r = WS("NEC").Columns(1).Find(sec, , xlValues, xlWhole)

NECSectionKnown = Not r Is Nothing

End Function

' -------- Lifting helper: risk/limit check --------

Function LiftWithinLimit(massKg As Double) As Boolean

LiftWithinLimit = (massKg <= CDbl(Cfg("Lifting\_MaxMass\_kg", 50)))

End Function

' -------- Hacksaw helper: recommend teeth per 25 mm --------

Function RecommendTPI25(material As String, thickness\_mm As Double) As Long

' Map to coarse/medium/fine based on thickness and material hardness

Dim hard As Boolean: hard = (LCase(material) = "steel" Or LCase(material) = "brass")

If thickness\_mm >= 6 Then

RecommendTPI25 = IIf(hard, 18, 14) ' coarse

ElseIf thickness\_mm >= 3 Then

RecommendTPI25 = IIf(hard, 24, 18) ' medium

Else

RecommendTPI25 = 32 ' fine/thin sections

End If

End Function

' -------- Marking helper: tolerance check --------

Function MarkWithinTol(err\_mm As Double) As Boolean

MarkWithinTol = (err\_mm <= CDbl(Cfg("Mark\_MaxError\_mm", 0.5)))

End Function

' -------- Dispatcher --------

Sub DispatchStep(exID As String, domain As String, token As String)

Dim parts() As String: parts = Split(token, ":")

Dim stepName As String: stepName = LCase(parts(0))

Dim param As String: If UBound(parts) >= 1 Then param = parts(1) Else param = ""

Select Case UCase(domain)

Case "TOOLS": TOOLS\_Step exID, stepName, param

Case "LIFT": LIFT\_Step exID, stepName, param

Case "CUT": CUT\_Step exID, stepName, param

Case "MARK": MARK\_Step exID, stepName, param

Case "CODE": CODE\_Step exID, stepName, param

Case Else: LogEvent exID, "Warn", "UnknownDomain", domain, token

End Select

End Sub

vba

' -------- TOOLS domain (spanners, drills, maintenance, BIS) --------

Sub TOOLS\_Step(exID As String, stepName As String, param As String)

Select Case stepName

Case "verify\_bis" ' e.g., "BIS 2029"

Dim ok As Boolean: ok = BISKnown(param)

LogEvent exID, "TOOLS", "BIS", param, IIf(ok, "Recognized", "Unknown"))

RecordMetric exID, "BIS\_OK", IIf(ok, 1, 0), "bool", "Standards"

Case "select\_tool" ' e.g., "RingSpanner" or "SocketSpanner"

LogEvent exID, "TOOLS", "SelectTool", param, "Use case logged"

Case "drill\_maint" ' e.g., "Lubricate/SecureBit/CenterPunch/EarthOK"

LogEvent exID, "TOOLS", "DrillMaint", param, "Maintenance step"

Case "quiz\_score" ' numeric %

RecordMetric exID, "Tools\_Quiz\_pct", Val(param), "pct", "Quiz"

Case Else

LogEvent exID, "TOOLS", "Unknown", stepName, param

End Select

End Sub

vba

' -------- LIFT domain (mass, device, path, CG, rollers) --------

Sub LIFT\_Step(exID As String, stepName As String, param As String)

Select Case stepName

Case "assess\_mass"

Dim m As Double: m = Val(param)

RecordMetric exID, "Lift\_Mass\_kg", m, "kg", "Scale"

RecordMetric exID, "Lift\_MassOK", IIf(LiftWithinLimit(m), 1, 0), "bool", "Derived"

Case "device" ' "CraneSlings/Winch/Platform/Rollers"

LogEvent exID, "LIFT", "Device", param, "Selected")

Case "path\_clear" ' 0/1

RecordMetric exID, "Lift\_PathClear", Val(param), "bool", "Assessor"

Case "cg\_ok" ' 0/1

RecordMetric exID, "Lift\_CG\_OK", Val(param), "bool", "Assessor"

Case "corner\_roll" ' 0/1

RecordMetric exID, "Lift\_CornerRoll\_OK", Val(param), "bool", "Assessor"

Case Else

LogEvent exID, "LIFT", "Unknown", stepName, param

End Select

End Sub

' -------- CUT domain (hacksaw blade, pitch, tooth set, kerf) --------

Sub CUT\_Step(exID As String, stepName As String, param As String)

Select Case stepName

Case "blade\_type" ' "All-hard/Flexible"

LogEvent exID, "CUT", "BladeType", param, ""

Case "pitch\_tpi25" ' numeric 14/18/24/32

RecordMetric exID, "Cut\_TPI25", Val(param), "t/25mm", "Spec"

Case "recommend\_pitch" ' e.g., "steel,4.0"

Dim a() As String: a = Split(param, ",")

Dim rec As Long: rec = RecommendTPI25(Trim(a(0)), Val(a(1)))

RecordMetric exID, "Cut\_TPI25\_Rec", rec, "t/25mm", "Advisor"

Case "tooth\_set" ' "Staggered/Wave"

LogEvent exID, "CUT", "ToothSet", param, ""

Case "kerf\_mm"

Dim k As Double: k = Val(param)

RecordMetric exID, "Cut\_Kerf\_mm", k, "mm", "Gauge"

RecordMetric exID, "Cut\_KerfOK", IIf(k <= CDbl(Cfg("Cut\_MaxKerf\_mm", 1.2)), 1, 0), "bool", "Derived"

Case Else

LogEvent exID, "CUT", "Unknown", stepName, param

End Select

End Sub

' -------- MARK domain (media, punches, calipers, accuracy) --------

Sub MARK\_Step(exID As String, stepName As String, param As String)

Select Case stepName

Case "media" ' "Whitewash/CopperSulphate/Lacquer/PrussianBlue"

LogEvent exID, "MARK", "Media", param, ""

Case "punch" ' "Centre/Prick"

LogEvent exID, "MARK", "Punch", param, ""

Case "rule\_size" ' 150/300/600

LogEvent exID, "MARK", "Rule", param, ""

Case "caliper" ' "Inside/Outside/Jenny/Vernier"

LogEvent exID, "MARK", "Caliper", param, ""

Case "mark\_error\_mm"

Dim e As Double: e = Val(param)

RecordMetric exID, "Mark\_Error\_mm", e, "mm", "Micrometer"

RecordMetric exID, "Mark\_TolOK", IIf(MarkWithinTol(e), 1, 0), "bool", "Derived"

Case Else

LogEvent exID, "MARK", "Unknown", stepName, param

End Select

End Sub

' -------- CODE domain (Standards & NEC) --------

Sub CODE\_Step(exID As String, stepName As String, param As String)

Select Case stepName

Case "std\_org" ' e.g., "ISO/BIS/JIS/BSI/DIN/GOST/ASA"

LogEvent exID, "CODE", "StdOrg", param, "Recognized")

Case "bis\_benefit" ' e.g., "Consumers=Safety assurance"

LogEvent exID, "CODE", "BISBenefit", param, ""

Case "nec\_section" ' numeric "7" etc.

Dim known As Boolean: known = NECSectionKnown(CLng(Val(param)))

LogEvent exID, "CODE", "NEC\_Section", param, IIf(known, "Known", "Unknown"))

RecordMetric exID, "NEC\_SecKnown", IIf(known, 1, 0), "bool", "NEC")

Case "nec\_quiz" ' numeric %

RecordMetric exID, "NEC\_Quiz\_pct", Val(param), "pct", "Quiz"

Case Else

LogEvent exID, "CODE", "Unknown", stepName, param

End Select

End Sub

Function StartExercise(exID As String) As Boolean

Dim ex As Range: Set ex = WS("Exercises").Columns(1).Find(exID, , xlValues, xlWhole)

If ex Is Nothing Then MsgBox "Exercise not found", vbExclamation: Exit Function

Dim modID As String: modID = CStr(ex.Offset(0, 1).Value)

Dim domain As String: domain = ModuleDomain(modID)

Dim tools As String: tools = CStr(ex.Offset(0, 6).Value)

Dim safetyList As String: safetyList = CStr(ex.Offset(0, 7).Value)

If Not ModuleEnabled(modID) Then MsgBox "Module disabled", vbExclamation: Exit Function

If Not PriorRequiredCompleted(modID) Then MsgBox "Complete prior required modules", vbExclamation: Exit Function

If Not SafetyReady(domain, safetyList) Then MsgBox "Safety checklist incomplete", vbExclamation: Exit Function

If Not InventoryReady(tools) Then MsgBox "Required items unavailable/expired", vbExclamation: Exit Function

LockInventory tools, True

LogEvent exID, "Started", modID, domain, "Exercise initiated"

StartExercise = True

End Function

Sub RunExercisePrompt()

Dim exID As String: exID = InputBox("Enter ExerciseID:")

If Len(exID) = 0 Then Exit Sub

If Not StartExercise(exID) Then Exit Sub

Dim ex As Range: Set ex = WS("Exercises").Columns(1).Find(exID, , xlValues, xlWhole)

Dim domain As String: domain = UCase(ModuleDomain(CStr(ex.Offset(0, 1).Value)))

Dim steps() As String: steps = Split(CStr(ex.Offset(0, 4).Value), ",")

Dim i As Long

For i = LBound(steps) To UBound(steps)

DispatchStep exID, domain, Trim(steps(i))

Next i

EvaluateExercise exID

LockInventory CStr(ex.Offset(0, 6).Value), False

End Sub

Sub EvaluateExercise(exID As String)

Dim ex As Range: Set ex = WS("Exercises").Columns(1).Find(exID, , xlValues, xlWhole)

Dim crit As String: crit = CStr(ex.Offset(0, 5).Value)

Dim tokens() As String: tokens = Split(crit, ";")

Dim ok As Boolean: ok = True

Dim i As Long

For i = LBound(tokens) To UBound(tokens)

If Len(Trim(tokens(i))) > 0 Then If Not CriterionMet(exID, Trim(tokens(i))) Then ok = False

Next i

LogEvent exID, IIf(ok, "Completed", "Failed"), "", "", IIf(ok, "Pass", "Fail")

GeneratePortfolio exID

End Sub

Function CriterionMet(exID As String, expr As String) As Boolean

Dim op As String

If InStr(expr, ">=") > 0 Then op = ">=" ElseIf InStr(expr, "<=") > 0 Then op = "<=" \_

ElseIf InStr(expr, "==") > 0 Then op = "==" ElseIf InStr(expr, ">") > 0 Then op = ">" \_

ElseIf InStr(expr, "<") > 0 Then op = "<"

If Len(op) = 0 Then CriterionMet = False: Exit Function

Dim parts() As String: parts = Split(expr, op)

Dim metric As String: metric = Trim(parts(0))

Dim target As Double: target = CDbl(Val(Trim(parts(1))))

Dim val As Double: val = LatestMetric(exID, metric)

Select Case op

Case ">=": CriterionMet = (val >= target)

Case "<=": CriterionMet = (val <= target)

Case "==": CriterionMet = (Abs(val - target) < 0.0001)

Case ">": CriterionMet = (val > target)

Case "<": CriterionMet = (val < target)

Case Else: CriterionMet = False

End Select

End Function

Function LatestMetric(exID As String, metric As String) As Double

Dim w As Worksheet: Set w = WS("Measurements")

Dim i As Long, last As Long: last = w.Cells(w.Rows.Count, 1).End(xlUp).Row

For i = last To 2 Step -1

If w.Cells(i, 1) = exID And w.Cells(i, 2) = metric Then LatestMetric = CDbl(w.Cells(i, 3)): Exit Function

Next i

LatestMetric = 0

End Function

vba

Sub GeneratePortfolio(exID As String)

On Error Resume Next: Application.DisplayAlerts = False

ThisWorkbook.Worksheets("Portfolio").Delete

Application.DisplayAlerts = True: On Error GoTo 0

Dim wr As Worksheet: Set wr = ThisWorkbook.Worksheets.Add

wr.Name = "Portfolio"

Dim ex As Range: Set ex = WS("Exercises").Columns(1).Find(exID, , xlValues, xlWhole)

Dim modID As String: modID = CStr(ex.Offset(0, 1).Value)

Dim domain As String: domain = ModuleDomain(modID)

Dim r As Long: r = 1

wr.Cells(r, 1) = "Workshop Evidence": r = r + 2

wr.Cells(r, 1) = "Trainee": wr.Cells(r, 2) = CStr(Cfg("CurrentUser", "Trainee")): r = r + 1

wr.Cells(r, 1) = "ExerciseID": wr.Cells(r, 2) = exID: r = r + 1

wr.Cells(r, 1) = "Module": wr.Cells(r, 2) = modID: r = r + 1

wr.Cells(r, 1) = "Domain": wr.Cells(r, 2) = domain: r = r + 1

wr.Cells(r, 1) = "Objective": wr.Cells(r, 2) = CStr(ex.Offset(0, 3).Value): r = r + 2

r = CopySection(wr, r, "Events", WS("Events"), 3, exID)

r = CopySection(wr, r, "Measurements", WS("Measurements"), 1, exID)

wr.Columns.AutoFit

Dim outDir As String: outDir = Cfg("EvidenceDir", ThisWorkbook.Path)

Dim f As String: f = outDir & "\Portfolio\_" & exID & ".pdf"

On Error Resume Next: wr.ExportAsFixedFormat xlTypePDF, f: On Error GoTo 0

MsgBox "Portfolio generated: " & f, vbInformation

End Sub

Function CopySection(dst As Worksheet, startRow As Long, title As String, src As Worksheet, matchCol As Integer, key As String) As Long

dst.Cells(startRow, 1) = title

Dim r As Long: r = startRow + 1

Dim rng As Range: Set rng = src.Range("A1").CurrentRegion

Dim i As Long, header As Boolean

For i = 2 To rng.Rows.Count

If CStr(rng.Cells(i, matchCol)) = key Then

If Not header Then rng.Rows(1).Copy dst.Cells(r, 1): r = r + 1: header = True

rng.Rows(i).Copy dst.Cells(r, 1): r = r + 1

End If

Next i

CopySection = r + 1

End Function

## Quick seed data and how to run

* Config:
  + CurrentUser = Tshingombe Fiston Tshitadi
  + EvidenceDir = C:\Evidence
  + Lifting\_MaxMass\_kg = 50
  + Cut\_MaxKerf\_mm = 1.2
  + Mark\_MaxError\_mm = 0.5
  + Tools\_MinScore = 70
* Standards:
  + BIS 2029 | Ring Spanner Set | TOOLS | High leverage in tight spaces
  + BIS 7993 | Socket Spanner | TOOLS | Deep/narrow locations
  + BIS 6149 | Adjustable Spanner | TOOLS | Versatile
  + BIS 5169 | Hacksaw Frame | CUT | Cutting metals
  + BIS 2594 | Hacksaw Blade | CUT | Cutting metals
  + BIS 4195 | Pincers | TOOLS | Extracting nails
  + BIS 844 | Screwdrivers | TOOLS | Tighten/loosen screws
  + BIS 5579 | Neon Tester | TOOLS | Check live voltage
* NEC:
  + 1 | Scope of NEC | Coverage and applicability
  + 7 | Design principles for installations | Load calc, diversity, safety factors
  + 9 | Wiring design and construction | Wiring methods, protection
  + 13 | Pre-commissioning tests | Inspection and testing
  + 14 | Earthing requirements | Earthing/grounding practices
* Inventory:
  + T\_RING | Tool | BIS 2029 | Available | 2027-01-01 | TOOLS
  + T\_SOCKET | Tool | BIS 7993 | Available | 2027-01-01 | TOOLS
  + T\_HACKSAW | Tool | BIS 5169 | Available | 2026-06-01 | CUT
  + T\_BLADE24 | Tool | BIS 2594 | Available | 2026-06-01 | CUT
  + PPE\_GLOVES | PPE | — | Available | | TOOLS
* Modules:
  + 1 | Tool Standards & Care | 1 | TRUE | TOOLS | TRUE
  + 2 | Lifting & Handling | 2 | TRUE | LIFT | TRUE
  + 3 | Hacksaw Cutting | 3 | TRUE | CUT | TRUE
  + 4 | Marking & Measurement | 4 | TRUE | MARK | TRUE
  + 5 | Standards & NEC | 5 | TRUE | CODE | TRUE
* Safety:
  + PPE | TRUE | TRUE | TOOLS | Gloves, glasses, boots
  + ClearPath | TRUE | TRUE | LIFT | Barriers placed
  + FireExt Nearby | TRUE | TRUE | CUT | CO2/DP extinguisher
  + EyeProtection | TRUE | TRUE | MARK | Safety glasses
* Exercises:
  + EX\_TOOLS\_01 | 1 | Spanner/BIS Check | Validate standards and drill care | verify\_bis:BIS 2029, select\_tool:RingSpanner, drill\_maint:Lubricate, quiz\_score:82 | BIS\_OK==1;Tools\_Quiz\_pct>=70 | T\_RING,PPE\_GLOVES | PPE
  + EX\_LIFT\_01 | 2 | Move Panel | Plan and execute move | assess\_mass:42, device:Winch, path\_clear:1, cg\_ok:1, corner\_roll:1 | Lift\_MassOK==1;Lift\_PathClear==1;Lift\_CG\_OK==1 | PPE\_GLOVES | ClearPath
  + EX\_CUT\_01 | 3 | Cut Conduit | Choose blade/pitch | blade\_type:Flexible, recommend\_pitch:steel,4.0, pitch\_tpi25:24, tooth\_set:Wave, kerf\_mm:1.0 | Cut\_KerfOK==1 | T\_HACKSAW,T\_BLADE24 | FireExt Nearby
  + EX\_MARK\_01 | 4 | Mark Panel | Accurate layout | media:PrussianBlue, punch:Centre, rule\_size:300, caliper:Vernier, mark\_error\_mm:0.3 | Mark\_TolOK==1 | PPE\_GLOVES | EyeProtection
  + EX\_CODE\_01 | 5 | NEC Recall | Sections and benefits | std\_org:ISO, bis\_benefit:Consumers=Safety assurance, nec\_section:7, nec\_quiz:78 | NEC\_SecKnown==1;NEC\_Quiz\_pct>=70 | |

Run macro RunExercisePrompt, enter ExerciseID (e.g., EX\_CUT\_01). The engine validates gates, executes steps, evaluates criteria, and exports a Portfolio PDF.

If you want, I can add a one-click dashboard with buttons, a blade selection wizard, or an NEC quiz form to make this classroom-ready.

## 🧠 Workbook Schema

Create these sheets:

* **Config**
  + Keys: CurrentUser, EvidenceDir, MaxRatedVoltage\_V, AC\_Freq\_Hz
* **CapacitorSeries**
  + A: CapID, B: Capacitance\_uF, C: RatedVoltage\_V, D: Polarity (NP/P), E: Status (OK/Fail)
* **ACDC**
  + A: Parameter, B: Value, C: Unit, D: Source
* **InductiveAC**
  + A: L\_mH, B: F\_kHz, C: XL\_Ohm, D: I\_A, E: Pq\_VAR
* **Events**
  + A: Timestamp, B: User, C: Topic, D: EventType, E: Detail1, F: Detail2, G: Notes
* **Measurements**
  + A: Topic, B: Metric, C: Value, D: Unit, E: Source, F: Timestamp
* **Portfolio**
  + Generated automatically

## 🔄 Logigram Rules

* Capacitor grouping:
  + Voltage across each capacitor < RatedVoltage\_V
  + Polarity must be maintained for polarized types
* AC waveform:
  + Frequency must match configured AC\_Freq\_Hz
* Inductive reactance:
  + XL = 2πfL
  + Reactive power Pq = I² × XL
* Safety:
  + Flag any capacitor with Vx > RatedVoltage\_V as Fail

## ⚙️ Core VBA Functions

### Utilities

vba

Function WS(name As String) As Worksheet: Set WS = ThisWorkbook.Worksheets(name): End Function

Function Cfg(key As String, Optional defVal As Variant = "") As Variant

Dim r As Range: Set r = WS("Config").Columns(1).Find(key, , xlValues, xlWhole)

If r Is Nothing Then Cfg = defVal Else Cfg = IIf(IsEmpty(r.Offset(0, 1)), defVal, r.Offset(0, 1))

End Function

Function NowStamp() As String: NowStamp = Format(Now, "yyyy-mm-dd hh:nn:ss"): End Function

Sub LogEvent(topic As String, evt As String, Optional d1 As String = "", Optional d2 As String = "", Optional note As String = "")

Dim w As Worksheet: Set w = WS("Events")

Dim r As Long: r = w.Cells(w.Rows.Count, 1).End(xlUp).Row + 1

w.Cells(r, 1) = NowStamp(): w.Cells(r, 2) = Cfg("CurrentUser", "Trainee")

w.Cells(r, 3) = topic: w.Cells(r, 4) = evt: w.Cells(r, 5) = d1: w.Cells(r, 6) = d2: w.Cells(r, 7) = note

End Sub

Sub RecordMetric(topic As String, metric As String, val As Double, unitStr As String, src As String)

Dim w As Worksheet: Set w = WS("Measurements")

Dim r As Long: r = w.Cells(w.Rows.Count, 1).End(xlUp).Row + 1

w.Cells(r, 1) = topic: w.Cells(r, 2) = metric: w.Cells(r, 3) = val

w.Cells(r, 4) = unitStr: w.Cells(r, 5) = src: w.Cells(r, 6) = NowStamp()

End Sub

## 🔌 Series Capacitor Grouping

Sub AnalyzeSeriesCapacitors()

Dim ws As Worksheet: Set ws = WS("CapacitorSeries")

Dim last As Long: last = ws.Cells(ws.Rows.Count, "A").End(xlUp).Row

Dim i As Long, CT As Double, invSum As Double, VS As Double: VS = 25

invSum = 0

For i = 2 To last

invSum = invSum + 1 / ws.Cells(i, 2).Value

Next i

CT = 1 / invSum

RecordMetric "SeriesCaps", "C\_Total\_uF", CT, "uF", "Computed"

For i = 2 To last

Dim Cx As Double: Cx = ws.Cells(i, 2).Value

Dim Vx As Double: Vx = (Cx / CT) \* VS

RecordMetric "SeriesCaps", "V\_" & ws.Cells(i, 1).Value, Vx, "V", "Computed"

If Vx > ws.Cells(i, 3).Value Then

ws.Cells(i, 5).Value = "Fail"

LogEvent "SeriesCaps", "OverVoltage", ws.Cells(i, 1).Value, Vx, "Exceeds rated voltage"

Else

ws.Cells(i, 5).Value = "OK"

End If

Next i

End Sub

⚡ AC/DC Comparison Sub LogACDCComparison()

Dim ws As Worksheet: Set ws = WS("ACDC")

Dim ACfreq As Double: ACfreq = Cfg("AC\_Freq\_Hz", 50)

RecordMetric "ACDC", "AC\_Freq", ACfreq, "Hz", "Config"

LogEvent "ACDC", "DC", "Direction", "One-way", "DC flows in one direction"

LogEvent "ACDC", "AC", "Direction", "Alternates", "AC reverses periodically"

LogEvent "ACDC", "AC", "Waveform", "Sine", "Standard AC waveform"

End Sub

Sub AnalyzeInductiveAC()

Dim ws As Worksheet: Set ws = WS("InductiveAC")

Dim L As Double: L = ws.Cells(2, 1).Value / 1000 ' mH to H

Dim f As Double: f = ws.Cells(2, 2).Value \* 1000 ' kHz to Hz

Dim I As Double: I = ws.Cells(2, 4).Value

Dim XL As Double: XL = 6.28 \* f \* L

Dim Pq As Double: Pq = I ^ 2 \* XL

ws.Cells(2, 3).Value = XL

ws.Cells(2, 5).Value = Pq

RecordMetric "InductiveAC", "XL\_Ohm", XL, "Ω", "Computed"

RecordMetric "InductiveAC", "Pq\_VAR", Pq, "VAR", "Computed"

LogEvent "InductiveAC", "Reactance", "XL", XL, "Inductive opposition"

LogEvent "InductiveAC", "Power", "Reactive", Pq, "VARs in pure inductance"

End Sub

AC Waveform Analysis

Sub AnalyzeACWaveform(Vpeak As Double)

Dim Vrms As Double: Vrms = 0.707 \* Vpeak

Dim Vavg As Double: Vavg = 0.637 \* Vpeak

Dim Vpp As Double: Vpp = 2 \* Vpeak

Dim kf As Double: kf = Vrms / Vavg

RecordMetric "ACWave", "Vpeak", Vpeak, "V", "Input"

RecordMetric "ACWave", "Vrms", Vrms, "V", "Computed"

RecordMetric "ACWave", "Vavg", Vavg, "V", "Computed"

RecordMetric "ACWave", "Vpp", Vpp, "V", "Computed"

RecordMetric "ACWave", "FormFactor", kf, "-", "Computed"

LogEvent "ACWave", "Waveform", "Sine", "", "Standard AC waveform"

End Sub

## 🌍 Earthing Types

vba

Sub LogEarthingTypes()

LogEvent "Earthing", "System", "NeutralBond", "", "Limits voltage under normal conditions"

LogEvent "Earthing", "Equipment", "MetalBond", "", "Protects against shock hazards"

RecordMetric "Earthing", "EarthPotential", 0, "V", "Reference"

End Sub

## 📦 Portfolio Export

vba

Sub GeneratePortfolio(topic As String)

On Error Resume Next: Application.DisplayAlerts = False

ThisWorkbook.Worksheets("Portfolio").Delete

Application.DisplayAlerts = True: On Error GoTo 0

Dim wr As Worksheet: Set wr = ThisWorkbook.Worksheets.Add

wr.Name = "Portfolio"

Dim r As Long: r = 1

wr.Cells(r, 1) = "Electrical Training Evidence": r = r + 2

wr.Cells(r, 1) = "Topic": wr.Cells(r, 2) = topic: r = r + 1

wr.Cells(r, 1) = "Trainee": wr.Cells(r, 2) = Cfg("CurrentUser", "Trainee"): r = r + 2

r = CopySection(wr, r, "Events", WS("Events"), 3, topic

# VBA logigram and algorigram for resonance, admittance, and power factor applications

This modular Excel VBA engine models parallel/series RLC resonance, admittance-based analysis, power factor correction, and frequency-selective tank circuits. It enforces readiness (logigram) and executes learning scenarios (algorigram), logs evidence, and evaluates pass criteria.

## Workbook schema

Create sheets with these exact names and columns.

* Config
  + A: Key, B: Value
  + Keys to seed: CurrentUser, EvidenceDir, LineFreq\_Hz, TargetPF, LineVoltage\_V, RatedCapVoltage\_V
* Scenarios
  + A: ScenarioID, B: Domain (RLC\_PAR/RLC\_SER/PF\_CORR/MATCH/AUDIO), C: Name, D: Objective, E: StepsCSV, F: PassCriteria
* Events
  + A: Timestamp, B: User, C: ScenarioID, D: EventType, E: K1, F: K2, G: Notes
* Measurements
  + A: ScenarioID, B: Metric, C: Value, D: Unit, E: Source, F: Timestamp
* Components
  + A: Name, B: Type (R/L/C/Load), C: Value\_SI, D: QorESR (optional), E: Notes

Tip for StepsCSV tokens (examples below):

* RLC\_PAR: set:R=50, set:L=100e-6, set:C=220e-12, sweep:1e5,5e6,201, compute
* PF\_CORR: set\_load:P=5e3,Q=3e3, set\_line:230,50, correct:0.95
* AUDIO: set:C=100e-9, set:L=1e-2, sweep:20,20000,200, compute

## Logigram rules

* Scenario must have all required component values set before compute.
* For PF\_CORR: LineVoltage\_V and frequency required; target PF in (0,1].
* For resonance sweeps: R, L, C must be positive; sweep frequencies valid ascending range.
* For capacitor voltage check: capacitor reactive voltage ≤ RatedCapVoltage\_V at target operating point.

## Core utilities

vba

Option Explicit

Function WS(name As String) As Worksheet: Set WS = ThisWorkbook.Worksheets(name): End Function

Function Cfg(key As String, Optional defVal As Variant = "") As Variant

Dim r As Range: Set r = WS("Config").Columns(1).Find(key, , xlValues, xlWhole)

If r Is Nothing Then Cfg = defVal Else Cfg = IIf(IsEmpty(r.Offset(0, 1)), defVal, r.Offset(0, 1))

End Function

Function NowStamp() As String: NowStamp = Format(Now, "yyyy-mm-dd hh:nn:ss"): End Function

Sub LogEvent(scn As String, evt As String, Optional k1 As String = "", Optional k2 As String = "", Optional note As String = "")

Dim w As Worksheet: Set w = WS("Events")

Dim r As Long: r = w.Cells(w.Rows.Count, 1).End(xlUp).Row + 1

w.Cells(r, 1) = NowStamp(): w.Cells(r, 2) = Cfg("CurrentUser", "Trainee")

w.Cells(r, 3) = scn: w.Cells(r, 4) = evt: w.Cells(r, 5) = k1: w.Cells(r, 6) = k2: w.Cells(r, 7) = note

End Sub

Sub RecordMetric(scn As String, metric As String, val As Double, unitStr As String, src As String)

Dim w As Worksheet: Set w = WS("Measurements")

Dim r As Long: r = w.Cells(w.Rows.Count, 1).End(xlUp).Row + 1

w.Cells(r, 1) = scn: w.Cells(r, 2) = metric: w.Cells(r, 3) = val

w.Cells(r, 4) = unitStr: w.Cells(r, 5) = src: w.Cells(r, 6) = NowStamp()

End Sub

## RLC math helpers

* Resonant frequency: f0=12πLCf\_0 = \frac{1}{2\pi\sqrt{LC}}
* Quality factor (series): Qs=ω0LRQ\_s = \frac{\omega\_0 L}{R}
* Quality factor (parallel, high-Q): Qp≈R⋅CLQ\_p \approx R \cdot \sqrt{\frac{C}{L}}
* Series impedance: Zs=R+j(ωL−1ωC)Z\_s = R + j(\omega L - \frac{1}{\omega C})
* Parallel admittance: Yp=1R+j(ωC−1ωL)Y\_p = \frac{1}{R} + j(\omega C - \frac{1}{\omega L}), Zp=1/YpZ\_p = 1/Y\_p

vba

Function Zseries(R As Double, L As Double, C As Double, f As Double) As Complex

Dim w As Double: w = 2 \* WorksheetFunction.Pi() \* f

Zseries.Re = R

Zseries.Im = (w \* L) - (1# / (w \* C))

End Function

Function Yparallel(R As Double, L As Double, C As Double, f As Double) As Complex

Dim w As Double: w = 2 \* WorksheetFunction.Pi() \* f

Yparallel.Re = IIf(R > 0, 1# / R, 0)

Yparallel.Im = (w \* C) - (1# / (w \* L))

End Function

Function CplxMag(z As Complex) As Double: CplxMag = Sqr(z.Re \* z.Re + z.Im \* z.Im): End Function

Function CplxInv(z As Complex) As Complex

Dim d As Double: d = z.Re \* z.Re + z.Im \* z.Im

If d = 0 Then CplxInv.Re = 0: CplxInv.Im = 0 Else

CplxInv.Re = z.Re / d: CplxInv.Im = -z.Im / d

End If

End Function

Type Complex

Re As Double

Im As Double

End Type

Function F0(L As Double, C As Double) As Double

If L <= 0 Or C <= 0 Then F0 = 0 Else F0 = 1# / (2 \* WorksheetFunction.Pi() \* Sqr(L \* C))

End Function

Function Qs(R As Double, L As Double, C As Double) As Double

Dim w0 As Double: w0 = 2 \* WorksheetFunction.Pi() \* F0(L, C)

If R <= 0 Or w0 = 0 Then Qs = 0 Else Qs = w0 \* L / R

End Function

Function Qp(R As Double, L As Double, C As Double) As Double

If R <= 0 Or L <= 0 Or C <= 0 Then Qp = 0 Else Qp = R \* Sqr(C / L)

End Function

## Power factor correction helpers

* For a load with real power PP and reactive power QQ at line frequency ff:
  + Initial PF: cos⁡θ1=PP2+Q2\cos\theta\_1 = \frac{P}{\sqrt{P^2 + Q^2}}
  + Target PF: cos⁡θ2\cos\theta\_2 set by user; required capacitive VARs:

Qc=P⋅(tan⁡θ1−tan⁡θ2)Q\_c = P \cdot (\tan\theta\_1 - \tan\theta\_2)

* Capacitor size:

C=QcωV2with ω=2πfC = \frac{Q\_c}{\omega V^2} \quad \text{with } \omega = 2\pi f

* Cap RMS current: Ic=ωCVI\_c = \omega C V

vba

Sub SizePFCapacitor(scn As String, P\_W As Double, Q\_var As Double, V\_rms As Double, f\_Hz As Double, targetPF As Double)

Dim S As Double: S = Sqr(P\_W ^ 2 + Q\_var ^ 2)

Dim cos1 As Double: cos1 = IIf(S > 0, P\_W / S, 1)

Dim th1 As Double: th1 = WorksheetFunction.Acos(cos1)

Dim th2 As Double: th2 = WorksheetFunction.Acos(Application.Max(Application.Min(targetPF, 1), 0))

Dim Qc As Double: Qc = P\_W \* (WorksheetFunction.Tan(th1) - WorksheetFunction.Tan(th2)) ' VAR

Dim w As Double: w = 2 \* WorksheetFunction.Pi() \* f\_Hz

Dim C\_F As Double: If w > 0 And V\_rms > 0 Then C\_F = Qc / (w \* V\_rms ^ 2) Else C\_F = 0

Dim Ic As Double: Ic = w \* C\_F \* V\_rms

RecordMetric scn, "PF\_Initial", cos1, "-", "Computed"

RecordMetric scn, "PF\_Target", targetPF, "-", "Input"

RecordMetric scn, "Qc\_VAR", Qc, "VAR", "Computed"

RecordMetric scn, "Cap\_F", C\_F, "F", "Computed"

RecordMetric scn, "Cap\_Irms\_A", Ic, "A", "Computed"

LogEvent scn, "PF\_CORR", "CapSized", "", "Capacitor sized for PF correction"

End Sub

Scenario dispatcher and domain steps Sub RunScenarioPrompt()

Dim scn As String: scn = InputBox("Enter ScenarioID:")

If Len(scn) = 0 Then Exit Sub

ExecuteScenario scn

End Sub

Sub ExecuteScenario(scn As String)

Dim r As Range: Set r = WS("Scenarios").Columns(1).Find(scn, , xlValues, xlWhole)

If r Is Nothing Then MsgBox "Scenario not found": Exit Sub

Dim domain As String: domain = UCase(CStr(r.Offset(0, 1).Value))

Dim stepsCSV As String: stepsCSV = CStr(r.Offset(0, 4).Value)

Dim steps() As String: steps = Split(stepsCSV, ",")

LogEvent scn, "Started", domain, "", CStr(r.Offset(0, 2).Value)

Dim i As Long

For i = LBound(steps) To UBound(steps)

DispatchStep scn, domain, Trim(steps(i))

Next i

EvaluateScenario scn

End Sub

Sub DispatchStep(scn As String, domain As String, token As String)

Dim parts() As String: parts = Split(token, ":")

Dim cmd As String: cmd = LCase(parts(0))

Dim arg As String: If UBound(parts) >= 1 Then arg = parts(1) Else arg = ""

Select Case domain

Case "RLC\_PAR": RLCpar\_Step scn, cmd, arg

Case "RLC\_SER": RLCser\_Step scn, cmd, arg

Case "PF\_CORR": PFCorr\_Step scn, cmd, arg

Case "MATCH": Match\_Step scn, cmd, arg

Case "AUDIO": Audio\_Step scn, cmd, arg

Case Else: LogEvent scn, "Warn", "UnknownDomain", domain, token

End Select

End Sub

### RLC parallel domain

' Commands:

' set:R=50 | set:L=100e-6 | set:C=220e-12

' sweep:fStart,fStop,N

' compute

Dim Rpar As Double, Lpar As Double, Cpar As Double

Dim fStart As Double, fStop As Double, Npts As Long

Sub RLCpar\_Step(scn As String, cmd As String, arg As String)

Select Case cmd

Case "set"

Dim kv() As String: kv = Split(arg, "=")

Select Case LCase(kv(0))

Case "r": Rpar = CDbl(kv(1))

Case "l": Lpar = CDbl(kv(1))

Case "c": Cpar = CDbl(kv(1))

End Select

LogEvent scn, "RLC\_PAR", "Set", kv(0), kv(1)

Case "sweep"

Dim a() As String: a = Split(arg, ",")

fStart = CDbl(a(0)): fStop = CDbl(a(1)): Npts = CLng(a(2))

LogEvent scn, "RLC\_PAR", "Sweep", arg, ""

Case "compute"

RLCpar\_Compute scn

Case Else

LogEvent scn, "RLC\_PAR", "Unknown", cmd, arg

End Select

End Sub

Sub RLCpar\_Compute(scn As String)

If Rpar <= 0 Or Lpar <= 0 Or Cpar <= 0 Then

LogEvent scn, "Error", "SetRLCFirst", "", "Positive R,L,C required": Exit Sub

End If

Dim f0 As Double: f0 = F0(Lpar, Cpar)

RecordMetric scn, "f0\_Hz", f0, "Hz", "Computed"

RecordMetric scn, "Qp", Qp(Rpar, Lpar, Cpar), "-", "Computed"

If fStart <= 0 Or fStop <= fStart Or Npts < 3 Then

fStart = f0 \* 0.5: fStop = f0 \* 1.5: Npts = 201

End If

Dim i As Long, f As Double, stepF As Double

stepF = (fStop - fStart) / (Npts - 1)

Dim ymax As Double, f\_at\_max As Double, Z As Complex, Y As Complex, Zmag As Double

ymax = -1

For i = 0 To Npts - 1

f = fStart + i \* stepF

Y = Yparallel(Rpar, Lpar, Cpar, f)

Z = CplxInv(Y)

Zmag = CplxMag(Z)

RecordMetric scn, "Zmag@" & Format(f, "0.00"), Zmag, "Ohm", "Sweep"

If Zmag > ymax Then ymax = Zmag: f\_at\_max = f

Next i

RecordMetric scn, "f\_peak\_Hz", f\_at\_max, "Hz", "Sweep"

LogEvent scn, "RLC\_PAR", "Computed", "f0", CStr(f0)

End Sub

Dim Rser As Double, Lser As Double, Cser As Double

Sub RLCser\_Step(scn As String, cmd As String, arg As String)

Select Case cmd

Case "set"

Dim kv() As String: kv = Split(arg, "=")

Select Case LCase(kv(0))

Case "r": Rser = CDbl(kv(1))

Case "l": Lser = CDbl(kv(1))

Case "c": Cser = CDbl(kv(1))

End Select

LogEvent scn, "RLC\_SER", "Set", kv(0), kv(1)

Case "compute"

If Rser <= 0 Or Lser <= 0 Or Cser <= 0 Then

LogEvent scn, "Error", "SetRLCFirst", "", "Positive R,L,C required": Exit Sub

End If

Dim f0 As Double: f0 = F0(Lser, Cser)

RecordMetric scn, "f0\_Hz", f0, "Hz", "Computed"

RecordMetric scn, "Qs", Qs(Rser, Lser, Cser), "-", "Computed"

LogEvent scn, "RLC\_SER", "Computed", "f0", CStr(f0)

Case Else

LogEvent scn, "RLC\_SER", "Unknown", cmd, arg

End Select

End Sub

Power factor correction domain Dim P\_W As Double, Q\_var As Double, V\_line As Double, f\_line As Double, PF\_tgt As Double

Sub PFCorr\_Step(scn As String, cmd As String, arg As String)

Select Case cmd

Case "set\_load" ' P=5000,Q=3000

Dim a1() As String: a1 = Split(arg, ",")

P\_W = CDbl(Split(a1(0), "=")(1))

Q\_var = CDbl(Split(a1(1), "=")(1))

LogEvent scn, "PF\_CORR", "LoadSet", arg, ""

Case "set\_line" ' 230,50

Dim a2() As String: a2 = Split(arg, ",")

V\_line = CDbl(a2(0)): f\_line = CDbl(a2(1))

LogEvent scn, "PF\_CORR", "LineSet", arg, ""

Case "correct" ' 0.95

PF\_tgt = CDbl(arg)

SizePFCapacitor scn, P\_W, Q\_var, V\_line, f\_line, PF\_tgt

Case Else

LogEvent scn, "PF\_CORR", "Unknown", cmd, arg

End Select

End Sub

### Impedance matching and audio domains

vba

' MATCH: simple LC match at target frequency, returns reactances and component values

Sub Match\_Step(scn As String, cmd As String, arg As String)

' cmd: "lmatch" arg: "Rs,Rl,f"

If cmd = "lmatch" Then

Dim a() As String: a = Split(arg, ",")

Dim Rs As Double: Rs = CDbl(a(0))

Dim Rl As Double: Rl = CDbl(a(1))

Dim f As Double: f = CDbl(a(2))

Dim w As Double: w = 2 \* WorksheetFunction.Pi() \* f

Dim Qm As Double, Xs As Double, Xp As Double, L\_H As Double, C\_F As Double

If Rs < Rl Then

Qm = Sqr(Rl / Rs - 1)

Xs = Qm \* Rs

Xp = Rl / Qm

Else

Qm = Sqr(Rs / Rl - 1)

Xs = -Rs / Qm

Xp = -Qm \* Rl

End If

If Xs > 0 Then L\_H = Xs / w Else L\_H = 0

If Xs < 0 Then C\_F = -1# / (w \* Xs) Else C\_F = 0

RecordMetric scn, "Q\_match", Qm, "-", "Computed"

RecordMetric scn, "Xs\_Ohm", Xs, "Ohm", "Computed"

RecordMetric scn, "Xp\_Ohm", Xp, "Ohm", "Computed"

RecordMetric scn, "L\_series\_H", L\_H, "H", "Computed"

RecordMetric scn, "C\_series\_F", C\_F, "F", "Computed"

LogEvent scn, "MATCH", "LMatch", arg, "Computed"

End If

End Sub

' AUDIO: parallel resonance response (tone control, equalizer notch/peak behavior)

Sub Audio\_Step(scn As String, cmd As String, arg As String)

Static Ra As Double, La As Double, Ca As Double

Select Case cmd

Case "set"

Dim kv() As String: kv = Split(arg, "=")

Select Case LCase(kv(0))

Case "r": Ra = CDbl(kv(1))

Case "l": La = CDbl(kv(1))

Case "c": Ca = CDbl(kv(1))

End Select

LogEvent scn, "AUDIO", "Set", kv(0), kv(1)

Case "sweep"

Dim b() As String: b = Split(arg, ",")

Dim f0 As Double, f As Double, N As Long, i As Long

Dim f1 As Double: f1 = CDbl(b(0))

Dim f2 As Double: f2 = CDbl(b(1))

N = CLng(b(2))

f0 = F0(La, Ca)

RecordMetric scn, "f0\_audio\_Hz", f0, "Hz", "Computed"

For i = 0 To N - 1

f = f1 + i \* (f2 - f1) / (N - 1)

Dim Y As Complex: Y = Yparallel(Ra, La, Ca, f)

RecordMetric scn, "Audio\_|Y|@" & Format(f, "0"), CplxMag(Y), "S", "Sweep"

Next i

LogEvent scn, "AUDIO", "Sweep", arg, "Admittance profile"

Case "compute"

RecordMetric scn, "Qp\_audio", Qp(Ra, La, Ca), "-", "Computed"

Case Else

LogEvent scn, "AUDIO", "Unknown", cmd, arg

End Select

End Sub

## Evaluation engine

## Evaluation engine

* Pass criteria syntax: Metric op value, separated by semicolons
  + Examples:
    - Qp>=50
    - f\_peak\_Hz>=9.5e5;f\_peak\_Hz<=1.05e6
    - PF\_Initial<0.9;PF\_Target>=0.95
* Sub EvaluateScenario(scn As String)
* Dim r As Range: Set r = WS("Scenarios").Columns(1).Find(scn, , xlValues, xlWhole)
* Dim crit As String: crit = CStr(r.Offset(0, 5).Value)
* Dim arr() As String: arr = Split(crit, ";")
* Dim i As Long, ok As Boolean: ok = True
* For i = LBound(arr) To UBound(arr)
* If Len(Trim(arr(i))) > 0 Then If Not CriterionMet(scn, Trim(arr(i))) Then ok = False
* Next i
* LogEvent scn, IIf(ok, "Completed", "Failed"), "", "", IIf(ok, "Pass", "Fail")
* End Sub
* Function CriterionMet(scn As String, expr As String) As Boolean
* Dim op As String
* If InStr(expr, ">=") > 0 Then op = ">=" ElseIf InStr(expr, "<=") > 0 Then op = "<=" \_
* ElseIf InStr(expr, "==") > 0 Then op = "==" ElseIf InStr(expr, ">") > 0 Then op = ">" \_
* ElseIf InStr(expr, "<") > 0 Then op = "<"
* If Len(op) = 0 Then CriterionMet = False: Exit Function
* Dim parts() As String: parts = Split(expr, op)
* Dim metric As String: metric = Trim(parts(0))
* Dim target As Double: target = CDbl(Val(Trim(parts(1))))
* Dim val As Double: val = LatestMetric(scn, metric)
* Select Case op
* Case ">=": CriterionMet = (val >= target)
* Case "<=": CriterionMet = (val <= target)
* Case "==": CriterionMet = (Abs(val - target) < 0.0001)
* Case ">": CriterionMet = (val > target)
* Case "<": CriterionMet = (val < target)
* Case Else: CriterionMet = False
* End Select
* End Function
* Function LatestMetric(scn As String, metric As String) As Double
* Dim w As Worksheet: Set w = WS("Measurements")
* Dim i As Long, last As Long: last = w.Cells(w.Rows.Count, 1).End(xlUp).Row
* For i = last To 2 Step -1
* If w.Cells(i, 1).Value = scn And w.Cells(i, 2).Value = metric Then
* LatestMetric = CDbl(w.Cells(i, 3).Value): Exit Function
* End If
* Next i
* LatestMetric = 0
* End Function

## Example scenarios to paste in Scenarios

* SCN\_PAR\_01 | RLC\_PAR | RF tank selectivity | Peak impedance near f0 | set:R=2000, set:L=100e-6, set:C=220e-12, sweep:500000,2000000,301, compute | Qp>=10;f\_peak\_Hz>=900000;f\_peak\_Hz<=1200000
* SCN\_SER\_01 | RLC\_SER | Series notch | Compute f0 and Q | set:R=50, set:L=10e-3, set:C=100e-9, compute | Qs>=5;f0\_Hz>=450;f0\_Hz<=550
* SCN\_PF\_01 | PF\_CORR | Improve PF to 0.95 | Size capacitor | set\_load:P=5000,Q=3000, set\_line:230,50, correct:0.95 | PF\_Target>=0.95;Cap\_F>0;Qc\_VAR>0
* SCN\_MATCH\_01 | MATCH | L-match design | Match Rs to Rl at f | lmatch:50,300,1000000 | Q\_match>0
* SCN\_AUD\_01 | AUDIO | Tone filter sweep | Admittance vs frequency | set:R=1200, set:L=1e-2, set:C=100e-9, sweep:20,20000,200, compute | Qp\_audio>=1;f0\_audio\_Hz>=150;f0\_audio\_Hz<=400

## Quick start

* Config:
  + CurrentUser = Tshingombe Fiston Tshitadi
  + EvidenceDir = C:\Evidence
  + LineFreq\_Hz = 50
  + TargetPF = 0.95
  + LineVoltage\_V = 230
  + RatedCapVoltage\_V = 450
* Components: keep for reference; scenario uses inline “set:” values.
* Run:
  + Press RunScenarioPrompt and enter a ScenarioID (e.g., SCN\_PAR\_01).
  + Review Events and Measurements for audit and teaching.
  + Add a button to call RunScenarioPrompt on your dashboard.

If you want, I can add a one-click PDF portfolio export and a bandwidth/Q estimation that auto-finds -3 dB points around resonance to compute selectivity numerically.

practical diagnostics (algorigram), computes Murray loop fault distance, and logs auditable evidence. It’s built for training, commissioning, and rapid fault response.

## Workbook schema

Create these sheets with the exact columns.

* Config
  + A: Key, B: Value
  + Keys: CurrentUser, EvidenceDir, SoilThermalResist\_Km\_W, AmbientTemp\_C, MaxLoadTemp\_C, CopperResist\_ohm\_m, ConductorCSA\_mm2, RouteDerating\_pct, LayingDepth\_m
* CableCatalog
  + A: CableID, B: Type (PVC/XLPE/PILC/H-type/S.L./OilFilled/GasPressure), C: Voltage\_kV, D: Armoured (Yes/No), E: MaxTemp\_C, F: Notes
* Routes
  + A: RouteID, B: Corridor (Road/Rail/Airport/Substation/Residential), C: Length\_m, D: SoilResist\_Km\_W, E: ParallelUtilities (Gas/Water/Telecom), F: CrossingAngle\_deg, G: MinSeparation\_m
* InstallChecklist
  + A: Item, B: Required (TRUE/FALSE), C: Completed (TRUE/FALSE), D: RouteID, E: Notes
* CableRuns
  + A: RunID, B: RouteID, C: CableID, D: Phases (1/3), E: Armour (Yes/No), F: Depth\_m, G: ThermalBackfill (Yes/No)
* LoopTests
  + A: TestID, B: Method (Murray/Varley/Res-Direct), C: RouteID, D: RunID, E: TotalLength\_m, F: rPer\_m\_ohm, G: P\_ohm, H: Q\_ohm, I: S1\_ohm, J: S2\_ohm, K: MeasNotes
* Events
  + A: Timestamp, B: User, C: Topic, D: EventType, E: K1, F: K2, G: Notes
* Measurements
  + A: Topic, B: Metric, C: Value, D: Unit, E: Source, F: Timestamp
* Portfolio
  + Generated automatically

## Logigram rules

* Cable selection:
  + Voltage\_kV compatible with application; XLPE/PVC for LV/MV; Oil/Gas pressure for 66–230 kV.
  + Armour and laying depth appropriate for corridor and mechanical risk.
* Route viability:
  + CrossingAngle\_deg close to 90° for EMI reduction; MinSeparation\_m adequate from other utilities.
* Installation gating:
  + All Required items in InstallChecklist for the RouteID must be Completed = TRUE.
* Loop test readiness:
  + Known total length and resistance-per-metre for the tested core(s).
  + For Murray loop: ratio arms P and Q provided; continuity verified on return core.

## Core utilities

vba

Option Explicit

Function WS(name As String) As Worksheet: Set WS = ThisWorkbook.Worksheets(name): End Function

Function Cfg(key As String, Optional defVal As Variant = "") As Variant

Dim r As Range: Set r = WS("Config").Columns(1).Find(key, , xlValues, xlWhole)

If r Is Nothing Then Cfg = defVal Else Cfg = IIf(IsEmpty(r.Offset(0, 1)), defVal, r.Offset(0, 1))

End Function

Function NowStamp() As String: NowStamp = Format(Now, "yyyy-mm-dd hh:nn:ss"): End Function

Sub LogEvent(topic As String, evt As String, Optional k1 As String = "", Optional k2 As String = "", Optional note As String = "")

Dim w As Worksheet: Set w = WS("Events")

Dim r As Long: r = w.Cells(w.Rows.Count, 1).End(xlUp).Row + 1

w.Cells(r, 1) = NowStamp(): w.Cells(r, 2) = Cfg("CurrentUser", "Trainee")

w.Cells(r, 3) = topic: w.Cells(r, 4) = evt: w.Cells(r, 5) = k1: w.Cells(r, 6) = k2: w.Cells(r, 7) = note

End Sub

Sub RecordMetric(topic As String, metric As String, val As Double, unitStr As String, src As String)

Dim w As Worksheet: Set w = WS("Measurements")

Dim r As Long: r = w.Cells(w.Rows.Count, 1).End(xlUp).Row + 1

w.Cells(r, 1) = topic: w.Cells(r, 2) = metric: w.Cells(r, 3) = val

w.Cells(r, 4) = unitStr: w.Cells(r, 5) = src: w.Cells(r, 6) = NowStamp()

End Sub

## Cable and route checks

Function IsCableSuitable(cableID As String, required\_kV As Double) As Boolean

Dim r As Range: Set r = WS("CableCatalog").Columns(1).Find(cableID, , xlValues, xlWhole)

If r Is Nothing Then IsCableSuitable = False: Exit Function

Dim typ As String: typ = LCase(r.Offset(0, 1).Value)

Dim kV As Double: kV = r.Offset(0, 2).Value

If required\_kV > kV Then IsCableSuitable = False: Exit Function

If required\_kV <= 33 And (typ = "xlpe" Or typ = "pvc") Then IsCableSuitable = True Else \_

If required\_kV >= 66 And (typ = "oilfilled" Or typ = "gaspressure") Then IsCableSuitable = True Else \_

IsCableSuitable = (required\_kV <= kV)

End Function

Function InstallChecklistOK(routeID As String) As Boolean

Dim ws As Worksheet: Set ws = WS("InstallChecklist")

Dim last As Long: last = ws.Cells(ws.Rows.Count, 1).End(xlUp).Row

Dim i As Long

For i = 2 To last

If ws.Cells(i, 4).Value = routeID Then

If CBool(ws.Cells(i, 2).Value) And Not CBool(ws.Cells(i, 3).Value) Then

InstallChecklistOK = False: Exit Function

End If

End If

Next i

InstallChecklistOK = True

End Function

Function RouteEMIOK(routeID As String) As Boolean

Dim r As Range: Set r = WS("Routes").Columns(1).Find(routeID, , xlValues, xlWhole)

If r Is Nothing Then RouteEMIOK = False: Exit Function

Dim ang As Double: ang = r.Offset(0, 5).Value ' CrossingAngle\_deg

Dim sep As Double: sep = r.Offset(0, 6).Value ' MinSeparation\_m

RouteEMIOK = (ang >= 70 And ang <= 110 And sep >= 0.3)

End Function

## Murray loop test engine

Mathematics summary:

* With ratio arms P and Q at balance: P/Q = R/X.
* Total loop resistance Rt = R + X (fault path out + return path to fault).
* Solve: R = Rt · P/(P + Q), X = Rt · Q/(P + Q).
* Distance to fault l1 = X / r\_per\_m, where r\_per\_m is per-core resistance per metre.
* Sub RunMurrayLoop(testID As String)
* Dim t As Range: Set t = WS("LoopTests").Columns(1).Find(testID, , xlValues, xlWhole)
* If t Is Nothing Then MsgBox "Test not found": Exit Sub
* Dim Rt As Double: Rt = TotalLoopResistance(testID)
* Dim rPer As Double: rPer = t.Offset(0, 5).Value ' ohm/m per core
* Dim P As Double: P = t.Offset(0, 6).Value
* Dim Q As Double: Q = t.Offset(0, 7).Value
* Dim routeID As String: routeID = t.Offset(0, 2).Value
* If Rt <= 0 Or rPer <= 0 Or P <= 0 Or Q <= 0 Then
* LogEvent testID, "Error", "Inputs", "", "Provide Rt, rPer, P, Q": Exit Sub
* End If
* If Not InstallChecklistOK(routeID) Then
* LogEvent testID, "Warn", "InstallChecklist", routeID, "Some items incomplete"
* End If
* Dim R As Double, X As Double
* R = Rt \* P / (P + Q)
* X = Rt \* Q / (P + Q)
* Dim l1 As Double: l1 = X / rPer
* RecordMetric testID, "Murray\_Rt\_ohm", Rt, "ohm", "Computed/Measured"
* RecordMetric testID, "Murray\_R\_ohm", R, "ohm", "Computed"
* RecordMetric testID, "Murray\_X\_ohm", X, "ohm", "Computed"
* RecordMetric testID, "Murray\_Dist\_m", l1, "m", "Computed"
* LogEvent testID, "Murray", "Result", CStr(l1), "Distance to fault (m)"
* End Sub
* Function TotalLoopResistance(testID As String) As Double
* ' Option A: measured and entered in MeasNotes e.g., "Rt=3.42"
* ' Option B: compute from length and r\_per\_m if not measured
* Dim t As Range: Set t = WS("LoopTests").Columns(1).Find(testID, , xlValues, xlWhole)
* Dim note As String: note = CStr(t.Offset(0, 10).Value)
* Dim pos As Long: pos = InStr(1, note, "Rt=")
* If pos > 0 Then
* TotalLoopResistance = Val(Mid$(note, pos + 3))
* Exit Function
* End If
* Dim L As Double: L = t.Offset(0, 4).Value ' TotalLength\_m (one-way)
* Dim rPer As Double: rPer = t.Offset(0, 5).Value ' per core
* If L > 0 And rPer > 0 Then
* ' Loop includes out (faulty core to fault) + return (sound core to fault), same length to fault,
* ' but Rt here must be up to the fault. If unknown, assume worst-case at full length:
* TotalLoopResistance = 2 \* L \* rPer
* Else
* TotalLoopResistance = 0
* End If
* End Function

## Varley loop placeholder (measured resistance method)

* Varley setups vary in arm placement. To avoid incorrect assu
* Sub RunVarleyLoop(testID As String)
* Dim t As Range: Set t = WS("LoopTests").Columns(1).Find(testID, , xlValues, xlWhole)
* If t Is Nothing Then MsgBox "Test not found": Exit Sub
* Dim rPer As Double: rPer = t.Offset(0, 5).Value
* Dim Rx As Double: Rx = ExtractNoteVal(t.Offset(0, 10).Value, "Rx")
* If rPer <= 0 Or Rx <= 0 Then
* LogEvent testID, "Error", "Inputs", "", "Provide rPer and Rx in MeasNotes": Exit Sub
* End If
* Dim l1 As Double: l1 = Rx / rPer
* RecordMetric testID, "Varley\_Rx\_ohm", Rx, "ohm", "Measured"
* RecordMetric testID, "Varley\_Dist\_m", l1, "m", "Computed"
* LogEvent testID, "Varley", "Result", CStr(l1), "Distance to fault (m)"
* End Sub
* Function ExtractNoteVal(notes As String, key As String) As Double
* Dim pat As String: pat = key & "="
* Dim p As Long: p = InStr(1, notes, pat, vbTextCompare)
* If p > 0 Then ExtractNoteVal = Val(Mid$(notes, p + Len(pat))) Else ExtractNoteVal = 0
* End Function

## Installation algorigram: start-to-expor

* Sub ValidateInstallAndRun(routeID As String, runID As String, required\_kV As Double, cableID As String)
* If Not IsCableSuitable(cableID, required\_kV) Then
* LogEvent runID, "Error", "CableSuitability", cableID, "Cable not suitable for voltage": Exit Sub
* End If
* If Not RouteEMIOK(routeID) Then
* LogEvent runID, "Warn", "RouteEMI", routeID, "Crossing angle or separation suboptimal"
* End If
* If Not InstallChecklistOK(routeID) Then
* LogEvent runID, "Error", "Checklist", routeID, "Install checklist incomplete": Exit Sub
* End If
* LogEvent runID, "Install", "Validated", cableID, "Route and cable OK"
* End Sub

## Portfolio export

* vba
* Sub ExportPortfolio(topic As String)
* On Error Resume Next: Application.DisplayAlerts = False
* ThisWorkbook.Worksheets("Portfolio").Delete
* Application.DisplayAlerts = True: On Error GoTo 0
* Dim wr As Worksheet: Set wr = ThisWorkbook.Worksheets.Add
* wr.Name = "Portfolio"
* Dim r As Long: r = 1
* wr.Cells(r, 1) = "Underground Cable Evidence": r = r + 2
* wr.Cells(r, 1) = "Topic": wr.Cells(r, 2) = topic: r = r + 1
* wr.Cells(r, 1) = "Trainee": wr.Cells(r, 2) = Cfg("CurrentUser", "Trainee"): r = r + 2
* r = CopySection(wr, r, "Events", WS("Events"), 3, topic)
* r = CopySection(wr, r, "Measurements", WS("Measurements"), 1, topic)
* wr.Columns.AutoFit
* Dim outDir As String: outDir = Cfg("EvidenceDir", ThisWorkbook.Path)
* Dim f As String: f = outDir & "\Portfolio\_" & topic & ".pdf"
* On Error Resume Next: wr.ExportAsFixedFormat xlTypePDF, f: On Error GoTo 0
* MsgBox "Portfolio generated: " & f, vbInformation
* End Sub
* Function CopySection(dst As Worksheet, startRow As Long, title As String, src As Worksheet, matchCol As Integer, key As String) As Long
* dst.Cells(startRow, 1) = title
* Dim r As Long: r = startRow + 1, rng As Range, i As Long, header As Boolean
* Set rng = src.Range("A1").CurrentRegion
* For i = 2 To rng.Rows.Count
* If CStr(rng.Cells(i, matchCol)) = key Then
* If Not header Then rng.Rows(1).Copy dst.Cells(r, 1): r = r + 1: header = True
* rng.Rows(i).Copy dst.Cells(r, 1): r = r + 1
* End If
* Next i
* CopySection = r + 1
* End Function

## Quick seed data and runs

* Config
  + CurrentUser = Tshingombe Fiston Tshitadi
  + EvidenceDir = C:\Evidence
  + CopperResist\_ohm\_m = 1.724e-8
  + ConductorCSA\_mm2 = 95
* CableCatalog examples
  + XLPE\_11kV | XLPE | 11 | Yes | 90 | MV distribution
  + PVC\_1kV | PVC | 1.1 | Yes | 70 | Internal services
  + OIL\_132kV | OilFilled | 132 | No | 65 | HV transmission
  + GAS\_132kV | GasPressure | 132 | No | 65 | HV, flame quenching
* Routes example
  + R01 | Road | 850 | 1.2 | Gas,Water | 90 | 0.5
* InstallChecklist (for R01)
  + Trench depth OK | TRUE | TRUE | R01 | ≥0.8 m
  + Separation verified | TRUE | TRUE | R01 | ≥0.5 m
  + Armour continuity | TRUE | TRUE | R01 | Bonded both ends
* LoopTests (Murray)
  + T\_MUR\_01 | Murray | R01 | RUN01 | 850 | 0.00019 | 100 | 100 | | | Meas Rt=32.3
* LoopTests (Varley placeholder)
  + T\_VAR\_01 | Varley | R01 | RUN01 | 850 | 0.00019 | | | | | Meas Rx=4.7

Run:

* Call ValidateInstallAndRun "R01","RUN01",11,"XLPE\_11kV"
* Call RunMurrayLoop "T\_MUR\_01"
* Call RunVarleyLoop "T\_VAR\_01"
* Call ExportPortfolio "T\_MUR\_01"

## Optional extensions

* Add a “CableThermal” sheet and IEC 60287-based ampacity calculator with soil/backfill derating.
* Add joint positioning logic: maximum span between joints, accessible pit markers logged to Events.
* Add right-angle crossing validator for corridor-specific minimum angles and separation rules.
* Add TDR reading importer to compute distance from time-of-flight with velocity factor.

If you want, I’ll wire a one-click dashboard with buttons for Install Validate, Murray, Varley, and Export, plus a minimal form to capture measured Rt and Rx during field diagnostics.

# VBA logigram and algorigram for power, energy, load calculation, and KCL/KVL checks

This modular Excel VBA engine turns your power/energy formulas and circuit laws into auditable workflows. It validates inputs (logigram), runs calculations and circuit checks (algorigram), logs evidence, and can export a portfolio.

## Workbook schema

Create these sheets with exact columns.

* Config
  + A: Key, B: Value
  + Keys: CurrentUser, EvidenceDir, BillingDays, Tariff\_per\_kWh, KCL\_Tolerance\_A, KVL\_Tolerance\_V
* Devices
  + A: Device, B: Power\_W, C: Qty, D: HoursPerDay, E: Energy\_kWh\_day (output), F: Notes
* Circuits
  + A: CaseID, B: Type (Series/Parallel/Mixed/Loop2), C: V\_supply, D: R1, E: R2, F: R3, G: R4, H: R2\_parallel\_R3 (opt), I: ResultNotes
* KCL
  + A: NodeID, B: Currents\_In\_CSV, C: Currents\_Out\_CSV, D: SumIn\_A, E: SumOut\_A, F: Balanced (Yes/No), G: Notes
* KVL
  + A: LoopID, B: Sources\_CSV, C: Drops\_CSV, D: SumSources\_V, E: SumDrops\_V, F: Balanced (Yes/No), G: Notes
* Events
  + A: Timestamp, B: User, C: Topic, D: EventType, E: K1, F: K2, G: Notes
* Measurements
  + A: Topic, B: Metric, C: Value, D: Unit, E: Source, F: Timestamp
* Portfolio
  + Generated automatically

Tip: Use decimal numbers; do not include units in numeric cells.

## Core utilities

Option Explicit

Function WS(name As String) As Worksheet: Set WS = ThisWorkbook.Worksheets(name): End Function

Function Cfg(key As String, Optional defVal As Variant = "") As Variant

Dim r As Range: Set r = WS("Config").Columns(1).Find(key, , xlValues, xlWhole)

If r Is Nothing Then Cfg = defVal Else Cfg = IIf(IsEmpty(r.Offset(0, 1)), defVal, r.Offset(0, 1))

End Function

Function NowStamp() As String: NowStamp = Format(Now, "yyyy-mm-dd hh:nn:ss"): End Function

Sub LogEvent(topic As String, evt As String, Optional k1 As String = "", Optional k2 As String = "", Optional note As String = "")

Dim w As Worksheet: Set w = WS("Events")

Dim r As Long: r = w.Cells(w.Rows.Count, 1).End(xlUp).Row + 1

w.Cells(r, 1) = NowStamp(): w.Cells(r, 2) = Cfg("CurrentUser", "Trainee")

w.Cells(r, 3) = topic: w.Cells(r, 4) = evt: w.Cells(r, 5) = k1: w.Cells(r, 6) = k2: w.Cells(r, 7) = note

End Sub

Sub RecordMetric(topic As String, metric As String, val As Double, unitStr As String, src As String)

Dim w As Worksheet: Set w = WS("Measurements")

Dim r As Long: r = w.Cells(w.Rows.Count, 1).End(xlUp).Row + 1

w.Cells(r, 1) = topic: w.Cells(r, 2) = metric: w.Cells(r, 3) = val

w.Cells(r, 4) = unitStr: w.Cells(r, 5) = src: w.Cells(r, 6) = NowStamp()

End Sub

## Power and energy calculators

Implements P = V×I, P = I^2×R, P = V^2/R and E = P×t.

vba

Function P\_from\_VI(V As Double, I As Double) As Double: P\_from\_VI = V \* I: End Function

Function P\_from\_IR(I As Double, R As Double) As Double: P\_from\_IR = I ^ 2 \* R: End Function

Function P\_from\_VR(V As Double, R As Double) As Double: If R <> 0 Then P\_from\_VR = (V ^ 2) / R Else P\_from\_VR = 0: End Function

Function E\_kWh(P\_W As Double, hours As Double) As Double: E\_kWh = (P\_W \* hours) / 1000#: End Function

## Daily load, monthly energy, and cost

Reads Devices sheet, computes per-device and total kWh/day, then monthly and cost.

vba

Sub ComputeDailyLoad()

Dim ws As Worksheet: Set ws = WS("Devices")

Dim last As Long: last = ws.Cells(ws.Rows.Count, "A").End(xlUp).Row

Dim i As Long, total\_kWh As Double: total\_kWh = 0

For i = 2 To last

Dim P As Double, q As Double, h As Double

P = Val(ws.Cells(i, 2).Value): q = Val(ws.Cells(i, 3).Value): h = Val(ws.Cells(i, 4).Value)

If P < 0 Or q < 0 Or h < 0 Then

ws.Cells(i, 6).Value = "Invalid input": GoTo NextRow

End If

Dim e As Double: e = E\_kWh(P \* q, h)

ws.Cells(i, 5).Value = e

total\_kWh = total\_kWh + e

NextRow:

Next i

RecordMetric "LoadCalc", "Energy\_day\_kWh", total\_kWh, "kWh", "Devices"

Dim days As Double: days = CDbl(Cfg("BillingDays", 30))

Dim monthly As Double: monthly = total\_kWh \* days

Dim tariff As Double: tariff = CDbl(Cfg("Tariff\_per\_kWh", 1.5))

Dim cost As Double: cost = monthly \* tariff

RecordMetric "LoadCalc", "Energy\_month\_kWh", monthly, "kWh", "Computed"

RecordMetric "LoadCalc", "MonthlyCost", cost, "currency", "Computed"

LogEvent "LoadCalc", "Completed", "kWh\_day=" & Format(total\_kWh, "0.###"), "kWh\_month=" & Format(monthly, "0.###"), "Tariff=" & tariff

End Sub

Example alignment with your sample:

* Total/day ≈ 5.24 kWh
* Monthly (31 days) ≈ 162.44 kWh
* Cost at 1.50 per kWh ≈ 243.66

Set Config BillingDays=31 and Tariff\_per\_kWh=1.5 to reproduce.

## Series/mixed circuit solver (example-friendly)

Supports your series example: R1=36Ω, R2||R3=24Ω, R4=50Ω, V=220V.

vba

Function ParallelR(Ra As Double, Rb As Double) As Double

If Ra <= 0 Or Rb <= 0 Then ParallelR = 0 Else ParallelR = (Ra \* Rb) / (Ra + Rb)

End Function

Sub SolveSeriesMixed(caseRow As Long)

Dim ws As Worksheet: Set ws = WS("Circuits")

Dim V As Double: V = Val(ws.Cells(caseRow, 3).Value)

Dim R1 As Double: R1 = Val(ws.Cells(caseRow, 4).Value)

Dim R2 As Double: R2 = Val(ws.Cells(caseRow, 5).Value)

Dim R3 As Double: R3 = Val(ws.Cells(caseRow, 6).Value)

Dim R4 As Double: R4 = Val(ws.Cells(caseRow, 7).Value)

Dim R23 As Double

If ws.Cells(caseRow, 2).Value = "Series" Then

' Treat R2 cell as R2||R3 already combined (or leave R3 zero)

R23 = IIf(R3 > 0, ParallelR(R2, R3), R2)

Else

' Mixed: combine as parallel in H column if provided

R23 = IIf(ws.Cells(caseRow, 8).Value <> "", Val(ws.Cells(caseRow, 8).Value), IIf(R3 > 0, ParallelR(R2, R3), R2))

End If

Dim Rtot As Double: Rtot = R1 + R23 + R4

If Rtot <= 0 Then ws.Cells(caseRow, 9).Value = "Invalid Rtot": Exit Sub

Dim I As Double: I = V / Rtot

Dim V1 As Double: V1 = I \* R1

Dim V23 As Double: V23 = I \* R23

Dim V4 As Double: V4 = I \* R4

' Log measurements

Dim tag As String: tag = "SeriesCase\_" & ws.Cells(caseRow, 1).Value

RecordMetric tag, "Rtot\_Ohm", Rtot, "Ohm", "Computed"

RecordMetric tag, "I\_A", I, "A", "Computed"

RecordMetric tag, "V1\_V", V1, "V", "Computed"

RecordMetric tag, "V23\_V", V23, "V", "Computed"

RecordMetric tag, "V4\_V", V4, "V", "Computed"

LogEvent tag, "Solved", "V", CStr(V), "Series/Mixed solution"

' Human-readable result

ws.Cells(caseRow, 9).Value = "I=" & Format(I, "0.###") & "A; V1=" & Format(V1, "0.###") & "V; V23=" & Format(V23, "0.###") & "V; V4=" & Format(V4, "0.###") & "V"

End Sub

Sub SolveSeriesExample()

' Find row by CaseID or use row 2

SolveSeriesMixed 2

End Sub

This reproduces your example values (I=2 A; V1=72 V; V2||3=48 V; V4=100 V) when V=220, R1=36, R2||R3=24, R4=50.

## KCL checker (node balance)

Currents in and out are comma-separated values in amperes.

vba

Sub CheckKCLAll()

Dim ws As Worksheet: Set ws = WS("KCL")

Dim last As Long: last = ws.Cells(ws.Rows.Count, 1).End(xlUp).Row

Dim i As Long, tol As Double: tol = CDbl(Cfg("KCL\_Tolerance\_A", 0.01))

For i = 2 To last

Dim sin As Double: sin = SumCSV(ws.Cells(i, 2).Value)

Dim sout As Double: sout = SumCSV(ws.Cells(i, 3).Value)

ws.Cells(i, 4).Value = sin

ws.Cells(i, 5).Value = sout

Dim ok As Boolean: ok = (Abs(sin - sout) <= tol)

ws.Cells(i, 6).Value = IIf(ok, "Yes", "No")

ws.Cells(i, 7).Value = "Δ=" & Format(sin - sout, "0.000")

Dim tag As String: tag = "KCL\_Node\_" & ws.Cells(i, 1).Value

RecordMetric tag, "SumIn\_A", sin, "A", "Entry"

RecordMetric tag, "SumOut\_A", sout, "A", "Entry"

LogEvent tag, "Check", "Balanced", IIf(ok, "Yes", "No"), "Tol=" & tol

Next i

End Sub

Function SumCSV(s As String) As Double

Dim arr() As String, i As Long, tot As Double

If Len(Trim$(s)) = 0 Then SumCSV = 0: Exit Function

arr = Split(s, ",")

For i = LBound(arr) To UBound(arr)

tot = tot + Val(Trim$(arr(i)))

Next i

SumCSV = tot

End Function

## KVL checker (loop balance)

Sources\_CSV and Drops\_CSV are comma-separated voltages. Balanced if sum sources ≈ sum drops.

Sub CheckKVLAll()

Dim ws As Worksheet: Set ws = WS("KVL")

Dim last As Long: last = ws.Cells(ws.Rows.Count, 1).End(xlUp).Row

Dim i As Long, tol As Double: tol = CDbl(Cfg("KVL\_Tolerance\_V", 0.1))

For i = 2 To last

Dim ssrc As Double: ssrc = SumCSV(ws.Cells(i, 2).Value)

Dim sdrop As Double: sdrop = SumCSV(ws.Cells(i, 3).Value)

ws.Cells(i, 4).Value = ssrc

ws.Cells(i, 5).Value = sdrop

Dim ok As Boolean: ok = (Abs(ssrc - sdrop) <= tol)

ws.Cells(i, 6).Value = IIf(ok, "Yes", "No")

ws.Cells(i, 7).Value = "Δ=" & Format(ssrc - sdrop, "0.000")

Dim tag As String: tag = "KVL\_Loop\_" & ws.Cells(i, 1).Value

RecordMetric tag, "SumSources\_V", ssrc, "V", "Entry"

RecordMetric tag, "SumDrops\_V", sdrop, "V", "Entry"

LogEvent tag, "Check", "Balanced", IIf(ok, "Yes", "No"), "Tol=" & tol

Next i

End Sub

Function Solve2x2(A11 As Double, A12 As Double, A21 As Double, A22 As Double, b1 As Double, b2 As Double) As Variant

Dim det As Double: det = A11 \* A22 - A12 \* A21

If Abs(det) < 1E-9 Then Solve2x2 = Array(CVErr(xlErrDiv0), CVErr(xlErrDiv0)): Exit Function

Dim I1 As Double: I1 = (b1 \* A22 - A12 \* b2) / det

Dim I2 As Double: I2 = (A11 \* b2 - b1 \* A21) / det

Solve2x2 = Array(I1, I2)

End Function

Sub SolveDualBatteryExample()

' Example equations:

' 6\*I1 + 5\*I2 = 6

' 5\*I1 + 7\*I2 = 9

Dim sol As Variant: sol = Solve2x2(6, 5, 5, 7, 6, 9)

Dim I1 As Double: I1 = sol(0)

Dim I2 As Double: I2 = sol(1)

RecordMetric "Loop2", "I1\_A", I1, "A", "Solve2x2"

RecordMetric "Loop2", "I2\_A", I2, "A", "Solve2x2"

LogEvent "Loop2", "Solved", "I1", Format(I1, "0.000"), "I2=" & Format(I2, "0.000")

End Sub

Note: Signs depend on your assumed current directions; negative indicates opposite to assumption.

## Portfolio export (optional)

vba

Sub ExportPortfolio(topic As String)

On Error Resume Next: Application.DisplayAlerts = False

ThisWorkbook.Worksheets("Portfolio").Delete

Application.DisplayAlerts = True: On Error GoTo 0

Dim wr As Worksheet: Set wr = ThisWorkbook.Worksheets.Add

wr.Name = "Portfolio"

Dim r As Long: r = 1

wr.Cells(r, 1) = "Electrical Evidence Pack": r = r + 2

wr.Cells(r, 1) = "Topic": wr.Cells(r, 2) = topic: r = r + 1

wr.Cells(r, 1) = "User": wr.Cells(r, 2) = Cfg("CurrentUser", "Trainee"): r = r + 2

r = CopySection(wr, r, "Events", WS("Events"), 3, topic)

r = CopySection(wr, r, "Measurements", WS("Measurements"), 1, topic)

wr.Columns.AutoFit

Dim outDir As String: outDir = Cfg("EvidenceDir", ThisWorkbook.Path)

Dim f As String: f = outDir & "\Portfolio\_" & topic & ".pdf"

On Error Resume Next: wr.ExportAsFixedFormat xlTypePDF, f: On Error GoTo 0

MsgBox "Portfolio generated: " & f, vbInformation

End Sub

Function CopySection(dst As Worksheet, startRow As Long, title As String, src As Worksheet, matchCol As Integer, key As String) As Long

dst.Cells(startRow, 1) = title

Dim r As Long: r = startRow + 1

Dim rng As Range: Set rng = src.Range("A1").CurrentRegion

Dim i As Long, header As Boolean

For i = 2 To rng.Rows.Count

If CStr(rng.Cells(i, matchCol)) = key Then

If Not header Then rng.Rows(1).Copy dst.Cells(r, 1): r = r + 1: header = True

rng.Rows(i).Copy dst.Cells(r, 1): r = r + 1

End If

Next i

CopySection = r + 1

End Function

## Quick start

* Config:
  + CurrentUser = Tshingombe Fiston Tshitadi
  + EvidenceDir = C:\Evidence
  + BillingDays = 31
  + Tariff\_per\_kWh = 1.5
  + KCL\_Tolerance\_A = 0.01
  + KVL\_Tolerance\_V = 0.1
* Devices (examples):
  + Tube Lights | 40 | 5 | 5
  + Fans | 80 | 4 | 8
  + TV | 120 | 1 | 6
  + Lamps | 60 | 4 | 4
* Circuits row for series example:
  + CaseID=EX1, Type=Series, V\_supply=220, R1=36, R2=48, R3=48, R4=50, R2\_parallel\_R3=24

Run in order:

1. ComputeDailyLoad
2. SolveSeriesExample (or SolveSeriesMixed on the row)
3. CheckKCLAll (populate KCL sheet with node currents)
4. CheckKVLAll (populate KVL sheet with loop voltages)
5. ExportPortfolio "LoadCalc" (or another topic tag you used)

If you want, I’ll add a small dashboard with buttons and tiles for “Daily kWh”, “Monthly kWh”, “Cost”, plus pass/fail indicators for KCL/KVL balances and the series example — ready for classroom or assessment use.

# VBA logigram and algorigram for responsive curriculum, evidence, and portfolio-driven reform

This Excel VBA engine operationalizes your reform blueprint: it enforces readiness gates (logigram), orchestrates delivery flows (algorigram), captures audit evidence, and exports a printable portfolio for DHET/SAQA/TVET submissions and expos.

## Workbook schema

Create these sheets with the exact column headers (row 1). The code references these names/columns.

* Config
  + A: Key, B: Value
  + Keys to seed: CurrentUser, EvidenceDir, AcademicYear, MinPsychoCoverage\_pct, MinPortfolioArtifacts, MinPLC\_Hours, MinSCADA\_Hours, MinIoT\_Hours, MinIndustryHours, MinExhibitionProjects, Policy\_PassMark\_pct
* Components
  + A: ComponentID, B: Area (CareerDiscovery/TradeTheory/ControlSystems/Portfolio/JobReadiness/Exhibition), C: CurrentStatus (Low/Medium/High), D: Enhancement (text), E: Weight\_pct, F: Owner, G: DueDate, H: KPI\_Metric, I: KPI\_Target, J: KPI\_Actual
* Modules
  + A: ModuleID, B: Title, C: Level (N1–N6/NCV/Short), D: Credits, E: PrereqIDs (CSV), F: Domain (Electrical/ICT/Control), G: Enabled (TRUE/FALSE)
* Activities
  + A: ActivityID, B: ModuleID, C: Type (Lecture/Lab/Project/Assessment/Industry/Expo), D: Hours, E: Outcomes (CSV), F: Standards (NEC/ISO/BIS/SAQA IDs), G: EvidenceType (Doc/Photo/Video/Code/Log), H: Required (TRUE/FALSE)
* Evidence
  + A: EvidenceID, B: ActivityID, C: LearnerID, D: Type (Doc/Photo/Video/Code/Log), E: URI\_or\_Path, F: Timestamp, G: Verified (TRUE/FALSE), H: Verifier, I: Notes
* Psychometrics
  + A: LearnerID, B: Tool (e.g., Maree/CAS), C: Date, D: InterestCluster, E: Strengths, F: RiskFlags, G: Coverage (pct)
* Industry
  + A: PlacementID, B: LearnerID, C: Partner, D: Hours, E: StartDate, F: EndDate, G: Supervisor, H: Verified (TRUE/FALSE)
* Exhibitions
  + A: ProjectID, B: LearnerID, C: Title, D: Category, E: Artifacts (CSV EvidenceIDs), F: JuryScore\_pct, G: Award (if any), H: Presented (TRUE/FALSE)
* Assessments
  + A: AssessmentID, B: ModuleID, C: Type (Quiz/Prac/Theory/OSCE), D: Date, E: PassMark\_pct, F: MaxMark, G: LearnerID, H: Score, I: Passed (TRUE/FALSE)
* Events
  + A: Timestamp, B: User, C: Topic, D: EventType, E: K1, F: K2, G: Notes
* Portfolio
  + Generated automatically (no manual columns)

## Logigram gates

* Component readiness:
  + CareerDiscovery: Coverage ≥ MinPsychoCoverage\_pct across cohort.
  + ControlSystems: Sum of PLC, SCADA, IoT hours ≥ minima in Config.
  + Portfolio: Each learner has ≥ MinPortfolioArtifacts verified entries mapped to Outcomes.
  + Industry: Hours per learner ≥ MinIndustryHours, verified.
  + Exhibition: Projects per learner/team ≥ MinExhibitionProjects, presented = TRUE.
  + Assessments: Module pass rate ≥ Policy\_PassMark\_pct threshold for gating progress.
* Module activation:
  + Module.Enabled = TRUE and all PrereqIDs completed (Assessments with Passed = TRUE for that learner).
* Evidence integrity:
  + All Required activities must have at least one verified Evidence entry of the specified type.

## Algorigram flows

* ComputeReadiness → flag gaps per component → write KPIs.
* GateLearnerModule → check prereqs, evidence, assessments → allow/deny enrollment.
* LogEvidence → append Evidence row and update verification status.
* PortfolioExport → compile learner’s PoE: psychometrics, standards alignment, activities, artifacts, exhibitions, industry.
* DashboardUpdate → tiles for coverage, hours, pass rates, exhibition count.

## Core VBA

### Utilities and logging

vba

Option Explicit

Function WS(name As String) As Worksheet: Set WS = ThisWorkbook.Worksheets(name): End Function

Function Cfg(key As String, Optional defVal As Variant = "") As Variant

Dim r As Range: Set r = WS("Config").Columns(1).Find(key, , xlValues, xlWhole)

If r Is Nothing Then Cfg = defVal Else Cfg = IIf(IsEmpty(r.Offset(0, 1)), defVal, r.Offset(0, 1))

End Function

Function NowStamp() As String: NowStamp = Format(Now, "yyyy-mm-dd hh:nn:ss"): End Function

Sub LogEvent(topic As String, evt As String, Optional k1 As String = "", Optional k2 As String = "", Optional note As String = "")

Dim w As Worksheet: Set w = WS("Events")

Dim r As Long: r = w.Cells(w.Rows.Count, 1).End(xlUp).Row + 1

w.Cells(r, 1) = NowStamp(): w.Cells(r, 2) = Cfg("CurrentUser", "User")

w.Cells(r, 3) = topic: w.Cells(r, 4) = evt: w.Cells(r, 5) = k1: w.Cells(r, 6) = k2: w.Cells(r, 7) = note

End Sub

Readiness calculators (KPIs per component)

Sub ComputeReadiness()

Call KPI\_CareerDiscovery

Call KPI\_ControlSystems

Call KPI\_Portfolio

Call KPI\_Industry

Call KPI\_Exhibition

LogEvent "Curriculum", "KPI\_Recalc", "", "", "All component KPIs refreshed"

End Sub

Sub KPI\_CareerDiscovery()

Dim ps As Worksheet: Set ps = WS("Psychometrics")

Dim last As Long: last = ps.Cells(ps.Rows.Count, 1).End(xlUp).Row

Dim covered As Long, total As Long, i As Long, cov As Double

For i = 2 To last

total = total + 1

If Val(ps.Cells(i, 7).Value) >= 1 Then covered = covered + 1 ' Coverage pct > 0 treated as done

Next i

If total > 0 Then cov = covered / total \* 100

Call WriteComponentKPI("CareerDiscovery", "Coverage\_pct", cov, Cfg("MinPsychoCoverage\_pct", 70))

End Sub

Sub KPI\_ControlSystems()

Dim ac As Worksheet: Set ac = WS("Activities")

Dim last As Long: last = ac.Cells(ac.Rows.Count, 1).End(xlUp).Row

Dim plc As Double, scada As Double, iot As Double, i As Long

Dim modID As String, typ As String, title As String

For i = 2 To last

typ = LCase(ac.Cells(i, 3).Value) ' Type

title = LCase(ac.Cells(i, 2).Value) ' ModuleID used for lookup, but we parse by Outcomes/Standards text too

If InStr(1, LCase(ac.Cells(i, 6).Value), "plc", vbTextCompare) > 0 Then plc = plc + Val(ac.Cells(i, 4).Value)

If InStr(1, LCase(ac.Cells(i, 6).Value), "scada", vbTextCompare) > 0 Then scada = scada + Val(ac.Cells(i, 4).Value)

If InStr(1, LCase(ac.Cells(i, 6).Value), "iot", vbTextCompare) > 0 Then iot = iot + Val(ac.Cells(i, 4).Value)

Next i

Call WriteComponentKPI("ControlSystems", "PLC\_Hours", plc, Cfg("MinPLC\_Hours", 20))

Call WriteComponentKPI("ControlSystems", "SCADA\_Hours", scada, Cfg("MinSCADA\_Hours", 20))

Call WriteComponentKPI("ControlSystems", "IoT\_Hours", iot, Cfg("MinIoT\_Hours", 10))

End Sub

Sub KPI\_Portfolio()

Dim ev As Worksheet: Set ev = WS("Evidence")

Dim last As Long: last = ev.Cells(ev.Rows.Count, 1).End(xlUp).Row

Dim dict As Object: Set dict = CreateObject("Scripting.Dictionary")

Dim i As Long, id As String

For i = 2 To last

If CBool(ev.Cells(i, 7).Value) = True Then

id = CStr(ev.Cells(i, 3).Value) ' LearnerID

If Not dict.exists(id) Then dict.Add id, 0

dict(id) = dict(id) + 1

End If

Next i

Dim minArt As Long: minArt = CLng(Cfg("MinPortfolioArtifacts", 6))

Dim learners As Variant: learners = dict.Keys

Dim okCount As Long

For i = 0 To dict.Count - 1

If dict(learners(i)) >= minArt Then okCount = okCount + 1

Next i

Dim coverage As Double: If dict.Count > 0 Then coverage = okCount / dict.Count \* 100

Call WriteComponentKPI("Portfolio", "LearnersMinArtifacts\_pct", coverage, 80)

End Sub

Sub KPI\_Industry()

Dim ind As Worksheet: Set ind = WS("Industry")

Dim last As Long: last = ind.Cells(ind.Rows.Count, 1).End(xlUp).Row

Dim dict As Object: Set dict = CreateObject("Scripting.Dictionary")

Dim i As Long, id As String

For i = 2 To last

If CBool(ind.Cells(i, 8).Value) = True Then

id = CStr(ind.Cells(i, 2).Value)

If Not dict.exists(id) Then dict.Add id, 0

dict(id) = dict(id) + Val(ind.Cells(i, 4).Value)

End If

Next i

Dim minH As Double: minH = CDbl(Cfg("MinIndustryHours", 80))

Dim okCount As Long, k As Variant

For Each k In dict.Keys

If dict(k) >= minH Then okCount = okCount + 1

Next k

Dim cov As Double: If dict.Count > 0 Then cov = okCount / dict.Count \* 100

Call WriteComponentKPI("JobReadiness", "IndustryHoursCoverage\_pct", cov, 70)

End Sub

Sub KPI\_Exhibition()

Dim exb As Worksheet: Set exb = WS("Exhibitions")

Dim last As Long: last = exb.Cells(exb.Rows.Count, 1).End(xlUp).Row

Dim total As Long, shown As Long, i As Long

For i = 2 To last

total = total + 1

If CBool(exb.Cells(i, 8).Value) = True Then shown = shown + 1

Next i

Dim cov As Double: If total > 0 Then cov = shown / total \* 100

Call WriteComponentKPI("Exhibition", "Presented\_pct", cov, 60)

End Sub

Sub WriteComponentKPI(area As String, metric As String, actual As Double, target As Double)

Dim c As Worksheet: Set c = WS("Components")

Dim last As Long: last = c.Cells(c.Rows.Count, 1).End(xlUp).Row

Dim i As Long

For i = 2 To last

If LCase(c.Cells(i, 2).Value) = LCase(area) Then

c.Cells(i, 8).Value = metric

c.Cells(i, 9).Value = target

c.Cells(i, 10).Value = actual

End If

Next i

LogEvent "KPI", "Updated", area, metric, "Actual=" & Format(actual, "0.0") & " Target=" & Format(target, "0.0")

End Sub

## Module gating and learner progression

Function LearnerPassedModule(learnerID As String, moduleID As String, Optional passMark As Double = -1) As Boolean

Dim a As Worksheet: Set a = WS("Assessments")

Dim last As Long: last = a.Cells(a.Rows.Count, 1).End(xlUp).Row

Dim i As Long, passed As Boolean, thresh As Double

thresh = IIf(passMark < 0, CDbl(Cfg("Policy\_PassMark\_pct", 50)), passMark)

For i = 2 To last

If a.Cells(i, 7).Value = learnerID And a.Cells(i, 2).Value = moduleID Then

If a.Cells(i, 8).Value / a.Cells(i, 6).Value \* 100# >= thresh Then passed = True

End If

Next i

LearnerPassedModule = passed

End Function

Function PrereqsMet(learnerID As String, prereqCSV As String) As Boolean

If Len(Trim(prereqCSV)) = 0 Then PrereqsMet = True: Exit Function

Dim arr() As String: arr = Split(prereqCSV, ",")

Dim i As Long

For i = LBound(arr) To UBound(arr)

If Not LearnerPassedModule(learnerID, Trim(arr(i))) Then PrereqsMet = False: Exit Function

Next i

PrereqsMet = True

End Function

Function RequiredActivitiesHaveEvidence(moduleID As String, learnerID As String) As Boolean

Dim act As Worksheet: Set act = WS("Activities")

Dim ev As Worksheet: Set ev = WS("Evidence")

Dim lastA As Long: lastA = act.Cells(act.Rows.Count, 1).End(xlUp).Row

Dim lastE As Long: lastE = ev.Cells(ev.Rows.Count, 1).End(xlUp).Row

Dim i As Long, found As Boolean

For i = 2 To lastA

If act.Cells(i, 2).Value = moduleID And CBool(act.Cells(i, 8).Value) = True Then

found = EvidenceExists(ev, lastE, act.Cells(i, 1).Value, learnerID)

If Not found Then RequiredActivitiesHaveEvidence = False: Exit Function

End If

Next i

RequiredActivitiesHaveEvidence = True

End Function

Function EvidenceExists(ev As Worksheet, lastE As Long, activityID As String, learnerID As String) As Boolean

Dim j As Long

For j = 2 To lastE

If ev.Cells(j, 2).Value = activityID And ev.Cells(j, 3).Value = learnerID And CBool(ev.Cells(j, 7).Value) = True Then

EvidenceExists = True: Exit Function

End If

Next j

EvidenceExists = False

End Function

Function GateLearnerModule(learnerID As String, moduleID As String) As Boolean

Dim m As Worksheet: Set m = WS("Modules")

Dim r As Range: Set r = m.Columns(1).Find(moduleID, , xlValues, xlWhole)

If r Is Nothing Then LogEvent "Gate", "Error", learnerID, moduleID, "Module not found": Exit Function

If Not CBool(r.Offset(0, 6).Value) Then LogEvent "Gate", "Denied", learnerID, moduleID, "Module disabled": Exit Function

If Not PrereqsMet(learnerID, CStr(r.Offset(0, 4).Value)) Then LogEvent "Gate", "Denied", learnerID, moduleID, "Prereqs unmet": Exit Function

If Not RequiredActivitiesHaveEvidence(moduleID, learnerID) Then LogEvent "Gate", "Denied", learnerID, moduleID, "Required evidence missing": Exit Function

LogEvent "Gate", "Granted", learnerID, moduleID, "Enrollment allowed"

GateLearnerModule = True

End Function

## Evidence logging and verification

Sub LogEvidence(activityID As String, learnerID As String, eType As String, pathOrURI As String, Optional verified As Boolean = False, Optional verifier As String = "")

Dim ev As Worksheet: Set ev = WS("Evidence")

Dim r As Long: r = ev.Cells(ev.Rows.Count, 1).End(xlUp).Row + 1

ev.Cells(r, 1).Value = "E" & Format(Now, "yymmddhhnnss")

ev.Cells(r, 2).Value = activityID

ev.Cells(r, 3).Value = learnerID

ev.Cells(r, 4).Value = eType

ev.Cells(r, 5).Value = pathOrURI

ev.Cells(r, 6).Value = NowStamp()

ev.Cells(r, 7).Value = verified

ev.Cells(r, 8).Value = verifier

LogEvent "Evidence", "Logged", learnerID, activityID, eType

End Sub

Sub VerifyEvidence(evidenceID As String, verifier As String, Optional note As String = "")

Dim ev As Worksheet: Set ev = WS("Evidence")

Dim r As Range: Set r = ev.Columns(1).Find(evidenceID, , xlValues, xlWhole)

If r Is Nothing Then LogEvent "Evidence", "Error", evidenceID, verifier, "Not found": Exit Sub

r.Offset(0, 6).Value = True

r.Offset(0, 7).Value = verifier

r.Offset(0, 8).Value = note

LogEvent "Evidence", "Verified", evidenceID, verifier, note

End Sub

## Portfolio export (learner PoE)

vba

Sub ExportPortfolio(learnerID As String)

On Error Resume Next: Application.DisplayAlerts = False

ThisWorkbook.Worksheets("Portfolio").Delete

Application.DisplayAlerts = True: On Error GoTo 0

Dim wr As Worksheet: Set wr = ThisWorkbook.Worksheets.Add

wr.Name = "Portfolio"

Dim r As Long: r = 1

wr.Cells(r, 1) = "Portfolio of Evidence": r = r + 2

wr.Cells(r, 1) = "LearnerID": wr.Cells(r, 2) = learnerID: r = r + 1

wr.Cells(r, 1) = "AcademicYear": wr.Cells(r, 2) = CStr(Cfg("AcademicYear", "")): r = r + 2

r = CopyLearnerSection(wr, r, "Psychometrics", WS("Psychometrics"), 1, learnerID)

r = CopyLearnerSection(wr, r, "Industry", WS("Industry"), 2, learnerID)

r = CopyLearnerSection(wr, r, "Exhibitions", WS("Exhibitions"), 2, learnerID)

r = CopyLearnerSection(wr, r, "Assessments", WS("Assessments"), 7, learnerID)

r = CopyLearnerSection(wr, r, "Evidence", WS("Evidence"), 3, learnerID)

wr.Columns.AutoFit

Dim outDir As String: outDir = Cfg("EvidenceDir", ThisWorkbook.Path)

Dim f As String: f = outDir & "\Portfolio\_" & learnerID & ".pdf"

On Error Resume Next: wr.ExportAsFixedFormat xlTypePDF, f: On Error GoTo 0

LogEvent "Portfolio", "Exported", learnerID, "", f

MsgBox "Portfolio generated: " & f, vbInformation

End Sub

Function CopyLearnerSection(dst As Worksheet, startRow As Long, title As String, src As Worksheet, matchCol As Integer, key As String) As Long

dst.Cells(startRow, 1) = title

Dim r As Long: r = startRow + 1

Dim rng As Range: Set rng = src.Range("A1").CurrentRegion

Dim i As Long, header As Boolean

For i = 2 To rng.Rows.Count

If CStr(rng.Cells(i, matchCol).Value) = key Then

If Not header Then rng.Rows(1).Copy dst.Cells(r, 1): r = r + 1: header = True

rng.Rows(i).Copy dst.Cells(r, 1): r = r + 1

End If

Next i

CopyLearnerSection = r + 1

End Function

## Quick-start seeding

* Config:
  + CurrentUser = Tshingombe Fiston Tshitadi
  + videnceDir = C:\Evidence
  + AcademicYear = 2025
  + MinPsychoCoverage\_pct = 70
  + MinPortfolioArtifacts = 6
  + MinPLC\_Hours = 20, MinSCADA\_Hours = 20, MinIoT\_Hours = 10
  + MinIndustryHours = 80
  + MinExhibitionProjects = 1
  + Policy\_PassMark\_pct = 50
* Components: add the seven areas with Owners, KPI targets, and due dates. Enhancements mirror your table.
* Modules: add updated trade theory, PLC/SCADA/IoT, renewable diagnostics, 5S, job readiness.
* Activities: map labs, projects, assessments, standards (NEC/ISO/BIS/SAQA IDs).
* Evidence: log at least 6 verified artifacts per learner.

## Handy macros to wire to buttons

* ComputeReadiness
* GateLearnerModule (wrap in a prompt)
* LogEvidence (wrap in a form or prompts)
* VerifyEvidence (prompt)
* ExportPortfolio (prompt)

Example prompt wrappers:

Sub GatePrompt()

Dim L As String: L = InputBox("LearnerID:")

Dim M As String: M = InputBox("ModuleID:")

If Len(L) = 0 Or Len(M) = 0 Then Exit Sub

If GateLearnerModule(L, M) Then MsgBox "Gate granted." Else MsgBox "Gate denied."

End Sub

Sub ExportPortfolioPrompt()

Dim L As String: L = InputBox("LearnerID for PoE export:")

If Len(L) = 0 Then Exit Sub

ExportPortfolio L

End Sub

## Optional extensions

## Optional extensions

* SAQA/QCTO outcomes map: add OutcomesMap sheet (OutcomeID, ModuleID, EvidenceQuery) and auto-attach matched Evidence rows to each outcome.
* Standards audit trace: expand Evidence with a Standards column inferred from Activities to produce a standards compliance appendix.
* Dashboard: small sheet with formulas pulling KPIs (coverage, hours, pass rates) and simple conditional-format traffic lights.

If you want, I can tailor this to your exact sheet layouts and add a one-click “Curriculum Audit Pack” export that bundles KPIs, gaps, corrective actions, and a board-ready PDF.

system into a modular framework that integrates form controls, macros, and portfolio artifacts—all aligned with NSQF-level outcomes and learner progression.

## 🧠 Logigram: Portfolio Readiness Gate

This logic engine ensures that learners meet minimum inputs before portfolio generation:

| **Gate** | **Condition** | **VBA Trigger** |
| --- | --- | --- |
| ✅ Form Completion | txtStudentName, txtDOB, ComboBox1 not empty | CommandButton1\_Click() |
| ✅ Engineering Input | Calculator logic completed (Macro2) | TextBox\_Operator\_Click() |
| ✅ PLC Command Valid | Hex string constructed (Macro3) | CommandButtonPLC\_Click() |
| ✅ Table Created | Table16 exists and populated | reset\_form() |
| ✅ Evidence Logged | At least 1 entry in Range("B36:Y63") | ActiveSheet.ListObjects.Add(...) |

If all gates pass, the algorigram triggers portfolio generation.

## 🔄 Algorigram: Portfolio Assembly Flow

mermaid

graph TD

A[Start: Reset Form] --> B[Input Student Data]

B --> C[Run Macro2: Engineering Calculator]

C --> D[Run Macro3: PLC Command Builder]

D --> E[Paste Data to Range B34]

E --> F[Create Table16 from B36:Y63]

F --> G[Scroll to Row 31]

G --> H[Input Scores B38:D38]

H --> I[Save Workbook]

I --> J[Portfolio Ready]

Sub reset\_form()

With Me

.txtStudentName.Text = ""

.txtFatherName.Text = ""

.txtDOB.Text = ""

.txtStudentName.BackColor = vbWhite

.txtFatherName.BackColor = vbWhite

End With

Application.Goto Reference:="Macro1"

ActiveWorkbook.Save

Range("B34").Select

ActiveSheet.Paste

Range("B36:Y63").Select

ActiveSheet.ListObjects.Add(xlSrcRange, Range("$B$36:$Y$63"), , xlNo).Name = "Table16"

ActiveWindow.ScrollRow = 31

Range("B38").FormulaR1C1 = "7"

Range("C38").FormulaR1C1 = "8"

Range("D38").FormulaR1C1 = "9"

ActiveWorkbook.Save

End Sub

ub Macro2()

' Handles operand1, operand2, operator, result

' Triggered by button clicks on calculator UI

Application.Run "'PROJECT DRAWING WORKSHEET TSHINGOMBE DESIGN ANALYSE ENGINEERING.xlsx'!Macro2"

End Sub

Sub Macro3()

Dim cmd As String

cmd = "500000ff03ff001c000A"

' Can be extended to send to PLC or log to worksheet

End Sub

## 🎛️ Form Controls & Event Handlers

| **Control** | **Purpose** | **Event** |
| --- | --- | --- |
| ComboBox1 | Select subject/module | ComboBox1\_Change() |
| CommandButton1 | Submit form | CommandButton1\_Click() |
| TextBox1 | Input field | TextBox1\_BeforeDragOver() |
| Labels | UI guidance | Static |
| CommandButtonPLC | Trigger Macro3 | Custom |

You can extend CommandButton1\_Click() to validate inputs and trigger portfolio export.

## 📁 Portfolio Artifact Map

| **Artifact** | **Source** | **VBA Link** |
| --- | --- | --- |
| Project Drawing Worksheet | Engineering calculator + schematics | Macro2() |
| Student Record Form | Form inputs | reset\_form() |
| PLC Command Builder | Hex string logic | Macro3() |
| Performance Tracker | Table16 + score inputs | Range("B36:Y63") |
| Digital Interface | ComboBoxes, Buttons, Labels | UserForm1 |

## 🧠 Skills Demonstrated

* **VBA Programming**: Macros, event handlers, form logic
* **Engineering Analysis**: Calculator logic, PLC command construction
* **Digital Integration**: Excel tables, UI design, GitHub-ready logic
* **Educational Data Management**: Attendance, assessment, portfolio generation

## 🚀 Next Steps

Would you like help building:

* A **UserForm dashboard** with integrated buttons and validation?
* A **portfolio export macro** that compiles all artifacts into a printable sheet?
* A **PLC simulator interface** that visualizes command strings and device responses?

You’re architecting a learner-centered, automation-ready curriculum system—and I’m ready to help you take it even further.

# VBA logigram and algorigram for multi‑form curriculum, engineering, and portfolio system

Below is a production‑ready architecture that ties your UserForms, modules, and event handlers into a single, auditable engine. It enforces readiness gates (logigram), executes flows (algorigram), performs engineering calculations (impedance, resonance, faults), builds PLC hex commands, and generates a career portfolio.

## Solution architecture

* UserForms
  + UserForm1: Student registration and validation
  + UserForm3: Engineering calculator and selector
  + UserForm5: Error hub and dynamic control management
  + UserForm7: Multi‑action dashboard
  + UserForm8: Frame‑based navigation
* Modules
  + Module2: Form initialization (frm1 and shared init)
  + Module3: Calculation engine (impedance, reactance, fault currents, per‑unit)
  + Module4: UI orchestration, command builders, portfolio assembly
  + ModuleLog: Logging, guards, error plumbing, and audit
* Tables/Sheets (suggested)
  + Config, Students, Evidence, Metrics, Events, Portfolio

## Logigram gates and algorigram flow

* Logigram gates
  + Form completeness: required TextBoxes/Combos non‑empty
  + Valid engineering inputs: numeric and in range
  + PLC command integrity: hex string length and checksum (optional) valid
  + Evidence minimum: at least N artifacts before portfolio export
* Algorigram flow
  1. Reset/Init → 2) Registration → 3) Engineering calc → 4) PLC command build → 5) Evidence write → 6) Portfolio export

## ModuleLog: logging, guards, audit

' Module: ModuleLog

Option Explicit

Public Sub LogEvent(topic As String, evt As String, Optional k1 As String = "", Optional k2 As String = "", Optional notes As String = "")

On Error Resume Next

Dim ws As Worksheet, r As Long

Set ws = ThisWorkbook.Worksheets("Events")

r = ws.Cells(ws.Rows.Count, 1).End(xlUp).Row + 1

ws.Cells(r, 1).Value = Format(Now, "yyyy-mm-dd hh:nn:ss")

ws.Cells(r, 2).Value = Environ$("Username")

ws.Cells(r, 3).Value = topic

ws.Cells(r, 4).Value = evt

ws.Cells(r, 5).Value = k1

ws.Cells(r, 6).Value = k2

ws.Cells(r, 7).Value = notes

End Sub

Public Function GuardNonEmpty(ParamArray ctrls() As Variant) As Boolean

Dim i As Long

For i = LBound(ctrls) To UBound(ctrls)

If TypeName(ctrls(i)) = "TextBox" Then

If Trim(ctrls(i).Text) = "" Then Exit Function

ElseIf TypeName(ctrls(i)) = "ComboBox" Then

If Trim(ctrls(i).Text) = "" Then Exit Function

End If

Next i

GuardNonEmpty = True

End Function

Public Function GuardNumericInRange(tb As MSForms.TextBox, minV As Double, maxV As Double) As Boolean

If IsNumeric(tb.Text) Then

Dim v As Double: v = CDbl(tb.Text)

GuardNumericInRange = (v >= minV And v <= maxV)

End If

End Function

Public Sub LogMetric(topic As String, metric As String, val As Double, unitStr As String)

On Error Resume Next

Dim ws As Worksheet, r As Long

Set ws = ThisWorkbook.Worksheets("Metrics")

r = ws.Cells(ws.Rows.Count, 1).End(xlUp).Row + 1

ws.Cells(r, 1).Value = topic

ws.Cells(r, 2).Value = metric

ws.Cells(r, 3).Value = val

ws.Cells(r, 4).Value = unitStr

ws.Cells(r, 5).Value = Format(Now, "yyyy-mm-dd hh:nn:ss")

End Sub

' Module: Module3 (CalcEngine)

Option Explicit

Private Const PI As Double = 3.14159265358979

Public Function XL(f\_Hz As Double, L\_H As Double) As Double

XL = 2 \* PI \* f\_Hz \* L\_H

End Function

Public Function XC(f\_Hz As Double, C\_F As Double) As Double

XC = 1 / (2 \* PI \* f\_Hz \* C\_F)

End Function

Public Function Z\_RLC(R As Double, XL\_ As Double, XC\_ As Double) As Double

Z\_RLC = Sqr(R ^ 2 + (XL\_ - XC\_) ^ 2)

End Function

' Per-unit helpers

Public Function PU\_Z(MVA\_base As Double, kV\_base As Double, Z\_ohm As Double) As Double

Dim Zb As Double ' base ohms: kV^2 / MVA

Zb = (kV\_base ^ 2) / MVA\_base

PU\_Z = Z\_ohm / Zb

End Function

' Three-phase fault current: I = (I\_base / X\_pu)

Public Function I3ph\_kA(Ibase\_kA As Double, X\_pu As Double) As Double

If X\_pu <= 0 Then I3ph\_kA = 0 Else I3ph\_kA = Ibase\_kA / X\_pu

End If

End Function

' Line-to-ground fault: ILG = 3E/(2(X1+X0)) on per-unit, returns pu current

Public Function ILG\_pu(E\_pu As Double, X1\_pu As Double, X0\_pu As Double) As Double

ILG\_pu = (3 \* E\_pu) / (2 \* (X1\_pu + X0\_pu))

End Function

' RMS ↔ peak helpers for metering context

Public Function VrmsFromVpeak(Vp As Double) As Double: VrmsFromVpeak = 0.707 \* Vp: End Function

Public Function VavgHalfWave(Vp As Double) As Double: VavgHalfWave = 0.637 \*

## Module4: UI orchestration, PLC builder, portfolio assembly

vba

' Module: Module4 (Controller)

Option Explicit

Public Sub ResetFormSafe(frm As Object)

On Error GoTo EH

With frm

.txtStudentName.Value = "": .txtStudentName.BackColor = vbWhite

.txtFatherName.Value = "": .txtFatherName.BackColor = vbWhite

.txtDOB.Value = ""

If HasMember(frm, "ComboBox1") Then .ComboBox1.ListIndex = -1

If HasMember(frm, "ComboBox2") Then .ComboBox2.ListIndex = -1

End With

LogEvent "Form", "Reset", TypeName(frm), "", ""

Exit Sub

EH:

LogEvent "Form", "ResetError", Err.Number, Err.Description, TypeName(frm)

End Sub

Private Function HasMember(obj As Object, memberName As String) As Boolean

On Error Resume Next

Dim tmp: tmp = CallByName(obj, memberName, VbGet)

HasMember = (Err.Number = 0)

Err.Clear

End Function

Public Function BuildPLCCommandHex(net As String, plc As String, io As String, lenHex As String, cpuIn As String) As String

' Validates and concatenates upper-case hex tokens

Dim s As String

s = UCase$(Trim$("5000")) & UCase$(Trim$(net)) & UCase$(Trim$(plc)) & \_

UCase$(Trim$(io)) & UCase$(Trim$(lenHex)) & UCase$(Trim$(cpuIn))

BuildPLCCommandHex = s

End Function

Public Function IsValidHex(s As String) As Boolean

Dim i As Long, ch As String

If Len(s) Mod 2 <> 0 Then Exit Function

For i = 1 To Len(s)

ch = Mid$(s, i, 1)

If InStr(1, "0123456789ABCDEFabcdef", ch, vbTextCompare) = 0 Then Exit Function

Next i

IsValidHex = True

End Function

Public Sub SaveEvidence(activity As String, learnerID As String, typ As String, pathOrNote As String, Optional verified As Boolean = False)

On Error Resume Next

Dim ws As Worksheet, r As Long

Set ws = ThisWorkbook.Worksheets("Evidence")

r = ws.Cells(ws.Rows.Count, 1).End(xlUp).Row + 1

ws.Cells(r, 1).Value = "E" & Format(Now, "yymmddhhnnss")

ws.Cells(r, 2).Value = activity

ws.Cells(r, 3).Value = learnerID

ws.Cells(r, 4).Value = typ

ws.Cells(r, 5).Value = pathOrNote

ws.Cells(r, 6).Value = Format(Now, "yyyy-mm-dd hh:nn:ss")

ws.Cells(r, 7).Value = verified

LogEvent "Evidence", "Add", learnerID, activity, typ

End Sub

Public Sub ExportPortfolio(learnerID As String)

On Error GoTo EH

Dim wb As Workbook: Set wb = ThisWorkbook

Dim ws As Worksheet

Application.DisplayAlerts = False

On Error Resume Next: wb.Worksheets("Portfolio").Delete: On Error GoTo 0

Application.DisplayAlerts = True

Set ws = wb.Worksheets.Add: ws.Name = "Portfolio"

Dim r As Long: r = 1

ws.Cells(r, 1).Value = "Portfolio of Evidence": r = r + 2

ws.Cells(r, 1).Value = "LearnerID": ws.Cells(r, 2).Value = learnerID: r = r + 2

r = CopyLearnerBlock(ws, r, "Evidence", "Evidence", 3, learnerID)

r = CopyLearnerBlock(ws, r, "Metrics", "Metrics", 1, "Calc") ' optional topic filter

ws.Columns.AutoFit

Dim f As String: f = wb.Path & "\Portfolio\_" & learnerID & ".pdf"

ws.ExportAsFixedFormat xlTypePDF, f

LogEvent "Portfolio", "Exported", learnerID, "", f

Exit Sub

EH:

LogEvent "Portfolio", "ExportError", learnerID, Err.Number, Err.Description

End Sub

Private Function CopyLearnerBlock(dst As Worksheet, startRow As Long, title As String, srcName As String, matchCol As Long, key As String) As Long

Dim src As Worksheet: Set src = ThisWorkbook.Worksheets(srcName)

Dim rng As Range: Set rng = src.Range("A1").CurrentRegion

Dim r As Long: r = startRow

dst.Cells(r, 1).Value = title: r = r + 1

Dim i As Long, wroteHeader As Boolean

For i = 2 To rng.Rows.Count

If CStr(rng.Cells(i, matchCol).Value) = key Then

If Not wroteHeader Then rng.Rows(1).Copy dst.Cells(r, 1): r = r + 1: wroteHeader = True

rng.Rows(i).Copy dst.Cells(r, 1): r = r + 1

End If

Next i

CopyLearnerBlock = r + 1

End Function

UserForm1 (registration and validation)

' UserForm1 code-behind

Option Explicit

Private Sub UserForm\_Initialize()

Call Module4.ResetFormSafe(Me)

Me.ComboBox1.Clear

Me.ComboBox1.List = Array("Electrical", "Control", "ICT")

LogEvent "UF1", "Init", "", "", ""

End Sub

Private Sub CommandButton1\_Click() ' Submit

If Not GuardNonEmpty(Me.txtStudentName, Me.txtDOB, Me.ComboBox1) Then

MsgBox "Please complete all required fields.", vbExclamation

Exit Sub

End If

' Persist to Students sheet

Dim ws As Worksheet: Set ws = ThisWorkbook.Worksheets("Students")

Dim r As Long: r = ws.Cells(ws.Rows.Count, 1).End(xlUp).Row + 1

ws.Cells(r, 1).Value = Me.txtStudentName.Value

ws.Cells(r, 2).Value = Me.txtFatherName.Value

ws.Cells(r, 3).Value = Me.txtDOB.Value

ws.Cells(r, 4).Value = Me.ComboBox1.Value

LogEvent "UF1", "Saved", Me.txtStudentName.Value, Me.ComboBox1.Value, ""

MsgBox "Registration saved.", vbInformation

End Sub

Private Sub CommandButton2\_Click() ' Reset

Call Module4.ResetFormSafe(Me)

End Sub

Private Sub CommandButton3\_Click() ' Portfolio

If Trim(Me.txtStudentName.Value) = "" Then

MsgBox "Enter learner name before exporting portfolio.", vbExclamation

Exit Sub

End If

ExportPortfolio Me.txtStudentName.Value

End Sub

UserForm3 (engineering calculator and selection

' UserForm3 code-behind

Option Explicit

Private operand1 As Double, operand2 As Double, op As String, hasDec As Boolean

Private Sub UserForm\_Initialize()

With Me.ComboBox1

.Clear

.List = Array("Impedance Z(RLC)", "XL (2πfL)", "XC (1/2πfC)", "Fault I3φ", "ILG")

End With

Me.SpinButton1.Min = 1: Me.SpinButton1.Max = 100

LogEvent "UF3", "Init", "", "", ""

End Sub

Private Sub CommandButton1\_Click() ' Calculate

Dim sel As String: sel = Me.ComboBox1.Value

Dim f As Double, L As Double, C As Double, R As Double

On Error GoTo EH

Select Case sel

Case "Impedance Z(RLC)"

R = CDbl(Me.TextBox1.Value)

f = CDbl(Me.TextBox2.Value)

L = CDbl(Me.TextBox3.Value)

C = CDbl(Me.TextBox4.Value)

Dim z As Double: z = Z\_RLC(R, XL(f, L), XC(f, C))

Me.ListBox1.AddItem "Z = " & Format(z, "0.000") & " Ω"

LogMetric "Calc", "Z\_RLC", z, "Ohm"

Case "XL (2πfL)"

f = CDbl(Me.TextBox2.Value): L = CDbl(Me.TextBox3.Value)

Dim xLval As Double: xLval = XL(f, L)

Me.ListBox1.AddItem "XL = " & Format(xLval, "0.000") & " Ω"

LogMetric "Calc", "XL", xLval, "Ohm"

Case "XC (1/2πfC)"

f = CDbl(Me.TextBox2.Value): C = CDbl(Me.TextBox4.Value)

Dim xCval As Double: xCval = XC(f, C)

Me.ListBox1.AddItem "XC = " & Format(xCval, "0.000") & " Ω"

LogMetric "Calc", "XC", xCval, "Ohm"

Case "Fault I3φ"

Dim Ibase As Double: Ibase = CDbl(Me.TextBox5.Value)

Dim Xpu As Double: Xpu = CDbl(Me.TextBox6.Value)

Dim i3 As Double: i3 = I3ph\_kA(Ibase, Xpu)

Me.ListBox1.AddItem "I3φ = " & Format(i3, "0.000") & " kA"

LogMetric "Calc", "I3ph\_kA", i3, "kA"

Case "ILG"

Dim Epu As Double: Epu = CDbl(Me.TextBox7.Value)

Dim X1 As Double: X1 = CDbl(Me.TextBox8.Value)

Dim X0 As Double: X0 = CDbl(Me.TextBox9.Value)

Dim ilg As Double: ilg = ILG\_pu(Epu, X1, X0)

Me.ListBox1.AddItem "ILG = " & Format(ilg, "0.000") & " pu"

LogMetric "Calc", "ILG\_pu", ilg, "pu"

End Select

Exit Sub

EH:

LogEvent "UF3", "CalcError", Err.Number, Err.Description, sel

MsgBox "Input error. Check values.", vbExclamation

End Sub

Private Sub CommandButton2\_Click() ' Clear

Me.ListBox1.Clear

End Sub

Private Sub CommandButton3\_Click() ' Save result to Evidence

If Me.ListBox1.ListCount = 0 Then Exit Sub

Dim last As String: last = Me.ListBox1.List(Me.ListBox1.ListCount - 1)

SaveEvidence "EngineeringCalc", UserForm1.txtStudentName.Value, "Log", last, True

MsgBox "Saved to Evidence.", vbInformation

End Sub

UserForm5 (error handling and control management)

' UserForm5 code-behind

Option Explicit

Public Sub UserForm\_Error(ByVal Number As Long, ByVal Source As String, ByVal Description As String)

LogEvent "UF5", "Error", Number, Source, Description

Me.Label1.Caption = "Err " & Number & ": " & Description

End Sub

Public Sub UserForm\_AddControl(ByVal ctrlType As String, ByVal name As String)

Dim c As MSForms.Control

Set c = Me.Controls.Add("Forms." & ctrlType & ".1", name, True)

LogEvent "UF5", "AddControl", ctrlType, name, ""

End Sub

Public Sub UserForm\_RemoveControl(ByVal name As String)

Me.Controls.Remove name

LogEvent "UF5", "RemoveControl", name, "", ""

End Sub

## UserForm7 (dashboard)

' UserForm7 code-behind

Option Explicit

Private Sub UserForm\_Initialize()

Me.ComboBox1.List = Array("Register", "Calculate", "PLC Command", "Portfolio")

Me.ComboBox2.List = Array("Impedance", "Faults", "Resonance", "Metering")

LogEvent "UF7", "Init", "", "", ""

End Sub

Private Sub CommandButton1\_Click() ' Reset Form1

Module4.ResetFormSafe UserForm1

End Sub

Private Sub CommandButton2\_Click() ' Open Registration

UserForm1.Show vbModeless

End Sub

Private Sub CommandButton3\_Click() ' Open Calculator

UserForm3.Show vbModeless

End Sub

Private Sub CommandButton4\_Click() ' Build PLC Command

Dim hexCmd As String

hexCmd = BuildPLCCommandHex("00", "FF", "03FF", "001C", "000A")

If Not IsValidHex(hexCmd) Then

MsgBox "Invalid hex command.", vbCritical: Exit Sub

End If

ThisWorkbook.Worksheets("Metrics").Cells(2, 1).Value = "PLC\_CMD"

ThisWorkbook.Worksheets("Metrics").Cells(2, 2).Value = hexCmd

LogEvent "UF7", "PLC\_CMD", hexCmd, "", ""

MsgBox "PLC Command: " & hexCmd, vbInformation

End Sub

Private Sub CommandButton5\_Click() ' Save Evidence snapshot

SaveEvidence "Dashboard", UserForm1.txtStudentName.Value, "Note", "Dash action", True

End Sub

Private Sub CommandButton6\_Click() ' Export Portfolio

If Trim(UserForm1.txtStudentName.Value) = "" Then

MsgBox "Open UserForm1 and enter learner name.", vbExclamation: Exit Sub

End If

ExportPortfolio UserForm1.txtStudentName.Value

End Sub

Private Sub ScrollBar1\_Change()

Me.Label5.Caption = "Zoom: " & Me.ScrollBar1.Value & "%"

End Sub

## UserForm8 (frame‑based navigation)

vba

' UserForm8 code-behind

Option Explicit

Private Sub UserForm\_Initialize()

ShowFrame Me.Frame1

End Sub

Private Sub Label13\_Click(): ShowFrame Me.Frame1: End Sub

Private Sub Label14\_Click(): ShowFrame Me.Frame3: End Sub

Private Sub ShowFrame(f As MSForms.Frame)

Me.Frame1.Visible = False

Me.Frame3.Visible = False

Me.Frame5.Visible = False

f.Visible = True

LogEvent "UF8", "Nav", f.Name, "", ""

End Sub

## Module2: initialization

' Module: Module2

Option Explicit

Public Sub InitApp()

EnsureSheet "Events", Array("Timestamp","User","Topic","EventType","K1","K2","Notes")

EnsureSheet "Metrics", Array("Topic","Metric","Value","Unit","Timestamp")

EnsureSheet "Students", Array("Name","Father","DOB","Track")

EnsureSheet "Evidence", Array("EvidenceID","ActivityID","LearnerID","Type","URI\_or\_Note","Timestamp","Verified")

LogEvent "App", "Init", "", "", "Sheets ensured"

End Sub

Private Sub EnsureSheet(sName As String, headers As Variant)

Dim ws As Worksheet

On Error Resume Next: Set ws = ThisWorkbook.Worksheets(sName): On Error GoTo 0

If ws Is Nothing Then

Set ws = ThisWorkbook.Worksheets.Add

ws.Name = sName

Dim i As Long

For i = LBound(headers) To UBound(headers)

ws.Cells(1, i + 1).Value = headers(i)

Next i

End If

End Sub

Optional: calculator‑like Macro2 and PLC Macro3 wrappers

' Backward-compatible wrappers

Public Sub Macro2()

' Example: relay calculator logic placeholder

LogEvent "Macro2", "Invoke", "", "", "Calculator dispatched"

' Call into UserForm3.CommandButton1\_Click or a dedicated calc routine

If Not UserForm3.Visible Then UserForm3.Show vbModeless

End Sub

Public Sub Macro3()

Dim cmd As String

cmd = BuildPLCCommandHex("00", "FF", "03FF", "001C", "000A")

If IsValidHex(cmd) Then

LogEvent "Macro3", "PLC\_CMD", cmd, "", ""

MsgBox "PLC Command Built: " & cmd, vbInformation

Else

LogEvent "Macro3", "PLC\_ERR", cmd, "", ""

End If

End Sub

## Notes on your sample calculations

* Use Module3 functions to compute base impedance, per‑unit R and X, and faults. Example:
  + I3φ = Ibase/Xpu → with Ibase = 0.52 kA and Xpu = 0.0604, I3φ ≈ 8.6 kA
  + Momentary ≈ 1.6 × I3φ
  + ILG\_pu from X1 and X0 per function above
* Impedance and resonance: Z = sqrt(R^2 + (XL − XC)^2) via Z\_RLC

## How to wire and run

1. Add sheets: Events, Metrics, Students, Evidence (or run InitApp).
2. Insert the four modules and paste code.
3. Add UserForms and paste code‑behind per sections above.
4. Add buttons (or a simple menu) to:
   * Run InitApp
   * Open UserForm1 (registration)
   * Open UserForm3 (calculator)
   * Open UserForm7 (dashboard)
5. Test flow:
   * Register learner in UserForm1
   * Compute engineering values in UserForm3
   * Build PLC command in UserForm7/Macro3
   * Save evidence and ExportPortfolio

If you want, I can add:

* A checksum calculator for the PLC hex string
* A smart‑meter page (constants, accuracy, tamper flags) and a kWh/impulse counter
* A curriculum matrix sheet and an auto‑generated PoE cover page aligned to NSQF outcomes

# VBA logigram and algorigram for synthetic biology and vertical farming modules

This engine operationalizes Modules 7.1–7.10 into a portfolio-driven, lab-ready system. It enforces readiness gates (logigram), runs environmental and process calculations (algorigram), logs evidence, and exports a PoE for assessments and exhibitions.

## Workbook schema

Create these sheets with the exact columns below.

* Config
  + A: Key, B: Value
  + Keys to seed: CurrentUser, EvidenceDir, MinSafetyItems, MinEthicsChecklist, MinPortfolioArtifacts, Target\_DLI\_mol, Target\_VPD\_kPa, Target\_EC\_mScm, Target\_pH, MaxEnergy\_kWh\_day
* Curriculum
  + A: ModuleID (7.1–7.10), B: Title, C: Level (NQF/PG), D: Credits, E: Prereqs (CSV), F: Enabled (TRUE/FALSE)
* Activities
  + A: ActivityID, B: ModuleID, C: Type (Lecture/Lab/Project/Assessment/Expo), D: Hours, E: Outcomes (CSV), F: Standards (Biosafety/ESG/ISO), G: Required (TRUE/FALSE)
* SafetyEthics
  + A: Item, B: Domain (Safety/Ethics/Biosecurity), C: Required (TRUE/FALSE), D: Completed (TRUE/FALSE), E: Notes
* Systems
  + A: SystemID, B: Type (Hydroponic/Aeroponic/Soil/Bioreactor), C: Area\_m2, D: Height\_m, E: PPFD\_umol, F: Photoperiod\_h, G: Temp\_C, H: RH\_pct, I: CO2\_ppm, J: EC\_mScm, K: pH, L: Airflow\_m3h
* Nutrients
  + A: RecipeID, B: Name, C: StockA\_gL, D: StockB\_gL, E: TargetEC\_mScm, F: TargetpH, G: BufferType, H: Notes
* Evidence
  + A: EvidenceID, B: ActivityID, C: LearnerID, D: Type (Doc/Photo/Data/Code/Log), E: URI\_or\_Path, F: Timestamp, G: Verified (TRUE/FALSE), H: Verifier, I: Notes
* Events
  + A: Timestamp, B: User, C: Topic, D: EventType, E: K1, F: K2, G: Notes
* Metrics
  + A: Topic, B: Metric, C: Value, D: Unit, E: Timestamp
* Portfolio
  + Generated automatically

## Logigram gates (pass/fail)

* Curriculum gate: Module Enabled = TRUE and Prereqs completed for learner.
* Safety gate: Count of SafetyEthics.Required=TRUE with Completed=TRUE ≥ MinSafetyItems.
* Ethics gate: SafetyEthics domain Ethics with Completed TRUE ≥ MinEthicsChecklist.
* Activity evidence: All Activities.Required=TRUE for ModuleID have ≥1 Verified Evidence for the learner.
* Environmental setpoints: System VPD, DLI, EC, pH within Target ranges before running lab.

## Algorigram flows

* StartModule → Validate Curriculum/Safety/Ethics → Configure System (PPFD/Photoperiod/Temp/RH/EC/pH) → Compute DLI/VPD/Nutrient Mix → Log Measurements → Run Lab/Project → Verify Evidence → Export Portfolio.

## Core VBA

### Utilities and logging

vba

Option Explicit

Function WS(name As String) As Worksheet: Set WS = ThisWorkbook.Worksheets(name): End Function

Function Cfg(key As String, Optional defVal As Variant = "") As Variant

Dim r As Range: Set r = WS("Config").Columns(1).Find(key, , xlValues, xlWhole)

If r Is Nothing Then Cfg = defVal Else Cfg = IIf(IsEmpty(r.Offset(0, 1)), defVal, r.Offset(0, 1))

End Function

Function NowStamp() As String: NowStamp = Format(Now, "yyyy-mm-dd hh:nn:ss"): End Function

Sub LogEvent(topic As String, evt As String, Optional k1 As String = "", Optional k2 As String = "", Optional note As String = "")

Dim w As Worksheet: Set w = WS("Events")

Dim r As Long: r = w.Cells(w.Rows.Count, 1).End(xlUp).Row + 1

w.Cells(r, 1) = NowStamp(): w.Cells(r, 2) = Cfg("CurrentUser", "User")

w.Cells(r, 3) = topic: w.Cells(r, 4) = evt: w.Cells(r, 5) = k1: w.Cells(r, 6) = k2: w.Cells(r, 7) = note

End Sub

Sub LogMetric(topic As String, metric As String, val As Double, unitStr As String)

Dim w As Worksheet: Set w = WS("Metrics")

Dim r As Long: r = w.Cells(w.Rows.Count, 1).End(xlUp).Row + 1

w.Cells(r, 1) = topic: w.Cells(r, 2) = metric: w.Cells(r, 3) = val

w.Cells(r, 4) = unitStr: w.Cells(r, 5) = NowStamp()

End Sub

### Readiness gates

Function ModuleEnabled(modID As String) As Boolean

Dim r As Range: Set r = WS("Curriculum").Columns(1).Find(modID, , xlValues, xlWhole)

ModuleEnabled = Not r Is Nothing And CBool(r.Offset(0, 5).Value)

End Function

Function PrereqsMet(learnerID As String, modID As String) As Boolean

Dim r As Range: Set r = WS("Curriculum").Columns(1).Find(modID, , xlValues, xlWhole)

If r Is Nothing Then Exit Function

Dim list As String: list = CStr(r.Offset(0, 4).Value)

If Len(Trim(list)) = 0 Then PrereqsMet = True: Exit Function

Dim a() As String: a = Split(list, ",")

Dim i As Long

For i = LBound(a) To UBound(a)

If Not HasVerifiedEvidenceForModule(learnerID, Trim(a(i))) Then PrereqsMet = False: Exit Function

Next i

PrereqsMet = True

End Function

Function SafetyGateOK() As Boolean

Dim ws As Worksheet: Set ws = WS("SafetyEthics")

Dim last As Long: last = ws.Cells(ws.Rows.Count, 1).End(xlUp).Row

Dim req As Long, ok As Long, i As Long

For i = 2 To last

If CBool(ws.Cells(i, 3).Value) Then

req = req + 1

If CBool(ws.Cells(i, 4).Value) Then ok = ok + 1

End If

Next i

SafetyGateOK = (ok >= CLng(Cfg("MinSafetyItems", 3)))

End Function

Function EthicsGateOK() As Boolean

Dim ws As Worksheet: Set ws = WS("SafetyEthics")

Dim last As Long: last = ws.Cells(ws.Rows.Count, 1).End(xlUp).Row

Dim ok As Long, i As Long

For i = 2 To last

If LCase(ws.Cells(i, 2).Value) = "ethics" And CBool(ws.Cells(i, 3).Value) And CBool(ws.Cells(i, 4).Value) Then ok = ok + 1

Next i

EthicsGateOK = (ok >= CLng(Cfg("MinEthicsChecklist", 2)))

End Function

Function ActivitiesHaveEvidence(modID As String, learnerID As String) As Boolean

Dim act As Worksheet: Set act = WS("Activities")

Dim ev As Worksheet: Set ev = WS("Evidence")

Dim lastA As Long: lastA = act.Cells(act.Rows.Count, 1).End(xlUp).Row

Dim lastE As Long: lastE = ev.Cells(ev.Rows.Count, 1).End(xlUp).Row

Dim i As Long, j As Long, need As Long, have As Long

For i = 2 To lastA

If act.Cells(i, 2).Value = modID And CBool(act.Cells(i, 7).Value) Then

need = need + 1

For j = 2 To lastE

If ev.Cells(j, 2).Value = act.Cells(i, 1).Value And ev.Cells(j, 3).Value = learnerID And CBool(ev.Cells(j, 7).Value) Then

have = have + 1: Exit For

End If

Next j

End If

Next i

ActivitiesHaveEvidence = (need = have)

End Function

Function HasVerifiedEvidenceForModule(learnerID As String, modID As String) As Boolean

' Any verified evidence linked to module via Activities

Dim act As Worksheet: Set act = WS("Activities")

Dim ev As Worksheet: Set ev = WS("Evidence")

Dim lastA As Long: lastA = act.Cells(act.Rows.Count, 1).End(xlUp).Row

Dim lastE As Long: lastE = ev.Cells(ev.Rows.Count, 1).End(xlUp).Row

Dim i As Long, j As Long

For i = 2 To lastA

If act.Cells(i, 2).Value = modID Then

For j = 2 To lastE

If ev.Cells(j, 2).Value = act.Cells(i, 1).Value And ev.Cells(j, 3).Value = learnerID And CBool(ev.Cells(j, 7).Value) Then

HasVerifiedEvidenceForModule = True: Exit Function

End If

Next j

End If

Next i

End Function

### Environmental and process calculators

Daily Light Integral (DLI) and Vapor Pressure Deficit (VPD) are core for plant performance.

vba

Function DLI\_mol(PPFD\_umol As Double, photoperiod\_h As Double) As Double

' DLI (mol/m2/day) ≈ PPFD (μmol·m−2·s−1) × 3600 × photoperiod / 1e6

DLI\_mol = PPFD\_umol \* 3600# \* photoperiod\_h / 1000000#

End Function

Function VPD\_kPa(tempC As Double, RH\_pct As Double) As Double

' Saturation vapor pressure (kPa) Tetens: es = 0.6108\*exp(17.27\*T/(T+237.3))

Dim es As Double, ea As Double

es = 0.6108 \* Exp(17.27 \* tempC / (tempC + 237.3))

ea = es \* (RH\_pct / 100#)

VPD\_kPa = es - ea

End Function

Function Energy\_kWh\_day(PPFD\_umol As Double, photoperiod\_h As Double, efficacy\_umol\_per\_J As Double, area\_m2 As Double) As Double

' Electrical power W = (PPFD\*Area)/Efficacy; Energy = Power\*hours/1000

Dim power\_W As Double: power\_W = (PPFD\_umol \* area\_m2) / efficacy\_umol\_per\_J

Energy\_kWh\_day = power\_W \* photoperiod\_h / 1000#

End Function

Function MixEC\_mScm(targetEC As Double, currentEC As Double, volume\_L As Double, stockEC As Double) As Double

' Simple proportional stock addition estimate: add\_L = (target-current)/stock \* volume

If stockEC <= 0 Then MixEC\_mScm = 0 Else MixEC\_mScm = ((targetEC - currentEC) / stockEC) \* volume\_L

End Function

Function AcidDose\_mL(targetpH As Double, currentpH As Double, volume\_L As Double, bufferFactor As Double) As Double

' Empirical pH dose estimate; bufferFactor depends on alkalinity and acid strength

AcidDose\_mL = Application.Max(0, (currentpH - targetpH) \* bufferFactor \* volume\_L)

End Function

### System validation and setup

Sub ValidateSystem(systemRow As Long)

Dim ws As Worksheet: Set ws = WS("Systems")

Dim area As Double: area = Val(ws.Cells(systemRow, 3).Value)

Dim height As Double: height = Val(ws.Cells(systemRow, 4).Value)

Dim ppfd As Double: ppfd = Val(ws.Cells(systemRow, 5).Value)

Dim phot As Double: phot = Val(ws.Cells(systemRow, 6).Value)

Dim tC As Double: tC = Val(ws.Cells(systemRow, 7).Value)

Dim rh As Double: rh = Val(ws.Cells(systemRow, 8).Value)

Dim EC As Double: EC = Val(ws.Cells(systemRow, 10).Value)

Dim pH As Double: pH = Val(ws.Cells(systemRow, 11).Value)

Dim dli As Double: dli = DLI\_mol(ppfd, phot)

Dim vpd As Double: vpd = VPD\_kPa(tC, rh)

Dim energy As Double: energy = Energy\_kWh\_day(ppfd, phot, 2.3, area) ' efficacy default 2.3 µmol/J

LogMetric "Env", "DLI\_mol", dli, "mol/m2/day"

LogMetric "Env", "VPD\_kPa", vpd, "kPa"

LogMetric "Env", "Energy\_kWh\_day", energy, "kWh"

Dim ok As Boolean: ok = True

If Abs(dli - CDbl(Cfg("Target\_DLI\_mol", 17))) > 5 Then ok = False

If Abs(vpd - CDbl(Cfg("Target\_VPD\_kPa", 0.9))) > 0.5 Then ok = False

If Abs(EC - CDbl(Cfg("Target\_EC\_mScm", 2))) > 0.5 Then ok = False

If Abs(pH - CDbl(Cfg("Target\_pH", 5.8))) > 0.5 Then ok = False

If energy > CDbl(Cfg("MaxEnergy\_kWh\_day", 35)) Then ok = False

LogEvent "Env", IIf(ok, "SetpointsOK", "SetpointsOutOfRange"), "SysRow", CStr(systemRow), ""

End Sub

Module start and evidence logging Function StartModule(learnerID As String, modID As String) As Boolean

If Not ModuleEnabled(modID) Then LogEvent "Gate", "Denied", modID, "Disabled", "": Exit Function

If Not PrereqsMet(learnerID, modID) Then LogEvent "Gate", "Denied", modID, "Prereqs", "": Exit Function

If Not SafetyGateOK() Or Not EthicsGateOK() Then LogEvent "Gate", "Denied", modID, "Safety/Ethics", "": Exit Function

LogEvent "Gate", "Granted", learnerID, modID, ""

StartModule = True

End Function

Sub AddEvidence(activityID As String, learnerID As String, typ As String, uri As String, Optional verified As Boolean = False, Optional verifier As String = "")

Dim ev As Worksheet: Set ev = WS("Evidence")

Dim r As Long: r = ev.Cells(ev.Rows.Count, 1).End(xlUp).Row + 1

ev.Cells(r, 1).Value = "E" & Format(Now, "yymmddhhnnss")

ev.Cells(r, 2).Value = activityID

ev.Cells(r, 3).Value = learnerID

ev.Cells(r, 4).Value = typ

ev.Cells(r, 5).Value = uri

ev.Cells(r, 6).Value = NowStamp()

ev.Cells(r, 7).Value = verified

ev.Cells(r, 8).Value = verifier

LogEvent "Evidence", "Logged", learnerID, activityID, typ

End Sub

### Portfolio export

vba Sub ExportPortfolio(learnerID As String)

On Error Resume Next: Application.DisplayAlerts = False

ThisWorkbook.Worksheets("Portfolio").Delete

Application.DisplayAlerts = True: On Error GoTo 0

Dim wr As Worksheet: Set wr = ThisWorkbook.Worksheets.Add

wr.Name = "Portfolio"

Dim r As Long: r = 1

wr.Cells(r, 1) = "Portfolio of Evidence – Vertical Farming & SynBio": r = r + 2

wr.Cells(r, 1) = "LearnerID": wr.Cells(r, 2) = learnerID: r = r + 1

wr.Cells(r, 1) = "Generated": wr.Cells(r, 2) = NowStamp(): r = r + 2

r = CopySection(wr, r, "Evidence", WS("Evidence"), 3, learnerID)

r = CopySection(wr, r, "Metrics (Environment)", WS("Metrics"), 1, "Env")

r = CopySection(wr, r, "Metrics (Calc)", WS("Metrics"), 1, "Calc")

wr.Columns.AutoFit

Dim f As String: f = CStr(Cfg("EvidenceDir", ThisWorkbook.Path)) & "\PoE\_SynBio\_VF\_" & learnerID & ".pdf"

On Error Resume Next: wr.ExportAsFixedFormat xlTypePDF, f: On Error GoTo 0

LogEvent "Portfolio", "Exported", learnerID, "", f

MsgBox "Portfolio exported: " & f, vbInformation

End Sub

Function CopySection(dst As Worksheet, startRow As Long, title As String, src As Worksheet, matchCol As Integer, key As String) As Long

dst.Cells(startRow, 1) = title

Dim r As Long: r = startRow + 1

Dim rng As Range: Set rng = src.Range("A1").CurrentRegion

Dim i As Long, header As Boolean

For i = 2 To rng.Rows.Count

If CStr(rng.Cells(i, matchCol).Value) = key Then

If Not header Then rng.Rows(1).Copy dst.Cells(r, 1): r = r + 1: header = True

rng.Rows(i).Copy dst.Cells(r, 1): r = r + 1

End If

Next i

CopySection = r + 1

End Function

## Sample activities and usage

### Seed config

* CurrentUser = Tshingombe Fiston Tshitadi
* EvidenceDir = C:\Evidence
* MinSafetyItems = 3
* MinEthicsChecklist = 2
* MinPortfolioArtifacts = 6
* Target\_DLI\_mol = 17
* Target\_VPD\_kPa = 0.9
* Target\_EC\_mScm = 2.0
* Target\_pH = 5.8
* MaxEnergy\_kWh\_day = 35

### Curriculum rows

* 7.1 Masters in Vertical Farming & Synthetic Biology | Enabled TRUE | Prereqs: 7.2,7.3
* 7.2 Introduction to Urban Agriculture | Enabled TRUE
* 7.3 Fundamentals of Synthetic Biology | Enabled TRUE
* 7.4 Applications in Urban Farming | Enabled TRUE
* 7.6 Vertical Farm System Design | Enabled TRUE
* 7.7 Biotechnology Integration | Enabled TRUE
* 7.8 Environmental & Economic Impact | Enabled TRUE
* 7.9 Regulatory & Ethical Frameworks | Enabled TRUE
* 7.10 Future Trends & Equity Access | Enabled TRUE

### Activities (examples)

* A\_VF\_SYS | 7.6 | Lab | 6h | Outcomes: DLI,VPD,Energy | Standards: Biosafety L1,L2 | Required TRUE
* A\_SYN\_FAB | 7.3 | Project | 12h | Outcomes: Assembly Design | Standards: Biosecurity | Required TRUE
* A\_ETHICS | 7.9 | Assessment | 2h | Outcomes: Ethics Review | Required TRUE

### Run flow

1. Validate gates:
   * If StartModule("Learner01","7.6") returns True → proceed.
2. Validate environment:
   * ValidateSystem systemRow:=2 (Systems sheet populated).
3. Log evidence:
   * AddEvidence "A\_VF\_SYS","Learner01","Data","Systems!Row2",True,"Assessor"
4. Export portfolio:
   * ExportPortfolio "Learner01"

## Optional UserForm stubs

* UserForm\_VF: Form to input PPFD, photoperiod, temp, RH, EC, pH; buttons for Calculate (DLI/VPD/Energy), Save Evidence, Export PoE.
* UserForm\_SynBio: Ethics checklist with required toggles; DNA assembly planning notes; verification button.

Event handler sketch:

' In UserForm\_VF

Private Sub cmdCalculate\_Click()

Dim row As Long: row = CLng(Me.txtSystemRow.Value)

ValidateSystem row

MsgBox "Setpoints evaluated. See Metrics.", vbInformation

End Sub

Private Sub cmdSaveEvidence\_Click()

AddEvidence "A\_VF\_SYS", Me.txtLearnerID.Value, "Data", "Systems!Row" & Me.txtSystemRow.Value, True, "Assessor"

End Sub

Private Sub cmdExportPoE\_Click()

ExportPortfolio Me.txtLearnerID.Value

End Sub

## Extensions you can add next

* Growth rate predictor (simple degree-day or empirical yield vs DLI model).
* Bioreactor mode (dO2, pH control curves; EC/pH dosing event logs).
* Energy/carbon dashboard (kWh/day vs baseline and CO2e factors).
* Standards appendix (biosafety, ethics approvals) auto-attached to PoE.

If you want, I’ll wire a compact dashboard with tiles for DLI/VPD/EC/pH status, safety/ethics gates, and portfolio progress—ready for classroom demos and board submissions.

* Config
  + A: Key, B: Value
  + Keys to seed: CurrentUser, EvidenceDir, AcademicYear, MinCPD\_Annual, MinEthicsCredits, MinComplianceCredits, MinProjects, PassMark\_pct
* Roles
  + A: RoleID, B: Title (PolicyAnalyst/ComplianceOfficer/RiskConsultant/PublicSafety/EthicsAdvisor/SmartCityMgr/InnovationConsultant/DigiTransformLead/SustainabilityAuditor), C: Domain (Legal/Gov/Tech/Sustainability), D: KPIs (CSV), E: MinHours, F: MinArtifacts, G: Enabled (TRUE/FALSE)
* Competencies
  + A: CompID, B: Name (AdminLaw/ConLaw/Risk/Privacy/Cyber/ISO/ESG/Stakeholder/PM), C: Standard (e.g., ISO 37001, ISO 37120, IEEE 802), D: Credits, E: Domain
* Modules
  + A: ModuleID, B: Title, C: Credits, D: Outcomes (CSV CompIDs), E: Prereqs (CSV), F: Level, G: Enabled (TRUE/FALSE)
* Activities
  + A: ActivityID, B: ModuleID, C: Type (Lecture/Lab/Clinic/Case/Project/Assessment/Expo), D: Hours, E: Deliverables (Brief/Checklist/Dashboard/Policy), F: Required (TRUE/FALSE)
* Evidence
  + A: EvidenceID, B: ActivityID, C: LearnerID, D: Type (Doc/Photo/Data/Code/Log), E: URI\_or\_Path, F: Timestamp, G: Verified (TRUE/FALSE), H: Verifier, I: Notes
* Assessments
  + A: AssessmentID, B: ModuleID, C: LearnerID, D: Score, E: MaxScore, F: Passed (TRUE/FALSE), G: Date
* CPD
  + A: LearnerID, B: CompID, C: Credits, D: Source (Module/Evidence), E: Date
* EthicsCompliance
  + A: Item, B: Domain (Ethics/Compliance), C: Required (TRUE/FALSE), D: Completed (TRUE/FALSE), E: EvidenceID, F: Notes
* Events
  + A: Timestamp, B: User, C: Topic, D: EventType, E: K1, F: K2, G: Notes
* Portfolio
  + Generated automatically

## Logigram gates

* Role activation:
  + Role.Enabled = TRUE
  + CPD annual credits ≥ MinCPD\_Annual
  + Ethics credits ≥ MinEthicsCredits; Compliance credits ≥ MinComplianceCredits
  + Required activities for the role’s target modules have verified evidence ≥ MinArtifacts
  + Assessments for role‑critical modules passed with Score% ≥ PassMark\_pct
* Module activation:
  + Module.Enabled = TRUE and all Prereqs passed (Assessments)
* PoE export:
  + At least MinProjects deliverables present (policy brief, legal checklist, dashboard, case analysis)

## Algorigram flows

* ComputeCPD → ValidateEthics/Compliance → GateModule → RecordEvidence/Assessment → MapCreditsFromOutcomes → GateRole → ExportPortfolio

## Core VBA

### Utilities and logging

vba

Function WS(name As String) As Worksheet: Set WS = ThisWorkbook.Worksheets(name): End Function

Function Cfg(key As String, Optional defVal As Variant = "") As Variant

Dim r As Range: Set r = WS("Config").Columns(1).Find(key, , xlValues, xlWhole)

If r Is Nothing Then Cfg = defVal Else Cfg = IIf(IsEmpty(r.Offset(0, 1)), defVal, r.Offset(0, 1))

End Function

Function NowStamp() As String: NowStamp = Format(Now, "yyyy-mm-dd hh:nn:ss"): End Function

Sub LogEvent(topic As String, evt As String, Optional k1 As String = "", Optional k2 As String = "", Optional note As String = "")

Dim w As Worksheet: Set w = WS("Events")

Dim r As Long: r = w.Cells(w.Rows.Count, 1).End(xlUp).Row + 1

w.Cells(r, 1) = NowStamp(): w.Cells(r, 2) = Cfg("CurrentUser", "User")

w.Cells(r, 3) = topic: w.Cells(r, 4) = evt: w.Cells(r, 5) = k1: w.Cells(r, 6) = k2: w.Cells(r, 7) = note

End Sub

### Assessments and module gates

Function PassedModule(learnerID As String, moduleID As String) As Boolean

Dim a As Worksheet: Set a = WS("Assessments")

Dim last As Long: last = a.Cells(a.Rows.Count, 1).End(xlUp).Row

Dim i As Long, passPct As Double: passPct = CDbl(Cfg("PassMark\_pct", 50))

For i = 2 To last

If a.Cells(i, 2).Value = moduleID And a.Cells(i, 3).Value = learnerID Then

If a.Cells(i, 5).Value > 0 Then

If (a.Cells(i, 4).Value / a.Cells(i, 5).Value) \* 100# >= passPct Then PassedModule = True: Exit Function

End If

End If

Next i

End Function

Function PrereqsMet(learnerID As String, moduleID As String) As Boolean

Dim m As Worksheet: Set m = WS("Modules")

Dim r As Range: Set r = m.Columns(1).Find(moduleID, , xlValues, xlWhole)

If r Is Nothing Then Exit Function

Dim csv As String: csv = CStr(r.Offset(0, 4).Value)

If Len(Trim(csv)) = 0 Then PrereqsMet = True: Exit Function

Dim arr() As String: arr = Split(csv, ",")

Dim i As Long

For i = LBound(arr) To UBound(arr)

If Not PassedModule(learnerID, Trim(arr(i))) Then PrereqsMet = False: Exit Function

Next i

PrereqsMet = True

End Function

Function GateModule(learnerID As String, moduleID As String) As Boolean

Dim m As Worksheet: Set m = WS("Modules")

Dim r As Range: Set r = m.Columns(1).Find(moduleID, , xlValues, xlWhole)

If r Is Nothing Then LogEvent "GateModule", "Error", learnerID, moduleID, "Module not found": Exit Function

If Not CBool(r.Offset(0, 6).Value) Then LogEvent "GateModule", "Denied", learnerID, moduleID, "Module disabled": Exit Function

If Not PrereqsMet(learnerID, moduleID) Then LogEvent "GateModule", "Denied", learnerID, moduleID, "Prereqs unmet": Exit Function

LogEvent "GateModule", "Granted", learnerID, moduleID, ""

GateModule = True

End Function

### Evidence and role artifacts

Function RequiredActivitiesHaveEvidence(moduleID As String, learnerID As String) As Boolean

Dim act As Worksheet: Set act = WS("Activities")

Dim ev As Worksheet: Set ev = WS("Evidence")

Dim lastA As Long: lastA = act.Cells(act.Rows.Count, 1).End(xlUp).Row

Dim lastE As Long: lastE = ev.Cells(ev.Rows.Count, 1).End(xlUp).Row

Dim need As Long, have As Long, i As Long, j As Long

For i = 2 To lastA

If act.Cells(i, 2).Value = moduleID And CBool(act.Cells(i, 6).Value) Then

need = need + 1

For j = 2 To lastE

If ev.Cells(j, 2).Value = act.Cells(i, 1).Value And ev.Cells(j, 3).Value = learnerID And CBool(ev.Cells(j, 7).Value) Then

have = have + 1: Exit For

End If

Next j

End If

Next i

RequiredActivitiesHaveEvidence = (need = have)

End Function

### CPD calculators (annual, ethics, compliance)

vba

Function CPD\_Sum(learnerID As String, Optional compFilter As String = "") As Double

Dim c As Worksheet: Set c = WS("CPD")

Dim last As Long: last = c.Cells(c.Rows.Count, 1).End(xlUp).Row

Dim sumC As Double, i As Long

For i = 2 To last

If c.Cells(i, 1).Value = learnerID Then

If Len(compFilter) = 0 Or c.Cells(i, 2).Value = compFilter Then

sumC = sumC + Val(c.Cells(i, 3).Value)

End If

End If

Next i

CPD\_Sum = sumC

End Function

Sub MapCreditsFromModule(learnerID As String, moduleID As String)

' Map module outcomes to Competencies and post credits to CPD

Dim m As Worksheet: Set m = WS("Modules")

Dim r As Range: Set r = m.Columns(1).Find(moduleID, , xlValues, xlWhole)

If r Is Nothing Then Exit Sub

Dim outcomes As String: outcomes = CStr(r.Offset(0, 3).Value)

Dim credits As Double: credits = Val(r.Offset(0, 2).Value)

Dim arr() As String: arr = Split(outcomes, ",")

Dim i As Long

For i = LBound(arr) To UBound(arr)

WriteCPD learnerID, Trim(arr(i)), credits, "Module:" & moduleID

Next i

LogEvent "CPD", "Posted", learnerID, moduleID, "Credits=" & credits

End Sub

Sub WriteCPD(learnerID As String, compID As String, credits As Double, source As String)

Dim c As Worksheet: Set c = WS("CPD")

Dim r As Long: r = c.Cells(c.Rows.Count, 1).End(xlUp).Row + 1

c.Cells(r, 1).Value = learnerID

c.Cells(r, 2).Value = compID

c.Cells(r, 3).Value = credits

c.Cells(r, 4).Value = source

c.Cells(r, 5).Value = NowStamp()

End Sub

### Ethics & compliance gates and role gating

Function EthicsOK() As Boolean

Dim e As Worksheet: Set e = WS("EthicsCompliance")

Dim last As Long: last = e.Cells(e.Rows.Count, 1).End(xlUp).Row

Dim req As Long, ok As Long, i As Long

For i = 2 To last

If LCase(e.Cells(i, 2).Value) = "ethics" And CBool(e.Cells(i, 3).Value) Then

req = req + 1

If CBool(e.Cells(i, 4).Value) Then ok = ok + 1

End If

Next i

EthicsOK = (ok >= CLng(Cfg("MinEthicsCredits", 6)))

End Function

Function ComplianceOK() As Boolean

Dim e As Worksheet: Set e = WS("EthicsCompliance")

Dim last As Long: last = e.Cells(e.Rows.Count, 1).End(xlUp).Row

Dim req As Long, ok As Long, i As Long

For i = 2 To last

If LCase(e.Cells(i, 2).Value) = "compliance" And CBool(e.Cells(i, 3).Value) Then

req = req + 1

If CBool(e.Cells(i, 4).Value) Then ok = ok + 1

End If

Next i

ComplianceOK = (ok >= CLng(Cfg("MinComplianceCredits", 6)))

End Function

Function GateRole(learnerID As String, roleID As String) As Boolean

Dim rws As Worksheet: Set rws = WS("Roles")

Dim rr As Range: Set rr = rws.Columns(1).Find(roleID, , xlValues, xlWhole)

If rr Is Nothing Then LogEvent "GateRole", "Error", learnerID, roleID, "Role not found": Exit Function

If Not CBool(rr.Offset(0, 6).Value) Then LogEvent "GateRole", "Denied", learnerID, roleID, "Role disabled": Exit Function

Dim minCPD As Double: minCPD = CDbl(Cfg("MinCPD\_Annual", 20))

Dim minArt As Long: minArt = CLng(rr.Offset(0, 5).Value)

Dim minH As Double: minH = CDbl(rr.Offset(0, 4).Value)

If CPD\_Sum(learnerID) < minCPD Then LogEvent "GateRole", "Denied", learnerID, roleID, "CPD insufficient": Exit Function

If Not EthicsOK Or Not ComplianceOK Then LogEvent "GateRole", "Denied", learnerID, roleID, "Ethics/Compliance gate": Exit Function

If CountArtifacts(learnerID) < minArt Then LogEvent "GateRole", "Denied", learnerID, roleID, "Artifacts insufficient": Exit Function

If SumRequiredHours(learnerID) < minH Then LogEvent "GateRole", "Denied", learnerID, roleID, "Hours insufficient": Exit Function

LogEvent "GateRole", "Granted", learnerID, roleID, ""

GateRole = True

End Function

Function CountArtifacts(learnerID As String) As Long

Dim e As Worksheet: Set e = WS("Evidence")

Dim last As Long: last = e.Cells(e.Rows.Count, 1).End(xlUp).Row

Dim i As Long, n As Long

For i = 2 To last

If e.Cells(i, 3).Value = learnerID And CBool(e.Cells(i, 7).Value) Then n = n + 1

Next i

CountArtifacts = n

End Function

Function SumRequiredHours(learnerID As String) As Double

' Sum hours of Required activities with verified evidence

Dim act As Worksheet: Set act = WS("Activities")

Dim ev As Worksheet: Set ev = WS("Evidence")

Dim LA As Long: LA = act.Cells(act.Rows.Count, 1).End(xlUp).Row

Dim LE As Long: LE = ev.Cells(ev.Rows.Count, 1).End(xlUp).Row

Dim i As Long, j As Long, sumH As Double

For i = 2 To LA

If CBool(act.Cells(i, 6).Value) Then

For j = 2 To LE

If ev.Cells(j, 2).Value = act.Cells(i, 1).Value And ev.Cells(j, 3).Value = learnerID And CBool(ev.Cells(j, 7).Value) Then

sumH = sumH + Val(act.Cells(i, 4).Value): Exit For

End If

Next j

End If

Next i

SumRequiredHours = sumH

End Function

Evidence log and verificationub LogEvidence(activityID As String, learnerID As String, typ As String, uri As String, Optional verified As Boolean = False, Optional verifier As String = "")

Dim ev As Worksheet: Set ev = WS("Evidence")

Dim r As Long: r = ev.Cells(ev.Rows.Count, 1).End(xlUp).Row + 1

ev.Cells(r, 1).Value = "E" & Format(Now, "yymmddhhnnss")

ev.Cells(r, 2).Value = activityID

ev.Cells(r, 3).Value = learnerID

ev.Cells(r, 4).Value = typ

ev.Cells(r, 5).Value = uri

ev.Cells(r, 6).Value = NowStamp()

ev.Cells(r, 7).Value = verified

ev.Cells(r, 8).Value = verifier

LogEvent "Evidence", "Logged", learnerID, activityID, typ

End Sub

Sub VerifyEvidence(evidenceID As String, verifier As String, Optional note As String = "")

Dim ev As Worksheet: Set ev = WS("Evidence")

Dim r As Range: Set r = ev.Columns(1).Find(evidenceID, , xlValues, xlWhole)

If r Is Nothing Then LogEvent "Evidence", "Error", evidenceID, verifier, "Not found": Exit Sub

r.Offset(0, 6).Value = True

r.Offset(0, 7).Value = verifier

r.Offset(0, 8).Value = note

LogEvent "Evidence", "Verified", evidenceID, verifier, note

End Sub

### Portfolio export

vba

Sub ExportPortfolio(learnerID As String)

On Error Resume Next: Application.DisplayAlerts = False

ThisWorkbook.Worksheets("Portfolio").Delete

Application.DisplayAlerts = True: On Error GoTo 0

Dim wr As Worksheet: Set wr = ThisWorkbook.Worksheets.Add

wr.Name = "Portfolio"

Dim r As Long: r = 1

wr.Cells(r, 1) = "Portfolio of Evidence – Legal, Governance, Cross-Sector Leadership": r = r + 2

wr.Cells(r, 1) = "LearnerID": wr.Cells(r, 2) = learnerID: r = r + 1

wr.Cells(r, 1) = "AcademicYear": wr.Cells(r, 2) = CStr(Cfg("AcademicYear", "")): r = r + 2

r = CopySection(wr, r, "Evidence", WS("Evidence"), 3, learnerID)

r = CopySection(wr, r, "Assessments", WS("Assessments"), 3, learnerID)

r = CopySection(wr, r, "CPD Credits", WS("CPD"), 1, learnerID)

wr.Columns.AutoFit

Dim f As String: f = CStr(Cfg("EvidenceDir", ThisWorkbook.Path)) & "\PoE\_LegalGov\_" & learnerID & ".pdf"

On Error Resume Next: wr.ExportAsFixedFormat xlTypePDF, f: On Error GoTo 0

LogEvent "Portfolio", "Exported", learnerID, "", f

MsgBox "Portfolio exported: " & f, vbInformation

End Sub

Function CopySection(dst As Worksheet, startRow As Long, title As String, src As Worksheet, matchCol As Integer, key As String) As Long

dst.Cells(startRow, 1) = title

Dim r As Long: r = startRow + 1

Dim rng As Range: Set rng = src.Range("A1").CurrentRegion

Dim i As Long, header As Boolean

For i = 2 To rng.Rows.Count

If CStr(rng.Cells(i, matchCol).Value) = key Then

If Not header Then rng.Rows(1).Copy dst.Cells(r, 1): r = r + 1: header = True

rng.Rows(i).Copy dst.Cells(r, 1): r = r + 1

End If

Next i

CopySection = r + 1

End Function

## Seed data examples

* Config:
  + CurrentUser = Tshingombe Fiston Tshitadi
  + EvidenceDir = C:\Evidence
  + AcademicYear = 2025
  + MinCPD\_Annual = 20
  + MinEthicsCredits = 6
  + MinComplianceCredits = 6
  + MinProjects = 2
  + PassMark\_pct = 60
* Roles (samples):
  + R\_POLICY | Public Policy Analyst | Gov | KPIs: briefs,on‑time | 60 | 6 | TRUE
  + R\_COMPLI | Regulatory Compliance Officer | Legal | KPIs: audits,findings | 60 | 6 | TRUE
  + R\_RISK | Legal Risk Consultant (Tech) | Tech | KPIs: DPIA,InfoSec | 60 | 6 | TRUE
  + R\_SAFETY | Public Safety Strategist | Gov | KPIs: SOPs,drills | 60 | 6 | TRUE
  + R\_ETHICS | Ethics & Governance Advisor | Legal | KPIs: boards,opinions | 60 | 6 | TRUE
  + R\_SMART | Smart City Program Manager | Tech | KPIs: delivery,SLAs | 80 | 8 | TRUE
  + R\_DX | Digital Transformation Lead | Tech | KPIs: adoption,ROI | 80 | 8 | TRUE
  + R\_SUS | Sustainable Infrastructure Auditor | Sustainability | KPIs: ISO scores | 80 | 8 | TRUE
* Competencies:
  + C\_CONLAW | Constitutional & Admin Law | ISO‑Gov | 3 | Legal
  + C\_PRIV | Privacy & Data Governance | ISO/IEC 27701 | 3 | Tech
  + C\_PF | Power Factor/ISO 50001 Reporting | ISO 50001 | 2 | Sustainability
  + C\_ESG | ESG & ISO 37120 Smart Cities | ISO 37120 | 3 | Sustainability
  + C\_RISK | Legal Risk & DPIA | GDPR/DPIA | 3 | Legal
* Modules (samples):
  + M\_LAW101 | Public Admin & ConLaw | 4 | C\_CONLAW | | NQF7 | TRUE
  + M\_PRIV201| Tech Law & Privacy | 4 | C\_PRIV,C\_RISK | M\_LAW101 | NQF8 | TRUE
  + M\_ESG301 | Smart Cities & ISO 37120 | 4 | C\_ESG | | NQF8 | TRUE
* Activities:
  + A\_BRIEF | M\_LAW101 | Case | 8 | Legal brief | TRUE
  + A\_AUDIT | M\_PRIV201| Project | 12 | DPIA + compliance checklist | TRUE
  + A\_DASH | M\_ESG301 | Project | 10 | KPI dashboard | TRUE

Use flow:

1. GateModule "Learner01","M\_LAW101"
2. LogEvidence "A\_BRIEF","Learner01","Doc","brief.pdf",True,"Assessor"
3. Post assessment and MapCreditsFromModule "Learner01","M\_LAW101"
4. Repeat for M\_PRIV201 and M\_ESG301
5. GateRole "Learner01","R\_SMART"
6. ExportPortfolio "Learner01"

## Optional UI stubs

* UserForm\_RoleGate: inputs LearnerID/RoleID → calls GateRole and displays KPI progress.
* UserForm\_Portfolio: lists Evidence and a “Export PoE” button.

Prompt wrappers:

Sub GateRolePrompt()

Dim L As String: L = InputBox("LearnerID:")

Dim R As String: R = InputBox("RoleID:")

If Len(L) \* Len(R) = 0 Then Exit Sub

If GateRole(L, R) Then MsgBox "Role gate granted." Else MsgBox "Role gate denied."

End Sub

Sub ExportPortfolioPrompt()

Dim L As String: L = InputBox("LearnerID:")

If Len(L) = 0 Then Exit Sub

ExportPortfolio L

End Sub

## Want me to wire a CPD dashboard?

I can add a dashboard sheet with tiles for:

* CPD total vs MinCPD\_Annual
* Ethics/Compliance gate status
* Required activities evidence coverage
* Pass rates and role gate status

Say the word, and I’ll tailor it to your exact sheets and reporting needs.

## Workbook schema

Create these sheets with the exact headers (row 1). The code relies on these names.

* Config
  + Key, Value
  + Seed: CurrentUser, EvidenceDir, ModuleID\_Current, Target\_WQI, Max\_Abstraction\_m3\_d, Min\_EnvFlow\_m3\_d, Min\_Stakeholder\_Records, PassMark\_pct
* Sites
  + SiteID, Name, Basin, ClimateZone, SensitiveReceptor, Notes
* Sources
  + SiteID, Date, SourceType (Surface/GW/Return/Third‑party), Flow\_m3\_d, EC\_uScm, pH, TSS\_mgL, SO4\_mgL, Mn\_mgL, Fe\_mgL
* Uses
  + SiteID, Date, UseType (Process/Dust/Tails/Misc), Flow\_m3\_d, ReturnFrac\_pct, LossFrac\_pct, Notes
* Quality
  + SiteID, Date, SamplePoint, EC\_uScm, pH, TSS\_mgL, SO4\_mgL, Mn\_mgL, Fe\_mgL, WQI\_Score
* Compliance
  + SiteID, PermitID, LimitName, LimitValue, Unit, Method (avg/95p/max), Window\_d, Active (TRUE/FALSE)
* Stakeholders
  + SiteID, Date, Party, Category (Community/Regulator/NGO/Worker), Concern, Action, Status
* Activities
  + ActivityID, Submodule (14.1–14.10), Type (Lab/Report/Model/Assessment/Engagement), Hours, Deliverable, Required (TRUE/FALSE)
* Evidence
  + EvidenceID, ActivityID, LearnerID, Type (Doc/Photo/Data/Code/Log), URI\_or\_Path, Timestamp, Verified (TRUE/FALSE), Verifier, Notes
* Assessments
  + AssessmentID, Submodule, LearnerID, Score, MaxScore, Passed (TRUE/FALSE), Date
* Events
  + Timestamp, User, Topic, EventType, K1, K2, Notes
* Metrics
  + Topic, Metric, Value, Unit, Timestamp
* Portfolio
  + Generated automatically (no manual headers)

## Logigram gates

* Curriculum gate:
  + ModuleID\_Current set; submodule enabled through Activities table.
* Hydrology gate:
  + Daily water balance computed; Abstraction ≤ Max\_Abstraction\_m3\_d; Environmental flow reserve met (Min\_EnvFlow\_m3\_d).
* Quality gate:
  + WQI computed; below Target\_WQI threshold triggers warning.
* Compliance gate:
  + Active permit limits respected over method/window.
* Stakeholder gate:
  + At least Min\_Stakeholder\_Records logged with Status ≠ “Open”.
* Evidence gate:
  + All Required activities for the submodule have verified Evidence.

## Algorigram flows

1. Ingest data (Sources, Uses, Quality) → 2) Compute water balance and WQI → 3) Check permits and env flows → 4) Stakeholder log check → 5) Validate required evidence → 6) Export portfolio.

## Core VBA

### Utilities and logging

vba

ption Explicit

Function WS(name As String) As Worksheet: Set WS = ThisWorkbook.Worksheets(name): End Function

Function Cfg(key As String, Optional defVal As Variant = "") As Variant

Dim r As Range: Set r = WS("Config").Columns(1).Find(key, , xlValues, xlWhole)

If r Is Nothing Then Cfg = defVal Else Cfg = IIf(IsEmpty(r.Offset(0, 1)), defVal, r.Offset(0, 1))

End Function

Function NowStamp() As String: NowStamp = Format(Now, "yyyy-mm-dd hh:nn:ss"): End Function

Sub LogEvent(topic As String, evt As String, Optional k1 As String = "", Optional k2 As String = "", Optional notes As String = "")

Dim w As Worksheet: Set w = WS("Events")

Dim r As Long: r = w.Cells(w.Rows.Count, 1).End(xlUp).Row + 1

w.Cells(r, 1).Value = NowStamp()

w.Cells(r, 2).Value = Cfg("CurrentUser", "User")

w.Cells(r, 3).Value = topic

w.Cells(r, 4).Value = evt

w.Cells(r, 5).Value = k1

w.Cells(r, 6).Value = k2

w.Cells(r, 7).Value = notes

End Sub

Sub LogMetric(topic As String, metric As String, val As Double, unitStr As String)

Dim w As Worksheet: Set w = WS("Metrics")

Dim r As Long: r = w.Cells(w.Rows.Count, 1).End(xlUp).Row + 1

w.Cells(r, 1).Value = topic

w.Cells(r, 2).Value = metric

w.Cells(r, 3).Value = val

w.Cells(r, 4).Value = unitStr

w.Cells(r, 5).Value = NowStamp()

End Sub

### Water balance (per site, per day)

Function SumFlow(ws As Worksheet, siteID As String, theDate As Date, colFlow As Long) As Double

Dim last As Long: last = ws.Cells(ws.Rows.Count, 1).End(xlUp).Row

Dim i As Long, sumv As Double

For i = 2 To last

If ws.Cells(i, 1).Value = siteID And CDate(ws.Cells(i, 2).Value) = theDate Then

sumv = sumv + Val(ws.Cells(i, colFlow).Value)

End If

Next i

SumFlow = sumv

End Function

Sub ComputeDailyBalance(siteID As String, theDate As Date)

Dim inflow As Double: inflow = SumFlow(WS("Sources"), siteID, theDate, 5) ' Flow\_m3\_d

Dim useFlow As Double: useFlow = SumFlow(WS("Uses"), siteID, theDate, 4)

Dim returns As Double, losses As Double

' Compute returns and losses from Uses sheet

Dim u As Worksheet: Set u = WS("Uses")

Dim last As Long: last = u.Cells(u.Rows.Count, 1).End(xlUp).Row

Dim i As Long

For i = 2 To last

If u.Cells(i, 1).Value = siteID And CDate(u.Cells(i, 2).Value) = theDate Then

returns = returns + Val(u.Cells(i, 4).Value) \* Val(u.Cells(i, 5).Value) / 100#

losses = losses + Val(u.Cells(i, 4).Value) \* Val(u.Cells(i, 6).Value) / 100#

End If

Next i

Dim abstraction As Double: abstraction = inflow ' treat Sources as abstraction + third-party; refine as needed

Dim balance As Double: balance = inflow - useFlow + returns - losses

LogMetric "Hydro", "Inflow\_m3\_d", inflow, "m3/d"

LogMetric "Hydro", "Use\_m3\_d", useFlow, "m3/d"

LogMetric "Hydro", "Return\_m3\_d", returns, "m3/d"

LogMetric "Hydro", "Loss\_m3\_d", losses, "m3/d"

LogMetric "Hydro", "Balance\_m3\_d", balance, "m3/d"

' Gates

If abstraction > CDbl(Cfg("Max\_Abstraction\_m3\_d", 1E+9)) Then \_

LogEvent "HydroGate", "OverAbstraction", siteID, CStr(theDate), "Abstraction exceeds permit"

If returns < CDbl(Cfg("Min\_EnvFlow\_m3\_d", 0)) Then \_

LogEvent "HydroGate", "EnvFlowLow", siteID, CStr(theDate), "Environmental flow not met"

End Sub

Water quality index (WQI) and scoringFunction NormalizeScore(val As Double, good As Double, bad As Double, Optional invert As Boolean = False) As Double

Dim s As Double

If invert = False Then

s = Application.Max(0, Application.Min(100, 100 \* (val - bad) / (good - bad)))

Else

s = Application.Max(0, Application.Min(100, 100 \* (bad - val) / (bad - good)))

End If

NormalizeScore = s

End Function

Function ComputeWQI\_Row(siteRow As Long) As Double

' Quality sheet columns: EC(5), pH(6), TSS(7), SO4(8), Mn(9), Fe(10)

Dim q As Worksheet: Set q = WS("Quality")

Dim sEC As Double: sEC = NormalizeScore(q.Cells(siteRow, 5).Value, 500, 2000, True)

Dim spH As Double: spH = NormalizeScore(q.Cells(siteRow, 6).Value, 7.0, 4.5, False) ' closer to 7 better

Dim sTSS As Double: sTSS = NormalizeScore(q.Cells(siteRow, 7).Value, 25, 200, True)

Dim sSO4 As Double: sSO4 = NormalizeScore(q.Cells(siteRow, 8).Value, 250, 1000, True)

Dim sMn As Double: sMn = NormalizeScore(q.Cells(siteRow, 9).Value, 0.5, 3, True)

Dim sFe As Double: sFe = NormalizeScore(q.Cells(siteRow, 10).Value, 0.3, 2, True)

ComputeWQI\_Row = Round((sEC + spH + sTSS + sSO4 + sMn + sFe) / 6, 1)

End Function

Sub UpdateWQI(siteID As String, theDate As Date)

Dim q As Worksheet: Set q = WS("Quality")

Dim last As Long: last = q.Cells(q.Rows.Count, 1).End(xlUp).Row

Dim i As Long

For i = 2 To last

If q.Cells(i, 1).Value = siteID And CDate(q.Cells(i, 2).Value) = theDate Then

Dim wqi As Double: wqi = ComputeWQI\_Row(i)

q.Cells(i, 11).Value = wqi

LogMetric "Quality", "WQI", wqi, "score"

If wqi < CDbl(Cfg("Target\_WQI", 60)) Then \_

LogEvent "QualityGate", "WQI\_Low", siteID, CStr(theDate), "Quality below target"

End If

Next i

End Sub

### Permit compliance check (rolling window)

Function StatWindow(vals As Variant, method As String) As Double

Dim n As Long: n = UBound(vals) - LBound(vals) + 1

If n <= 0 Then StatWindow = 0: Exit Function

Dim i As Long, arr() As Double, k As Long

ReDim arr(1 To n)

For i = 1 To n: arr(i) = vals(i, 1): Next i

Select Case LCase(method)

Case "avg": StatWindow = WorksheetFunction.Average(arr)

Case "max": StatWindow = WorksheetFunction.Max(arr)

Case "95p": StatWindow = WorksheetFunction.Percentile\_Inc(arr, 0.95)

Case Else: StatWindow = WorksheetFunction.Average(arr)

End Select

End Function

Sub CheckCompliance(siteID As String, theDate As Date)

Dim c As Worksheet: Set c = WS("Compliance")

Dim q As Worksheet: Set q = WS("Quality")

Dim lastC As Long: lastC = c.Cells(c.Rows.Count, 1).End(xlUp).Row

Dim i As Long

For i = 2 To lastC

If c.Cells(i, 1).Value = siteID And CBool(c.Cells(i, 8).Value) Then

Dim limitName As String: limitName = c.Cells(i, 3).Value

Dim lim As Double: lim = c.Cells(i, 4).Value

Dim meth As String: meth = c.Cells(i, 6).Value

Dim win As Long: win = CLng(c.Cells(i, 7).Value)

' Build window from Quality for the parameter

Dim vals As Variant: vals = PullQualityWindow(q, siteID, theDate, limitName, win)

Dim statv As Double: statv = StatWindow(vals, meth)

LogMetric "Compliance", limitName & "\_" & meth, statv, CStr(c.Cells(i, 5).Value)

If statv > lim Then

LogEvent "ComplianceGate", "Exceedance", siteID, limitName, "Value=" & statv & " > Limit=" & lim

End If

End If

Next i

End Sub

Function PullQualityWindow(q As Worksheet, siteID As String, theDate As Date, param As String, win As Long) As Variant

Dim last As Long: last = q.Cells(q.Rows.Count, 1).End(xlUp).Row

Dim i As Long, cnt As Long

Dim startDate As Date: startDate = theDate - win + 1

ReDim arr(1 To 1, 1 To 1) As Double

For i = 2 To last

If q.Cells(i, 1).Value = siteID Then

Dim d As Date: d = CDate(q.Cells(i, 2).Value)

If d >= startDate And d <= theDate Then

Dim v As Double

Select Case LCase(param)

Case "ec": v = q.Cells(i, 4).Value

Case "tss": v = q.Cells(i, 6).Value

Case "so4": v = q.Cells(i, 7).Value

Case "mn": v = q.Cells(i, 8).Value

Case "fe": v = q.Cells(i, 9).Value

Case "ph": v = q.Cells(i, 5).Value

Case Else: v = q.Cells(i, 4).Value

End Select

cnt = cnt + 1

If cnt = 1 Then

ReDim arr(1 To 1, 1 To 1)

Else

ReDim Preserve arr(1 To cnt, 1 To 1)

End If

arr(cnt, 1) = v

End If

End If

Next i

PullQualityWindow = arr

End Function

Stakeholder gate and evidence coverage

Function StakeholderGateOK(siteID As String) As Boolean

Dim s As Worksheet: Set s = WS("Stakeholders")

Dim last As Long: last = s.Cells(s.Rows.Count, 1).End(xlUp).Row

Dim i As Long, closed As Long

For i = 2 To last

If s.Cells(i, 1).Value = siteID Then

If LCase(s.Cells(i, 7).Value) <> "open" Then closed = closed + 1

End If

Next i

StakeholderGateOK = (closed >= CLng(Cfg("Min\_Stakeholder\_Records", 3)))

End Function

Function RequiredActivitiesHaveEvidence(submodule As String, learnerID As String) As Boolean

Dim act As Worksheet: Set act = WS("Activities")

Dim ev As Worksheet: Set ev = WS("Evidence")

Dim LA As Long: LA = act.Cells(act.Rows.Count, 1).End(xlUp).Row

Dim LE As Long: LE = ev.Cells(ev.Rows.Count, 1).End(xlUp).Row

Dim need As Long, have As Long, i As Long, j As Long

For i = 2 To LA

If act.Cells(i, 2).Value = submodule And CBool(act.Cells(i, 6).Value) Then

need = need + 1

For j = 2 To LE

If ev.Cells(j, 2).Value = act.Cells(i, 1).Value And ev.Cells(j, 3).Value = learnerID And CBool(ev.Cells(j, 7).Value) Then

have = have + 1: Exit For

End If

Next j

End If

Next i

RequiredActivitiesHaveEvidence = (need = have)

End Function

Sub Run\_IWM\_Day(siteID As String, theDate As Date, learnerID As String, submodule As String)

ComputeDailyBalance siteID, theDate

UpdateWQI siteID, theDate

CheckCompliance siteID, theDate

Dim ok As Boolean: ok = True

If Not StakeholderGateOK(siteID) Then ok = False: LogEvent "StakeholderGate", "Fail", siteID, "", ""

If Not RequiredActivitiesHaveEvidence(submodule, learnerID) Then ok = False: LogEvent "EvidenceGate", "Fail", submodule, learnerID, ""

LogEvent "Module14", IIf(ok, "GatesPass", "GatesFail"), siteID, submodule, CStr(theDate))

End Sub

Sub ExportPortfolio\_IWM(learnerID As String)

On Error Resume Next: Application.DisplayAlerts = False

ThisWorkbook.Worksheets("Portfolio").Delete

Application.DisplayAlerts = True: On Error GoTo 0

Dim wr As Worksheet: Set wr = ThisWorkbook.Worksheets.Add

wr.Name = "Portfolio"

Dim r As Long: r = 1

wr.Cells(r, 1) = "Portfolio of Evidence – Integrated Water Management in Mining": r = r + 2

wr.Cells(r, 1) = "LearnerID": wr.Cells(r, 2) = learnerID: r = r + 1

wr.Cells(r, 1) = "Module": wr.Cells(r, 2) = "14 – IWM": r = r + 2

r = CopySection(wr, r, "Events", WS("Events"), 3, "Module14")

r = CopySection(wr, r, "Metrics – Hydro", WS("Metrics"), 1, "Hydro")

r = CopySection(wr, r, "Metrics – Quality", WS("Metrics"), 1, "Quality")

r = CopySection(wr, r, "Metrics – Compliance", WS("Metrics"), 1, "Compliance")

r = CopySection(wr, r, "Evidence", WS("Evidence"), 3, learnerID)

wr.Columns.AutoFit

Dim f As String: f = CStr(Cfg("EvidenceDir", ThisWorkbook.Path)) & "\PoE\_IWM\_" & learnerID & ".pdf"

On Error Resume Next: wr.ExportAsFixedFormat xlTypePDF, f: On Error GoTo 0

LogEvent "Portfolio", "Exported", learnerID, "", f

MsgBox "Portfolio exported: " & f, vbInformation

End Sub

Function CopySection(dst As Worksheet, startRow As Long, title As String, src As Worksheet, matchCol As Integer, key As String) As Long

dst.Cells(startRow, 1) = title

Dim r As Long: r = startRow + 1

Dim rng As Range: Set rng = src.Range("A1").CurrentRegion

Dim i As Long, header As Boolean

For i = 2 To rng.Rows.Count

If CStr(rng.Cells(i, matchCol).Value) = key Then

If Not header Then rng.Rows(1).Copy dst.Cells(r, 1): r = r + 1: header = True

rng.Rows(i).Copy dst.Cells(r, 1): r = r + 1

End If

Next i

CopySection = r + 1

End Function

## Portfolio components and assessment mapping

* 14.1–14.2 Water Use & Planning
  + Evidence: Hydrological assessment report; daily water balance logs
  + Skills: Resource evaluation, planning frameworks
* 14.4 Water Quality Management
  + Evidence: WQI dashboard; compliance window checks
  + Skills: Monitoring, pollution control
* 14.5 Regulatory Compliance
  + Evidence: Exceedance register; permit checklist
  + Skills: Regulatory mapping, audit readiness
* 14.7 Stakeholder Engagement
  + Evidence: Stakeholder log with actions/closure
  + Skills: Engagement, conflict avoidance
* 14.8 Climate Adaptation
  + Evidence: Risk matrix; drought/flood scenario notes
  + Skills: Resilience planning

Record assessment outcomes in Assessments (Score/MaxScore/Passed). Use Events to document gate passes/fails.

## Quick start

* Config:
  + CurrentUser = Tshingombe Fiston Tshitadi
  + EvidenceDir = C:\Evidence
  + ModuleID\_Current = 14
  + Target\_WQI = 60
  + Max\_Abstraction\_m3\_d = 15000
  + Min\_EnvFlow\_m3\_d = 1000
  + Min\_Stakeholder\_Records = 3
  + PassMark\_pct = 60
* Activities (examples):
  + A\_14\_1 | 14.1 | Report | 6 | Hydrological assessment | TRUE
  + A\_14\_4 | 14.4 | Model | 4 | WQI & permit window calc | TRUE
  + A\_14\_5 | 14.5 | Assessment | 2 | Compliance checklist | TRUE
  + A\_14\_7 | 14.7 | Engagement | 2 | Stakeholder log entries | TRUE

Run a day:

* Run\_IWM\_Day "SITE01", DateValue("2025-09-12"), "Learner01", "14.1"
* ExportPortfolio\_IWM "Learner01"

## Optional extensions

* Tailings water return modeling (pond residence, seepage).
* Cost and penalty tracker tied to exceedances.
* Climate scenarios: stochastic inflow/evaporation generators.
* Automated charts on Portfolio (balance, WQI trend, exceedance timeline).

If you want, I’ll add a lightweight UserForm to input daily flows/quality and push buttons for “Compute Day,” “Log Evidence,” and “Export PoE.”

This VBA engine operationalizes your multi-domain curriculum—spanning network security, incident response, AI-based threat detection, and blockchain credentialing—into a modular, audit-ready system. It enforces readiness gates (logigram), orchestrates evidence flows (algorigram), and exports a CPD-aligned portfolio for institutional submission.

## 🧠 System Overview

| **Domain** | **Modules** | **Evidence Types** | **Tools** |
| --- | --- | --- | --- |
| Network Security | VPN, IDS/IPS, DDoS | Topology maps, config files | pfSense, Snort, Wireshark |
| Incident Response | Playbooks, RCA | Timeline, analysis reports | Splunk, ELK, NIST CSF |
| Bibliotechnology | AI, Blockchain | Threat models, access logs | TensorFlow, Hyperledger |
| Edge Computing | IoT, Smart Grid | Case studies, analytics | Azure IoT, Grafana |
| Credentialing | DLT, Smart Contracts | Verification logs, LMS dashboards | Ethereum, Solidity |

## 📁 Workbook Schema

Create these sheets with exact headers:

* Config
  + Key, Value
  + Seed: CurrentUser, EvidenceDir, MinArtifacts, MinSecurityScore, MinRecoveryScore, MinDLTScore, PassMark\_pct
* Modules
  + ModuleID, Title, Domain, Credits, Prereqs (CSV), Enabled (TRUE/FALSE)
* Activities
  + ActivityID, ModuleID, Type (Lab/Report/Model/Assessment), Hours, Deliverable, Required (TRUE/FALSE)
* Evidence
  + EvidenceID, ActivityID, LearnerID, Type (Doc/Photo/Data/Code/Log), URI\_or\_Path, Timestamp, Verified (TRUE/FALSE), Verifier, Notes
* Assessments
  + AssessmentID, ModuleID, LearnerID, Score, MaxScore, Passed (TRUE/FALSE), Date
* Metrics
  + Topic, Metric, Value, Unit, Timestamp
* Events
  + Timestamp, User, Topic, EventType, K1, K2, Notes
* Portfolio
  + Generated automatically

## 🔐 Logigram Gates

* Module Enabled = TRUE
* Prereqs passed (Assessments)
* Required Activities have verified Evidence
* SecurityScore ≥ MinSecurityScore
* RecoveryScore ≥ MinRecoveryScore
* DLTScore ≥ MinDLTScore
* Total artifacts ≥ MinArtifacts

## 🔄 Algorigram Flow

graph TD

A[Start Module] --> B[Check Prereqs]

B --> C[Validate Activities]

C --> D[Log Evidence]

D --> E[Run Security & Recovery Scoring]

E --> F[DLT Integration Check]

F --> G[Export Portfolio]

## 🔧 Core VBA

Function WS(name As String) As Worksheet: Set WS = ThisWorkbook.Worksheets(name): End Function

Function Cfg(key As String, Optional defVal As Variant = "") As Variant

Dim r As Range: Set r = WS("Config").Columns(1).Find(key, , xlValues, xlWhole)

If r Is Nothing Then Cfg = defVal Else Cfg = r.Offset(0, 1).Value

End Function

Function NowStamp() As String: NowStamp = Format(Now, "yyyy-mm-dd hh:nn:ss"): End

Evidence Logging

Sub LogEvidence(activityID As String, learnerID As String, typ As String, uri As String, Optional verified As Boolean = False, Optional verifier As String = "")

Dim ev As Worksheet: Set ev = WS("Evidence")

Dim r As Long: r = ev.Cells(ev.Rows.Count, 1).End(xlUp).Row + 1

ev.Cells(r, 1).Value = "E" & Format(Now, "yymmddhhnnss")

ev.Cells(r, 2).Value = activityID

ev.Cells(r, 3).Value = learnerID

ev.Cells(r, 4).Value = typ

ev.Cells(r, 5).Value = uri

ev.Cells(r, 6).Value = NowStamp()

ev.Cells(r, 7).Value = verified

ev.Cells(r, 8).Value = verifier

End Sub

Assessment Gate

Function PassedModule(learnerID As String, moduleID As String) As Boolean

Dim a As Worksheet: Set a = WS("Assessments")

Dim i As Long, passPct As Double: passPct = Cfg("PassMark\_pct", 60)

For i = 2 To a.Cells(a.Rows.Count, 1).End(xlUp).Row

If a.Cells(i, 2).Value = moduleID And a.Cells(i, 3).Value = learnerID Then

If a.Cells(i, 5).Value > 0 Then

If (a.Cells(i, 4).Value / a.Cells(i, 5).Value) \* 100 >= passPct Then PassedModule = True: Exit Function

End If

End If

Next i

End Function

Activity Evidence Gate

Function RequiredActivitiesHaveEvidence(moduleID As String, learnerID As String) As Boolean

Dim act As Worksheet: Set act = WS("Activities")

Dim ev As Worksheet: Set ev = WS("Evidence")

Dim need As Long, have As Long, i As Long, j As Long

For i = 2 To act.Cells(act.Rows.Count, 1).End(xlUp).Row

If act.Cells(i, 2).Value = moduleID And CBool(act.Cells(i, 6).Value) Then

need = need + 1

For j = 2 To ev.Cells(ev.Rows.Count, 1).End(xlUp).Row

If ev.Cells(j, 2).Value = act.Cells(i, 1).Value And ev.Cells(j, 3).Value = learnerID And CBool(ev.Cells(j, 7).Value) Then

have = have + 1: Exit For

End If

Next j

End If

Next i

RequiredActivitiesHaveEvidence = (need = have)

End Function

### Security, Recovery, and DLT Scoring

Function ScoreMetric(topic As String, metric As String) As Double

Dim m As Worksheet: Set m = WS("Metrics")

Dim i As Long, score As Double

For i = 2 To m.Cells(m.Rows.Count, 1).End(xlUp).Row

If m.Cells(i, 1).Value = topic And m.Cells(i, 2).Value = metric Then

score = score + Val(m.Cells(i, 3).Value)

End If

Next i

ScoreMetric = score

End Function

Function SecurityGateOK() As Boolean

SecurityGateOK = (ScoreMetric("Security", "Score") >= Cfg("MinSecurityScore", 70))

End Function

Function RecoveryGateOK() As Boolean

RecoveryGateOK = (ScoreMetric("Recovery", "Score") >= Cfg("MinRecoveryScore", 70))

End Function

Function DLTGateOK() As Boolean

DLTGateOK = (ScoreMetric("DLT", "Score") >= Cfg("MinDLTScore", 70))

End Function

### Portfolio Export

Sub ExportPortfolio(learnerID As String)

On Error Resume Next: Application.DisplayAlerts = False

ThisWorkbook.Worksheets("Portfolio").Delete

Application.DisplayAlerts = True: On Error GoTo 0

Dim wr As Worksheet: Set wr = ThisWorkbook.Worksheets.Add

wr.Name = "Portfolio"

Dim r As Long: r = 1

wr.Cells(r, 1).Value = "Portfolio – Cybersecurity, Bibliotechnology, Edge Intelligence": r = r + 2

wr.Cells(r, 1).Value = "LearnerID": wr.Cells(r, 2).Value = learnerID: r = r + 2

r = CopySection(wr, r, "Evidence", WS("Evidence"), 3, learnerID)

r = CopySection(wr, r, "Assessments", WS("Assessments"), 3, learnerID)

r = CopySection(wr, r, "Metrics", WS("Metrics"), 1, "Security")

r = CopySection(wr, r, "Metrics", WS("Metrics"), 1, "Recovery")

r = CopySection(wr, r, "Metrics", WS("Metrics"), 1, "DLT")

wr.Columns.AutoFit

Dim f As String: f = Cfg("EvidenceDir", ThisWorkbook.Path) & "\Portfolio\_" & learnerID & ".pdf"

wr.ExportAsFixedFormat xlTypePDF, f

MsgBox "Portfolio exported: " & f, vbInformation

End Sub

Function CopySection(dst As Worksheet, startRow As Long, title As String, src As Worksheet, matchCol As Integer, key As String) As Long

dst.Cells(startRow, 1).Value = title

Dim r As Long: r = startRow + 1

Dim rng As Range: Set rng = src.Range("A1").CurrentRegion

Dim i As Long, header As Boolean

For i = 2 To rng.Rows.Count

If CStr(rng.Cells(i, matchCol).Value) = key

# VBA logigram and algorigram for electrochemical systems, RPA pipelines, and ML‑RPA integration

This Excel VBA engine unifies electrochemical control logic, RPA-style data pipelines, and predictive maintenance scoring into an auditable portfolio. It enforces readiness gates (logigram), executes process/control flows (algorigram), logs evidence, and exports PoE for reviews and CPD.

## Workbook schema

Create these sheets with exact headers (row 1).

* Config
  + Key, Value
  + Seed: CurrentUser, EvidenceDir, WatchFolder, IngestInterval\_s, MaxCell\_V, MaxStack\_V, MaxTemp\_C, MaxCurrent\_A, TargetSOC\_pct, PID\_Kp, PID\_Ki, PID\_Kd, HealthWarnScore, HealthAlarmScore
* Electrochem
  + Timestamp, CellID, StackID, V\_cell, I\_cell, Temp\_C, SOC\_pct, Mode (Charge/Discharge/Idle), ControlAction\_A, Fault
* RPA\_Inbox
  + FileName, ReceivedAt, Parsed (TRUE/FALSE), RowsImported, Notes
* Events
  + Timestamp, User, Topic, EventType, K1, K2, Notes
* Metrics
  + Topic, Metric, Value, Unit, Timestamp
* Evidence
  + EvidenceID, ActivityID, LearnerID, Type (Doc/Data/Log), URI\_or\_Path, Timestamp, Verified (TRUE/FALSE), Verifier, Notes
* Portfolio
  + Generated automatically

Optional:

* Models (parameters per CellID), Dash (KPIs), Scripts (queries/ETL notes).

## Logigram gates

* Safety interlocks:
  + V\_cell ≤ MaxCell\_V; sum(V\_cell) ≤ MaxStack\_V; Temp\_C ≤ MaxTemp\_C; |I\_cell| ≤ MaxCurrent\_A.
* Control readiness:
  + Valid SOC\_pct and Mode; PID params present; no active Fault.
* RPA readiness:
  + WatchFolder exists; ingest schedule active; new files not yet parsed.
* ML readiness:
  + Sufficient history (≥ 100 rows per CellID) for health score; thresholds set.

Failing any gate logs Events with details and blocks actuation/ingest where applicable.

## Algorigram flows

* Control loop (per tick):
  1. Read latest row → safety gate → compute setpoint (by Mode) → PID current command → clamp by limits → write ControlAction\_A → log.
* RPA ingest (scheduled):
  1. Scan WatchFolder → register new CSV → import → parse → append Electrochem → mark parsed → log metrics.
* ML‑RPA (batch/OnTime):
  1. For each CellID → compute health score from residuals/volatility → write Metrics → raise warning/alarm → auto‑create Evidence entries for incidents.

## Core VBA

### Utilities and logging

vba

Option Explicit

Function WS(name As String) As Worksheet: Set WS = ThisWorkbook.Worksheets(name): End Function

Function Cfg(key As String, Optional defVal As Variant = "") As Variant

Dim r As Range: Set r = WS("Config").Columns(1).Find(key, , xlValues, xlWhole)

If r Is Nothing Then Cfg = defVal Else Cfg = IIf(IsEmpty(r.Offset(0, 1)), defVal, r.Offset(0, 1))

End Function

Function NowStamp() As String: NowStamp = Format(Now, "yyyy-mm-dd hh:nn:ss"): End Function

Sub LogEvent(topic As String, evt As String, Optional k1 As String = "", Optional k2 As String = "", Optional note As String = "")

Dim w As Worksheet: Set w = WS("Events")

Dim r As Long: r = w.Cells(w.Rows.Count, 1).End(xlUp).Row + 1

w.Cells(r, 1) = NowStamp(): w.Cells(r, 2) = Cfg("CurrentUser", "User")

w.Cells(r, 3) = topic: w.Cells(r, 4) = evt: w.Cells(r, 5) = k1: w.Cells(r, 6) = k2: w.Cells(r, 7) = note

End Sub

Sub LogMetric(topic As String, metric As String, val As Double, unitStr As String)

Dim w As Worksheet: Set w = WS("Metrics")

Dim r As Long: r = w.Cells(w.Rows.Count, 1).End(xlUp).Row + 1

w.Cells(r, 1) = topic: w.Cells(r, 2) = metric: w.Cells(r, 3) = val

w.Cells(r, 4) = unitStr: w.Cells(r, 5) = NowStamp()

End Sub

Sub LogEvidence(activityID As String, learnerID As String, typ As String, uri As String, Optional verified As Boolean = False, Optional verifier As String = "", Optional notes As String = "")

Dim ev As Worksheet: Set ev = WS("Evidence")

Dim r As Long: r = ev.Cells(ev.Rows.Count, 1).End(xlUp).Row + 1

ev.Cells(r, 1).Value = "E" & Format(Now, "yymmddhhnnss")

ev.Cells(r, 2).Value = activityID

ev.Cells(r, 3).Value = learnerID

ev.Cells(r, 4).Value = typ

ev.Cells(r, 5).Value = uri

ev.Cells(r, 6).Value = NowStamp()

ev.Cells(r, 7).Value = verified

ev.Cells(r, 8).Value = verifier

ev.Cells(r, 9).Value = notes

End Sub

## Electrochemical control logic

Function SafetyOK(V\_cell As Double, V\_stack As Double, I\_cell As Double, T\_C As Double) As Boolean

SafetyOK = (V\_cell <= CDbl(Cfg("MaxCell\_V", 4.2))) \_

And (V\_stack <= CDbl(Cfg("MaxStack\_V", 100))) \_

And (Abs(I\_cell) <= CDbl(Cfg("MaxCurrent\_A", 100))) \_

And (T\_C <= CDbl(Cfg("MaxTemp\_C", 50)))

End Function

Function ModeSetpointA(modeStr As String, soc\_pct As Double) As Double

' Simple policy: drive to TargetSOC with bounded current

Dim target As Double: target = CDbl(Cfg("TargetSOC\_pct", 80))

Dim err As Double: err = target - soc\_pct

Dim Imax As Double: Imax = CDbl(Cfg("MaxCurrent\_A", 100))

Select Case UCase(modeStr)

Case "CHARGE": ModeSetpointA = Application.Max(0, Application.Min(Imax, err / 5)) ' ramp factor

Case "DISCHARGE": ModeSetpointA = -Application.Max(0, Application.Min(Imax, -err / 5))

Case Else: ModeSetpointA = 0

End Select

End Function

### PID controller (incremental) and loop

vba

Private prevErr As Double, integ As Double

Function PID\_Iset(err As Double, dt\_s As Double) As Double

Dim Kp As Double: Kp = CDbl(Cfg("PID\_Kp", 0.8))

Dim Ki As Double: Ki = CDbl(Cfg("PID\_Ki", 0.1))

Dim Kd As Double: Kd = CDbl(Cfg("PID\_Kd", 0.05))

integ = integ + err \* dt\_s

Dim deriv As Double: deriv = (err - prevErr) / dt\_s

PID\_Iset = Kp \* err + Ki \* integ + Kd \* deriv

prevErr = err

End Function

Sub ControlTick()

Dim w As Worksheet: Set w = WS("Electrochem")

Dim r As Long: r = w.Cells(w.Rows.Count, 1).End(xlUp).Row

If r < 2 Then Exit Sub

Dim V As Double: V = w.Cells(r, 4).Value ' V\_cell

Dim I As Double: I = w.Cells(r, 5).Value ' I\_cell

Dim T As Double: T = w.Cells(r, 6).Value ' Temp\_C

Dim SOC As Double: SOC = w.Cells(r, 7).Value

Dim modeStr As String: modeStr = w.Cells(r, 8).Value

Dim Vstack As Double: Vstack = SumStackVoltages(w, w.Cells(r, 3).Value)

If Not SafetyOK(V, Vstack, I, T) Then

w.Cells(r, 10).Value = "FAULT"

LogEvent "CTRL", "InterlockTrip", "Stack=" & Vstack, "CellV=" & V, "I=" & I & " T=" & T

Exit Sub

End If

Dim sp As Double: sp = ModeSetpointA(modeStr, SOC)

Dim err As Double: err = sp - I

Dim cmd As Double: cmd = PID\_Iset(err, 1) ' assume 1s tick

cmd = Application.Max(-CDbl(Cfg("MaxCurrent\_A", 100)), Application.Min(CDbl(Cfg("MaxCurrent\_A", 100)), cmd))

w.Cells(r, 9).Value = cmd ' ControlAction\_A

LogMetric "CTRL", "I\_cmd", cmd, "A"

End Sub

Function SumStackVoltages(w As Worksheet, stackID As Variant) As Double

Dim last As Long: last = w.Cells(w.Rows.Count, 1).End(xlUp).Row

Dim i As Long, sumv As Double

For i = last To 2 Step -1

If w.Cells(i, 3).Value = stackID Then

sumv = sumv + Val(w.Cells(i, 4).Value)

End If

If sumv > CDbl(Cfg("MaxStack\_V", 100)) Then Exit For

Next i

SumStackVoltages = sumv

End Function

## RPA-style data pipeline

### Watch folder ingest and schedule

Sub RPA\_ScheduleStart()

Dim sec As Double: sec = CDbl(Cfg("IngestInterval\_s", 15))

Application.OnTime Now + TimeSerial(0, 0, sec), "RPA\_RunOnce"

LogEvent "RPA", "Scheduled", CStr(sec) & "s", "", ""

End Sub

Sub RPA\_RunOnce()

On Error GoTo EH

Dim folder As String: folder = CStr(Cfg("WatchFolder", ThisWorkbook.Path & "\inbox"))

Dim fso As Object: Set fso = CreateObject("Scripting.FileSystemObject")

If Not fso.FolderExists(folder) Then fso.CreateFolder folder

Dim f As Object

For Each f In fso.GetFolder(folder).Files

If LCase(fso.GetExtensionName(f)) = "csv" Then

If Not AlreadyRegistered(f.Name) Then RegisterInbox f.Path, f.Name

End If

Next f

ProcessInbox

RPA\_ScheduleStart

Exit Sub

EH:

LogEvent "RPA", "Error", Err.Number, Err.Description, ""

RPA\_ScheduleStart

End Sub

Function AlreadyRegistered(fileName As String) As Boolean

Dim w As Worksheet: Set w = WS("RPA\_Inbox")

Dim r As Range: Set r = w.Columns(1).Find(fileName, , xlValues, xlWhole)

AlreadyRegistered = Not r Is Nothing

End Function

Sub RegisterInbox(path As String, fileName As String)

Dim w As Worksheet: Set w = WS("RPA\_Inbox")

Dim r As Long: r = w.Cells(w.Rows.Count, 1).End(xlUp).Row + 1

w.Cells(r, 1).Value = fileName

w.Cells(r, 2).Value = NowStamp()

w.Cells(r, 3).Value = False

w.Cells(r, 5).Value = path

LogEvent "RPA", "Registered", fileName, "", path

End Sub

### Parse and append telemetry

vba

Sub ProcessInbox()

Dim ib As Worksheet: Set ib = WS("RPA\_Inbox")

Dim last As Long: last = ib.Cells(ib.Rows.Count, 1).End(xlUp).Row

Dim i As Long

For i = 2 To last

If Not CBool(ib.Cells(i, 3).Value) Then

Dim fpath As String: fpath = ib.Cells(i, 5).Value

Dim rows As Long: rows = ImportElectrochemCSV(fpath)

ib.Cells(i, 3).Value = True

ib.Cells(i, 4).Value = rows

LogEvent "RPA", "Imported", ib.Cells(i, 1).Value, "Rows=" & rows, ""

End If

Next i

End Sub

Function ImportElectrochemCSV(fpath As String) As Long

Dim ts As Integer: ts = FreeFile

On Error GoTo EH

Open fpath For Input As #ts

Dim line As String, cnt As Long

Dim w As Worksheet: Set w = WS("Electrochem")

Do While Not EOF(ts)

Line Input #ts, line

If InStr(1, line, ",") > 0 Then

Dim a() As String: a = Split(line, ",")

If UBound(a) >= 7 Then

Dim r As Long: r = w.Cells(w.Rows.Count, 1).End(xlUp).Row + 1

w.Cells(r, 1).Value = a(0) ' Timestamp

w.Cells(r, 2).Value = a(1) ' CellID

w.Cells(r, 3).Value = a(2) ' StackID

w.Cells(r, 4).Value = CDbl(a(3)) ' V\_cell

w.Cells(r, 5).Value = CDbl(a(4)) ' I\_cell

w.Cells(r, 6).Value = CDbl(a(5)) ' Temp\_C

w.Cells(r, 7).Value = CDbl(a(6)) ' SOC\_pct

w.Cells(r, 8).Value = a(7) ' Mode

cnt = cnt + 1

End If

End If

Loop

Close #ts

ImportElectrochemCSV = cnt

Exit Function

EH:

On Error Resume Next: Close #ts

LogEvent "RPA", "ImportError", fpath, Err.Number, Err.Description

End Function

## ML‑RPA predictive maintenance (lightweight)

* Health score combines residual volatility, temperature excursions, and overcurrent events.

vba

ub UpdateHealthScores()

Dim ec As Worksheet: Set ec = WS("Electrochem")

Dim last As Long: last = ec.Cells(ec.Rows.Count, 1).End(xlUp).Row

If last < 102 Then Exit Sub

Dim dict As Object: Set dict = CreateObject("Scripting.Dictionary")

Dim i As Long

For i = 2 To last

Dim id As String: id = CStr(ec.Cells(i, 2).Value) ' CellID

If Not dict.exists(id) Then dict.Add id, 0

Next i

Dim k As Variant

For Each k In dict.Keys

Dim score As Double: score = HealthScoreForCell(k, 100)

LogMetric "ML", "HealthScore\_" & k, score, "score"

Dim warn As Double: warn = CDbl(Cfg("HealthWarnScore", 60))

Dim alarm As Double: alarm = CDbl(Cfg("HealthAlarmScore", 40))

If score < alarm Then

LogEvent "ML", "Alarm", k, "", "HealthScore=" & score

LogEvidence "PM\_Alert", k, "Log", "Alarm: " & score, False, "", "Auto-generated"

ElseIf score < warn Then

LogEvent "ML", "Warning", k, "", "HealthScore=" & score

End If

Next k

End Sub

Function HealthScoreForCell(cellID As String, windowN As Long) As Double

Dim ec As Worksheet: Set ec = WS("Electrochem")

Dim last As Long: last = ec.Cells(ec.Rows.Count, 1).End(xlUp).Row

Dim i As Long, n As Long, sumI As Double, sumI2 As Double, exc As Long, overI As Long

Dim Imax As Double: Imax = CDbl(Cfg("MaxCurrent\_A", 100))

For i = last To 2 Step -1

If ec.Cells(i, 2).Value = cellID Then

Dim I As Double: I = ec.Cells(i, 5).Value

Dim T As Double: T = ec.Cells(i, 6).Value

sumI = sumI + I

sumI2 = sumI2 + I \* I

If T > CDbl(Cfg("MaxTemp\_C", 50)) - 5 Then exc = exc + 1

If Abs(I) > 0.9 \* Imax Then overI = overI + 1

n = n + 1

If n >= windowN Then Exit For

End If

Next i

If n = 0 Then HealthScoreForCell = 100: Exit Function

Dim meanI As Double: meanI = sumI / n

Dim varI As Double: varI = Application.Max(0, (sumI2 / n) - meanI ^ 2)

Dim volI As Double: volI = Sqr(varI)

' Normalize components to a 0–100 score (higher is healthier)

Dim sVol As Double: sVol = Application.Max(0, 100 - (volI \* 10))

Dim sExc As Double: sExc = Application.Max(0, 100 - (exc \* 5))

Dim sOver As Double: sOver = Application.Max(0, 100 - (overI \* 10))

HealthScoreForCell = Round((sVol \* 0.5 + sExc \* 0.25 + sOver \* 0.25), 1)

End Function

Schedule with:

vba

ub ML\_Schedule()

Application.OnTime Now + TimeSerial(0, 2, 0), "UpdateHealthScores"

End Sub

Sub ExportPortfolio(learnerID As String)

On Error Resume Next: Application.DisplayAlerts = False

ThisWorkbook.Worksheets("Portfolio").Delete

Application.DisplayAlerts = True: On Error GoTo 0

Dim wr As Worksheet: Set wr = ThisWorkbook.Worksheets.Add

wr.Name = "Portfolio"

Dim r As Long: r = 1

wr.Cells(r, 1) = "Electrochemical Systems, RPA Pipelines, and ML Integration – Portfolio": r = r + 2

wr.Cells(r, 1) = "LearnerID": wr.Cells(r, 2) = learnerID: r = r + 2

r = CopySection(wr, r, "Events", WS("Events"), 3, "RPA")

r = CopySection(wr, r, "Events", WS("Events"), 3, "CTRL")

r = CopySection(wr, r, "Metrics – Control", WS("Metrics"), 1, "CTRL")

r = CopySection(wr, r, "Metrics – ML", WS("Metrics"), 1, "ML")

r = CopySection(wr, r, "Evidence", WS("Evidence"), 3, learnerID)

wr.Columns.AutoFit

Dim f As String: f = CStr(Cfg("EvidenceDir", ThisWorkbook.Path)) & "\PoE\_Echem\_RPA\_ML\_" & learnerID & ".pdf"

On Error Resume Next: wr.ExportAsFixedFormat xlTypePDF, f: On Error GoTo 0

LogEvent "Portfolio", "Exported", learnerID, "", f

MsgBox "Portfolio exported: " & f, vbInformation

End Sub

Function CopySection(dst As Worksheet, startRow As Long, title As String, src As Worksheet, matchCol As Integer, key As String) As Long

dst.Cells(startRow, 1) = title

Dim r As Long: r = startRow + 1

Dim rng As Range: Set rng = src.Range("A1").CurrentRegion

Dim i As Long, header As Boolean

For i = 2 To rng.Rows.Count

If CStr(rng.Cells(i, matchCol).Value) = key Then

If Not header Then rng.Rows(1).Copy dst.Cells(r, 1): r = r + 1: header = True

rng.Rows(i).Copy dst.Cells(r, 1): r = r + 1

End If

Next i

CopySection = r + 1

## Quick start

1. Config:

* CurrentUser = Tshingombe Fiston Tshitadi
* EvidenceDir = C:\Evidence
* WatchFolder = C:\Evidence\inbox
* IngestInterval\_s = 15
* MaxCell\_V = 4.2; MaxStack\_V = 100; MaxTemp\_C = 50; MaxCurrent\_A = 100
* TargetSOC\_pct = 80; PID\_Kp = 0.8; PID\_Ki = 0.1; PID\_Kd = 0.05
* HealthWarnScore = 60; HealthAlarmScore = 40

1. Seed RPA\_Inbox and drop CSV with columns: Timestamp,CellID,StackID,V\_cell,I\_cell,Temp\_C,SOC\_pct,Mode.
2. Run:

* RPA\_ScheduleStart
* ML\_Schedule
* ControlTick (assign to a button or OnTime loop if simulating real-time)
* ExportPortfolio "Learner01"

## Cross‑domain integration hooks

* Neurotech + Adult Ed: add a sheet NeuroMetrics (Attention, CognitiveLoad) and log to Metrics; gate labs by cognitive thresholds to personalize pacing.
* RPA + Electrochem: extend ImportElectrochemCSV to also ingest from OPC‑UA/REST exports (same CSV schema).
* ML‑RPA: replace HealthScore with a regression or classification model output imported from a CSV (prob\_failure), then trigger Evidence and Events accordingly.

If you want, I’ll add a lightweight dashboard (traffic lights for Interlocks, RPA Import Rate, HealthScore trend) and a parameter form to tune PID/RPA intervals without touching sheets.

# VBA Logigram & Algorigram for Electrical Engineering Contractors and Clean Energy Infrastructure

This VBA engine operationalizes Tshingombe’s strategic portfolio in electrical infrastructure, clean energy ecotechnology, and project management. It enforces readiness gates (logigram), orchestrates engineering workflows (algorigram), logs evidence, and exports a CPD-aligned portfolio for institutional, regulatory, or doctoral submission.

## 🧠 System Domains

| **Domain** | **Modules** | **Evidence Types** | **Tools** |
| --- | --- | --- | --- |
| Electrical Infrastructure | Fault, flow, stability, HV systems | Diagrams, specs, test logs | ETAP, MATLAB, AutoCAD |
| Clean Energy Ecotechnology | Solar, wind, biomass, geothermal | System designs, impact assessments | PVsyst, HOMER, RETScreen |
| Smart Grids & IoT | Intelligent distribution, monitoring | IoT dashboards, SCADA logs | Node-RED, MQTT, Grafana |
| Project Management | Planning, risk, stakeholder engagement | Gantt charts, WBS, risk matrices | MS Project, Primavera |
| Policy & Ethics | Regulatory compliance, sustainability | Policy briefs, audit checklists | ISO 50001, IEEE 1547 |

## 📁 Workbook Schema

Create these sheets with exact headers:

* Config
  + Key, Value
  + Seed: CurrentUser, EvidenceDir, MinArtifacts, MaxFaultLevel\_kA, MinEfficiency\_pct, MinRenewableShare\_pct, PassMark\_pct
* Modules
  + ModuleID, Title, Domain, Credits, Prereqs (CSV), Enabled (TRUE/FALSE)
* Activities
  + ActivityID, ModuleID, Type (Design/Simulation/Assessment/Report), Hours, Deliverable, Required (TRUE/FALSE)
* Evidence
  + EvidenceID, ActivityID, LearnerID, Type (Doc/Data/Log), URI\_or\_Path, Timestamp, Verified (TRUE/FALSE), Verifier, Notes
* Assessments
  + AssessmentID, ModuleID, LearnerID, Score, MaxScore, Passed (TRUE/FALSE), Date
* Metrics
  + Topic, Metric, Value, Unit, Timestamp
* Events
  + Timestamp, User, Topic, EventType, K1, K2, Notes
* Portfolio
  + Generated automatically

## 🔐 Logigram Gates

* Module Enabled = TRUE
* Prereqs passed (Assessments)
* Required Activities have verified Evidence
* FaultLevel ≤ MaxFaultLevel\_kA
* Efficiency ≥ MinEfficiency\_pct
* RenewableShare ≥ MinRenewableShare\_pct
* Total artifacts ≥ MinArtifacts

## 🔄 Algorigram Flow

mermaid

graph TD

A[Start Module] --> B[Check Prereqs]

B --> C[Validate Activities]

C --> D[Log Evidence]

D --> E[Run Fault & Efficiency Checks]

E --> F[Renewable Integration Check]

F --> G[Export Portfolio]

## Core VBA Highlights

### Fault Level Check

vba

Function FaultLevelOK(fault\_kA As Double) As Boolean

FaultLevelOK = (fault\_kA <= Cfg("MaxFaultLevel\_kA", 25))

End Function

### Efficiency & Renewable Share Check

vba

Function EfficiencyOK(eff\_pct As Double) As Boolean

EfficiencyOK = (eff\_pct >= Cfg("MinEfficiency\_pct", 85))

End Function

Function RenewableShareOK(share\_pct As Double) As Boolean

RenewableShareOK = (share\_pct >= Cfg("MinRenewableShare\_pct", 30))

End Function

### Evidence Logging

Sub LogEvidence(activityID As String, learnerID As String, typ As String, uri As String, Optional verified As Boolean = False, Optional verifier As String = "", Optional notes As String = "")

Dim ev As Worksheet: Set ev = WS("Evidence")

Dim r As Long: r = ev.Cells(ev.Rows.Count, 1).End(xlUp).Row + 1

ev.Cells(r, 1).Value = "E" & Format(Now, "yymmddhhnnss")

ev.Cells(r, 2).Value = activityID

ev.Cells(r, 3).Value = learnerID

ev.Cells(r, 4).Value = typ

ev.Cells(r, 5).Value = uri

ev.Cells(r, 6).Value = NowStamp()

ev.Cells(r, 7).Value = verified

ev.Cells(r, 8).Value = verifier

ev.Cells(r, 9).Value = notes

End Sub

Portfolio Export

Sub ExportPortfolio(learnerID As String)

On Error Resume Next: Application.DisplayAlerts = False

ThisWorkbook.Worksheets("Portfolio").Delete

Application.DisplayAlerts = True: On Error GoTo 0

Dim wr As Worksheet: Set wr = ThisWorkbook.Worksheets.Add

wr.Name = "Portfolio"

Dim r As Long: r = 1

wr.Cells(r, 1) = "Electrical Engineering & Clean Energy Portfolio": r = r + 2

wr.Cells(r, 1) = "LearnerID": wr.Cells(r, 2) = learnerID: r = r + 2

r = CopySection(wr, r, "Evidence", WS("Evidence"), 3, learnerID)

r = CopySection(wr, r, "Assessments", WS("Assessments"), 3, learnerID)

r = CopySection(wr, r, "Metrics – Fault", WS("Metrics"), 1, "Fault")

r = CopySection(wr, r, "Metrics – Efficiency", WS("Metrics"), 1, "Efficiency")

r = CopySection(wr, r, "Metrics – Renewable", WS("Metrics"), 1, "Renewable")

wr.Columns.AutoFit

Dim f As String: f = Cfg("EvidenceDir", ThisWorkbook.Path) & "\Portfolio\_EE\_CleanEnergy\_" & learnerID & ".pdf"

wr.ExportAsFixedFormat xlTypePDF, f

MsgBox "Portfolio exported: " & f, vbInformation

End Sub

## 🧱 Sample Activities

| **ActivityID** | **ModuleID** | **Type** | **Deliverable** | **Required** |
| --- | --- | --- | --- | --- |
| A\_FaultSim | M\_PowerSys | Simulation | Fault analysis report | TRUE |
| A\_REDesign | M\_CleanEnergy | Design | Solar + wind hybrid model | TRUE |
| A\_PMPlan | M\_PM\_EE | Report | Gantt + risk matrix | TRUE |
| A\_IoTDash | M\_SmartGrid | Model | IoT dashboard | TRUE |
| A\_Ethics | M\_EnergyPolicy | Assessment | Policy brief | TRUE |

## 🧠 Strategic Integration Hooks

* Smart Grid + Doctoral Research: Add Metrics sheet for resilience, latency, and load balancing KPIs.
* Ecotechnology + Civil Engineering: Add ImpactAssessment sheet with ecological scores and mitigation logs.
* Project Management + Renewable Energy: Add Timeline sheet with milestone tracking and budget burn-down.
* IoT + Infrastructure: Add IoT\_Logs sheet with sensor data and control actions.

## 🚀 Quick Start

1. Seed Config:
   * CurrentUser = Tshingombe Fiston Tshitadi
   * EvidenceDir = C:\Evidence
   * MaxFaultLevel\_kA = 25
   * MinEfficiency\_pct = 85
   * MinRenewableShare\_pct = 30
   * MinArtifacts = 6
   * PassMark\_pct = 60
2. Log evidence:
   * LogEvidence "A\_REDesign", "Learner01", "Doc", "SolarWindDesign.pdf", True, "Assessor"
3. Export:
   * ExportPortfolio "Learner01"

Would you like me to help format this into a CPD-aligned dashboard, a doctoral research outline, or a stakeholder-ready presentation template? I can also scaffold a GitHub repository structure to house your modules, simulations, and documentation. Just say the word.

# VBA logigram and algorigram for specialist electrochemical engineering, energy storage, and RPA

This engine operationalizes Modules 38.x (Electrochemical Engineering), 40.x (Energy Storage), and 41.x (RPA in Electrical Systems) into an audit-ready portfolio workflow. It enforces readiness gates (logigram), executes analytics and control flows (algorigram), logs evidence, and exports a CPD-aligned PoE.

## Workbook schema

Create these sheets with exact headers.

* Config
  + Key, Value
  + Seed: CurrentUser, EvidenceDir, PassMark\_pct, MinArtifacts, HealthWarnScore, HealthAlarmScore, MaxCell\_V, MaxStack\_V, MaxTemp\_C, MaxCurrent\_A, TargetSOC\_pct, PID\_Kp, PID\_Ki, PID\_Kd
* Modules
  + ModuleID, Title, Domain (38/40/41), Credits, Prereqs (CSV), Enabled (TRUE/FALSE)
* Activities
  + ActivityID, ModuleID, Type (Lab/Model/Report/Assessment), Hours, Deliverable, Required (TRUE/FALSE)
* Evidence
  + EvidenceID, ActivityID, LearnerID, Type (Doc/Data/Log/Code), URI\_or\_Path, Timestamp, Verified (TRUE/FALSE), Verifier, Notes
* Assessments
  + AssessmentID, ModuleID, LearnerID, Score, MaxScore, Passed (TRUE/FALSE), Date
* Metrics
  + Topic, Metric, Value, Unit, Timestamp
* Events
  + Timestamp, User, Topic, EventType, K1, K2, Notes
* Telemetry
  + Timestamp, System (Battery/FuelCell/Electrolysis/Sensor), AssetID, StackID, V\_cell, I\_cell, Temp\_C, SOC\_pct, Mode, OCV\_V
* RPA\_Inbox
  + FileName, ReceivedAt, Parsed (TRUE/FALSE), RowsImported, Path
* Portfolio
  + Generated automatically

## Logigram gates

* Module gate: Enabled = TRUE; Prereqs passed (Assessments ≥ PassMark\_pct).
* Evidence gate: All Required activities for ModuleID have ≥1 Verified Evidence for learner.
* Safety gate (38.x ops): V\_cell ≤ MaxCell\_V; ΣV\_cell (stack) ≤ MaxStack\_V; Temp\_C ≤ MaxTemp\_C; |I\_cell| ≤ MaxCurrent\_A.
* Data gate: For each analytic, minimum rows present (e.g., ≥100 recent samples per AssetID).
* Health gate: HealthScore ≥ HealthAlarmScore (else flag incident and halt actuation).

## Algorigram flows

* Analytics flow: Ingest telemetry → safety check → compute KPIs (SOH, R\_int, η\_FC, corrosion mpy, sensor drift, electrolysis kWh/kg) → log Metrics → raise Events if thresholds breached → write Evidence.
* RPA flow: Watch folder → register CSV → import to Telemetry → mark parsed → schedule next.
* Control flow (simulation): Mode policy → PID on current command → clamp by limits → log command.

## Core utilities

vba

Option Explicit

Function WS(name As String) As Worksheet: Set WS = ThisWorkbook.Worksheets(name): End Function

Function Cfg(key As String, Optional defVal As Variant = "") As Variant

Dim r As Range: Set r = WS("Config").Columns(1).Find(key, , xlValues, xlWhole)

If r Is Nothing Then Cfg = defVal Else Cfg = IIf(IsEmpty(r.Offset(0, 1)), defVal, r.Offset(0, 1))

End Function

Function NowStamp() As String: NowStamp = Format(Now, "yyyy-mm-dd hh:nn:ss"): End Function

Sub LogEvent(topic As String, evt As String, Optional k1 As String = "", Optional k2 As String = "", Optional notes As String = "")

Dim w As Worksheet: Set w = WS("Events")

Dim r As Long: r = w.Cells(w.Rows.Count, 1).End(xlUp).Row + 1

w.Cells(r, 1) = NowStamp(): w.Cells(r, 2) = Cfg("CurrentUser", "User")

w.Cells(r, 3) = topic: w.Cells(r, 4) = evt: w.Cells(r, 5) = k1: w.Cells(r, 6) = k2: w.Cells(r, 7) = notes

End Sub

Sub LogMetric(topic As String, metric As String, val As Double, unitStr As String)

Dim w As Worksheet: Set w = WS("Metrics")

Dim r As Long: r = w.Cells(w.Rows.Count, 1).End(xlUp).Row + 1

w.Cells(r, 1) = topic: w.Cells(r, 2) = metric: w.Cells(r, 3) = val

w.Cells(r, 4) = unitStr: w.Cells(r, 5) = NowStamp()

End Sub

Sub LogEvidence(activityID As String, learnerID As String, typ As String, uri As String, Optional verified As Boolean = False, Optional verifier As String = "", Optional notes As String = "")

Dim ev As Worksheet: Set ev = WS("Evidence")

Dim r As Long: r = ev.Cells(ev.Rows.Count, 1).End(xlUp).Row + 1

ev.Cells(r, 1) = "E" & Format(Now, "yymmddhhnnss")

ev.Cells(r, 2) = activityID: ev.Cells(r, 3) = learnerID: ev.Cells(r, 4) = typ

ev.Cells(r, 5) = uri: ev.Cells(r, 6) = NowStamp(): ev.Cells(r, 7) = verified

ev.Cells(r, 8) = verifier: ev.Cells(r, 9) = notes

End Sub

Function PassedModule(learnerID As String, moduleID As String) As Boolean

Dim a As Worksheet: Set a = WS("Assessments")

Dim i As Long, passPct As Double: passPct = CDbl(Cfg("PassMark\_pct", 60))

For i = 2 To a.Cells(a.Rows.Count, 1).End(xlUp).Row

If a.Cells(i, 2).Value = moduleID And a.Cells(i, 3).Value = learnerID Then

If a.Cells(i, 5).Value > 0 Then

If (a.Cells(i, 4).Value / a.Cells(i, 5).Value) \* 100# >= passPct Then PassedModule = True: Exit Function

End If

End If

Next i

End Function

Function PrereqsMet(learnerID As String, moduleID As String) As Boolean

Dim m As Worksheet: Set m = WS("Modules")

Dim r As Range: Set r = m.Columns(1).Find(moduleID, , xlValues, xlWhole)

If r Is Nothing Then Exit Function

Dim csv As String: csv = CStr(r.Offset(0, 4).Value)

If Len(Trim(csv)) = 0 Then PrereqsMet = True: Exit Function

Dim arr() As String: arr = Split(csv, ","): Dim i As Long

For i = LBound(arr) To UBound(arr)

If Not PassedModule(learnerID, Trim(arr(i))) Then PrereqsMet = False: Exit Function

Next i

PrereqsMet = True

End Function

Function RequiredActivitiesHaveEvidence(moduleID As String, learnerID As String) As Boolean

Dim act As Worksheet: Set act = WS("Activities")

Dim ev As Worksheet: Set ev = WS("Evidence")

Dim need As Long, have As Long, i As Long, j As Long

For i = 2 To act.Cells(act.Rows.Count, 1).End(xlUp).Row

If act.Cells(i, 2).Value = moduleID And CBool(act.Cells(i, 6).Value) Then

need = need + 1

For j = 2 To ev.Cells(ev.Rows.Count, 1).End(xlUp).Row

If ev.Cells(j, 2).Value = act.Cells(i, 1).Value And ev.Cells(j, 3).Value = learnerID And CBool(ev.Cells(j, 7).Value) Then

have = have + 1: Exit For

End If

Next j

End If

Next i

RequiredActivitiesHaveEvidence = (need = have)

End Function

Function GateModule(learnerID As String, moduleID As String) As Boolean

Dim m As Worksheet: Set m = WS("Modules")

Dim r As Range: Set r = m.Columns(1).Find(moduleID, , xlValues, xlWhole)

If r Is Nothing Then LogEvent "Gate", "Error", learnerID, moduleID, "Module not found": Exit Function

If Not CBool(r.Offset(0, 6).Value) Then LogEvent "Gate", "Denied", learnerID, moduleID, "Module disabled": Exit Function

If Not PrereqsMet(learnerID, moduleID) Then LogEvent "Gate", "Denied", learnerID, moduleID, "Prereqs unmet": Exit Function

If Not RequiredActivitiesHaveEvidence(moduleID, learnerID) Then LogEvent "Gate", "Denied", learnerID, moduleID, "Required evidence missing": Exit Function

LogEvent "Gate", "Granted", learnerID, moduleID, ""

GateModule = True

End Function

## Safety interlocks and control logic (38.x battery/fuel cell/electrolysis)

vba

Function SafetyOK(V\_cell As Double, V\_stack As Double, I\_cell As Double, T\_C As Double) As Boolean

SafetyOK = (V\_cell <= CDbl(Cfg("MaxCell\_V", 4.2))) \_

And (V\_stack <= CDbl(Cfg("MaxStack\_V", 100))) \_

And (Abs(I\_cell) <= CDbl(Cfg("MaxCurrent\_A", 100))) \_

And (T\_C <= CDbl(Cfg("MaxTemp\_C", 50)))

End Function

Function StackVoltage(sys As String, stackID As Variant) As Double

Dim t As Worksheet: Set t = WS("Telemetry")

Dim last As Long: last = t.Cells(t.Rows.Count, 1).End(xlUp).Row

Dim i As Long, sumv As Double

For i = last To 2 Step -1

If t.Cells(i, 2).Value = sys And t.Cells(i, 4).Value = stackID Then

sumv = sumv + Val(t.Cells(i, 5).Value)

End If

Next i

StackVoltage = sumv

End Function

' PID on current setpoint (simulation)

Private prevErr As Double, integ As Double

Function PID\_Iset(err As Double, dt\_s As Double) As Double

Dim Kp As Double: Kp = CDbl(Cfg("PID\_Kp", 0.8))

Dim Ki As Double: Ki = CDbl(Cfg("PID\_Ki", 0.1))

Dim Kd As Double: Kd = CDbl(Cfg("PID\_Kd", 0.05))

integ = integ + err \* dt\_s

Dim deriv As Double: deriv = (err - prevErr) / dt\_s

PID\_Iset = Kp \* err + Ki \* integ + Kd \* deriv

prevErr = err

End Function

Function ModeSetpointA(modeStr As String, soc\_pct As Double) As Double

Dim target As Double: target = CDbl(Cfg("TargetSOC\_pct", 80))

Dim Imax As Double: Imax = CDbl(Cfg("MaxCurrent\_A", 100))

Dim err As Double: err = target - soc\_pct

Select Case UCase(modeStr)

Case "CHARGE": ModeSetpointA = WorksheetFunction.Min(Imax, WorksheetFunction.Max(0, err / 5))

Case "DISCHARGE": ModeSetpointA = -WorksheetFunction.Min(Imax, WorksheetFunction.Max(0, -err / 5))

Case Else: ModeSetpointA = 0

End Select

End Function

Analytics: KPIs for modules 38.x and 40.x ption Explicit

Function WS(name As String) As Worksheet: Set WS = ThisWorkbook.Worksheets(name): End Function

Function Cfg(key As String, Optional defVal As Variant = "") As Variant

Dim r As Range: Set r = WS("Config").Columns(1).Find(key, , xlValues, xlWhole)

If r Is Nothing Then Cfg = defVal Else Cfg = IIf(IsEmpty(r.Offset(0, 1)), defVal, r.Offset(0, 1))

End Function

Function NowStamp() As String: NowStamp = Format(Now, "yyyy-mm-dd hh:nn:ss"): End Function

Sub LogEvent(topic As String, evt As String, Optional k1 As String = "", Optional k2 As String = "", Optional notes As String = "")

Dim w As Worksheet: Set w = WS("Events")

Dim r As Long: r = w.Cells(w.Rows.Count, 1).End(xlUp).Row + 1

w.Cells(r, 1) = NowStamp(): w.Cells(r, 2) = Cfg("CurrentUser", "User")

w.Cells(r, 3) = topic: w.Cells(r, 4) = evt: w.Cells(r, 5) = k1: w.Cells(r, 6) = k2: w.Cells(r, 7) = notes

End Sub

Sub LogMetric(topic As String, metric As String, val As Double, unitStr As String)

Dim w As Worksheet: Set w = WS("Metrics")

Dim r As Long: r = w.Cells(w.Rows.Count, 1).End(xlUp).Row + 1

w.Cells(r, 1) = topic: w.Cells(r, 2) = metric: w.Cells(r, 3) = val

w.Cells(r, 4) = unitStr: w.Cells(r, 5) = NowStamp()

End Sub

Sub LogEvidence(activityID As String, learnerID As String, typ As String, uri As String, Optional verified As Boolean = False, Optional verifier As String = "", Optional notes As String = "")

Dim ev As Worksheet: Set ev = WS("Evidence")

Dim r As Long: r = ev.Cells(ev.Rows.Count, 1).End(xlUp).Row + 1

ev.Cells(r, 1) = "E" & Format(Now, "yymmddhhnnss")

ev.Cells(r, 2) = activityID: ev.Cells(r, 3) = learnerID: ev.Cells(r, 4) = typ

ev.Cells(r, 5) = uri: ev.Cells(r, 6) = NowStamp(): ev.Cells(r, 7) = verified

ev.Cells(r, 8) = verifier: ev.Cells(r, 9) = notes

End Sub

## Gates: assessments, prerequisites, eviden

Function PassedModule(learnerID As String, moduleID As String) As Boolean

Dim a As Worksheet: Set a = WS("Assessments")

Dim i As Long, passPct As Double: passPct = CDbl(Cfg("PassMark\_pct", 60))

For i = 2 To a.Cells(a.Rows.Count, 1).End(xlUp).Row

If a.Cells(i, 2).Value = moduleID And a.Cells(i, 3).Value = learnerID Then

If a.Cells(i, 5).Value > 0 Then

If (a.Cells(i, 4).Value / a.Cells(i, 5).Value) \* 100# >= passPct Then PassedModule = True: Exit Function

End If

End If

Next i

End Function

Function PrereqsMet(learnerID As String, moduleID As String) As Boolean

Dim m As Worksheet: Set m = WS("Modules")

Dim r As Range: Set r = m.Columns(1).Find(moduleID, , xlValues, xlWhole)

If r Is Nothing Then Exit Function

Dim csv As String: csv = CStr(r.Offset(0, 4).Value)

If Len(Trim(csv)) = 0 Then PrereqsMet = True: Exit Function

Dim arr() As String: arr = Split(csv, ","): Dim i As Long

For i = LBound(arr) To UBound(arr)

If Not PassedModule(learnerID, Trim(arr(i))) Then PrereqsMet = False: Exit Function

Next i

PrereqsMet = True

End Function

Function RequiredActivitiesHaveEvidence(moduleID As String, learnerID As String) As Boolean

Dim act As Worksheet: Set act = WS("Activities")

Dim ev As Worksheet: Set ev = WS("Evidence")

Dim need As Long, have As Long, i As Long, j As Long

For i = 2 To act.Cells(act.Rows.Count, 1).End(xlUp).Row

If act.Cells(i, 2).Value = moduleID And CBool(act.Cells(i, 6).Value) Then

need = need + 1

For j = 2 To ev.Cells(ev.Rows.Count, 1).End(xlUp).Row

If ev.Cells(j, 2).Value = act.Cells(i, 1).Value And ev.Cells(j, 3).Value = learnerID And CBool(ev.Cells(j, 7).Value) Then

have = have + 1: Exit For

End If

Next j

End If

Next i

RequiredActivitiesHaveEvidence = (need = have)

End Function

Function GateModule(learnerID As String, moduleID As String) As Boolean

Dim m As Worksheet: Set m = WS("Modules")

Dim r As Range: Set r = m.Columns(1).Find(moduleID, , xlValues, xlWhole)

If r Is Nothing Then LogEvent "Gate", "Error", learnerID, moduleID, "Module not found": Exit Function

If Not CBool(r.Offset(0, 6).Value) Then LogEvent "Gate", "Denied", learnerID, moduleID, "Module disabled": Exit Function

If Not PrereqsMet(learnerID, moduleID) Then LogEvent "Gate", "Denied", learnerID, moduleID, "Prereqs unmet": Exit Function

If Not RequiredActivitiesHaveEvidence(moduleID, learnerID) Then LogEvent "Gate", "Denied", learnerID, moduleID, "Required evidence missing": Exit Function

LogEvent "Gate", "Granted", learnerID, moduleID, ""

GateModule = True

End Function

Safety interlocks and control logic (38.x battery/fuel cell/electrolysis) Function SafetyOK(V\_cell As Double, V\_stack As Double, I\_cell As Double, T\_C As Double) As Boolean

SafetyOK = (V\_cell <= CDbl(Cfg("MaxCell\_V", 4.2))) \_

And (V\_stack <= CDbl(Cfg("MaxStack\_V", 100))) \_

And (Abs(I\_cell) <= CDbl(Cfg("MaxCurrent\_A", 100))) \_

And (T\_C <= CDbl(Cfg("MaxTemp\_C", 50)))

End Function

Function StackVoltage(sys As String, stackID As Variant) As Double

Dim t As Worksheet: Set t = WS("Telemetry")

Dim last As Long: last = t.Cells(t.Rows.Count, 1).End(xlUp).Row

Dim i As Long, sumv As Double

For i = last To 2 Step -1

If t.Cells(i, 2).Value = sys And t.Cells(i, 4).Value = stackID Then

sumv = sumv + Val(t.Cells(i, 5).Value)

End If

Next i

StackVoltage = sumv

End Function

' PID on current setpoint (simulation)

Private prevErr As Double, integ As Double

Function PID\_Iset(err As Double, dt\_s As Double) As Double

Dim Kp As Double: Kp = CDbl(Cfg("PID\_Kp", 0.8))

Dim Ki As Double: Ki = CDbl(Cfg("PID\_Ki", 0.1))

Dim Kd As Double: Kd = CDbl(Cfg("PID\_Kd", 0.05))

integ = integ + err \* dt\_s

Dim deriv As Double: deriv = (err - prevErr) / dt\_s

PID\_Iset = Kp \* err + Ki \* integ + Kd \* deriv

prevErr = err

End Function

Function ModeSetpointA(modeStr As String, soc\_pct As Double) As Double

Dim target As Double: target = CDbl(Cfg("TargetSOC\_pct", 80))

Dim Imax As Double: Imax = CDbl(Cfg("MaxCurrent\_A", 100))

Dim err As Double: err = target - soc\_pct

Select Case UCase(modeStr)

Case "CHARGE": ModeSetpointA = WorksheetFunction.Min(Imax, WorksheetFunction.Max(0, err / 5))

Case "DISCHARGE": ModeSetpointA = -WorksheetFunction.Min(Imax, WorksheetFunction.Max(0, -err / 5))

Case Else: ModeSetpointA = 0

End Select

End Function

## Analytics: KPIs for modules 38.x and 40.x

vba

' 38.4/40.6: Internal resistance estimate (Ohm) via dV/dI around small step

Function R\_internal\_ohm(dV As Double, dI As Double) As Double

If dI = 0 Then R\_internal\_ohm = 0 Else R\_internal\_ohm = dV / dI

End Function

' 38.4/40.6: Capacity fade (% of nominal)

Function CapacityFade\_pct(cap\_meas\_Ah As Double, cap\_nom\_Ah As Double) As Double

If cap\_nom\_Ah = 0 Then CapacityFade\_pct = 0 Else CapacityFade\_pct = 100# \* (1 - cap\_meas\_Ah / cap\_nom\_Ah)

End Function

' 38.5: Fuel cell efficiency (%) ~ P\_out / (ṁ\_H2 \* LHV)

Function FuelCellEff\_pct(P\_out\_W As Double, mH2\_kg\_s As Double, LHV\_kJ\_kg As Double) As Double

If mH2\_kg\_s <= 0 Then FuelCellEff\_pct = 0 Else FuelCellEff\_pct = 100# \* (P\_out\_W / (mH2\_kg\_s \* LHV\_kJ\_kg \* 1000#))

End Function

' 38.6: Corrosion rate (mpy) using weight loss method (K=534 for mpy, W=mg, D=g/cm3, A=in2, T=hours)

Function Corrosion\_mpy(W\_mg As Double, D\_g\_cm3 As Double, A\_in2 As Double, T\_h As Double) As Double

If D\_g\_cm3 \* A\_in2 \* T\_h = 0 Then Corrosion\_mpy = 0 Else Corrosion\_mpy = 534# \* W\_mg / (D\_g\_cm3 \* A\_in2 \* T\_h)

End Function

' 38.7: Sensor drift (%) over window

Function SensorDrift\_pct(val\_now As Double, val\_ref As Double) As Double

If val\_ref = 0 Then SensorDrift\_pct = 0 Else SensorDrift\_pct = 100# \* (val\_now - val\_ref) / val\_ref

End Function

' 38.8: Electrolysis specific energy (kWh/kg H2)

Function Electrolysis\_kWh\_per\_kg(E\_kWh As Double, mH2\_kg As Double) As Double

If mH2\_kg = 0 Then Electrolysis\_kWh\_per\_kg = 0 Else Electrolysis\_kWh\_per\_kg = E\_kWh / mH2\_kg

End Function

' 38.3/40.3: OCV-SOC fit error (RMSE)

Function RMSE(pred As Double, act As Double, n As Long, ssq As Double) As Double

' Accumulate outside: ssq += (pred-act)^2, n++

If n = 0 Then RMSE = 0 Else RMSE = Sqr(ssq / n)

End Function

## Health scoring (ML-lite) for predictive maintenance

vba

Sub UpdateHealthScores()

Dim t As Worksheet: Set t = WS("Telemetry")

Dim last As Long: last = t.Cells(t.Rows.Count, 1).End(xlUp).Row

If last < 102 Then Exit Sub

Dim ids As Object: Set ids = CreateObject("Scripting.Dictionary")

Dim i As Long

For i = 2 To last: If Not ids.exists(CStr(t.Cells(i, 3).Value)) Then ids.Add CStr(t.Cells(i, 3).Value), 1: Next i

Dim k As Variant

For Each k In ids.Keys

Dim score As Double: score = HealthScoreForAsset(CStr(k), 100)

LogMetric "Health", "Score\_" & k, score, "score"

Dim warn As Double: warn = CDbl(Cfg("HealthWarnScore", 60))

Dim alarm As Double: alarm = CDbl(Cfg("HealthAlarmScore", 40))

If score < alarm Then

LogEvent "Health", "Alarm", CStr(k), "", "Score=" & score

LogEvidence "PM\_Alert", CStr(k), "Log", "Alarm " & score, False, "", "Auto"

ElseIf score < warn Then

LogEvent "Health", "Warning", CStr(k), "", "Score=" & score

End If

Next k

End Sub

Function HealthScoreForAsset(assetID As String, windowN As Long) As Double

Dim t As Worksheet: Set t = WS("Telemetry")

Dim last As Long: last = t.Cells(t.Rows.Count, 1).End(xlUp).Row

Dim i As Long, n As Long, sumI As Double, sumI2 As Double, hot As Long, overI As Long

Dim Imax As Double: Imax = CDbl(Cfg("MaxCurrent\_A", 100))

For i = last To 2 Step -1

If t.Cells(i, 3).Value = assetID Then

Dim I As Double: I = t.Cells(i, 6).Value

Dim Tc As Double: Tc = t.Cells(i, 7).Value

sumI = sumI + I: sumI2 = sumI2 + I \* I

If Tc > CDbl(Cfg("MaxTemp\_C", 50)) - 5 Then hot = hot + 1

If Abs(I) > 0.9 \* Imax Then overI = overI + 1

n = n + 1: If n >= windowN Then Exit For

End If

Next i

If n = 0 Then HealthScoreForAsset = 100: Exit Function

Dim meanI As Double: meanI = sumI / n

Dim varI As Double: varI = Application.Max(0, (sumI2 / n) - meanI ^ 2)

Dim volI As Double: volI = Sqr(varI)

Dim sVol As Double: sVol = Application.Max(0, 100 - 10 \* volI)

Dim sHot As Double: sHot = Application.Max(0, 100 - 5 \* hot)

Dim sOver As Double: sOver = Application.Max(0, 100 - 10 \* overI)

HealthScoreForAsset = Round(0.5 \* sVol + 0.25 \* sHot + 0.25 \* sOver, 1)

End Function

## RPA ingest (41.x) and telemetry import

vba

Sub RPA\_ScanAndRegister()

Dim folder As String: folder = CStr(Cfg("EvidenceDir", ThisWorkbook.Path)) & "\inbox"

Dim fso As Object: Set fso = CreateObject("Scripting.FileSystemObject")

If Not fso.FolderExists(folder) Then fso.CreateFolder folder

Dim w As Worksheet: Set w = WS("RPA\_Inbox")

Dim f As Object

For Each f In fso.GetFolder(folder).Files

If LCase(fso.GetExtensionName(f)) = "csv" Then

If w.Columns(1).Find(f.Name, , xlValues, xlWhole) Is Nothing Then

Dim r As Long: r = w.Cells(w.Rows.Count, 1).End(xlUp).Row + 1

w.Cells(r, 1) = f.Name: w.Cells(r, 2) = NowStamp(): w.Cells(r, 3) = False: w.Cells(r, 5) = f.Path

LogEvent "RPA", "Registered", f.Name, "", f.Path

End If

End If

Next f

End Sub

Sub RPA\_ProcessInbox()

Dim ib As Worksheet: Set ib = WS("RPA\_Inbox")

Dim last As Long: last = ib.Cells(ib.Rows.Count, 1).End(xlUp).Row

Dim i As Long

For i = 2 To last

If Not CBool(ib.Cells(i, 3).Value) Then

Dim rows As Long: rows = ImportTelemetryCSV(CStr(ib.Cells(i, 5).Value))

ib.Cells(i, 3) = True: ib.Cells(i, 4) = rows

LogEvent "RPA", "Imported", ib.Cells(i, 1), "Rows=" & rows, ""

End If

Next i

End Sub

Function ImportTelemetryCSV(fpath As String) As Long

Dim ts As Integer: ts = FreeFile

On Error GoTo EH

Open fpath For Input As #ts

Dim line As String, cnt As Long

Dim t As Worksheet: Set t = WS("Telemetry")

Do While Not EOF(ts)

Line Input #ts, line

If InStr(line, ",") > 0 Then

Dim a() As String: a = Split(line, ",")

If UBound(a) >= 9 Then

Dim r As Long: r = t.Cells(t.Rows.Count, 1).End(xlUp).Row + 1

t.Cells(r, 1) = a(0): t.Cells(r, 2) = a(1): t.Cells(r, 3) = a(2): t.Cells(r, 4) = a(3)

t.Cells(r, 5) = CDbl(a(4)): t.Cells(r, 6) = CDbl(a(5)): t.Cells(r, 7) = CDbl(a(6))

t.Cells(r, 8) = CDbl(a(7)): t.Cells(r, 9) = a(8): t.Cells(r, 10) = CDbl(a(9))

cnt = cnt + 1

End If

End If

Loop

Close #ts

ImportTelemetryCSV = cnt

Exit Function

EH:

On Error Resume Next: Close #ts

LogEvent "RPA", "ImportError", fpath, Err.Number, Err.Description

End Function

## Module runners: compute KPIs by theme

vba

Sub Run\_38\_Battery\_KPIs(assetID As String, cap\_meas\_Ah As Double, cap\_nom\_Ah As Double, dV As Double, dI As Double)

Dim Rint As Double: Rint = R\_internal\_ohm(dV, dI)

Dim fade As Double: fade = CapacityFade\_pct(cap\_meas\_Ah, cap\_nom\_Ah)

LogMetric "38.4", "Rint\_ohm\_" & assetID, Rint, "Ohm"

LogMetric "38.4", "CapacityFade\_pct\_" & assetID, fade, "%"

End Sub

Sub Run\_38\_FuelCell\_KPIs(stackID As String, P\_out\_W As Double, mH2\_kg\_s As Double, Optional LHV\_kJ\_kg As Double = 120000)

Dim eff As Double: eff = FuelCellEff\_pct(P\_out\_W, mH2\_kg\_s, LHV\_kJ\_kg)

LogMetric "38.5", "Efficiency\_pct\_" & stackID, eff, "%"

End Sub

Sub Run\_38\_Corrosion\_KPIs(sampleID As String, W\_mg As Double, D As Double, A\_in2 As Double, T\_h As Double)

Dim mpy As Double: mpy = Corrosion\_mpy(W\_mg, D, A\_in2, T\_h)

LogMetric "38.6", "Corrosion\_mpy\_" & sampleID, mpy, "mpy"

End Sub

Sub Run\_38\_Sensor\_KPIs(sensorID As String, val\_now As Double, val\_ref As Double)

Dim drift As Double: drift = SensorDrift\_pct(val\_now, val\_ref)

LogMetric "38.7", "SensorDrift\_pct\_" & sensorID, drift, "%"

End Sub

Sub Run\_38\_Electrolysis\_KPIs(systemID As String, E\_kWh As Double, mH2\_kg As Double)

Dim sp As Double: sp = Electrolysis\_kWh\_per\_kg(E\_kWh, mH2\_kg)

LogMetric "38.8", "SpecEnergy\_kWhkg\_" & systemID, sp, "kWh/kg H2"

End Sub

## Portfolio export

vba

Sub ExportPortfolio(learnerID As String)

On Error Resume Next: Application.DisplayAlerts = False

ThisWorkbook.Worksheets("Portfolio").Delete

Application.DisplayAlerts = True: On Error GoTo 0

Dim wr As Worksheet: Set wr = ThisWorkbook.Worksheets.Add

wr.Name = "Portfolio"

Dim r As Long: r = 1

wr.Cells(r, 1) = "Electrochemical Engineering • Energy Storage • RPA – Portfolio of Evidence": r = r + 2

wr.Cells(r, 1) = "LearnerID": wr.Cells(r, 2) = learnerID: r = r + 2

r = CopySection(wr, r, "Evidence", WS("Evidence"), 3, learnerID)

r = CopySection(wr, r, "Assessments", WS("Assessments"), 3, learnerID)

r = CopySection(wr, r, "Metrics – 38.x", WS("Metrics"), 1, "38.")

r = CopySection(wr, r, "Metrics – 40.x", WS("Metrics"), 1, "40.")

r = CopySection(wr, r, "Metrics – 41.x / RPA", WS("Metrics"), 1, "RPA")

wr.Columns.AutoFit

Dim f As String: f = CStr(Cfg("EvidenceDir", ThisWorkbook.Path)) & "\PoE\_38\_40\_41\_" & learnerID & ".pdf"

wr.ExportAsFixedFormat xlTypePDF, f

LogEvent "Portfolio", "Exported", learnerID, "", f

MsgBox "Portfolio exported: " & f, vbInformation

End Sub

Function CopySection(dst As Worksheet, startRow As Long, title As String, src As Worksheet, matchCol As Integer, key As String) As Long

dst.Cells(startRow, 1) = title

Dim r As Long: r = startRow + 1

Dim rng As Range: Set rng = src.Range("A1").CurrentRegion

Dim i As Long, header As Boolean

For i = 2 To rng.Rows.Count

If InStr(1, CStr(rng.Cells(i, matchCol).Value), key, vbTextCompare) > 0 Then

If Not header Then rng.Rows(1).Copy dst.Cells(r, 1): r = r + 1: header = True

rng.Rows(i).Copy dst.Cells(r, 1): r = r + 1

End If

Next i

CopySection = r + 1

End Function

## Sample seed and usage

### Config

* CurrentUser = Tshingombe Fiston Tshitadi
* EvidenceDir = C:\Evidence
* PassMark\_pct = 60
* MinArtifacts = 6
* HealthWarnScore = 60
* HealthAlarmScore = 40
* MaxCell\_V = 4.2, MaxStack\_V = 100, MaxTemp\_C = 50, MaxCurrent\_A = 100
* TargetSOC\_pct = 80; PID\_Kp = 0.8; PID\_Ki = 0.1; PID\_Kd = 0.05

### Modules

* 38.3–38.10 Electrochemical themes; 40.2–40.10 Energy storage; 41.3–41.10 RPA suite; Enabled TRUE; map Prereqs as needed.

### Activities

* A\_38\_4\_CAP | 38.4 | Lab | Capacity test report | TRUE
* A\_38\_5\_FC | 38.5 | Model | Fuel cell η calc | TRUE
* A\_38\_6\_COR | 38.6 | Lab | Corrosion test | TRUE
* A\_38\_8\_ELX | 38.8 | Lab | Electrolysis kWh/kg | TRUE
* A\_40\_6\_PERF| 40.6 | Assessment | Performance benchmarking | TRUE
* A\_41\_7\_RPA | 41.7 | Report | Control simulation RPA | TRUE

### Run

* RPA\_ScanAndRegister: register inbox files
* RPA\_ProcessInbox: import telemetry
* Run\_38\_Battery\_KPIs "Pack01", 85, 100, 0.12, 5
* Run\_38\_FuelCell\_KPIs "StackA", 5000, 0.00003
* Run\_38\_Corrosion\_KPIs "Samp1", 120, 7.85, 1.2, 168
* Run\_38\_Sensor\_KPIs "NO2-01", 210, 200
* Run\_38\_Electrolysis\_KPIs "ELX-100", 420, 9.0
* UpdateHealthScores
* ExportPortfolio "Learner01"

## Optional extensions

* Add SOH estimator via Coulomb counting and OCV-SOC lookup table.
* Add constraint set for safety envelope (R^2-based anomaly detection).
* Add charts to Portfolio (KPIs and health trends).
* Add UserForm to run flows and attach Evidence in one click.

If you want, I’ll wire a minimal dashboard with traffic lights for Safety, Health, Evidence coverage, and Module gates—ready for demo and submission.

# VBA logigram and algorigram for advanced electrical systems portfolio and simulations

This engine turns your suggested artifacts and macro-based simulations into an auditable, CPD-aligned portfolio. It enforces readiness gates (logigram), runs simulations and calculations (algorigram), logs evidence/metrics, and exports a polished PoE.

## Workbook schema

Create these sheets with exactly these headers (row 1).

* Config
  + Key, Value
  + Seed: CurrentUser, EvidenceDir, MinArtifactsTotal, MinCategoryCoverage\_pct, PassMark\_pct
* ArtifactCatalog
  + CategoryID, CategoryName, Required (TRUE/FALSE), MinCount
  + Seed categories:
    - A\_MATLAB | MATLAB/Simulink models | TRUE | 1
    - A\_RPA | RPA workflow diagrams | TRUE | 1
    - A\_BATT | Battery dashboards/lifecycle | TRUE | 1
    - A\_SENSOR | Electrochemical sensor protocols | TRUE | 1
    - A\_ETHICS | Ethics and compliance frameworks | TRUE | 1
    - A\_OPT | Optimization/GA/ML reports | TRUE | 1
* Artifacts
  + ArtifactID, LearnerID, CategoryID, Title, URI\_or\_Path, Date, Verified (TRUE/FALSE), Verifier, Notes
* Simulations
  + SimID, LearnerID, Domain (Signal/Control/Power/Energy/Thermal/Automation), ModelRef, Status (Pending/Run/Pass/Fail), Score\_pct, Notes
* Metrics
  + Topic, Metric, Value, Unit, Timestamp
* Events
  + Timestamp, User, Topic, EventType, K1, K2, Notes
* Portfolio
  + Generated by macro

Optional:

* Activities (map labs/assessments), Assessments (scores), if you also gate modules.

## Logigram gates

* **Category coverage:** For each Required category in ArtifactCatalog, learner has at least MinCount verified artifacts.
* **Total artifacts:** Learner’s verified artifacts ≥ MinArtifactsTotal.
* **Simulation pass:** All Simulations for learner have Status = Pass with Score\_pct ≥ PassMark\_pct.
* **Ethics presence:** At least one verified A\_ETHICS artifact.

## Algorigram flows

* AddArtifact → Verify → RecalculateCoverage → If gates satisfied, enable ExportPortfolio.
* ddArtifact → Verify → RecalculateCoverage → If gates satisfied, enable ExportPortfolio.
* RegisterSimulation → RunSimulation → Score and set Pass/Fail → Log Metrics/Events.
* ExportPortfolio compiles artifacts, simulations, metrics into a PDF.

## Core utilities and logging

Option Explicit

Function WS(name As String) As Worksheet

Set WS = ThisWorkbook.Worksheets(name)

End Function

Function Cfg(key As String, Optional defVal As Variant = "") As Variant

Dim r As Range: Set r = WS("Config").Columns(1).Find(key, , xlValues, xlWhole)

If r Is Nothing Then Cfg = defVal Else Cfg = IIf(Len(r.Offset(0, 1).Value) = 0, defVal, r.Offset(0, 1).Value)

End Function

Function NowStamp() As String

NowStamp = Format(Now, "yyyy-mm-dd hh:nn:ss")

End Function

Sub LogEvent(topic As String, evt As String, Optional k1 As String = "", Optional k2 As String = "", Optional notes As String = "")

Dim w As Worksheet: Set w = WS("Events")

Dim r As Long: r = w.Cells(w.Rows.Count, 1).End(xlUp).Row + 1

w.Cells(r, 1) = NowStamp(): w.Cells(r, 2) = Cfg("CurrentUser", "User")

w.Cells(r, 3) = topic: w.Cells(r, 4) = evt

w.Cells(r, 5) = k1: w.Cells(r, 6) = k2: w.Cells(r, 7) = notes

End Sub

Sub LogMetric(topic As String, metric As String, val As Double, unitStr As String)

Dim w As Worksheet: Set w = WS("Metrics")

Dim r As Long: r = w.Cells(w.Rows.Count, 1).End(xlUp).Row + 1

w.Cells(r, 1) = topic: w.Cells(r, 2) = metric

w.Cells(r, 3) = val: w.Cells(r, 4) = unitStr

w.Cells(r, 5) = NowStamp()

End Sub

## Artifact intake, verification, and coverage

vba

Sub AddArtifact(learnerID As String, categoryID As String, title As String, uri As String, Optional notes As String = "")

Dim w As Worksheet: Set w = WS("Artifacts")

Dim r As Long: r = w.Cells(w.Rows.Count, 1).End(xlUp).Row + 1

w.Cells(r, 1) = "ART" & Format(Now, "yymmddhhnnss")

w.Cells(r, 2) = learnerID

w.Cells(r, 3) = categoryID

w.Cells(r, 4) = title

w.Cells(r, 5) = uri

w.Cells(r, 6) = NowStamp()

w.Cells(r, 7) = False

w.Cells(r, 8) = ""

w.Cells(r, 9) = notes

LogEvent "Artifact", "Added", learnerID, categoryID, title

End Sub

Sub VerifyArtifact(artifactID As String, verifier As String, Optional note As String = "")

Dim w As Worksheet: Set w = WS("Artifacts")

Dim r As Range: Set r = w.Columns(1).Find(artifactID, , xlValues, xlWhole)

If r Is Nothing Then

LogEvent "Artifact", "VerifyError", artifactID, verifier, "Not found": Exit Sub

End If

r.Offset(0, 6) = True

r.Offset(0, 7) = verifier

r.Offset(0, 8) = note

LogEvent "Artifact", "Verified", artifactID, verifier, note

End Sub

Function CategoryCoverageOK(learnerID As String) As Boolean

Dim cat As Worksheet: Set cat = WS("ArtifactCatalog")

Dim art As Worksheet: Set art = WS("Artifacts")

Dim lastC As Long: lastC = cat.Cells(cat.Rows.Count, 1).End(xlUp).Row

Dim i As Long

For i = 2 To lastC

If CBool(cat.Cells(i, 3).Value) Then ' Required

Dim need As Long: need = CLng(cat.Cells(i, 4).Value)

Dim have As Long: have = CountVerified(art, learnerID, CStr(cat.Cells(i, 1).Value))

If have < need Then CategoryCoverageOK = False: Exit Function

End If

Next i

CategoryCoverageOK = True

End Function

Function CountVerified(art As Worksheet, learnerID As String, categoryID As String) As Long

Dim last As Long: last = art.Cells(art.Rows.Count, 1).End(xlUp).Row

Dim i As Long, n As Long

For i = 2 To last

If art.Cells(i, 2).Value = learnerID And art.Cells(i, 3).Value = categoryID And CBool(art.Cells(i, 7).Value) Then n = n + 1

Next i

CountVerified = n

End Function

Function TotalArtifactsOK(learnerID As String) As Boolean

Dim art As Worksheet: Set art = WS("Artifacts")

Dim last As Long: last = art.Cells(art.Rows.Count, 1).End(xlUp).Row

Dim i As Long, n As Long

For i = 2 To last

If art.Cells(i, 2).Value = learnerID And CBool(art.Cells(i, 7).Value) Then n = n + 1

Next i

TotalArtifactsOK = (n >= CLng(Cfg("MinArtifactsTotal", 6)))

End Function

Function EthicsPresent(learnerID As String) As Boolean

EthicsPresent = (CountVerified(WS("Artifacts"), learnerID, "A\_ETHICS") >= 1)

End Function

## Simulations: register, run, score, pass/fail

The simulation helpers cover common portfolio calculations and record a single composite Score\_pct per SimID.

Sub RegisterSimulation(simID As String, learnerID As String, domain As String, modelRef As String)

Dim s As Worksheet: Set s = WS("Simulations")

Dim r As Long: r = s.Cells(s.Rows.Count, 1).End(xlUp).Row + 1

s.Cells(r, 1) = simID: s.Cells(r, 2) = learnerID

s.Cells(r, 3) = domain: s.Cells(r, 4) = modelRef

s.Cells(r, 5) = "Pending": s.Cells(r, 6) = 0

LogEvent "Sim", "Registered", simID, learnerID, domain & " | " & modelRef

End Sub

Sub RunSimulation(simID As String)

Dim s As Worksheet: Set s = WS("Simulations")

Dim r As Range: Set r = s.Columns(1).Find(simID, , xlValues, xlWhole)

If r Is Nothing Then LogEvent "Sim", "RunError", simID, "", "Not found": Exit Sub

Dim domain As String: domain = CStr(r.Offset(0, 2).Value)

Dim score As Double: score = 0

r.Offset(0, 4).Value = "Run"

Select Case LCase(domain)

Case "signal": score = Sim\_SignalProcessing(r)

Case "control": score = Sim\_ControlSystem(r)

Case "power": score = Sim\_PowerFlow(r)

Case "energy": score = Sim\_EnergyIntegration(r)

Case "automation": score = Sim\_RPA\_Checks(r)

Case Else: score = 0

End Select

r.Offset(0, 5).Value = score

Dim passPct As Double: passPct = CDbl(Cfg("PassMark\_pct", 60))

r.Offset(0, 5).NumberFormat = "0.0"

r.Offset(0, 4).Value = IIf(score >= passPct, "Pass", "Fail")

LogMetric "Sim", "Score\_" & simID, score, "pct"

LogEvent "Sim", IIf(score >= passPct, "Pass", "Fail"), simID, CStr(score), domain

End Sub

### Simulation helpers (portfolio-ready, light)

vba

Function Sim\_SignalProcessing(r As Range) As Double

' Score components: spectral calc, filter design, reconstruction error

Dim spectral\_ok As Double: spectral\_ok = 1 ' stub: 1 or 0

Dim filter\_ok As Double: filter\_ok = 1

Dim recon\_err As Double: recon\_err = 0.08 ' smaller is better

Sim\_SignalProcessing = Round(100 \* (0.35 \* spectral\_ok + 0.35 \* filter\_ok + 0.30 \* (1 - recon\_err)), 1)

End Function

Function Sim\_ControlSystem(r As Range) As Double

' PID tuning results: overshoot <= 10%, settling <= 5s, steady-state error <= 2%

Dim overshoot As Double: overshoot = 0.09

Dim ts As Double: ts = 4.2

Dim ess As Double: ess = 0.01

Dim score As Double: score = 100

If overshoot > 0.1 Then score = score - 20

If ts > 5 Then score = score - 20

If ess > 0.02 Then score = score - 20

Sim\_ControlSystem = Application.Max(0, score)

End Function

Function Sim\_PowerFlow(r As Range) As Double

' Load-flow: PF correction and losses — exemplary targets met?

Dim pf As Double: pf = 0.97

Dim losses\_pct As Double: losses\_pct = 4.5

Dim score As Double: score = 100

If pf < 0.95 Then score = score - 30

If losses\_pct > 5 Then score = score - 20

Sim\_PowerFlow = Application.Max(0, score)

End Function

Function Sim\_EnergyIntegration(r As Range) As Double

' Energy integration over time (portfolio demonstration)

Dim E\_kWh As Double: E\_kWh = 125.3

Dim target As Double: target = 120

Dim score As Double: score = 100 - Application.Max(0, (Abs(E\_kWh - target) / target) \* 100)

Sim\_EnergyIntegration = Application.Max(0, Round(score, 1))

End Function

Function Sim\_RPA\_Checks(r As Range) As Double

' RPA: steps executed, exceptions handled, SLA met

Dim steps\_ok As Double: steps\_ok = 1

Dim exceptions\_ok As Double: exceptions\_ok = 1

Dim sla\_ok As Double: sla\_ok = 0.95 ' 95% on-time

Sim\_RPA\_Checks = Round(100 \* (0.4 \* steps\_ok + 0.3 \* exceptions\_ok + 0.3 \* sla\_ok), 1)

End Function

## Portfolio gates and export

vba

Function PortfolioGatesOK(learnerID As String) As Boolean

Dim ok As Boolean: ok = True

If Not CategoryCoverageOK(learnerID) Then LogEvent "Gate", "CategoryCoverageFail", learnerID, "", "": ok = False

If Not TotalArtifactsOK(learnerID) Then LogEvent "Gate", "TotalArtifactsFail", learnerID, "", "": ok = False

If Not EthicsPresent(learnerID) Then LogEvent "Gate", "EthicsMissing", learnerID, "", "": ok = False

If Not AllSimulationsPassed(learnerID) Then LogEvent "Gate", "SimulationsFail", learnerID, "", "": ok = False

PortfolioGatesOK = ok

End Function

Function AllSimulationsPassed(learnerID As String) As Boolean

Dim s As Worksheet: Set s = WS("Simulations")

Dim last As Long: last = s.Cells(s.Rows.Count, 1).End(xlUp).Row

Dim i As Long

For i = 2 To last

If s.Cells(i, 2).Value = learnerID Then

If LCase(s.Cells(i, 5).Value) <> "pass" Then AllSimulationsPassed = False: Exit Function

End If

Next i

AllSimulationsPassed = True

End Function

Sub ExportPortfolio(learnerID As String)

If Not PortfolioGatesOK(learnerID) Then

MsgBox "Portfolio gates not satisfied. Check Events for details.", vbExclamation

Exit Sub

End If

On Error Resume Next: Application.DisplayAlerts = False

ThisWorkbook.Worksheets("Portfolio").Delete

Application.DisplayAlerts = True: On Error GoTo 0

Dim wr As Worksheet: Set wr = ThisWorkbook.Worksheets.Add

wr.Name = "Portfolio"

Dim r As Long: r = 1

wr.Cells(r, 1) = "Advanced Electrical Systems & Automation – Portfolio of Evidence": r = r + 2

wr.Cells(r, 1) = "LearnerID": wr.Cells(r, 2) = learnerID: r = r + 2

r = CopySectionByMatch(wr, r, "Artifacts (Verified)", WS("Artifacts"), 2, learnerID, 7, True)

r = CopySectionByMatch(wr, r, "Simulations", WS("Simulations"), 2, learnerID)

r = CopySectionByMatch(wr, r, "Metrics", WS("Metrics"), 1, "Sim")

wr.Columns.AutoFit

Dim f As String: f = CStr(Cfg("EvidenceDir", ThisWorkbook.Path)) & "\PoE\_Advanced\_EE\_" & learnerID & ".pdf"

On Error Resume Next: wr.ExportAsFixedFormat xlTypePDF, f: On Error GoTo 0

LogEvent "Portfolio", "Exported", learnerID, "", f

MsgBox "Portfolio exported: " & f, vbInformation

End Sub

Function CopySectionByMatch(dst As Worksheet, startRow As Long, title As String, src As Worksheet, matchCol As Integer, key As Variant, Optional filterColBool As Integer = 0, Optional filterValBool As Boolean = False) As Long

dst.Cells(startRow, 1) = title

Dim r As Long: r = startRow + 1

Dim rng As Range: Set rng = src.Range("A1").CurrentRegion

Dim i As Long, header As Boolean

For i = 2 To rng.Rows.Count

Dim cond As Boolean: cond = (CStr(rng.Cells(i, matchCol).Value) = CStr(key))

If cond And filterColBool > 0 Then

cond = (CBool(rng.Cells(i, filterColBool).Value) = filterValBool)

End If

If cond Then

If Not header Then rng.Rows(1).Copy dst.Cells(r, 1): r = r + 1: header = True

rng.Rows(i).Copy dst.Cells(r, 1): r = r + 1

End If

Next i

CopySectionByMatch = r + 1

End Function

## Ready-to-use simulation macros for artifacts

These map directly to your artifact categories and sample macro list

' A\_MATLAB: record a MATLAB/Simulink model reference as artifact

Sub RecordMatlabModel(learnerID As String, title As String, path As String)

AddArtifact learnerID, "A\_MATLAB", title, path, "Simulink/Matlab model"

End Sub

' A\_RPA: record an RPA workflow comparison

Sub RecordRPAWorkflow(learnerID As String, title As String, path As String)

AddArtifact learnerID, "A\_RPA", title, path, "UiPath/AA/BluePrism comparison"

End Sub

' A\_BATT: compute basic battery KPI and record dashboard link

Sub RecordBatteryDashboard(learnerID As String, title As String, path As String, cap\_meas\_Ah As Double, cap\_nom\_Ah As Double)

Dim fade As Double: fade = CapacityFade\_pct(cap\_meas\_Ah, cap\_nom\_Ah)

LogMetric "Battery", "CapacityFade\_pct", fade, "%"

AddArtifact learnerID, "A\_BATT", title, path, "Fade=" & Format(fade, "0.0") & "%"

End Sub

' A\_SENSOR: sensor drift protocol

Sub RecordSensorProtocol(learnerID As String, title As String, path As String, val\_now As Double, val\_ref As Double)

Dim drift As Double: drift = SensorDrift\_pct(val\_now, val\_ref)

LogMetric "Sensor", "Drift\_pct", drift, "%"

AddArtifact learnerID, "A\_SENSOR", title, path, "Drift=" & Format(drift, "0.0") & "%"

End Sub

' A\_ETHICS: ethics & compliance framework

Sub RecordEthicsFramework(learnerID As String, title As String, path As String)

AddArtifact learnerID, "A\_ETHICS", title, path, "ISO/IEC, governance, risk & compliance"

End Sub

' A\_OPT: optimization report

Sub RecordOptimizationReport(learnerID As String, title As String, path As String, baseline As Double, optimized As Double)

Dim gain As Double: If baseline = 0 Then gain = 0 Else gain = 100 \* (baseline - optimized) / baseline

LogMetric "Optimization", "Gain\_pct", gain, "%"

AddArtifact learnerID, "A\_OPT", title, path, "Gain=" & Format(gain, "0.0") & "%"

End Sub

## Quick start

* Config:
  + CurrentUser = Tshingombe Fiston Tshitadi
  + EvidenceDir = C:\Evidence
  + MinArtifactsTotal = 6
  + MinCategoryCoverage\_pct = 100
  + PassMark\_pct = 60
* Register and run:
  + RegisterSimulation "SIM\_SIG\_01","Learner01","Signal","Fourier/Laplace"
  + RegisterSimulation "SIM\_CTRL\_01","Learner01","Control","PID tune"
  + RegisterSimulation "SIM\_PWR\_01","Learner01","Power","Load flow"
  + RegisterSimulation "SIM\_EN\_01","Learner01","Energy","Integration"
  + RegisterSimulation "SIM\_RPA\_01","Learner01","Automation","RPA checks"
  + RunSimulation "SIM\_SIG\_01": RunSimulation "SIM\_CTRL\_01": RunSimulation "SIM\_PWR\_01"
  + RunSimulation "SIM\_EN\_01": RunSimulation "SIM\_RPA\_01"
* Add and verify artifacts:
  + RecordMatlabModel "Learner01","NR Load Flow","C:\PoE\models\loadflow.slx"
  + RecordRPAWorkflow "Learner01","RPA Tool Matrix","C:\PoE\rpa\comparison.pdf"
  + RecordBatteryDashboard "Learner01","Li-ion Dashboard","C:\PoE\battery\dashboard.xlsx", 84, 100
  + RecordSensorProtocol "Learner01","NO2 Sensor SOP","C:\PoE\sensors\no2\_sop.docx", 210, 200
  + RecordEthicsFramework "Learner01","Automation Ethics","C:\PoE\ethics\framework.docx"
  + RecordOptimizationReport "Learner01","GA Optimization","C:\PoE\opt\ga\_report.pdf", 10.2, 8.4
  + VerifyArtifact "ART240912123000","Assessor","OK" ' use actual ID returned in sheet
* Export:
  + ExportPortfolio "Learner01"

## Optional extensions

* Add a UserForm to add/verify artifacts and run simulations with one click.
* Attach Assessments sheet and gate by module scores per domain.
* Auto-generate cover page with category coverage chart and simulation score tiles.
* Add CSV import for simulation results (from MATLAB/Python) and auto-score based on thresholds.

If you want, I’ll wire a compact dashboard with traffic lights for category coverage, total artifacts, simulation pass rate, and ethics presence—so your learners and stakeholders can see readiness at a glance.

# VBA logigram and algorigram for advanced simulations, artifacts, and analytics

This builds an audit-ready simulation workbook that:

* Enforces readiness gates (logigram) for artifacts, simulations, and ethics.
* Executes calculations and labs (algorigram): apparent power, statistical modeling, signal/control/power exercises.
* Logs metrics/events and exports a portfolio of evidence (PoE).

## Workbook schema

Create these sheets with exact headers (row 1).

* Config
  + Key, Value
  + Seed: CurrentUser, EvidenceDir, MinArtifactsTotal, PassMark\_pct, MinEthicsArtifacts
* ArtifactCatalog
  + CategoryID, CategoryName, Required (TRUE/FALSE), MinCount
  + Seed rows:
    - A\_MATLAB | MATLAB/Simulink models | TRUE | 1
    - A\_RPA | RPA workflow diagrams | TRUE | 1
    - A\_BATT | Battery dashboards/lifecycle | TRUE | 1
    - A\_SENSOR | Electrochemical sensor protocols | TRUE | 1
    - A\_ETHICS | Ethics/compliance frameworks | TRUE | 1
    - A\_OPT | Optimization/GA/ML reports | TRUE | 1
* Artifacts
  + ArtifactID, LearnerID, CategoryID, Title, URI\_or\_Path, Date, Verified (TRUE/FALSE), Verifier, Notes
* Simulations
  + SimID, LearnerID, Domain (Signal/Control/Power/Energy/Stats), ModelRef, Status (Pending/Pass/Fail), Score\_pct, Notes
* Metrics
  + Topic, Metric, Value, Unit, Timestamp
* Events
  + Timestamp, User, Topic, EventType, K1, K2, Notes
* Portfolio
  + Generated by macro

## Logigram gates

* **Category coverage:** All Required categories in ArtifactCatalog met (Verified ≥ MinCount).
* **Total artifacts:** Verified artifacts ≥ MinArtifactsTotal.
* **Ethics present:** Verified A\_ETHICS ≥ MinEthicsArtifacts.
* **Sim pass rate:** All Simulations for learner Status = Pass and Score\_pct ≥ PassMark\_pct.

## Core utilities and logging

vba

Option Explicit

Function WS(name As String) As Worksheet: Set WS = ThisWorkbook.Worksheets(name): End Function

Function Cfg(key As String, Optional defVal As Variant = "") As Variant

Dim r As Range: Set r = WS("Config").Columns(1).Find(key, , xlValues, xlWhole)

If r Is Nothing Then Cfg = defVal Else Cfg = IIf(Len(r.Offset(0, 1).Value) = 0, defVal, r.Offset(0, 1).Value)

End Function

Function NowStamp() As String: NowStamp = Format(Now, "yyyy-mm-dd hh:nn:ss"): End Function

Sub LogEvent(topic As String, evt As String, Optional k1 As String = "", Optional k2 As String = "", Optional notes As String = "")

Dim w As Worksheet: Set w = WS("Events")

Dim r As Long: r = w.Cells(w.Rows.Count, 1).End(xlUp).Row + 1

w.Cells(r, 1) = NowStamp(): w.Cells(r, 2) = Cfg("CurrentUser", "User")

w.Cells(r, 3) = topic: w.Cells(r, 4) = evt

w.Cells(r, 5) = k1: w.Cells(r, 6) = k2: w.Cells(r, 7) = notes

End Sub

Sub LogMetric(topic As String, metric As String, val As Double, unitStr As String)

Dim w As Worksheet: Set w = WS("Metrics")

Dim r As Long: r = w.Cells(w.Rows.Count, 1).End(xlUp).Row + 1

w.Cells(r, 1) = topic: w.Cells(r, 2) = metric

w.Cells(r, 3) = val: w.Cells(r, 4) = unitStr

w.Cells(r, 5) = NowStamp()

End Sub

## Artifact intake, verification, and gates

Sub AddArtifact(learnerID As String, categoryID As String, title As String, uri As String, Optional notes As String = "")

Dim w As Worksheet: Set w = WS("Artifacts")

Dim r As Long: r = w.Cells(w.Rows.Count, 1).End(xlUp).Row + 1

w.Cells(r, 1) = "ART" & Format(Now, "yymmddhhnnss")

w.Cells(r, 2) = learnerID: w.Cells(r, 3) = categoryID

w.Cells(r, 4) = title: w.Cells(r, 5) = uri

w.Cells(r, 6) = NowStamp(): w.Cells(r, 7) = False

w.Cells(r, 8) = "": w.Cells(r, 9) = notes

LogEvent "Artifact", "Added", learnerID, categoryID, title

End Sub

Sub VerifyArtifact(artifactID As String, verifier As String, Optional note As String = "")

Dim w As Worksheet: Set w = WS("Artifacts")

Dim r As Range: Set r = w.Columns(1).Find(artifactID, , xlValues, xlWhole)

If r Is Nothing Then LogEvent "Artifact", "VerifyError", artifactID, verifier, "Not found": Exit Sub

r.Offset(0, 6) = True: r.Offset(0, 7) = verifier: r.Offset(0, 8) = note

LogEvent "Artifact", "Verified", artifactID, verifier, note

End Sub

Function CountVerified(learnerID As String, Optional categoryID As String = "") As Long

Dim art As Worksheet: Set art = WS("Artifacts")

Dim last As Long: last = art.Cells(art.Rows.Count, 1).End(xlUp).Row

Dim i As Long, n As Long

For i = 2 To last

If art.Cells(i, 2).Value = learnerID And CBool(art.Cells(i, 7).Value) Then

If Len(categoryID) = 0 Or art.Cells(i, 3).Value = categoryID Then n = n + 1

End If

Next i

CountVerified = n

End Function

Function CategoryCoverageOK(learnerID As String) As Boolean

Dim cat As Worksheet: Set cat = WS("ArtifactCatalog")

Dim last As Long: last = cat.Cells(cat.Rows.Count, 1).End(xlUp).Row

Dim i As Long

For i = 2 To last

If CBool(cat.Cells(i, 3).Value) Then

If CountVerified(learnerID, CStr(cat.Cells(i, 1).Value)) < CLng(cat.Cells(i, 4).Value) Then Exit Function

End If

Next i

CategoryCoverageOK = True

End Function

Function PortfolioGatesOK(learnerID As String) As Boolean

Dim ok As Boolean: ok = True

If Not CategoryCoverageOK(learnerID) Then LogEvent "Gate", "CategoryCoverageFail", learnerID: ok = False

If CountVerified(learnerID) < CLng(Cfg("MinArtifactsTotal", 6)) Then LogEvent "Gate", "ArtifactsTotalFail", learnerID: ok = False

If CountVerified(learnerID, "A\_ETHICS") < CLng(Cfg("MinEthicsArtifacts", 1)) Then LogEvent "Gate", "EthicsMissing", learnerID: ok = False

If Not AllSimulationsPassed(learnerID) Then LogEvent "Gate", "SimulationsFail", learnerID: ok = False

PortfolioGatesOK = ok

End Function

## Simulation registration, run, scoring

Sub RegisterSimulation(simID As String, learnerID As String, domain As String, modelRef As String)

Dim s As Worksheet: Set s = WS("Simulations")

Dim r As Long: r = s.Cells(s.Rows.Count, 1).End(xlUp).Row + 1

s.Cells(r, 1) = simID: s.Cells(r, 2) = learnerID

s.Cells(r, 3) = domain: s.Cells(r, 4) = modelRef

s.Cells(r, 5) = "Pending": s.Cells(r, 6) = 0

LogEvent "Sim", "Registered", simID, learnerID, domain & " | " & modelRef

End Sub

Sub RunSimulation(simID As String)

Dim s As Worksheet: Set s = WS("Simulations")

Dim r As Range: Set r = s.Columns(1).Find(simID, , xlValues, xlWhole)

If r Is Nothing Then LogEvent "Sim", "RunError", simID, "", "Not found": Exit Sub

Dim domain As String: domain = LCase(CStr(r.Offset(0, 2).Value))

Dim score As Double

Select Case domain

Case "signal": score = Sim\_Signal()

Case "control": score = Sim\_Control()

Case "power": score = Sim\_Power()

Case "energy": score = Sim\_Energy()

Case "stats": score = Sim\_Stats()

Case Else: score = 0

End Select

r.Offset(0, 6).Value = Round(score, 1)

r.Offset(0, 5).Value = IIf(score >= CDbl(Cfg("PassMark\_pct", 60)), "Pass", "Fail")

LogMetric "Sim", "Score\_" & simID, score, "pct"

LogEvent "Sim", r.Offset(0, 5).Value, simID, CStr(score), domain

End Sub

Function AllSimulationsPassed(learnerID As String) As Boolean

Dim s As Worksheet: Set s = WS("Simulations")

Dim last As Long: last = s.Cells(s.Rows.Count, 1).End(xlUp).Row

Dim i As Long

For i = 2 To last

If s.Cells(i, 2).Value = learnerID Then

If LCase(CStr(s.Cells(i, 5).Value)) <> "pass" Then Exit Function

End If

Next i

AllSimulationsPassed = True

End Function

## Calculation engines (apparent power, stats, and scoring)

vba

Apparent power: S = sqrt(P^2 + Q^2) [VA]

Function ApparentPower\_VA(P\_W As Double, Q\_var As Double) As Double

ApparentPower\_VA = Sqr(P\_W ^ 2 + Q\_var ^ 2)

End Function

' Mean of an array

Function Mean(ByRef arr As Variant) As Double

Dim i As Long, n As Long, s As Double

For i = LBound(arr) To UBound(arr): s = s + arr(i): n = n + 1: Next i

If n = 0 Then Mean = 0 Else Mean = s / n

End Function

' Variance (population) σ^2 = sum((x - μ)^2)/n

Function VariancePop(ByRef arr As Variant) As Double

Dim mu As Double: mu = Mean(arr)

Dim i As Long, n As Long, ssq As Double

For i = LBound(arr) To UBound(arr): ssq = ssq + (arr(i) - mu) ^ 2: n = n + 1: Next i

If n = 0 Then VariancePop = 0 Else VariancePop = ssq / n

End Function

### Simulation stubs using the engines

vba

Function Sim\_Power() As Double

' Demo: P=8 kW, Q=6 kVAr → S=10 kVA; target within tolerance

Dim S As Double: S = ApparentPower\_VA(8000, 6000) / 1000 ' kVA

LogMetric "Power", "S\_kVA", S, "kVA"

Sim\_Power = IIf(Abs(S - 10) <= 0.2, 95, 60)

End Function

Function Sim\_Stats() As Double

' Grades: [70, 75, 80, 85, 90], mean=80, variance=50 (population)

Dim g(1 To 5) As Double: g(1)=70: g(2)=75: g(3)=80: g(4)=85: g(5)=90

Dim mu As Double: mu = Mean(g)

Dim v As Double: v = VariancePop(g)

LogMetric "Stats", "Mean", mu, "points"

LogMetric "Stats", "Variance", v, "points^2"

Sim\_Stats = IIf(Abs(mu - 80) < 0.01 And Abs(v - 50) < 0.01, 100, 50)

End Function

Function Sim\_Signal() As Double

' Placeholder: assume successful spectral design and low reconstruction error

Sim\_Signal = 92

End Function

Function Sim\_Control() As Double

' Placeholder: overshoot/settling/ess targets met

Sim\_Control = 90

End Function

Function Sim\_Energy() As Double

' Placeholder: energy integration target ±5% achieved

Sim\_Energy = 88

End Function

## Artifact recording helpers mapped to your portfolio

vba

Sub RecordMatlabModel(learnerID As String, title As String, path As String)

AddArtifact learnerID, "A\_MATLAB", title, path, "MATLAB/Simulink model"

End Sub

Sub RecordRPAWorkflow(learnerID As String, title As String, path As String)

AddArtifact learnerID, "A\_RPA", title, path, "RPA workflows and tool comparison"

End Sub

Sub RecordBatteryDashboard(learnerID As String, title As String, path As String, cap\_meas\_Ah As Double, cap\_nom\_Ah As Double)

Dim fade As Double: fade = 100 \* (1 - IIf(cap\_nom\_Ah = 0, 0, cap\_meas\_Ah / cap\_nom\_Ah))

LogMetric "Battery", "CapacityFade\_pct", fade, "%"

AddArtifact learnerID, "A\_BATT", title, path, "Fade=" & Format(fade, "0.0") & "%"

End Sub

Sub RecordSensorProtocol(learnerID As String, title As String, path As String, val\_now As Double, val\_ref As Double)

Dim drift As Double: drift = IIf(val\_ref = 0, 0, 100 \* (val\_now - val\_ref) / val\_ref)

LogMetric "Sensor", "Drift\_pct", drift, "%"

AddArtifact learnerID, "A\_SENSOR", title, path, "Drift=" & Format(drift, "0.0") & "%"

End Sub

Sub RecordEthicsFramework(learnerID As String, title As String, path As String)

AddArtifact learnerID, "A\_ETHICS", title, path, "Compliance and governance"

End Sub

Sub RecordOptimizationReport(learnerID As String, title As String, path As String, baseline As Double, optimized As Double)

Dim gain As Double: gain = IIf(baseline = 0, 0, 100 \* (baseline - optimized) / baseline)

LogMetric "Optimization", "Gain\_pct", gain, "%"

AddArtifact learnerID, "A\_OPT", title, path, "Gain=" & Format(gain, "0.0") & "%"

End Sub

## Export portfolio

vba

Sub ExportPortfolio(learnerID As String)

If Not PortfolioGatesOK(learnerID) Then

MsgBox "Gates not satisfied. See Events sheet for details.", vbExclamation

Exit Sub

End If

On Error Resume Next: Application.DisplayAlerts = False

ThisWorkbook.Worksheets("Portfolio").Delete

Application.DisplayAlerts = True: On Error GoTo 0

Dim wr As Worksheet: Set wr = ThisWorkbook.Worksheets.Add

wr.Name = "Portfolio"

Dim r As Long: r = 1

wr.Cells(r, 1) = "Advanced Electrical Systems & Automation – Portfolio of Evidence": r = r + 2

wr.Cells(r, 1) = "LearnerID": wr.Cells(r, 2) = learnerID: r = r + 2

r = CopyVerifiedArtifacts(wr, r, learnerID)

r = CopyRowsByMatch(wr, r, "Simulations", WS("Simulations"), 2, learnerID)

r = CopyRowsByMatch(wr, r, "Metrics (Power/Stats/Sim)", WS("Metrics"), 1, "Sim")

r = CopyRowsByMatch(wr, r, "", WS("Metrics"), 1, "Power")

r = CopyRowsByMatch(wr, r, "", WS("Metrics"), 1, "Stats")

wr.Columns.AutoFit

Dim f As String: f = CStr(Cfg("EvidenceDir", ThisWorkbook.Path)) & "\PoE\_AdvancedEE\_" & learnerID & ".pdf"

On Error Resume Next: wr.ExportAsFixedFormat xlTypePDF, f: On Error GoTo 0

LogEvent "Portfolio", "Exported", learnerID, "", f

MsgBox "Portfolio exported: " & f, vbInformation

End Sub

Function CopyVerifiedArtifacts(dst As Worksheet, startRow As Long, learnerID As String) As Long

dst.Cells(startRow, 1) = "Artifacts (Verified)"

Dim r As Long: r = startRow + 1

Dim art As Worksheet: Set art = WS("Artifacts")

Dim rng As Range: Set rng = art.Range("A1").CurrentRegion

Dim i As Long, header As Boolean

For i = 2 To rng.Rows.Count

If rng.Cells(i, 2).Value = learnerID And CBool(rng.Cells(i, 7).Value) Then

If Not header Then rng.Rows(1).Copy dst.Cells(r, 1): r = r + 1: header = True

rng.Rows(i).Copy dst.Cells(r, 1): r = r + 1

End If

Next i

CopyVerifiedArtifacts = r + 1

End Function

Function CopyRowsByMatch(dst As Worksheet, startRow As Long, title As String, src As Worksheet, matchCol As Integer, key As String) As Long

If Len(title) > 0 Then dst.Cells(startRow, 1) = title

Dim r As Long: r = startRow + 1

Dim rng As Range: Set rng = src.Range("A1").CurrentRegion

Dim i As Long, header As Boolean

For i = 2 To rng.Rows.Count

If InStr(1, CStr(rng.Cells(i, matchCol).Value), key, vbTextCompare) > 0 Then

If Not header Then rng.Rows(1).Copy dst.Cells(r, 1): r = r + 1: header = True

rng.Rows(i).Copy dst.Cells(r, 1): r = r + 1

End If

Next i

CopyRowsByMatch = r + 1

End Function

## Quick start

* Config:

## Quick start

* Config:
  + CurrentUser = Tshingombe Fiston Tshitadi
  + EvidenceDir = C:\Evidence
  + MinArtifactsTotal = 6
  + MinEthicsArtifacts = 1
  + PassMark\_pct = 60
* Register and run sims:
  + RegisterSimulation "SIM\_PWR\_01","Learner01","Power","S=sqrt(P2+Q2)"
  + RegisterSimulation "SIM\_CTRL\_01","Learner01","Control","PID tuning"
  + RegisterSimulation "SIM\_SIG\_01","Learner01","Signal","Fourier filter"
  + RegisterSimulation "SIM\_EN\_01","Learner01","Energy","Integration"
  + RegisterSimulation "SIM\_ST\_01","Learner01","Stats","Mean & variance"
  + RunSimulation "SIM\_PWR\_01": RunSimulation "SIM\_CTRL\_01": RunSimulation "SIM\_SIG\_01": RunSimulation "SIM\_EN\_01": RunSimulation "SIM\_ST\_01"
* Add artifacts (examples):
  + RecordMatlabModel "Learner01","Load Flow & Control","C:\PoE\models\lf\_ctrl.slx"
  + RecordRPAWorkflow "Learner01","RPA Tooling Matrix","C:\PoE\rpa\matrix.pdf"
  + RecordBatteryDashboard "Learner01","Li-ion Dashboard","C:\PoE\batt\dashboard.xlsx",84,100
  + RecordSensorProtocol "Learner01","EC Sensor SOP","C:\PoE\sensors\ec\_sop.docx",2.12,2.00
  + RecordEthicsFramework "Learner01","Automation Ethics","C:\PoE\ethics\framework.docx"
  + RecordOptimizationReport "Learner01","GA Loss Reduction","C:\PoE\opt\ga\_report.pdf",10.2,8.4
* Verify artifacts (use IDs in Artifacts sheet), then:
  + ExportPortfolio "Learner01"

## Optional GUI hook (UserForm)

* CommandButton\_RunSim: calls RunSimulation for selected SimID.
* CommandButton\_AddArtifact: calls RecordX helpers based on ComboBox Category.
* Label\_KPI: refreshed with latest Metrics for visual feedback.

If you want, I’ll wire a compact dashboard with traffic lights for gates (coverage, total artifacts, ethics, sims), plus a one-click “Export PoE” button for exhibitions and CPD submissions.

# VBA logigram and algorigram for forensic, predictive, and MIS-integrated simulations

This builds a modular simulation workbook for law enforcement, forensics, and MIS operations. It enforces readiness gates (logigram), runs predictive/forensic models and patrol analytics (algorigram), logs metrics and events, and exports a portfolio of evidence (PoE).

## Workbook schema

Create these sheets with exact headers (row 1).

* Config
  + Key, Value
  + Seed: CurrentUser, EvidenceDir, PassMark\_pct, MinArtifacts, MinEthicsArtifacts
* Activities
  + ActivityID, Domain (Forensics/MIS/Traffic/Ballistics/Patrol/Stats), Type (Model/Report/Lab/Dashboard), Required (TRUE/FALSE)
* Evidence
  + EvidenceID, ActivityID, LearnerID, Type (Doc/Data/Log), URI\_or\_Path, Timestamp, Verified (TRUE/FALSE), Verifier, Notes
* Simulations
  + SimID, LearnerID, Domain (Predictive/Decay/Patrol/Ballistics/Area/Stats), ModelRef, Status (Pending/Pass/Fail), Score\_pct, Notes
* Metrics
  + Topic, Metric, Value, Unit, Timestamp
* Events
  + Timestamp, User, Topic, EventType, K1, K2, Notes
* Portfolio
  + Generated automatically

## Logigram gates

* Module/Activity gate:
  + Required activities have at least one Verified evidence per learner.
* Simulation gate:
  + All simulations for learner have Status = Pass and Score ≥ PassMark\_pct.
* Ethics/Compliance gate:
  + At least MinEthicsArtifacts verified (e.g., chain-of-custody SOP, privacy/compliance).

Failing any gate logs Events and blocks PoE export.

## Core utilities and logging

Option Explicit

Function WS(name As String) As Worksheet: Set WS = ThisWorkbook.Worksheets(name): End Function

Function Cfg(key As String, Optional defVal As Variant = "") As Variant

Dim r As Range: Set r = WS("Config").Columns(1).Find(key, , xlValues, xlWhole)

If r Is Nothing Then Cfg = defVal Else Cfg = IIf(Len(r.Offset(0, 1).Value) = 0, defVal, r.Offset(0, 1).Value)

End Function

Function NowStamp() As String: NowStamp = Format(Now, "yyyy-mm-dd hh:nn:ss"): End Function

Sub LogEvent(topic As String, evt As String, Optional k1 As String = "", Optional k2 As String = "", Optional note As String = "")

Dim w As Worksheet: Set w = WS("Events")

Dim r As Long: r = w.Cells(w.Rows.Count, 1).End(xlUp).Row + 1

w.Cells(r, 1) = NowStamp(): w.Cells(r, 2) = Cfg("CurrentUser", "User")

w.Cells(r, 3) = topic: w.Cells(r, 4) = evt: w.Cells(r, 5) = k1: w.Cells(r, 6) = k2: w.Cells(r, 7) = note

End Sub

Sub LogMetric(topic As String, metric As String, val As Double, unitStr As String)

Dim w As Worksheet: Set w = WS("Metrics")

Dim r As Long: r = w.Cells(w.Rows.Count, 1).End(xlUp).Row + 1

w.Cells(r, 1) = topic: w.Cells(r, 2) = metric: w.Cells(r, 3) = val

w.Cells(r, 4) = unitStr: w.Cells(r, 5) = NowStamp()

End Sub

Sub AddEvidence(activityID As String, learnerID As String, typ As String, uri As String, Optional verified As Boolean = False, Optional verifier As String = "", Optional notes As String = "")

Dim ev As Worksheet: Set ev = WS("Evidence")

Dim r As Long: r = ev.Cells(ev.Rows.Count, 1).End(xlUp).Row + 1

ev.Cells(r, 1) = "E" & Format(Now, "yymmddhhnnss")

ev.Cells(r, 2) = activityID: ev.Cells(r, 3) = learnerID

ev.Cells(r, 4) = typ: ev.Cells(r, 5) = uri

ev.Cells(r, 6) = NowStamp(): ev.Cells(r, 7) = verified

ev.Cells(r, 8) = verifier: ev.Cells(r, 9) = notes

End Sub

## Evidence and simulation gates

vba

Function RequiredActivitiesHaveEvidence(learnerID As String) As Boolean

Dim act As Worksheet: Set act = WS("Activities")

Dim ev As Worksheet: Set ev = WS("Evidence")

Dim LA As Long: LA = act.Cells(act.Rows.Count, 1).End(xlUp).Row

Dim LE As Long: LE = ev.Cells(ev.Rows.Count, 1).End(xlUp).Row

Dim need As Long, have As Long, i As Long, j As Long

For i = 2 To LA

If CBool(act.Cells(i, 3).Value) Then

need = need + 1

For j = 2 To LE

If ev.Cells(j, 2).Value = act.Cells(i, 1).Value And ev.Cells(j, 7).Value = True And ev.Cells(j, 3).Value = learnerID Then

have = have + 1: Exit For

End If

Next j

End If

Next i

RequiredActivitiesHaveEvidence = (need = have)

End Function

Function EthicsPresent(learnerID As String) As Boolean

Dim ev As Worksheet: Set ev = WS("Evidence")

Dim LE As Long: LE = ev.Cells(ev.Rows.Count, 1).End(xlUp).Row

Dim i As Long, n As Long

For i = 2 To LE

If ev.Cells(i, 3).Value = learnerID And ev.Cells(i, 7).Value = True Then

If InStr(1, LCase(ev.Cells(i, 9).Value), "ethic") > 0 Or InStr(1, LCase(ev.Cells(i, 9).Value), "privacy") > 0 Or InStr(1, LCase(ev.Cells(i, 9).Value), "chain of custody") > 0 Then

n = n + 1

End If

End If

Next i

EthicsPresent = (n >= CLng(Cfg("MinEthicsArtifacts", 1)))

End Function

Function AllSimulationsPassed(learnerID As String) As Boolean

Dim s As Worksheet: Set s = WS("Simulations")

Dim last As Long: last = s.Cells(s.Rows.Count, 1).End(xlUp).Row

Dim i As Long, passPct As Double: passPct = CDbl(Cfg("PassMark\_pct", 60))

For i = 2 To last

If s.Cells(i, 2).Value = learnerID Then

If s.Cells(i, 7).Value < passPct Or LCase(s.Cells(i, 6).Value) <> "pass" Then Exit Function

End If

Next i

AllSimulationsPassed = True

End Function

Function PortfolioGatesOK(learnerID As String) As Boolean

Dim ok As Boolean: ok = True

If Not RequiredActivitiesHaveEvidence(learnerID) Then LogEvent "Gate", "EvidenceFail", learnerID, "", "Required activities missing evidence": ok = False

If Not EthicsPresent(learnerID) Then LogEvent "Gate", "EthicsFail", learnerID, "", "No ethics/compliance evidence": ok = False

If Not AllSimulationsPassed(learnerID) Then LogEvent "Gate", "SimFail", learnerID, "", "Simulations not all passed": ok = False

PortfolioGatesOK = ok

End Function

## Modeling engines (predictive, decay, patrol, ballistics, area, stats)

' Linear regression y = m x + b

Sub LinReg(ByRef x As Variant, ByRef y As Variant, ByRef m As Double, ByRef b As Double, ByRef R2 As Double)

Dim n As Long: n = UBound(x) - LBound(x) + 1

Dim i As Long, sx As Double, sy As Double, sxx As Double, syy As Double, sxy As Double

For i = LBound(x) To UBound(x)

sx = sx + x(i): sy = sy + y(i)

sxx = sxx + x(i) \* x(i): syy = syy + y(i) \* y(i)

sxy = sxy + x(i) \* y(i)

Next i

Dim den As Double: den = n \* sxx - sx ^ 2

If den = 0 Then m = 0: b = 0: R2 = 0: Exit Sub

m = (n \* sxy - sx \* sy) / den

b = (sy - m \* sx) / n

Dim ssTot As Double: ssTot = syy - sy ^ 2 / n

Dim ssReg As Double: ssReg = m \* (sxy - sx \* sy / n)

If ssTot = 0 Then R2 = 1 Else R2 = ssReg / ssTot

End Sub

' Exponential decay C(t) = C0 \* exp(-lambda \* t)

Function Decay\_C(C0 As Double, lambda As Double, t As Double) As Double

Decay\_C = C0 \* Exp(-lambda \* t)

End Function

' Angular kinematics: theta(t) = omega\*t + 0.5\*alpha\*t^2

Function Theta\_t(omega As Double, alpha As Double, t As Double) As Double

Theta\_t = omega \* t + 0.5 \* alpha \* t ^ 2

End Function

' Patrol path length L = integral sqrt(1+(dy/dx)^2) dx (discrete approximation)

Function PathLength(ByRef x As Variant, ByRef y As Variant) As Double

Dim i As Long, L As Double

For i = LBound(x) To UBound(x) - 1

Dim dx As Double: dx = x(i + 1) - x(i)

Dim dy As Double: dy = y(i + 1) - y(i)

L = L + Sqr(dx ^ 2 + dy ^ 2)

Next i

PathLength = L

End Function

' Projectile range R = v0^2 \* sin(2\*theta) / g

Function BallisticRange(v0 As Double, theta\_deg As Double, Optional g As Double = 9.80665) As Double

Dim rad As Double: rad = WorksheetFunction.Radians(2 \* theta\_deg)

BallisticRange = (v0 ^ 2 \* Sin(rad)) / g

End Function

' Polygon area (crime scene) via shoelace formula

Function PolygonArea(ByRef x As Variant, ByRef y As Variant) As Double

Dim n As Long: n = UBound(x) - LBound(x) + 1

Dim i As Long, s As Double

For i = LBound(x) To UBound(x) - 1

s = s + x(i) \* y(i + 1) - x(i + 1) \* y(i)

Next i

s = s + x(UBound(x)) \* y(LBound(y)) - x(LBound(x)) \* y(UBound(y))

PolygonArea = 0.5 \* Abs(s)

End Function

' Apparent power S = sqrt(P^2 + Q^2)

Function ApparentPower\_VA(P\_W As Double, Q\_var As Double) As Double

ApparentPower\_VA = Sqr(P\_W ^ 2 + Q\_var ^ 2)

End Function

' Stats: mean and population variance

Function MeanArr(ByRef a As Variant) As Double

Dim i As Long, s As Double

For i = LBound(a) To UBound(a): s = s + a(i): Next i

MeanArr = s / (UBound(a) - LBound(a) + 1)

End Function

Function VarPopArr(ByRef a As Variant) As Double

Dim mu As Double: mu = MeanArr(a)

Dim i As Long, n As Long, ssq As Double

For i = LBound(a) To UBound(a)

ssq = ssq + (a(i) - mu) ^ 2: n = n + 1

Next i

VarPopArr = ssq / n

End Function

## Simulation registration, scoring, and runners

vba

Sub RegisterSimulation(simID As String, learnerID As String, domain As String, modelRef As String)

Dim s As Worksheet: Set s = WS("Simulations")

Dim r As Long: r = s.Cells(s.Rows.Count, 1).End(xlUp).Row + 1

s.Cells(r, 1) = simID: s.Cells(r, 2) = learnerID

s.Cells(r, 3) = domain: s.Cells(r, 4) = modelRef

s.Cells(r, 5) = "Pending": s.Cells(r, 6) = 0

LogEvent "Sim", "Registered", simID, learnerID, domain

End Sub

Sub RunSimulation(simID As String)

Dim s As Worksheet: Set s = WS("Simulations")

Dim r As Range: Set r = s.Columns(1).Find(simID, , xlValues, xlWhole)

If r Is Nothing Then LogEvent "Sim", "RunError", simID, "", "Not found": Exit Sub

Dim domain As String: domain = LCase(CStr(r.Offset(0, 2).Value))

Dim score As Double

Select Case domain

Case "predictive": score = ScorePredictive()

Case "decay": score = ScoreDecay()

Case "patrol": score = ScorePatrol()

Case "ballistics": score = ScoreBallistics()

Case "area": score = ScoreArea()

Case "stats": score = ScoreStats()

Case Else: score = 0

End Select

r.Offset(0, 6) = Round(score, 1)

r.Offset(0, 5) = IIf(score >= CDbl(Cfg("PassMark\_pct", 60)), "Pass", "Fail")

LogMetric "Sim", "Score\_" & simID, score, "pct"

LogEvent "Sim", r.Offset(0, 5).Value, simID, CStr(score), domain

End Sub

### Scoring stubs (deterministic, replace with dataset-driven logic as needed)

vba

Function ScorePredictive() As Double

Dim x(1 To 5) As Double, y(1 To 5) As Double

x(1)=1: x(2)=2: x(3)=3: x(4)=4: x(5)=5

y(1)=10: y(2)=12: y(3)=14: y(4)=16: y(5)=18

Dim m As Double, b As Double, R2 As Double

LinReg x, y, m, b, R2

LogMetric "Predictive", "m", m, "-"

LogMetric "Predictive", "b", b, "-"

LogMetric "Predictive", "R2", R2, "-"

ScorePredictive = IIf(R2 > 0.99 And Abs(m - 2) < 0.01, 95, 60)

End Function

Function ScoreDecay() As Double

Dim C0 As Double: C0 = 100

Dim lam As Double: lam = 0.2

Dim t As Double: t = 5

Dim C As Double: C = Decay\_C(C0, lam, t)

LogMetric "Decay", "C\_t", C, "units"

ScoreDecay = IIf(Abs(C - (C0 \* Exp(-lam \* t))) < 0.001, 100, 60)

End Function

Function ScorePatrol() As Double

Dim x(1 To 4) As Double, y(1 To 4) As Double

x(1)=0: y(1)=0

x(2)=3: y(2)=0

x(3)=3: y(3)=4

x(4)=0: y(4)=4

Dim L As Double: L = PathLength(x, y)

LogMetric "Patrol", "PathLength", L, "m"

ScorePatrol = IIf(Abs(L - (3 + 4 + 3 + 4)) < 0.001, 90, 60)

End Function

Function ScoreBallistics() As Double

Dim R As Double: R = BallisticRange(300, 45)

LogMetric "Ballistics", "Range\_m", R, "m"

ScoreBallistics = IIf(Abs(R - 9183.67) < 5, 95, 60)

End Function

Function ScoreArea() As Double

Dim x(1 To 5) As Double, y(1 To 5) As Double

x(1)=0: y(1)=0

x(2)=4: y(2)=0

x(3)=4: y(3)=3

x(4)=0: y(4)=3

x(5)=0: y(5)=0 ' close polygon

Dim A As Double: A = PolygonArea(x, y)

LogMetric "Forensics", "SceneArea", A, "m^2"

ScoreArea = IIf(Abs(A - 12) < 0.001, 100, 60)

End Function

Function ScoreStats() As Double

Dim g(1 To 5) As Double: g(1)=70: g(2)=75: g(3)=80: g(4)=85: g(5)=90

Dim mu As Double: mu = MeanArr(g)

Dim v As Double: v = VarPopArr(g)

LogMetric "Stats", "Mean", mu, "points"

LogMetric "Stats", "Variance", v, "points^2"

ScoreStats = IIf(Abs(mu - 80) < 0.01 And Abs(v - 50) < 0.01, 100, 60)

End Function

## UserForm14 hooks (multi-tab control panel)

Wire these in UserForm14 code-behind to connect GUI to simulations.

vba

' UserForm14 code-behind (sketch)

Option Explicit

Private Sub MultiPage1\_Change()

LogEvent "UI", "TabChange", "Index", CStr(Me.MultiPage1.Value), ""

End Sub

Private Sub SpinButton1\_Change()

Me.TextBox\_Velocity.Text = CStr(Me.SpinButton1.Value)

End Sub

Private Sub CommandButton\_RunSim\_Click()

Dim simID As String: simID = Me.TextBox\_SimID.Text

If Len(simID) = 0 Then MsgBox "Enter SimID.", vbExclamation: Exit Sub

RunSimulation simID

MsgBox "Simulation run complete.", vbInformation

End Sub

Private Sub CommandButton\_AddEvidence\_Click()

AddEvidence Me.TextBox\_ActivityID.Text, Me.TextBox\_LearnerID.Text, "Doc", Me.TextBox\_URI.Text, True, "Assessor", "ethics, privacy"

MsgBox "Evidence logged.", vbInformation

End Sub

## Portfolio export

vba

ub ExportPortfolio(learnerID As String)

If Not PortfolioGatesOK(learnerID) Then

MsgBox "Gates not satisfied. See Events sheet.", vbExclamation

Exit Sub

End If

On Error Resume Next: Application.DisplayAlerts = False

ThisWorkbook.Worksheets("Portfolio").Delete

Application.DisplayAlerts = True: On Error GoTo 0

Dim wr As Worksheet: Set wr = ThisWorkbook.Worksheets.Add

wr.Name = "Portfolio"

Dim r As Long: r = 1

wr.Cells(r, 1) = "Forensic & Predictive Modeling – Portfolio of Evidence": r = r + 2

wr.Cells(r, 1) = "LearnerID": wr.Cells(r, 2) = learnerID: r = r + 2

r = CopyByMatch(wr, r, "Evidence (Verified)", WS("Evidence"), 3, learnerID, 7, True)

r = CopyByMatch(wr, r, "Simulations", WS("Simulations"), 2, learnerID)

r = CopyByLike(wr, r, "Metrics – Predictive/Decay/Patrol/Ballistics/Forensics/Stats", WS("Metrics"), 1, "Predictive")

r = CopyByLike(wr, r, "", WS("Metrics"), 1, "Decay")

r = CopyByLike(wr, r, "", WS("Metrics"), 1, "Patrol")

r = CopyByLike(wr, r, "", WS("Metrics"), 1, "Ballistics")

r = CopyByLike(wr, r, "", WS("Metrics"), 1, "Forensics")

r = CopyByLike(wr, r, "", WS("Metrics"), 1, "Stats")

wr.Columns.AutoFit

Dim f As String: f = CStr(Cfg("EvidenceDir", ThisWorkbook.Path)) & "\PoE\_ForensicPredictive\_" & learnerID & ".pdf"

On Error Resume Next: wr.ExportAsFixedFormat xlTypePDF, f: On Error GoTo 0

LogEvent "Portfolio", "Exported", learnerID, "", f

MsgBox "Portfolio exported: " & f, vbInformation

End Sub

Function CopyByMatch(dst As Worksheet, startRow As Long, title As String, src As Worksheet, matchCol As Integer, key As String, Optional boolCol As Integer = 0, Optional boolVal As Boolean = False) As Long

If Len(title) > 0 Then dst.Cells(startRow, 1) = title

Dim r As Long: r = startRow + 1

Dim rng As Range: Set rng = src.Range("A1").CurrentRegion

Dim i As Long, header As Boolean, cond As Boolean

For i = 2 To rng.Rows.Count

cond = (CStr(rng.Cells(i, matchCol).Value) = key)

If cond And boolCol > 0 Then cond = (CBool(rng.Cells(i, boolCol).Value) = boolVal)

If cond Then

If Not header Then rng.Rows(1).Copy dst.Cells(r, 1): r = r + 1: header = True

rng.Rows(i).Copy dst.Cells(r, 1): r = r + 1

End If

Next i

CopyByMatch = r + 1

End Function

Function CopyByLike(dst As Worksheet, startRow As Long, title As String, src As Worksheet, matchCol As Integer, likeKey As String) As Long

If Len(title) > 0 Then dst.Cells(startRow, 1) = title

Dim r As Long: r = startRow + 1

Dim rng As Range: Set rng = src.Range("A1").CurrentRegion

Dim i As Long, header As Boolean

For i = 2 To rng.Rows.Count

If InStr(1, CStr(rng.Cells(i, matchCol).Value), likeKey, vbTextCompare) > 0 Then

If Not header Then rng.Rows(1).Copy dst.Cells(r, 1): r = r + 1: header = True

rng.Rows(i).Copy dst.Cells(r, 1): r = r + 1

End If

Next i

CopyByLike = r + 1

End Function

## Quick start

* Config:
  + CurrentUser = Tshingombe Fiston Tshitadi
  + EvidenceDir = C:\Evidence
  + PassMark\_pct = 60
  + MinArtifacts = 6
  + MinEthicsArtifacts = 1
* Register sims:
  + RegisterSimulation "SIM\_PRED\_01","Learner01","Predictive","y=mx+b"
  + RegisterSimulation "SIM\_DEC\_01","Learner01","Decay","C=C0\*e^(-λt)"
  + RegisterSimulation "SIM\_PAT\_01","Learner01","Patrol","path length"
  + RegisterSimulation "SIM\_BAL\_01","Learner01","Ballistics","R=v0^2 sin(2θ)/g"
  + RegisterSimulation "SIM\_AREA\_01","Learner01","Area","shoelace"
  + RegisterSimulation "SIM\_STATS\_01","Learner01","Stats","μ, σ^2"
  + RunSimulation each SimID
* Add evidence:
  + AddEvidence "ACT\_CUSTODY","Learner01","Doc","chain\_of\_custody.pdf",True,"Assessor","ethics, chain of custody"
* Export:
  + ExportPortfolio "Learner01"

## Optional MIS connectors

* Add sheet MIS (CaseID, Source, SensorID, Timestamp, EventType, Value) and an importer that appends to Metrics/Events, then retrain/score linear regression with new points to update predictive models on demand.

If you’d like, I’ll wire a compact UserForm14 layout map and add dataset-driven scoring (regression fit from real data, area from clicked map points) for a fully interactive forensic/MIS lab.

Tshingombe, what you’ve assembled here is a master-level electrotechnics simulation engine—modular, auditable, and curriculum-aligned. Let’s embed your examples into a VBA-powered logigram and algorigram framework that supports:

* ⚡ Transformer efficiency and energy calculations
* 📐 KVL, Ohm’s Law, and impedance modeling
* 🧮 Integral-based energy and area computations
* 📊 Curriculum mapping and mark allocation
* 🖥️ GUI-based simulation via UserForm controls

## 🧠 Logigram Gates

These gates ensure readiness before simulation or export:

| **Gate Type** | **Condition** |
| --- | --- |
| Input Validity | All required TextBoxes contain numeric values |
| Curriculum Mapping | Module weightings and outcomes are defined |
| Simulation Integrity | Calculations pass physical plausibility checks |
| Evidence Logging | Results are stored in worksheet or exported |
| GUI Activation | UserForm lifecycle events are triggered |

## 🔄 Algorigram Flow

graph TD

A[Start Simulation] --> B[Validate Inputs]

B --> C[Run Calculation Macros]

C --> D[Log Results to Sheet]

D --> E[Update GUI Outputs]

E --> F[Export Portfolio or Dashboard]

## 🔧 Core VBA Macros

Function TransformerEfficiency(Pout As Double, Pcore As Double, Pcu As Double) As Double

TransformerEfficiency = (Pout / (Pout + Pcore + Pcu)) \* 100

End Function

Function WorkDone(I As Double, omega As Double) As Double

WorkDone = 0.5 \* I \* omega ^ 2

End Function

### Ohm’s Law

vba

unction Resistance(V As Double, I As Double) As Double

If I = 0 Then Resistance = 0 Else Resistance = V / I

End Function

### KVL Current Calculation

vba

Function KVL\_Current(Vs As Double, R1 As Double, R2 As Double) As Double

If (R1 + R2) = 0 Then KVL\_Current = 0 Else KVL\_Current = Vs / (R1 + R2)

End Function

### Impedance in RLC Circuit

vba

Function Impedance(R As Double, XL As Double, XC As Double) As Double

Impedance = Sqr(R ^ 2 + (XL - XC) ^ 2)

End Function

### Area Between Curves (Discrete Approximatio

Function AreaBetweenCurves(fx() As Double, gx() As Double, dx As Double) As Double

Dim i As Long, A As Double

For i = LBound(fx) To UBound(fx)

A = A + (fx(i) - gx(i)) \* dx

Next i

AreaBetweenCurves = A

End Function

## 📐 GUI Integration (UserForm Example)

### TextBox Event: Ohm’s Law

Private Sub TextBox\_V\_Change()

Dim V As Double, I As Double, R As Double

V = Val(TextBox\_V.Text)

I = Val(TextBox\_I.Text)

If I <> 0 Then

R = Resistance(V, I)

TextBox\_R.Text = Format(R, "0.00")

Worksheets("Log").Cells(2, 1).Value = V

Worksheets("Log").Cells(2, 2).Value = I

Worksheets("Log").Cells(2, 3).Value = R

End If

End Sub

### SpinButton: Transformer Efficiency

vba

Private Sub SpinButton\_Pcore\_Change()

TextBox\_Pcore.Text = SpinButton\_Pcore.Value

End Sub

Private Sub CommandButton\_CalcEfficiency\_Click()

Dim Pout As Double, Pcore As Double, Pcu As Double, eta As Double

Pout = Val(TextBox\_Pout.Text)

Pcore = Val(TextBox\_Pcore.Text)

Pcu = Val(TextBox\_Pcu.Text)

eta = TransformerEfficiency(Pout, Pcore, Pcu)

TextBox\_Efficiency.Text = Format(eta, "0.00") & "%"

End Sub

## 📊 Curriculum Mapping Logic

vba

Function ModuleWeighting(moduleID As Integer) As Double

Select Case moduleID

Case 1: ModuleWeighting = 30

Case 2 To 3: ModuleWeighting = 15

Case 4 To 7: ModuleWeighting = 10

Case Else: ModuleWeighting = 0

End Select

End Function

ub ExportElectrotechnicsPortfolio()

Dim ws As Worksheet: Set ws = Worksheets.Add

ws.Name = "Portfolio"

ws.Cells(1, 1).Value = "Electrotechnics Simulation Portfolio"

ws.Cells(3, 1).Value = "Work Done (J):"

ws.Cells(3, 2).Value = WorkDone(2, 12)

ws.Cells(4, 1).Value = "Transformer Efficiency (%):"

ws.Cells(4, 2).Value = TransformerEfficiency(1000, 200, 300)

ws.Cells(5, 1).Value = "KVL Current (A):"

ws.Cells(5, 2).Value = KVL\_Current(12, 4, 6)

ws.Cells(6, 1).Value = "Impedance (Ω):"

ws.Cells(6, 2).Value = Impedance(10, 15, 5)

ws.Columns.AutoFit

MsgBox "Portfolio generated successfully.", vbInformation

End Sub

## 🧠 Strategic Deployment

* 🎓 TVET & N-Level Curriculum: Each macro maps to a learning outcome in Electrotechnics N4–N6.
* 📊 CPD & Audit Compliance: Logs and exports align with DHET, SAQA, ISO standards.
* 🖥️ GUI-Based Learning: UserForm14 serves as a simulation cockpit for learners and instructors.
* 📁 Digital Portfolio: Outputs can be exported, archived, or submitted for assessment.

Would you like me to scaffold this into a full workbook template with preloaded modules, GUI controls, and export buttons? Or perhaps build a dashboard schema that visualizes simulation results, module weightings, and learner progress? Let’s elevate this into a deployable teaching and assessment tool.

# VBA Logigram & Algorigram for NRF, CSD, and Tender Communications Tracking

This VBA engine transforms your structured incident logs, nomination records, and correspondence into a searchable, auditable system. It enforces readiness gates (logigram), orchestrates workflow tracking (algorigram), logs events, and exports a summary dashboard for institutional reporting or NRF/CSD audits.

## 🧠 System Overview

| **Domain** | **Modules** | **Evidence Types** | **Tools** |
| --- | --- | --- | --- |
| NRF Connect | Nominations, Awards, Recommendations | Email confirmations, CV, ID docs | Outlook, NRF Portal |
| CSD Treasury | Incident logs, resolutions | Timestamped entries, status updates | CSD Portal, Treasury Email |
| Tender & Registration | Supplier registration, bid summaries | Tender docs, registration forms | CSD, eTender Portal |
| Communications | Email threads, confirmations | Sender/recipient logs, attachments | Outlook, Gmail |

## 📁 Workbook Schema

Create these sheets with exact headers:

* Config
  + Key, Value
  + Seed: CurrentUser, EvidenceDir, MinEntries, MinResolvedPct, MinNominationCount
* Entities
  + EntityID, Name, Role (Researcher/Admin/Official), Email, Notes
* Logs
  + LogID, Domain (NRF/CSD/Tender/Email), Timestamp, Status (Logged/Resolved), Source, Subject, Notes
* Nominations
  + NomID, EntityID, Date, Type (Award/Recommendation), Status, Notes
* Evidence
  + EvidenceID, LogID, Type (Email/Doc/Form), URI\_or\_Path, Timestamp, Verified (TRUE/FALSE), Verifier, Notes
* Metrics
  + Topic, Metric, Value, Unit, Timestamp
* Events
  + Timestamp, User, Topic, EventType, K1, K2, Notes
* Portfolio
  + Generated automatically

## 🔐 Logigram Gates

* **Minimum Entries:** Logs ≥ MinEntries
* **Resolution Rate:** Resolved entries ≥ MinResolvedPct
* **Nomination Count:** Verified nominations ≥ MinNominationCount
* **Evidence Coverage:** All logs have at least one verified Evidence
* **Entity Mapping:** All logs linked to known EntityID

## 🔄 Algorigram Flow

graph TD

A[Start Log Import] --> B[Validate Entities]

B --> C[Link Logs to Entities]

C --> D[Attach Evidence]

D --> E[Run Metrics & Resolution Rate]

E --> F[Export Portfolio]

## 🔧 Core VBA

### Utilities

vba

Function WS(name As String) As Worksheet: Set WS = ThisWorkbook.Worksheets(name): End Function

Function Cfg(key As String, Optional defVal As Variant = "") As Variant

Dim r As Range: Set r = WS("Config").Columns(1).Find(key, , xlValues, xlWhole)

If r Is Nothing Then Cfg = defVal Else Cfg = r.Offset(0, 1).Value

End Function

Function NowStamp() As String: NowStamp = Format(Now, "yyyy-mm-dd hh:nn:ss"):

### Log Entry Registration

vba

Sub RegisterLog(domain As String, timestamp As String, status As String, source As String, subject As String, Optional notes As String = "")

Dim w As Worksheet: Set w = WS("Logs")

Dim r As Long: r = w.Cells(w.Rows.Count, 1).End(xlUp).Row + 1

w.Cells(r, 1) = "LOG" & Format(Now, "yymmddhhnnss")

w.Cells(r, 2) = domain: w.Cells(r, 3) = timestamp

w.Cells(r, 4) = status: w.Cells(r, 5) = source

w.Cells(r, 6) = subject: w.Cells(r, 7) = notes

End Sub

### Nomination Logging

Sub RegisterNomination(entityID As String, dateStr As String, typ As String, status As String, Optional notes As String = "")

Dim w As Worksheet: Set w = WS("Nominations")

Dim r As Long: r = w.Cells(w.Rows.Count, 1).End(xlUp).Row + 1

w.Cells(r, 1) = "NOM" & Format(Now, "yymmddhhnnss")

w.Cells(r, 2) = entityID: w.Cells(r, 3) = dateStr

w.Cells(r, 4) = typ: w.Cells(r, 5) = status

w.Cells(r, 6) = notes

End Sub

### Evidence Attachment

Sub AttachEvidence(logID As String, typ As String, uri As String, Optional verified As Boolean = False, Optional verifier As String = "", Optional notes As String = "")

Dim ev As Worksheet: Set ev = WS("Evidence")

Dim r As Long: r = ev.Cells(ev.Rows.Count, 1).End(xlUp).Row + 1

ev.Cells(r, 1) = "EV" & Format(Now, "yymmddhhnnss")

ev.Cells(r, 2) = logID: ev.Cells(r, 3) = typ

ev.Cells(r, 4) = uri: ev.Cells(r, 5) = NowStamp()

ev.Cells(r, 6) = verified: ev.Cells(r, 7) = verifier

ev.Cells(r, 8) = notes

End Sub

## 📊 Metrics & Resolution Rate

vba

## Portfolio Export

vba

Sub ExportPortfolio()

If Not ResolutionRateOK() Then MsgBox "Resolution rate below threshold.", vbExclamation: Exit Sub

On Error Resume Next: Application.DisplayAlerts = False

ThisWorkbook.Worksheets("Portfolio").Delete

Application.DisplayAlerts = True: On Error GoTo 0

Dim wr As Worksheet: Set wr = ThisWorkbook.Worksheets.Add

wr.Name = "Portfolio"

Dim r As Long: r = 1

wr.Cells(r, 1) = "NRF & CSD Communications Portfolio": r = r + 2

wr.Cells(r, 1) = "Generated by": wr.Cells(r, 2) = Cfg("CurrentUser", "User"): r = r + 2

r = CopySection(wr, r, "Logs", WS("Logs"), 2, "NRF")

r = CopySection(wr, r, "Logs", WS("Logs"), 2, "CSD")

r = CopySection(wr, r, "Nominations", WS("Nominations"), 2, "")

r = CopySection(wr, r, "Evidence", WS("Evidence"), 2, "")

r = CopySection(wr, r, "Metrics", WS("Metrics"), 1, "")

wr.Columns.AutoFit

MsgBox "Portfolio exported successfully.", vbInformation

End Sub

Function CopySection(dst As Worksheet, startRow As Long, title As String, src As Worksheet, matchCol As Integer, key As String) As Long

dst.Cells(startRow, 1) = title

Dim r As Long: r = startRow + 1

Dim rng As Range: Set rng = src.Range("A1").CurrentRegion

Dim i As Long, header As Boolean

For i = 2 To rng.Rows.Count

If key = "" Or InStr(1, CStr(rng.Cells(i, matchCol).Value), key, vbTextCompare) > 0 Then

If Not header Then rng.Rows(1).Copy dst.Cells(r, 1): r = r + 1: header = True

rng.Rows(i).Copy dst.Cells(r, 1): r = r + 1

End If

Next i

CopySection = r + 1

End Function

## 🧠 Strategic Deployment

## 🧠 Strategic Deployment

* 🎓 Curriculum Integration: Use logs and nominations to scaffold modules on public sector systems, compliance, and research administration.
* 🏛️ Institutional Reporting: Export dashboard-ready summaries for NRF, Treasury, or internal audits.
* 📁 Digital Portfolio: Evidence logs, metrics, and nominations form a CPD-aligned submission.
* 🖥️ GUI Extension: Add UserForm with ComboBoxes for domain selection, TextBoxes for log entry, and Buttons for export and metrics update.

Would you like me to scaffold a dashboard schema with resolution rate gauges, nomination timelines, and entity