?? 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 ? 8? ? 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 Bool
ean) As Boolean
    ValidateProcedure = evidenceLabeled And storedProperly And custodyIntact
End Function
```

?? Deployment Strategy

"   Integrate these modules into a multi-sheet Excel workbook with:
o   Dashboard for input/output
o   Logigram sheet for procedural flow
o   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 ballist
ic incident with decay tracking and courtroom validation? We can modularize it into a certifiable trai
ning tool.

Workbook structure and named ranges

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

sheets

"   Dashboard
o Inputs: Progress actions, buttons
o   Outputs: Status, next module, warnings
"   Modules

o Columns: A: moduleID , b: moduleName , C: Sequence , D: required (True / False)
"   Progress
o Columns: A: DateTime , b: User , C: moduleID , D: EventType , e: score , f: Duration
"   Quizzes
o Columns: A: moduleID , b: quizName , C: maxScore , D: PassingScore
"   Procurement
o   Columns: A: Field, B: Value (Department, ProjectCode, EstimatedCostPerDay, FundingSource, Resources)
"   Quotation
o   Columns: A: Field, B: Value (FullName, Company, Contact, VAT_BEE, Delivery)
"   Intake
o   Columns: A: Field, B: Value (Name, Institution, Contact, Reason)
"   Workshops
o   Columns: A: ModuleName, B: Type, C: Date, D: Facilitator, E: Room, F: Notes
"   CareerMap
o Columns: A: Position , b: Requirements , C: TimeFrame , D: Mentoring
"   RAndD
o Columns: A: topic , b: FocusArea , C: output , D: status
"   Config
o   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):
o   On "Complete Module": validate sequencing ? log event ? recompute status ? update Dashboard.
o   On "Record Quiz": validate module exists ? log score/time ? recompute module readiness.
o   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:
o   Start Module: Assign macro StartModulePrompt (below).
o   Record Quiz: Assign RecordQuizPrompt.
o   Complete Module: Assign CompleteModulePrompt.
o   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:
o   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
o   Set Required = TRUE for all.
"   Set policy and user:
o Config:
"   CurrentUser = "Tshingombe Fiston Tshitadi"
"   PassingPolicy = 50 (or your desired threshold)
"   Record your existing quiz result:
o   Use Record Quiz button for "Introduction to AI", score 3, max defined on Quizzes, duration 39 seco
nds.
"   Generate your submission pack:
o   Fill Procurement, Quotation, Intake, Workshops, CareerMap, RAndD.

```
o   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 dat
a quality.
"   Validation rules: Data Validation lists for Departments (Education, Career Center, ICT) and SETA s
ectors (MERSETA, TETA, HASTA, SASSETA).
"   KPI tiles on Dashboard: Completed count, next module, average quiz score, procurement readiness fl
ag.
"   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
o   Inputs: Buttons, current session status
o   Outputs: Next step, safety status, calibration status, data quality flags
"   Assets
o Columns: A: assetID , b: Category , C: Model , D: SerialNo , e: Location , f: status
"   Sessions
o Columns: A: sessionID , b: Trainee , C: Module , D: StartTime , e: EndTime , f: Facilitator , g: sta
tus
"   Safety
o Columns: A: ChecklistItem , b: required , C: completed , D: notes
"   Calibration
o   Columns: A: SensorID, B: Type, C: Date, D: ReadingKnown, E: ForceKnownN, F: Scale, G: Offset
"   Measurements
o Columns: A: sessionID , b: testType , C: X_Pos_m , D: LoadType , e: LoadValue , f: DynoLeft_N , g: D
ynoRight_N , h: dial1_mm , i: Dial2_mm , j: Temp_C
"   Analysis
o Columns: A: sessionID , b: Computation , C: Param1 , D: Param2 , e: Param3 , f: result
"   Procurement
o   Columns: A: Field, B: Value (Department, ProjectCode, EstimatedCostPerDay, FundingSource, Resource
s)
"   Config
o   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 calibratio
n.
"   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 D
ouble)
    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 offs
et ~ 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 Do
uble) 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", 200000000000#))

    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, ses
sionID 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:
o Add Measurements columns: Frequency_Hz, Level_dB, Output_Vpp, Thresholds.
o Add Calibration type: "audio" with Scale in dB per volt.
o Add Analysis: hearing curve plotting (store session computations in Analysis).
" Procurement integration:
o Use the procurement sheet to auto-validate session resource costs.
o 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
o A: StationID , b: role (Teacher / Student), C: Hostname , D: IP , e: Connectivity (LAN / WiFi), f: status (online / Offline), g: DL_WORKSPACE (Yes / No)
"   ModuleCatalog
o   A: ModuleID, B: Name (CIMSIM/IoT/DATA/CYBER), C: RequiredAssets (comma list), D: PrereqModules (comma list), E: Enabled (TRUE/FALSE)
"   DeviceRegistry
o   A: DeviceID, B: Type (PLC/DevIoT/Sensor/Actuator), C: Model, D: PortMap, E: Protocols (MQTT/Modbus), F: AssignedStation, G: Status
"   ScenarioBook
o   A: ScenarioID, B: ModuleID, C: Name, D: Objective, E: Steps (CSV), F: PassCriteria
"   Events
o A: timestamp , b: User , C: scenarioID , D: EventType , e: Payload1 , f: Payload2 , g: notes
"   Measurements
o A: scenarioID , b: metric , C: Value , D: Unit , e: SourceDevice , f: timestamp
"   Config
o   A: Key, B: Value (CurrentUser, MinStudents, RequireDL_WORKSPACE, MQTT_Topic_OnOff, SafetyPolicy, EvidenceDir)
"   Safety
o A: ChecklistItem , b: required (True / False), C: completed (True / False), D: notes
"   Portfolio
o   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:
o   Teacher station Online and DL_WORKSPACE = Yes.
o   Count(Students Online) ? MinStudents.
o   Required assets for the selected module are Status = Available/Online.
"   Safety gating:
o   All Required items in Safety are Completed = TRUE.
"   Module prerequisites:
o   All PrereqModules are Enabled and previously run (in Events as Completed).
"   Scenario approval:
o   Scenario steps are executable with current Devices and Protocols.
Algorigram flows
"   StartScenario:
o   Validate infrastructure ? Validate safety ? Check module prereqs ? Lock resources ? Log Started.
"   RunStep:
o   Execute step dispatcher (CIMSIM | IoT | DATA | CYBER) ? Capture measurement(s) ? Log checkpoint.
"   EvaluateScenario:
o   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 Stri
ng)
    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: E
xit 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:=x
lWhole)
    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, "pa
ssed", "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, match
Col 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
o   CurrentUser = Tshingombe Fiston Tshitadi
o MinStudents = 1
o RequireDL_WORKSPACE = True
o MQTT_Topic_OnOff = OnOff
o   EvidenceDir = C:\Evidence (or your path)
"   StationRegistry
o   One Teacher row: Status=Online, DL_WORKSPACE=Yes
o   One or more Student rows: Status=Online
"   DeviceRegistry
o PLC_1(plc, …, status = online)
o DevIoT_1(DevIoT, Protocols = MQTT, AssignedStation = Student1, status = online)
o   Sensors/Actuators as needed (IR_1, RGB_1, Lamp_1)
"   ModuleCatalog
o   CIMSIM: RequiredAssets = PLC_1,IR_1,RGB_1
o   IOT: RequiredAssets = DevIoT_1,Lamp_1
o Data: RequiredAssets = (blank)
o Cyber: RequiredAssets = (blank)
o   Set Enabled=TRUE
"   ScenarioBook (examples)
o   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)
o   SCN_IOT_01 | IOT | Switch-Lamp via MQTT | Switch controls remote lamp | publish_switch:ON,subscribe_lamp:ON | LampState==1
o   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
o A: StationID , b: role (Teacher / Student), C: Hostname , D: IP , e: Link (LAN / WiFi), f: status (online / Offline), g: WorkspaceOK (Yes / No)
"   DeviceRegistry
o   A: DeviceID, B: Domain (IoT/DM91/HYD/HVAC/CHEM/REN), C: Type, D: Model, E: Protocols, F: AssignedStation, G: Status (Online/InUse/Available)
"   ScenarioBook
o   A: ScenarioID, B: Domain, C: Name, D: Objective, E: StepsCSV, F: PassCriteria, G: RequiredDevicesCSV, H: SafetyChecklistCSV
"   Safety
o A: item , b: required (True / False), C: completed (True / False), D: notes
"   Measurements
o A: scenarioID , b: metric , C: Value , D: Unit , e: source , f: timestamp
"   Events
o A: timestamp , b: User , C: scenarioID , D: EventType , e: k1 , f: k2 , g: notes
"   Config
o A: key , b: Value
o keys: CurrentUser , MinStudents, RequireWorkspace, EvidenceDir, MQTT_Topic, DM91_SparkThreshold_kV, WaterHammer_MaxBar, HVAC_TempSet_C, PV_STC_W, Wind_Rated_W
"   Portfolio
o   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:
o   Teacher station Online and (if required) WorkspaceOK=Yes.
o   Students Online ? MinStudents.
"   Safety gate:
o   Every Safety item listed in ScenarioBook.H (if Required) must be Completed.
"   Device availability:
o   All ScenarioBook.G devices found in DeviceRegistry and Status=Online/Available.
"   Domain-specific prechecks:
o   IoT: MQTT topic configured.
o   DM91: Lab power OK, panel interlocks ready.
o   Hydraulics: Bench present, compressed air available device flag.
o   HVAC: Sensor calibration present (temp/RH/anemometer).
o   Chemical: Reactor sensors online (T, flow, cond).
o   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 = "", Opt
ional 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: Ex
it 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 DevicesRead
y = 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", vbExclam
ation: 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), "b
ool", "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 A
s 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:
- o CurrentUser = Tshingombe Fiston Tshitadi
- o MinStudents = 1
- o RequireWorkspace = True
- o EvidenceDir = C:\Evidence
- o MQTT_Topic = OnOff
- o DM91_SparkThreshold_kV = 12
- o WaterHammer_MaxBar = 8
- o HVAC_TempSet_C = 22
- o PV_STC_W = 200, Wind_Rated_W = 300

" ScenarioBook examples:
- o 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
- o 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
- o SCN_HYD_01 | HYD | Water hammer safe op | Limit peak | pulse_close:fast, measure_p_peekbar:6.8 | PeakPressureBar<=8 | Transducer_A,Bench,AirSupply
- o 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
- o SCN_REN_01 | REN | PV curve point | Ratio check | pv_set_irr:800, pv_measure:165 | PV_Ratio>=0.7 | PV_Emu

" Run:
- o Press RunScenarioPrompt and enter ScenarioID.
- o 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
o   A: Field, B: Value (Name, ID, Trade Level, Workplace, Assessor)
"   Modules
o A: moduleID , b: moduleName , C: Sequence , D: required (True / False), e: domain (ELEC / Road / Fire / aid), f: Enabled (True / False)
"   Safety
o A: ChecklistItem , b: required (True / False), C: completed (True / False), D: domain , e: notes
"   Exercises
o   A: ExerciseID, B: ModuleID, C: Name, D: Objective, E: StepsCSV, F: PassCriteria, G: RequiredToolsCSV, H: RiskTagsCSV
"   Tools
o   A: ToolID, B: Name, C: Status (Available/InUse/Out), D: CalDueDate, E: Domain
"   Events
o A: timestamp , b: User , C: ExerciseID , D: EventType , e: k1 , f: k2 , g: notes
"   Measurements
o A: ExerciseID , b: metric , C: Value , D: Unit , e: source , f: timestamp
"   Config
o A: key , b: Value
o keys: CurrentUser , MinPPE, EvidenceDir, Elec_IsolationTimeout_s, Fire_MAX_Risk, FirstAid_TrainerPresent(True / False)
"   Portfolio
o   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:
o   ELEC: Verified isolation/LOTO and zero-voltage check within Elec_IsolationTimeout_s.
o   FIRE: Risk tag sum must be ? Fire_MAX_Risk (or mitigations documented).
o   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 IsModuleComple
ted = 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: E
xit Function
    If Not ToolsReady(tools) Then MsgBox "Tools unavailable or calibration expired", vbExclamation: Ex
it 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)): Ex
it 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 M?
            RecordMetric exID, "IR_MOhm", val(param), "M?", "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", "D
erived"
        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 A
s 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 A
s 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:
o   CurrentUser = Tshingombe Fiston Tshitadi
o MinPPE = 3
o   EvidenceDir = C:\Evidence
o Elec_IsolationTimeout_s = 300
o Fire_MAX_Risk = 5
o FirstAid_TrainerPresent = True
"   Modules:
o   1 | Electrical Safety & LOTO | Seq 1 | TRUE | ELEC | TRUE
o   2 | Generators & Relays | Seq 2 | TRUE | ELEC | TRUE
o   3 | Road Safety & Signals | Seq 3 | FALSE | ROAD | TRUE
o   4 | Fire Safety | Seq 4 | TRUE | FIRE | TRUE
o   5 | First Aid & CPR | Seq 5 | TRUE | AID | TRUE
"   Tools:
o   T_DMM, T_Megger, T_RelayBench, T_CO2_Ext, T_DryPowder, T_FirstAidManikin (all Available; set CalDueDate accordingly)
"   Exercises:
o   EX_ELEC_01 | 1 | LOTO + ZVV | Isolate and verify zero voltage | loto_apply, zvv_test:0.3 | ZVV_V<=1 | T_DMM | PPE,InsulatedGloves
o   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 |
o   EX_ROAD_01 | 3 | Signals quiz | Identify signals | signal_meaning:Red=Stop, sign_identify:Triangular=Warning, quiz_score:85 | RoadQuizScore>=70 | | HiVis
o   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
o   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
o   A: ModuleID, B: Name, C: Sequence, D: Required (TRUE/FALSE), E: Domain (FIVES/TOOLS/LIFT/CUT/MARK), F: Enabled (TRUE/FALSE)
"   Exercises

o   A: ExerciseID, B: ModuleID, C: Name, D: Objective, E: StepsCSV, F: PassCriteria, G: RequiredItemsCSV, H: SafetyChecklistCSV

"   Safety

o A: item , b: required (True / False), C: completed (True / False), D: domain , e: notes

"   Inventory

o   A: ItemID, B: Category (Tool/PPE/Fixture), C: StandardCode (e.g., BIS 3650), D: Status (Available/InUse/Out), E: CalDueDate, F: Domain

"   Standards

o   A: Code, B: Title, C: Domain, D: Notes (e.g., BIS 3650 Combination Pliers)

"   Measurements

o A: ExerciseID , b: metric , C: Value , D: Unit , e: source , f: timestamp

"   Events

o A: timestamp , b: User , C: ExerciseID , D: EventType , e: k1 , f: k2 , g: notes

"   Config

o A: key , b: Value

o   Keys: CurrentUser, EvidenceDir, MinPPE, Lifting_MaxMass_kg, Cut_MaxKerf_mm, Mark_MaxError_mm, 5S_MinScore, Tools_MinScore

"   Portfolio

o   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:

o   FIVES: Minimum PPE present; 5S audit will compute score target.

o   TOOLS: Tool standard codes must be known in Standards table; tool care checks recorded.

o   LIFT: Mass ? Lifting_MaxMass_kg; path clear; proper devices selected.

o   CUT: Blade type/pitch aligned to material and thickness; kerf and edge quality recorded.

o   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 IsModuleComple
ted = 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 = Fals
e: 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: E
xit 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: E
xit Function
    If Not InventoryReady(tools) Then MsgBox "Required items unavailable or expired", vbExclamation: E
xit 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)): Ex
it 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 A
s 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:
o   CurrentUser: Tshingombe Fiston Tshitadi
o   EvidenceDir: C:\Evidence
o   MinPPE: 3
o   Lifting_MaxMass_kg: 50
o   Cut_MaxKerf_mm: 1.2
o   Mark_MaxError_mm: 0.5
o   5S_MinScore: 70
o   Tools_MinScore: 70
"   Modules:
o   1 | 5S Workplace Audit | 1 | TRUE | FIVES | TRUE
o   2 | Tool Standards & Care | 2 | TRUE | TOOLS | TRUE
o   3 | Lifting & Handling | 3 | TRUE | LIFT | TRUE
o   4 | Hacksaw Cutting | 4 | TRUE | CUT | TRUE
o   5 | Marking & Measurement | 5 | TRUE | MARK | TRUE
"   Standards (examples):
o   BIS 3650 | Combination Pliers | TOOLS | Cutting, twisting, gripping
o   BIS 844 | Screwdrivers | TOOLS | Tightening/loosening screws
o   BIS 5579 | Neon Tester | TOOLS | Checking live voltage
"   Inventory (examples):
o   T_PLIERS | Tool | BIS 3650 | Available | 2026-12-31 | TOOLS
o   T_SCREW | Tool | BIS 844 | Available | 2027-01-01 | TOOLS
o   PPE_GLOVES | PPE | - | Available | | FIVES
o   L_WINCH | Fixture | - | Available | | LIFT
o   CUT_SAW | Tool | - | Available | 2026-05-01 | CUT
o   MARK_RULE300 | Tool | - | Available | 2028-01-01 | MARK
"   Safety (examples):
o   PPE | TRUE | TRUE | FIVES | Gloves, goggles, boots
o   ClearPath | TRUE | TRUE | LIFT | Barriers placed
o   FireExt Nearby | TRUE | TRUE | CUT | CO2 or DP extinguisher
o   EyeProtection | TRUE | TRUE | MARK | Safety glasses
"   Exercises:
o   EX_5S_01 | 1 | 5S Quick Kaizen | Improve bench area | seiri:12,seiton:38,seiso:4,seiketsu:3,shitsu
ke:4,total_score:76 | 5S_TotalScore>=70 | PPE_GLOVES | PPE
o   EX_TOOL_01 | 2 | Tool Standards Check | Verify BIS + care | verify_bis:BIS 3650,inspect_tool:Combi
nationPliers=OK,care_task:LubricateHinge=Done,selection_quiz:85,neon_tester_use:1 | Tools_BIS_Recogniz
ed==1;Tools_QuizScore>=70 | T_PLIERS,T_SCREW | PPE
o   EX_LIFT_01 | 3 | Move Transformer | Safe relocation | assess_mass:42,device_select:Winch,path_clea
r:1,center_gravity:1,corner_roll:1 | Lift_MassOK==1;Lift_PathClear==1;Lift_CG_OK==1 | L_WINCH,PPE_GLOV
```

ES | ClearPath
o   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
o   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
o   A: ModuleID, B: Name, C: Sequence, D: Required (TRUE/FALSE), E: Domain (TOOLS/LIFT/CUT/MARK/CODE), F: Enabled (TRUE/FALSE)
"   Exercises
o   A: ExerciseID, B: ModuleID, C: Name, D: Objective, E: StepsCSV, F: PassCriteria, G: RequiredItemsCSV, H: SafetyChecklistCSV
"   Safety
o A: item , b: required (True / False), C: completed (True / False), D: domain , e: notes
"   Inventory
o   A: ItemID, B: Category (Tool/PPE/Fixture), C: StandardCode (e.g., BIS 2029), D: Status (Available/InUse/Out), E: CalDueDate, F: Domain
"   Standards
o   A: Code (e.g., BIS 2029), B: Title, C: Domain, D: Notes
"   NEC
o A: Section (1 - 20), b: title , C: Focus
"   Events
o A: timestamp , b: User , C: ExerciseID , D: EventType , e: k1 , f: k2 , g: notes
"   Measurements
o A: ExerciseID , b: metric , C: Value , D: Unit , e: source , f: timestamp
"   Config
o A: key , b: Value
o keys: CurrentUser , EvidenceDir, Lifting_MaxMass_kg, Cut_MaxKerf_mm, Mark_MaxError_mm, Tools_MinScore
"   Portfolio
o   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 IsModuleComple
ted = 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 = Fals
e: 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: E
xit 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: E
xit 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)): Ex
it 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 A
s 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:
o   CurrentUser = Tshingombe Fiston Tshitadi
o   EvidenceDir = C:\Evidence
o Lifting_MaxMass_kg = 50
o Cut_MaxKerf_mm = 1.2
o Mark_MaxError_mm = 0.5
o Tools_MinScore = 70
"   Standards:
o   BIS 2029 | Ring Spanner Set | TOOLS | High leverage in tight spaces

Module5 - 45

o   BIS 7993 | Socket Spanner | TOOLS | Deep/narrow locations
o   BIS 6149 | Adjustable Spanner | TOOLS | Versatile
o   BIS 5169 | Hacksaw Frame | CUT | Cutting metals
o   BIS 2594 | Hacksaw Blade | CUT | Cutting metals
o   BIS 4195 | Pincers | TOOLS | Extracting nails
o   BIS 844 | Screwdrivers | TOOLS | Tighten/loosen screws
o   BIS 5579 | Neon Tester | TOOLS | Check live voltage
"   NEC:
o   1 | Scope of NEC | Coverage and applicability
o   7 | Design principles for installations | Load calc, diversity, safety factors
o   9 | Wiring design and construction | Wiring methods, protection
o   13 | Pre-commissioning tests | Inspection and testing
o   14 | Earthing requirements | Earthing/grounding practices
"   Inventory:
o   T_RING | Tool | BIS 2029 | Available | 2027-01-01 | TOOLS
o   T_SOCKET | Tool | BIS 7993 | Available | 2027-01-01 | TOOLS
o   T_HACKSAW | Tool | BIS 5169 | Available | 2026-06-01 | CUT
o   T_BLADE24 | Tool | BIS 2594 | Available | 2026-06-01 | CUT
o   PPE_GLOVES | PPE | - | Available | | TOOLS
"   Modules:
o   1 | Tool Standards & Care | 1 | TRUE | TOOLS | TRUE
o   2 | Lifting & Handling | 2 | TRUE | LIFT | TRUE
o   3 | Hacksaw Cutting | 3 | TRUE | CUT | TRUE
o   4 | Marking & Measurement | 4 | TRUE | MARK | TRUE
o   5 | Standards & NEC | 5 | TRUE | CODE | TRUE
"   Safety:
o   PPE | TRUE | TRUE | TOOLS | Gloves, glasses, boots
o   ClearPath | TRUE | TRUE | LIFT | Barriers placed
o   FireExt Nearby | TRUE | TRUE | CUT | CO2/DP extinguisher
o   EyeProtection | TRUE | TRUE | MARK | Safety glasses
"   Exercises:
o   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
o   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
o   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
o   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
o   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
o keys: CurrentUser , EvidenceDir, MaxRatedVoltage_V, AC_Freq_Hz
"   CapacitorSeries
o A: CapID , b: Capacitance_uF , C: RatedVoltage_V , D: Polarity (NP / P), e: status (ok / Fail)
"   ACDC
o A: Parameter , b: Value , C: Unit , D: source
"   InductiveAC
o A: L_mH , b: F_kHz , C: XL_Ohm , D: I_A , e: Pq_VAR
"   Events
o A: timestamp , b: User , C: topic , D: EventType , e: Detail1 , f: Detail2 , g: notes
"   Measurements
o A: topic , b: metric , C: Value , D: Unit , e: source , f: timestamp
"   Portfolio
o   Generated automatically
?? Logigram Rules
"   Capacitor grouping:
o   Voltage across each capacitor < RatedVoltage_V
o   Polarity must be maintained for polarized types
"   AC waveform:
o   Frequency must match configured AC_Freq_Hz
"   Inductive reactance:
o   XL = 2?fL
o   Reactive power Pq = $I^2 \times XL$
"   Safety:
o   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 = "", O
ptional 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, "?", "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
o A: key , b: Value
o   Keys to seed: CurrentUser, EvidenceDir, LineFreq_Hz, TargetPF, LineVoltage_V, RatedCapVoltage_V
"   Scenarios
o   A: ScenarioID, B: Domain (RLC_PAR/RLC_SER/PF_CORR/MATCH/AUDIO), C: Name, D: Objective, E: StepsCSV, F: PassCriteria
"   Events
o A: timestamp , b: User , C: scenarioID , D: EventType , e: k1 , f: k2 , g: notes
"   Measurements
o A: scenarioID , b: metric , C: Value , D: Unit , e: source , f: timestamp
"   Components
o   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=12?LCf_0 = \frac{1}{2\pi\sqrt{LC}}
"   Quality factor (series): Qs=?0LRQ_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(?L?1?C)Z_s = R + j(\omega L - \frac{1}{\omega C})
"   Parallel admittance: Yp=1R+j(?C?1?L)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:
o   Initial PF: cos??1=PP2+Q2\cos\theta_1 = \frac{P}{\sqrt{P^2 + Q^2}}
o   Target PF: cos??2\cos\theta_2 set by user; required capacitive VARs:
Qc=P?(tan??1?tan??2)Q_c = P \cdot (\tan\theta_1 - \tan\theta_2)
"   Capacitor size:
C=Qc?V2with ?=2?fC = \frac{Q_c}{\omega V^2} \quad \text{with } \omega = 2\pi f
"   Cap RMS current: Ic=?CVI_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, ta
rgetPF 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
o 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, se
t: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, comp
ute | 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:230,5
0, 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=1
```

00e-9, sweep:20,20000,200, compute | Qp_audio>=1;f0_audio_Hz>=150;f0_audio_Hz<=400
Quick Start
"   Config:
o   CurrentUser = Tshingombe Fiston Tshitadi
o   EvidenceDir = C:\Evidence
o LineFreq_Hz = 50
o targetPF = 0.95
o LineVoltage_V = 230
o RatedCapVoltage_V = 450
"   Components: keep for reference; scenario uses inline "set:" values.
"   Run:
o   Press RunScenarioPrompt and enter a ScenarioID (e.g., SCN_PAR_01).
o   Review Events and Measurements for audit and teaching.
o   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
o A: key , b: Value
o keys: CurrentUser , EvidenceDir, SoilThermalResist_Km_W, AmbientTemp_C, MaxLoadTemp_C, CopperResist_ohm_m, ConductorCSA_mm2, RouteDerating_pct, LayingDepth_m
"   CableCatalog
o   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
o 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
o A: item , b: required (True / False), C: completed (True / False), D: routeID , e: notes
"   CableRuns
o A: RunID , b: routeID , C: cableID , D: Phases (1 / 3), e: Armour (Yes / No), f: Depth_m , g: ThermalBackfill (Yes / No)
"   LoopTests
o 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
o A: timestamp , b: User , C: topic , D: EventType , e: k1 , f: k2 , g: notes
"   Measurements
o A: topic , b: metric , C: Value , D: Unit , e: source , f: timestamp
"   Portfolio
o   Generated automatically
Logigram rules
"   Cable selection:
o   Voltage_kV compatible with application; XLPE/PVC for LV/MV; Oil/Gas pressure for 66-230 kV.
o   Armour and laying depth appropriate for corridor and mechanical risk.
"   Route viability:
o   CrossingAngle_deg close to 90° for EMI reduction; MinSeparation_m adequate from other utilities.
"   Installation gating:
o   All Required items in InstallChecklist for the RouteID must be Completed = TRUE.
"   Loop test readiness:
o   Known total length and resistance-per-metre for the tested core(s).
o   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 El
se _
    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 St
ring)
"        If Not IsCableSuitable(cableID, required_kV) Then
"            LogEvent runID, "Error", "CableSuitability", cableID, "Cable not suitable for voltage": Ex
it 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, matchC
ol 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
o   CurrentUser = Tshingombe Fiston Tshitadi
o   EvidenceDir = C:\Evidence
o CopperResist_ohm_m = 0.00000001724
o ConductorCSA_mm2 = 95
"   CableCatalog examples
o   XLPE_11kV | XLPE | 11 | Yes | 90 | MV distribution
o   PVC_1kV | PVC | 1.1 | Yes | 70 | Internal services
o   OIL_132kV | OilFilled | 132 | No | 65 | HV transmission
o   GAS_132kV | GasPressure | 132 | No | 65 | HV, flame quenching
"   Routes example
o   R01 | Road | 850 | 1.2 | Gas,Water | 90 | 0.5
"   InstallChecklist (for R01)
o   Trench depth OK | TRUE | TRUE | R01 | ?0.8 m
o   Separation verified | TRUE | TRUE | R01 | ?0.5 m
o   Armour continuity | TRUE | TRUE | R01 | Bonded both ends
"   LoopTests (Murray)
o   T_MUR_01 | Murray | R01 | RUN01 | 850 | 0.00019 | 100 | 100 | | | Meas Rt=32.3
"   LoopTests (Varley placeholder)
o   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
o A: key , b: Value
o keys: CurrentUser , EvidenceDir, BillingDays, Tariff_per_kWh, KCL_Tolerance_A, KVL_Tolerance_V
" Devices
o A: Device , b: power_W , C: Qty , D: HoursPerDay , e: Energy_kWh_day (output), f: notes

"   Circuits
o   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
o A: NodeID , b: Currents_In_CSV , C: Currents_Out_CSV , D: SumIn_A , e: SumOut_A , f: Balanced (Yes / No), g: notes
"   KVL
o A: LoopID , b: Sources_CSV , C: Drops_CSV , D: SumSources_V , e: SumDrops_V , f: Balanced (Yes / No) , g: notes
"   Events
o A: timestamp , b: User , C: topic , D: EventType , e: k1 , f: k2 , g: notes
"   Measurements
o A: topic , b: metric , C: Value , D: Unit , e: source , f: timestamp
"   Portfolio
o   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(m
onthly, "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=36?, R2||R3=24?, R4=50?, 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, Paral
lelR(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=2
4, 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 = "?=" & 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 = "?=" & 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 Doub
le) As Variant
    Dim det As Double: det = A11 * A22 - A12 * A21
    If Abs(det) < 0.000000001 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 A
s 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:
o   CurrentUser = Tshingombe Fiston Tshitadi
o   EvidenceDir = C:\Evidence
o BillingDays = 31
o Tariff_per_kWh = 1.5
o KCL_Tolerance_A = 0.01
o KVL_Tolerance_V = 0.1
"   Devices (examples):
o   Tube Lights | 40 | 5 | 5
o   Fans | 80 | 4 | 8
o   TV | 120 | 1 | 6
o   Lamps | 60 | 4 | 4
"   Circuits row for series example:
o   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
o A: key , b: Value
o   Keys to seed: CurrentUser, EvidenceDir, AcademicYear, MinPsychoCoverage_pct, MinPortfolioArtifacts, MinPLC_Hours, MinSCADA_Hours, MinIoT_Hours, MinIndustryHours, MinExhibitionProjects, Policy_PassMark_pct
"   Components
o 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
o 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
o   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
o   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
o   A: LearnerID, B: Tool (e.g., Maree/CAS), C: Date, D: InterestCluster, E: Strengths, F: RiskFlags, G: Coverage (pct)
"   Industry
o A: PlacementID , b: learnerID , C: Partner , D: hours , e: startDate , f: EndDate , g: Supervisor , h: verified (True / False)
"   Exhibitions
o   A: ProjectID, B: LearnerID, C: Title, D: Category, E: Artifacts (CSV EvidenceIDs), F: JuryScore_pct, G: Award (if any), H: Presented (TRUE/FALSE)
"   Assessments
o   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
o A: timestamp , b: User , C: topic , D: EventType , e: k1 , f: k2 , g: notes
"   Portfolio
o   Generated automatically (no manual columns)
Logigram Gates
"   Component readiness:
o   CareerDiscovery: Coverage ? MinPsychoCoverage_pct across cohort.
o   ControlSystems: Sum of PLC, SCADA, IoT hours ? minima in Config.
o   Portfolio: Each learner has ? MinPortfolioArtifacts verified entries mapped to Outcomes.
o   Industry: Hours per learner ? MinIndustryHours, verified.
o   Exhibition: Projects per learner/team ? MinExhibitionProjects, presented = TRUE.
o   Assessments: Module pass rate ? Policy_PassMark_pct threshold for gating progress.
"   Module activation:
o   Module.Enabled = TRUE and all PrereqIDs completed (Assessments with Passed = TRUE for that learner).
"   Evidence integrity:
o   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 don
e
    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/Stand
ards text too
        If InStr(1, LCase(ac.Cells(i, 6).Value), "plc", vbTextCompare) > 0 Then plc = plc + val(ac.Cel
ls(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.Cel
ls(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(t
arget, "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 Funct
ion
    If Not CBool(r.Offset(0, 6).Value) Then LogEvent "Gate", "Denied", learnerID, moduleID, "Module di
sabled": 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", learner
ID, 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, Optio
nal 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, mat
chCol 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:
o   CurrentUser = Tshingombe Fiston Tshitadi
"
o   videnceDir = C:\Evidence
o AcademicYear = 2025
o MinPsychoCoverage_pct = 70
o MinPortfolioArtifacts = 6
o MinPLC_Hours = 20, MinSCADA_Hours = 20, MinIoT_Hours = 10
o MinIndustryHours = 80
o MinExhibitionProjects = 1
o 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
Print Form; Completion; txtStudentName, txtDOB, Not ComboBox1; CommandButton1_Click()
? Engineering Input Calculator logic completed (Macro2) TextBox_Operator_Click()
? PLC Command Valid Hex string constructed (Macro3) CommandButtonPLC_Click()
Print 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
o UserForm1: Student Registration And validation
o UserForm3: engineering calculator And selector
o   UserForm5: Error hub and dynamic control management
o   UserForm7: Multi action dashboard
o   UserForm8: Frame based navigation
"   Modules
o   Module2: Form initialization (frm1 and shared init)
o   Module3: Calculation engine (impedance, reactance, fault currents, per unit)
o   Module4: UI orchestration, command builders, portfolio assembly
o   ModuleLog: Logging, guards, error plumbing, and audit
"   Tables/Sheets (suggested)
o Config, Students, Evidence, Metrics, Events, Portfolio
Logigram gates and algorigram flow
"   Logigram gates
o   Form completeness: required TextBoxes/Combos non empty
o   Valid engineering inputs: numeric and in range
o   PLC command integrity: hex string length and checksum (optional) valid
o   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 Stri
ng, 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 (2?fL)", "XC (1/2?fC)", "Fault I3?", "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") & " ?"
            LogMetric "Calc", "Z_RLC", z, "Ohm"
        Case "XL (2?fL)"
            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") & " ?"
            LogMetric "Calc", "XL", xLval, "Ohm"
        Case "XC (1/2?fC)"
            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") & " ?"
            LogMetric "Calc", "XC", xCval, "Ohm"
        Case "Fault I3?"
            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 "I3? = " & 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", "Tim
estamp", "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:
o I3? = Ibase/Xpu ? with Ibase = 0.52 kA and Xpu = 0.0604, I3? ? 8.6 kA
o Momentary ? 1.6 × I3?
o 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:
o Run InitApp
o Open UserForm1 (registration)
o Open UserForm3 (calculator)
o Open UserForm7 (dashboard)
5. Test flow:
o Register learner in UserForm1
o Compute engineering values in UserForm3
o Build PLC command in UserForm7/Macro3
o 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
o A: key , b: Value
o Keys to seed: CurrentUser, EvidenceDir, MinSafetyItems, MinEthicsChecklist, MinPortfolioArtifacts, Target_DLI_mol, Target_VPD_kPa, Target_EC_mScm, Target_pH, MaxEnergy_kWh_day
" Curriculum
o A: moduleID (7.1 - 7.1), b: title , C: Level (NQF / PG), D: credits , e: Prereqs (csv), f: Enabled (True / False)
" Activities
o 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
o A: item , b: domain (Safety / Ethics / Biosecurity), C: required (True / False), D: completed (True / False), e: notes
" Systems
o 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
o A: RecipeID, B: Name, C: StockA_gL, D: StockB_gL, E: TargetEC_mScm, F: TargetpH, G: BufferType, H: Notes
" Evidence
o 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
o A: timestamp , b: User , C: topic , D: EventType , e: k1 , f: k2 , g: notes
" Metrics
o A: topic , b: metric , C: Value , D: Unit , e: timestamp
" Portfolio
o 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 A
nd 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 A
nd 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 (?mol·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, a
rea_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 Doub
le) 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 Boolea
n
    If Not ModuleEnabled(modID) Then LogEvent "Gate", "Denied", modID, "Disabled", "": Exit Function
    If Not PrereqsMet(learnerID, modID) Then LogEvent "Gate", "Denied", modID, "Prereqs", "": Exit Fun
ction
    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 veri
fied 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 A
s 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 TR
UE
"   A_ETHICS | 7.9 | Assessment | 2h | Outcomes: Ethics Review | Required TRUE
Run flow
1. Validate gates:
o   If StartModule("Learner01","7.6") returns True ? proceed.
2. Validate environment:
o   ValidateSystem systemRow:=2 (Systems sheet populated).
3. Log evidence:
o   AddEvidence "A_VF_SYS","Learner01","Data","Systems!Row2",True,"Assessor"
4. Export portfolio:
o   ExportPortfolio "Learner01"
Optional UserForm stubs
"   UserForm_VF: Form to input PPFD, photoperiod, temp, RH, EC, pH; buttons for Calculate (DLI/VPD/Ene
rgy), 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
o A: key , b: Value
o   Keys to seed: CurrentUser, EvidenceDir, AcademicYear, MinCPD_Annual, MinEthicsCredits, MinComplianceCredits, MinProjects, PassMark_pct
"   Roles
o 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
o   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
o   A: ModuleID, B: Title, C: Credits, D: Outcomes (CSV CompIDs), E: Prereqs (CSV), F: Level, G: Enabled (TRUE/FALSE)
"   Activities
o   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
o   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
o A: AssessmentID , b: moduleID , C: learnerID , D: score , e: maxScore , f: passed (True / False), g: Date
"   CPD
o A: learnerID , b: compID , C: credits , D: source (Module / Evidence), e: Date
"   EthicsCompliance
o A: item , b: domain (Ethics / compliance), C: required (True / False), D: completed (True / False), e: evidenceID , f: notes
"   Events
o A: timestamp , b: User , C: topic , D: EventType , e: k1 , f: k2 , g: notes
"   Portfolio
o   Generated automatically

Logigram Gates
"   Role activation:
o role.Enabled = True
o   CPD annual credits ? MinCPD_Annual
o   Ethics credits ? MinEthicsCredits; Compliance credits ? MinComplianceCredits
o   Required activities for the role's target modules have verified evidence ? MinArtifacts
o   Assessments for role critical modules passed with Score% ? PassMark_pct
"   Module activation:
o   Module.Enabled = TRUE and all Prereqs passed (Assessments)
"   PoE export:
o   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 = "", O
ptional 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 = T
rue: 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, "Mod
ule 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 A
nd 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 Func
tion
    If Not CBool(rr.Offset(0, 6).Value) Then LogEvent "GateRole", "Denied", learnerID, roleID, "Role d
isabled": 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 insuffi
cient": 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, "Arti
facts insufficient": Exit Function
    If SumRequiredHours(learnerID) < minH Then LogEvent "GateRole", "Denied", learnerID, roleID, "Hour
s 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 A
nd 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 A
s 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:
o   CurrentUser = Tshingombe Fiston Tshitadi
o   EvidenceDir = C:\Evidence
o AcademicYear = 2025
o MinCPD_Annual = 20
o MinEthicsCredits = 6
o MinComplianceCredits = 6
o MinProjects = 2
o PassMark_pct = 60
"   Roles (samples):
o   R_POLICY | Public Policy Analyst | Gov | KPIs: briefs,on time | 60 | 6 | TRUE
o   R_COMPLI | Regulatory Compliance Officer | Legal | KPIs: audits,findings | 60 | 6 | TRUE
o   R_RISK | Legal Risk Consultant (Tech) | Tech | KPIs: DPIA,InfoSec | 60 | 6 | TRUE
o   R_SAFETY | Public Safety Strategist | Gov | KPIs: SOPs,drills | 60 | 6 | TRUE
o   R_ETHICS | Ethics & Governance Advisor | Legal | KPIs: boards,opinions | 60 | 6 | TRUE
o   R_SMART | Smart City Program Manager | Tech | KPIs: delivery,SLAs | 80 | 8 | TRUE
o   R_DX | Digital Transformation Lead | Tech | KPIs: adoption,ROI | 80 | 8 | TRUE
o   R_SUS | Sustainable Infrastructure Auditor | Sustainability | KPIs: ISO scores | 80 | 8 | TRUE
"   Competencies:
o   C_CONLAW | Constitutional & Admin Law | ISO Gov | 3 | Legal
o   C_PRIV | Privacy & Data Governance | ISO/IEC 27701 | 3 | Tech
o   C_PF | Power Factor/ISO 50001 Reporting | ISO 50001 | 2 | Sustainability
o   C_ESG | ESG & ISO 37120 Smart Cities | ISO 37120 | 3 | Sustainability
o   C_RISK | Legal Risk & DPIA | GDPR/DPIA | 3 | Legal
"   Modules (samples):
o   M_LAW101 | Public Admin & ConLaw | 4 | C_CONLAW | | NQF7 | TRUE

o M_PRIV201| Tech Law & Privacy | 4 | C_PRIV,C_RISK | M_LAW101 | NQF8 | TRUE
o M_ESG301 | Smart Cities & ISO 37120 | 4 | C_ESG | | NQF8 | TRUE
" Activities:
o A_BRIEF | M_LAW101 | Case | 8 | Legal brief | TRUE
o A_AUDIT | M_PRIV201| Project | 12 | DPIA + compliance checklist | TRUE
o 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
o key, Value
o Seed: CurrentUser , EvidenceDir, ModuleID_Current, Target_WQI, Max_Abstraction_m3_d, Min_EnvFlow_m3_d, Min_Stakeholder_Records, PassMark_pct
" Sites
o siteID, name, Basin, ClimateZone, SensitiveReceptor, notes
" Sources
o SiteID, Date, SourceType (Surface/GW/Return/Third party), Flow_m3_d, EC_uScm, pH, TSS_mgL, SO4_mgL, Mn_mgL, Fe_mgL
" Uses
o siteID, Date, UseType(Process / Dust / Tails / Misc), Flow_m3_d, ReturnFrac_pct, LossFrac_pct, notes
" Quality
o siteID, Date, SamplePoint, EC_uScm, pH, TSS_mgL, SO4_mgL, Mn_mgL, Fe_mgL, WQI_Score
" Compliance
o SiteID, PermitID, LimitName, LimitValue, Unit, Method (avg/95p/max), Window_d, Active (TRUE/FALSE)
" Stakeholders
o siteID, Date, Party, Category(Community / Regulator / NGO / Worker), Concern, Action, status
" Activities
o ActivityID, Submodule (14.1-14.10), Type (Lab/Report/Model/Assessment/Engagement), Hours, Deliverable, Required (TRUE/FALSE)
" Evidence
o EvidenceID, ActivityID, LearnerID, Type (Doc/Photo/Data/Code/Log), URI_or_Path, Timestamp, Verified (TRUE/FALSE), Verifier, Notes
" Assessments
o AssessmentID, submodule, learnerID, score, maxScore, passed(True / False), Date
" Events
o timestamp, User, topic, EventType, k1, k2, notes
" Metrics
o topic, metric, Value, Unit, timestamp
" Portfolio
o Generated automatically (no manual headers)

Logigram Gates

" Curriculum gate:
o ModuleID_Current set; submodule enabled through Activities table.
" Hydrology gate:

o   Daily water balance computed; Abstraction ? Max_Abstraction_m3_d; Environmental flow reserve met (
Min_EnvFlow_m3_d).
"   Quality gate:
o   WQI computed; below Target_WQI threshold triggers warning.
"   Compliance gate:
o   Active permit limits respected over method/window.
"   Stakeholder gate:
o   At least Min_Stakeholder_Records logged with Status ? "Open".
"   Evidence gate:
o   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 = "", O
ptional 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; refi
ne 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", 1000000000#)) 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 Dou
ble, 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#, 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 & " > Lim
it=" & 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 A
nd 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 "EvidenceGat
e", "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 A
s 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

o   Evidence: Hydrological assessment report; daily water balance logs

o   Skills: Resource evaluation, planning frameworks

"   14.4 Water Quality Management

o   Evidence: WQI dashboard; compliance window checks

o Skills: Monitoring, pollution control
" 14.5 Regulatory Compliance
o Evidence: Exceedance register; permit checklist
o Skills: Regulatory mapping, audit readiness
" 14.7 Stakeholder Engagement
o Evidence: Stakeholder log with actions/closure
o Skills: Engagement, conflict avoidance
" 14.8 Climate Adaptation
o Evidence: Risk matrix; drought/flood scenario notes
o Skills: Resilience Planning
Record assessment outcomes in Assessments (Score/MaxScore/Passed). Use Events to document gate passes/fails.
Quick Start
" Config:
o CurrentUser = Tshingombe Fiston Tshitadi
o EvidenceDir = C:\Evidence
o ModuleID_Current = 14
o Target_WQI = 60
o Max_Abstraction_m3_d = 15000
o Min_EnvFlow_m3_d = 1000
o Min_Stakeholder_Records = 3
o PassMark_pct = 60
" Activities (examples):
o A_14_1 | 14.1 | Report | 6 | Hydrological assessment | TRUE
o A_14_4 | 14.4 | Model | 4 | WQI & permit window calc | TRUE
o A_14_5 | 14.5 | Assessment | 2 | Compliance checklist | TRUE
o 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
o key, Value
o Seed: CurrentUser , EvidenceDir, MinArtifacts, MinSecurityScore, MinRecoveryScore, MinDLTScore, PassMark_pct
" Modules
o moduleID, title, domain, credits, Prereqs(csv), Enabled(True / False)
" Activities
o ActivityID, ModuleID, Type (Lab/Report/Model/Assessment), Hours, Deliverable, Required (TRUE/FALSE)
" Evidence
o EvidenceID, ActivityID, LearnerID, Type (Doc/Photo/Data/Code/Log), URI_or_Path, Timestamp, Verified (TRUE/FALSE), Verifier, Notes
" Assessments
o AssessmentID, moduleID, learnerID, score, maxScore, passed(True / False), Date
" Metrics
o topic, metric, Value, Unit, timestamp
" Events
o timestamp, User, topic, EventType, k1, k2, notes
" Portfolio
o 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 veri
fied 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 = Tr
ue: 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 A
nd 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 A
s 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
o key, Value
o 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
o timestamp, cellID, stackID, V_cell, I_cell, Temp_C, soc_pct, Mode(Charge / Discharge / Idle), ControlAction_A, Fault
"   RPA_Inbox
o fileName, ReceivedAt, Parsed(True / False), RowsImported, notes
"   Events
o timestamp, User, topic, EventType, k1, k2, notes
"   Metrics
o topic, metric, Value, Unit, timestamp
"   Evidence
o   EvidenceID, ActivityID, LearnerID, Type (Doc/Data/Log), URI_or_Path, Timestamp, Verified (TRUE/FALSE), Verifier, Notes
"   Portfolio
o   Generated automatically

Optional:
"   Models (parameters per CellID), Dash (KPIs), Scripts (queries/ETL notes).
Logigram Gates
"   Safety interlocks:
o   V_cell ? MaxCell_V; sum(V_cell) ? MaxStack_V; Temp_C ? MaxTemp_C; |I_cell| ? MaxCurrent_A.
"   Control readiness:
o   Valid SOC_pct and Mode; PID params present; no active Fault.
"   RPA readiness:
o   WatchFolder exists; ingest schedule active; new files not yet parsed.
"   ML readiness:
o   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 facto
r
        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", 10
0)), 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_" & learnerI
D & ".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 A
s 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
2. Seed RPA_Inbox and drop CSV with columns: Timestamp,CellID,StackID,V_cell,I_cell,Temp_C,SOC_pct,Mode.
3. 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
o key, Value
o Seed: CurrentUser , EvidenceDir, MinArtifacts, MaxFaultLevel_kA, MinEfficiency_pct, MinRenewableShare_pct, PassMark_pct
"   Modules
o moduleID, title, domain, credits, Prereqs(csv), Enabled(True / False)
"   Activities
o   ActivityID, ModuleID, Type (Design/Simulation/Assessment/Report), Hours, Deliverable, Required (TRUE/FALSE)
"   Evidence
o   EvidenceID, ActivityID, LearnerID, Type (Doc/Data/Log), URI_or_Path, Timestamp, Verified (TRUE/FALSE), Verifier, Notes
"   Assessments
o AssessmentID, moduleID, learnerID, score, maxScore, passed(True / False), Date
"   Metrics
o topic, metric, Value, Unit, timestamp
"   Events
o timestamp, User, topic, EventType, k1, k2, notes
"   Portfolio
o   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_" & learne
rID & ".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:
   - o CurrentUser = Tshingombe Fiston Tshitadi
   - o EvidenceDir = C:\Evidence
   - o MaxFaultLevel_kA = 25
   - o MinEfficiency_pct = 85
   - o MinRenewableShare_pct = 30
   - o MinArtifacts = 6
   - o PassMark_pct = 60
2. Log evidence:
   - o LogEvidence "A_REDesign", "Learner01", "Doc", "SolarWindDesign.pdf", True, "Assessor"
3. Export:
   - o 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
   - o key, Value
   - o 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
o moduleID, title, domain(38 / 40 / 41), credits, Prereqs(csv), Enabled(True / False)
"   Activities
o   ActivityID, ModuleID, Type (Lab/Model/Report/Assessment), Hours, Deliverable, Required (TRUE/FALSE)
"   Evidence
o   EvidenceID, ActivityID, LearnerID, Type (Doc/Data/Log/Code), URI_or_Path, Timestamp, Verified (TRUE/FALSE), Verifier, Notes
"   Assessments
o AssessmentID, moduleID, learnerID, score, maxScore, passed(True / False), Date
"   Metrics
o topic, metric, Value, Unit, timestamp
"   Events
o timestamp, User, topic, EventType, k1, k2, notes
"   Telemetry
o timestamp, System(Battery / FuelCell / Electrolysis / Sensor), assetID, stackID, V_cell, I_cell, Temp_C, soc_pct, Mode, OCV_V
"   RPA_Inbox
o fileName, ReceivedAt, Parsed(True / False), RowsImported, path
"   Portfolio
o   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; ?V_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, ?_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 = T
rue: 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 A
nd 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 Funct
ion
    If Not CBool(r.Offset(0, 6).Value) Then LogEvent "Gate", "Denied", learnerID, moduleID, "Module di
sabled": Exit Function
    If Not PrereqsMet(learnerID, moduleID) Then LogEvent "Gate", "Denied", learnerID, moduleID, "Prere
qs unmet": Exit Function
    If Not RequiredActivitiesHaveEvidence(moduleID, learnerID) Then LogEvent "Gate", "Denied", learner
ID, 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 = "", O
ptional 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 veri
fied 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 = T
rue: 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 A
nd 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 Funct
ion
    If Not CBool(r.Offset(0, 6).Value) Then LogEvent "Gate", "Denied", learnerID, moduleID, "Module di
sabled": Exit Function
    If Not PrereqsMet(learnerID, moduleID) Then LogEvent "Gate", "Denied", learnerID, moduleID, "Prere
qs unmet": Exit Function
    If Not RequiredActivitiesHaveEvidence(moduleID, learnerID) Then LogEvent "Gate", "Denied", learner
ID, 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 / (?_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_re
f
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).Va
lue), 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 Dou
ble)
    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 o Energy Storage o 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 A
s 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 ? 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
o key, Value
o Seed: CurrentUser , EvidenceDir, MinArtifactsTotal, MinCategoryCoverage_pct, PassMark_pct
"   ArtifactCatalog
o categoryID, CategoryName, required(True / False), MinCount

o   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
o artifactID, learnerID, categoryID, title, URI_or_Path, Date, verified(True / False), verifier, notes
"   Simulations
o simID, learnerID, domain(Signal / control / power / energy / Thermal / Automation), modelRef, status (Pending / Run / Pass / Fail), Score_pct, notes
"   Metrics
o topic, metric, Value, Unit, timestamp
"   Events
o timestamp, User, topic, EventType, k1, k2, notes
"   Portfolio
o   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.Cell
s(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 + 0.3 * (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 = Fal
se
    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, mat
chCol As Integer, key As Variant, Optional filterColBool As Integer = 0, Optional filterValBool As Boo
lean = 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) / basel
ine
    LogMetric "Optimization", "Gain_pct", gain, "%"
    AddArtifact learnerID, "A_OPT", title, path, "Gain=" & Format(gain, "0.0") & "%"
End Sub
```

Quick Start

```
"   Config:
o   CurrentUser = Tshingombe Fiston Tshitadi
o   EvidenceDir = C:\Evidence
o MinArtifactsTotal = 6
o MinCategoryCoverage_pct = 100
o PassMark_pct = 60
"   Register and run:
o   RegisterSimulation "SIM_SIG_01","Learner01","Signal","Fourier/Laplace"
o   RegisterSimulation "SIM_CTRL_01","Learner01","Control","PID tune"
o   RegisterSimulation "SIM_PWR_01","Learner01","Power","Load flow"
o   RegisterSimulation "SIM_EN_01","Learner01","Energy","Integration"
o   RegisterSimulation "SIM_RPA_01","Learner01","Automation","RPA checks"
o   RunSimulation "SIM_SIG_01": RunSimulation "SIM_CTRL_01": RunSimulation "SIM_PWR_01"
o   RunSimulation "SIM_EN_01": RunSimulation "SIM_RPA_01"
"   Add and verify artifacts:
o   RecordMatlabModel "Learner01","NR Load Flow","C:\PoE\models\loadflow.slx"
o   RecordRPAWorkflow "Learner01","RPA Tool Matrix","C:\PoE\rpa\comparison.pdf"
o   RecordBatteryDashboard "Learner01","Li-ion Dashboard","C:\PoE\battery\dashboard.xlsx", 84, 100
o   RecordSensorProtocol "Learner01","NO2 Sensor SOP","C:\PoE\sensors\no2_sop.docx", 210, 200
o   RecordEthicsFramework "Learner01","Automation Ethics","C:\PoE\ethics\framework.docx"
o   RecordOptimizationReport "Learner01","GA Optimization","C:\PoE\opt\ga_report.pdf", 10.2, 8.4
o   VerifyArtifact "ART240912123000","Assessor","OK" ' use actual ID returned in sheet
"   Export:
o   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 gl
ance.

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
o key, Value
o Seed: CurrentUser , EvidenceDir, MinArtifactsTotal, PassMark_pct, MinEthicsArtifacts
" ArtifactCatalog
o categoryID, CategoryName, required(True / False), MinCount
o 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
o artifactID, learnerID, categoryID, title, URI_or_Path, Date, verified(True / False), verifier, notes
" Simulations
o simID, learnerID, domain(Signal / control / power / energy / stats), modelRef, status(Pending / Pass / Fail), Score_pct, notes
" Metrics
o topic, metric, Value, Unit, timestamp
" Events
o timestamp, User, topic, EventType, k1, k2, notes
" Portfolio
o 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 S
ub
    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) The
n 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", "ArtifactsTo
talFail", 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 = Fa
lse
    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) ?^2 = sum((x - ?)^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, matchC
ol 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:
o   CurrentUser = Tshingombe Fiston Tshitadi
o   EvidenceDir = C:\Evidence
o MinArtifactsTotal = 6
o MinEthicsArtifacts = 1
o PassMark_pct = 60
"   Register and run sims:
o   RegisterSimulation "SIM_PWR_01","Learner01","Power","S=sqrt(P2+Q2)"
o   RegisterSimulation "SIM_CTRL_01","Learner01","Control","PID tuning"
o   RegisterSimulation "SIM_SIG_01","Learner01","Signal","Fourier filter"
o   RegisterSimulation "SIM_EN_01","Learner01","Energy","Integration"
o   RegisterSimulation "SIM_ST_01","Learner01","Stats","Mean & variance"
o   RunSimulation "SIM_PWR_01": RunSimulation "SIM_CTRL_01": RunSimulation "SIM_SIG_01": RunSimulation "SIM_EN_01": RunSimulation "SIM_ST_01"
"   Add artifacts (examples):
o   RecordMatlabModel "Learner01","Load Flow & Control","C:\PoE\models\lf_ctrl.slx"
o   RecordRPAWorkflow "Learner01","RPA Tooling Matrix","C:\PoE\rpa\matrix.pdf"
o   RecordBatteryDashboard "Learner01","Li-ion Dashboard","C:\PoE\batt\dashboard.xlsx",84,100
o   RecordSensorProtocol "Learner01","EC Sensor SOP","C:\PoE\sensors\ec_sop.docx",2.12,2.00
o   RecordEthicsFramework "Learner01","Automation Ethics","C:\PoE\ethics\framework.docx"
o   RecordOptimizationReport "Learner01","GA Loss Reduction","C:\PoE\opt\ga_report.pdf",10.2,8.4
"   Verify artifacts (use IDs in Artifacts sheet), then:
o   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
o key, Value
o Seed: CurrentUser , EvidenceDir, PassMark_pct, MinArtifacts, MinEthicsArtifacts
"   Activities
o   ActivityID, Domain (Forensics/MIS/Traffic/Ballistics/Patrol/Stats), Type (Model/Report/Lab/Dashboard), Required (TRUE/FALSE)
"   Evidence
o   EvidenceID, ActivityID, LearnerID, Type (Doc/Data/Log), URI_or_Path, Timestamp, Verified (TRUE/FALSE), Verifier, Notes
"   Simulations
o simID, learnerID, domain(predictive / decay / patrol / ballistics / area / stats), modelRef, status(Pending / Pass / Fail), Score_pct, notes

"   Metrics
o topic, metric, Value, Unit, timestamp
"   Events
o timestamp, User, topic, EventType, k1, k2, notes
"   Portfolio
o   Generated automatically
Logigram Gates
"   Module/Activity gate:
o   Required activities have at least one Verified evidence per learner.
"   Simulation gate:
o   All simulations for learner have Status = Pass and Score ? PassMark_pct.
"   Ethics/Compliance gate:
o   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 = "", O
ptional 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 veri
fied 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).Val
ue), "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 Functio
n
        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/comp
liance evidence": ok = False
    If Not AllSimulationsPassed(learnerID) Then LogEvent "Gate", "SimFail", learnerID, "", "Simulation
s 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 D
ouble)
    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, Tru
e, "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_" & le
arnerID & ".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 A
s Integer, key As String, Optional boolCol As Integer = 0, Optional boolVal As Boolean = False) As Lon
g
    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:
"
o   CurrentUser = Tshingombe Fiston Tshitadi
o   EvidenceDir = C:\Evidence
o PassMark_pct = 60
o MinArtifacts = 6
o MinEthicsArtifacts = 1
"   Register sims:
o   RegisterSimulation "SIM_PRED_01","Learner01","Predictive","y=mx+b"
o   RegisterSimulation "SIM_DEC_01","Learner01","Decay","C=C0*e^(-?t)"
o   RegisterSimulation "SIM_PAT_01","Learner01","Patrol","path length"
o   RegisterSimulation "SIM_BAL_01","Learner01","Ballistics","R=v0^2 sin(2?)/g"
o   RegisterSimulation "SIM_AREA_01","Learner01","Area","shoelace"
o   RegisterSimulation "SIM_STATS_01","Learner01","Stats","?, ?^2"
o   RunSimulation each SimID
"   Add evidence:
o   AddEvidence "ACT_CUSTODY","Learner01","Doc","chain_of_custody.pdf",True,"Assessor","ethics, chain of custody"
"   Export:
o   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 (?):"
    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
o key, Value
o Seed: CurrentUser , EvidenceDir, MinEntries, MinResolvedPct, MinNominationCount
" Entities
o entityID, name, role(Researcher / Admin / Official), Email, notes
" Logs
o logID, domain(NRF / CSD / Tender / Email), timestamp, status(Logged / Resolved), source, subject, notes
" Nominations
o NomID, EntityID, Date, Type (Award/Recommendation), Status, Notes
" Evidence
o EvidenceID, LogID, Type (Email/Doc/Form), URI_or_Path, Timestamp, Verified (TRUE/FALSE), Verifier, Notes
" Metrics
o topic, metric, Value, Unit, timestamp
" Events
o timestamp, User, topic, EventType, k1, k2, notes
" Portfolio
o 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, Optiona
l 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 A
s 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 audit
s.
"   ?? 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