EN0202272ID

No bio

(/account/user)

0 | Follow user

My Groups

Coming soon

Add Group (https://www.elektormagazine.com/labs/204047/en0202272id/groups)

My Stats

0 Published projects

Views

0 Followers

0 Star(s) on average

0 Comments

0 Comments

View as visitor:

Add Project

Cancel

Welcome at Elektor Labs! Here you can add your own project to share within our community. You may receive input from your peers when you need help or feedback. Also, the Elektor Labs professionals are also closely watching this section of our website to help out or on occasion select projects that could even be published in our magazine.
CONTEST PARTICIPANTS: Please use this form to submit entries to design contests managed by Elektor, such as the STM32 Wireless Innovation Design Contest (2023-2024). To do so, please fill in the fields and provide project files (photos, schematics, code, documentation, etc). Make sure the appropriate contest is selected in the "Contest" drop-down list below

Title

APPLICATION TRADE MATERIAL DIDACTIC

Video

(https://www.elektormagazine.com)

Teaser

APLICATION

Please add a short 'elevator pitch' for the project. Note: the teaser must be in English language in order for the project to be accepted.

Project image

Add your project image here
JPG, PNG or GIF file (5 MB max)
400 x 225 pixels minimum (16:9 aspect ratio)

Description

Please add the full description for the project. Important: the text must be in English language in order for the project to be accepted.

Gmail

| **TSHINGOMBEKB TSHITADI <tshingombekb@gmail.com>**
eginering
TSHINGOMBEKB TSHITADI <tshingombekb@gmail.com> | Tue, Sep 23, 2025 at 3:27 PM
To: tshingombe fiston <tshingombefiston@gmail.com>, TSHINGOMBEKB TSHITADI <tshingombekb@gmail.com>

On Tue, Sep 23, 2025 at 2:39 PM TSHINGOMBEKB TSHITADI <tshingombekb@gmail.com (mailto:tshingombekb@gmail.com)> wrote:

On Tue, Sep 23, 2025 at 12:52 PM TSHINGOMBEKB TSHITADI <tshingombekb@gmail.com (mailto:tshingombekb@gmail.com)> wrote:

On Mon, Sep 22, 2025 at 4:12 PM TSHINGOMBEKB TSHITADI <tshingombekb@gmail.com (mailto:tshingombekb@gmail.com)> wrote:

On Mon, Sep 22, 2025 at 2:24 PM TSHINGOMBEKB TSHITADI <tshingombekb@gmail.com (mailto:tshingombekb@gmail.com)> wrote:

On Fri, Sep 19, 2025 at 3:40 PM TSHINGOMBEKB TSHITADI <tshingombekb@gmail.com (mailto:tshingombekb@gmail.com)> wrote:

```
'
' engi Macro
'
'


Option Explicit

Private Sub UserForm_Initialize()
    ' Initialize defaults
    Me.MultiPage1.Value = 0 ' First tab
    Me.optMale.Value = False
    Me.optFemale.Value = False
    Me.txtPassword.PasswordChar = "o"
End Sub

Private Sub cmdNext_Click()
    ' Toggle between tabs
    If Me.MultiPage1.Value < Me.MultiPage1.Pages.count - 1 Then
        Me.MultiPage1.Value = Me.MultiPage1.Value + 1
    Else
        Me.MultiPage1.Value = 0
    End If
End Sub

Private Sub cmdCancel_Click()
    If MsgBox("Cancel registration?", vbQuestion + vbYesNo, "Confirm") = vbYes Then
        Unload Me
    End If
End Sub

Private Sub cmdOK_Click()
    Dim errMsg As String
    If Not ValidateInputs(errMsg) Then
        MsgBox errMsg, vbExclamation, "Validation"
        Exit Sub
    End If

    ' Simulated save; replace with your persistence logic
    ' e.g., write to worksheet/database/API
    ' Example (Excel): WriteToSheet
```

```
    MsgBox "Registration successful.", vbInformation, "Success"
    Unload Me
End Sub

Private Function ValidateInputs(ByRef errMsg As String) As Boolean
    Dim dt As Date
    Dim genderSelected As Boolean

    ' Basic required fields
    If Trim$(Me.txtFirstName.text) = "" Then
        errMsg = "First name is required."
        ValidateInputs = False
        Exit Function
    End If

    If Trim$(Me.txtSurname.text) = "" Then
        errMsg = "Surname is required."
        ValidateInputs = False
        Exit Function
    End If

    If Trim$(Me.txtDOB.text) = "" Then
        errMsg = "Birth date is required (YYYY-MM-DD)."
        ValidateInputs = False
        Exit Function
    End If

    ' Date validation (expects a valid date; adjust to your locale/format)
    On Error GoTo BadDate
    dt = CDate(Me.txtDOB.text)
    On Error GoTo 0

    If dt > Date Then
        errMsg = "Birth date cannot be in the future."
        ValidateInputs = False
        Exit Function
    End If

    ' Gender
    genderSelected = (Me.optMale.Value Or Me.optFemale.Value)
    If Not genderSelected Then
        errMsg = "Please select a gender."
        ValidateInputs = False
        Exit Function
    End If

    ' Account page checks
    If Trim$(Me.txtUsername.text) = "" Then
        errMsg = "Username is required."
        ValidateInputs = False
        Exit Function
```

```
    End If
    If Len(Me.txtPassword.text) < 6 Then
        errMsg = "Password must be at least 6 characters."
        ValidateInputs = False
        Exit Function
    End If

    ValidateInputs = True
    Exit Function

BadDate:
    errMsg = "Invalid birth date. Use a valid date (e.g., 2001-05-17)."
    ValidateInputs = False
End Function

' Optional: Excel example of saving to a sheet
Private Sub WriteToSheet()
    Dim ws As Worksheet
    Dim nextRow As Long
    Dim gender As String

    Set ws = ThisWorkbook.Worksheets("Registrations")

    If Me.optMale.Value Then
        gender = "Male"
    ElseIf Me.optFemale.Value Then
        gender = "Female"
    Else
        gender = ""
    End If

    nextRow = ws.Cells(ws.Rows.count, "A").End(xlUp).row + 1
    With ws
        .Cells(nextRow, 1).Value = Me.txtFirstName.text
        .Cells(nextRow, 2).Value = Me.txtSurname.text
        .Cells(nextRow, 3).Value = Me.txtDOB.text
        .Cells(nextRow, 4).Value = gender
        .Cells(nextRow, 5).Value = Me.txtUsername.text
        .Cells(nextRow, 6).Value = MaskPassword(Me.txtPassword.text) ' Store hashed/hidden if possible
        .Cells(nextRow, 7).Value = Now
    End With
End Sub

Private Function MaskPassword(ByVal pwd As String) As String
    ' Placeholder for hashing/encryption
    MaskPassword = String$(Len(pwd), "*")
End Function
```

Minimal module to show the form (Excel/VBA)

Place this in a standard module to open the form.

vb

```
End Sub
Quick build steps
"   Add a UserForm: Insert ? UserForm ? Rename to frmRegistration.
"   Add a MultiPage: Insert MultiPage ? Name: MultiPage1 ? Set Captions to "Personal Info" and "Account".
"   Add a Frame to Page(0): Name: fraPersonal ? Caption: Personal details.
"   Add labels/textboxes:
o lblFirstName + txtFirstName
o lblSurname + txtSurname
o lblDOB + txtDOB
o lblGender + optMale + optFemale
"   Add Account page fields: lblUsername + txtUsername, lblPassword + txtPassword (set PasswordChar =
o).
"   Add buttons: cmdOK, cmdCancel, cmdNext; set their captions.
"   Paste code: Into frmRegistration code-behind and optional module.
"   Run: Call ShowRegistration.
Optional enhancements
"   Input masks: Enforce date format with a simple mask or a calendar control.
"   Gender options: Add "Other/Prefer not to say."
"   Validation UI: Highlight invalid fields by changing BackColor.
"   Persistence: Replace WriteToSheet with a database call or JSON file write.
"   Navigation: Disable cmdOK until all validations pass.
If you want, tell me your target environment (Excel VBA, Access, VB6, or VB.NET (http://VB.NET)
WinForms), and I'll tailor the exact control names and code syntax accordingly.
Absolutely , Tshingombe.Here 's a structured Visual Basic (VBA) framework for a National Diploma
Examination Result Statement Generator tailored to engineering studies. It includes:
"   Student info fields
"   Course modules across terms and semesters
"   Assessment breakdown (homework, classwork, exams)
"   Score scaling and qualification levels
"   Print-ready marksheet logic
"   Award and pass/fail classification
?? UserForm Layout Overview
?? Student Info Section
Control Type    Name    Caption
Label   lblStudentName  Student Name
TextBox txtStudentName  -
Label   lblSurname  Surname
TextBox txtSurname  -
Label   lblAchievementYear  Achievement Year
TextBox txtAchievementYear  -
Label   lblExperienceYears  Work Experience (Years)
TextBox txtExperienceYears  -
?? Academic Record Section
Use a MultiPage or TabStrip to organize:
?? Page 1: Terms & Semesters
Term Controls
Term 1-4    txtTerm1, txtTerm2, txtTerm3, txtTerm4
Semester 1-2    txtSem1, txtSem2
?? Page 2: Course Modules & Assessment
```

Field Controls
Course Topic lstCourseTopics (List/ComboBox)
Homework txtHomeworkScore
Classwork txtClassworkScore
Exams txtExamScore
Total Score lblTotalScore (calculated)
Rating (%) lblRating (calculated)
?? Qualification & Scaling
Field Controls
Final Qualification cboQualification (e.g., 1st, 2nd, 3rd, 4th)
Level cboLevel(1 - 9)
Course Weight txtCourseWeight
Scaling Factor txtScalingFactor
Final Score lblFinalScore (calculated)
?? Buttons
Button Function
cmdCalculate Compute total, rating, qualification
cmdPrint Print formatted marksheet
cmdClear Reset form
cmdExit Close form
?? Calculation Logic (VBA)

```
Private Sub cmdCalculate_Click()
    Dim Homework As Double, Classwork As Double, Exam As Double
    Dim total As Double, rating As Double, scaledScore As Double
    Dim weight As Double, scaleFactor As Double

    ' Get scores
    Homework = Val(txtHomeworkScore.text)
    Classwork = Val(txtClassworkScore.text)
    Exam = Val(txtExamScore.text)

    ' Compute total
    total = Homework + Classwork + Exam
    lblTotalScore.Caption = total & " / 100"

    ' Rating
    rating = (total / 100) * 100
    lblRating.Caption = Format(rating, "0.00") & "%"

    ' Scaling
    weight = Val(txtCourseWeight.text)
    scaleFactor = Val(txtScalingFactor.text)
    scaledScore = total * (weight / 100) * scaleFactor
    lblFinalScore.Caption = Format(scaledScore, "0.00")

    ' Qualification logic
    Select Case rating
        Case Is >= 50
            cboQualification.text = "Pass"
        Case 40 To 49
            cboQualification.text = "Conditional Pass"
```

```
    Case 20 To 39
        cboQualification.text = "Fail"
    Case Is < 20
        cboQualification.text = "Severe Fail"
  End Select
End Sub
??? Print Statement Logic
Private Sub cmdPrint_Click()
  Dim msg As String
  msg = "National Diploma Result Statement" & vbCrLf & vbCrLf
  msg = msg & "Student: " & txtStudentName.text & " " & txtSurname.text & vbCrLf
  msg = msg & "Achievement Year: " & txtAchievementYear.text & vbCrLf
  msg = msg & "Experience: " & txtExperienceYears.text & " years" & vbCrLf & vbCrLf

  msg = msg & "Course Topics: " & lstCourseTopics.text & vbCrLf
  msg = msg & "Homework: " & txtHomeworkScore.text & vbCrLf
  msg = msg & "Classwork: " & txtClassworkScore.text & vbCrLf
  msg = msg & "Exam: " & txtExamScore.text & vbCrLf
  msg = msg & "Total Score: " & lblTotalScore.Caption & vbCrLf
  msg = msg & "Rating: " & lblRating.Caption & vbCrLf
  msg = msg & "Final Qualification: " & cboQualification.text & vbCrLf
  msg = msg & "Level: " & cboLevel.text & vbCrLf
  msg = msg & "Final Scaled Score: " & lblFinalScore.Caption & vbCrLf

  MsgBox msg, vbInformation, "Print Preview"
  ' Replace with actual print logic if needed
End Sub
?? Award Logic (Optional)
vb
If rating >= 85 Then
  MsgBox "Student qualifies for Distinction Award!", vbInformation
ElseIf rating >= 70 Then
  MsgBox "Student qualifies for Merit Award.", vbInformation
ElseIf rating >= 50 Then
  MsgBox "Student passed successfully.", vbInformation
Else
  MsgBox "Student did not meet pass criteria.", vbExclamation
End If
Visual Basic framework for reprint, release, and recertification of result statements
Below is a practical Visual Basic/VBA scaffold to manage reprint and release workflows for electrical trade theory result statements, including backlog checks, irregularity flags, insurance/quality-body attestations, and reconciliation of internal vs external assessment. It covers student identity, term/semester records, combination/replace results, and recertification.
UserForm Structure And Fields
"  Form name: frmResultRelease
"  Pages: MultiPage1 with tabs: Identity, Assessments, Quality, Actions
Identity Page
"  Student ID: txtStudentID
"  Username: txtUsername
"  Surname: txtSurname
"  Year of birth: txtYOB
```

" Admin year: txtAdminYear
" Program: cboProgramme (ADC, Advanced Diploma, etc.)
" Level: cboLevel (1-8)
" Trade: cboTrade (Electrical, Instrumentation, etc.)

Assessments Page
" Internal assessment total (0-100): txtInternal
" External assessment total (0-100): txtExternal
" Exam type: cboExamType (Main, Rewrite, Supplementary)
" Attempt count: txtAttempt
" Backlog credits outstanding: txtBacklogCredits
" Combination/replace source ID: txtCombineWithResultID

Quality Page
" Irregularity flag: chkIrregularity
" Irregularity note: txtIrregularityNote
" Insurance/QA body clearance: chkQACleared
" QA reference number: txtQARef
" Material/proctor issue flag: chkProctorIssue
" Material batch ref: txtMaterialBatch

Actions Page
" Status label: lblReleaseStatus
" Buttons: cmdReconcile, cmdEvaluate, cmdRelease, cmdReprint, cmdRecertify, cmdSave, cmdExportPDF, cmdClose

Business rules
" Pass thresholds:
o Pass ? 50%; Conditional pass 40-49%; Fail 20-39%; Severe fail < 20.
" Variance check internal vs external:
o If absolute difference > 20 percentage points, set ReviewRequired.
" Irregularity or QA not cleared:
o Hold release until cleared.
" Backlog credits > 0:
o Hold certificate; allow statement with "Provisional" if enabled.
" Rewrite attempt logic:
o If cboExamType = "Rewrite", mark AttemptedRewrite = True; allow combination/replace if improved.
" Combination and replace result:
o If txtCombineWithResultID not empty and new score higher, replace; else keep best.

status model
" EligibleForRelease
" HoldIrregularity
" HoldBacklog
" HoldQANotCleared
" ReviewVariance
" RecertificationRequired
" ReprintAllowed

Code: Core types And utilities

```
Option Explicit

Private Enum ReleaseStatus
    EligibleForRelease = 0
    HoldIrregularity = 1
    HoldBacklog = 2
    HoldQANotCleared = 3
```

```
    ReviewVariance = 4
    RecertificationRequired = 5
    ReprintAllowed = 6
End Enum

Private Type StudentRecord
    StudentID As String
    Username As String
    Surname As String
    YOB As Integer
    AdminYear As Integer
    programme As String
    Level As Integer
    Trade As String
    internalScore As Double
    externalScore As Double
    ExamType As String
    Attempt As Integer
    BacklogCredits As Integer
    CombineWithID As String
    Irregularity As Boolean
    IrregularityNote As String
    QACleared As Boolean
    QARef As String
    ProctorIssue As Boolean
    MaterialBatch As String
    finalScore As Double
    rating As Double
End Type

Private Const PASS_THRESHOLD As Double = 50#
Private Const CONDITIONAL_LOW As Double = 40#
Private Const FAIL_LOW As Double = 20#
Private Const VARIANCE_THRESHOLD As Double = 20#   'percentage points
Code: Data capture And reconciliation

    Dim r As StudentRecord
    r.StudentID = Trim$(txtStudentID.text)
    r.Username = Trim$(txtUsername.text)
    r.Surname = Trim$(txtSurname.text)
    r.YOB = Val(txtYOB.text)
    r.AdminYear = Val(txtAdminYear.text)
    r.programme = cboProgramme.text
    r.Level = Val(cboLevel.text)
    r.Trade = cboTrade.text
    r.internalScore = Val(txtInternal.text)
    r.externalScore = Val(txtExternal.text)
    r.ExamType = cboExamType.text
    r.Attempt = Val(txtAttempt.text)
    r.BacklogCredits = Val(txtBacklogCredits.text)
    r.CombineWithID = Trim$(txtCombineWithResultID.text)
```

```
    r.Irregularity = chkIrregularity.Value
    r.IrregularCode = Trim$(txtIrregularity.text)
    r.QACleared = chkQACleared.Value
    r.QARef = Trim$(txtQARef.text)
    r.ProctorIssue = chkProctorIssue.Value
    r.MaterialBatch = Trim$(txtMaterialBatch.text)
    ReadForm = r
End Function


    ' Weighted blend: external prioritized; adjust as needed
    Dim blended As Double
    blended = (0.4 * r.internalScore) + (0.6 * r.externalScore)
    r.finalScore = blended
    r.rating = blended ' out of 100
End Sub


    Dim variance As Double
    variance = Abs(r.internalScore - r.externalScore)

    If r.Irregularity Then
        EvaluateStatus = HoldIrregularity: Exit Function
    End If
    If Not r.QACleared Then
        EvaluateStatus = HoldQANotCleared: Exit Function
    End If
    If r.BacklogCredits > 0 Then
        EvaluateStatus = HoldBacklog: Exit Function
    End If
    If variance > VARIANCE_THRESHOLD Then
        EvaluateStatus = ReviewVariance: Exit Function
    End If

    ' Recertification if severe fail on external or repeated attempts
    If r.externalScore < FAIL_LOW Or r.Attempt >= 3 Then
        EvaluateStatus = RecertificationRequired: Exit Function
    End If

    EvaluateStatus = EligibleForRelease
End Function
Code: combination/replace and award logic
vb
Private Function BestOf(oldScore As Double, newScore As Double) As Double
    If newScore > oldScore Then
        BestOf = newScore Else BestOf = oldScore
    End If
End Function

Private Function AwardText(ByVal rating As Double) As String
    If rating >= 85 Then
```

```
    AwardText = "Distinction"
  ElseIf rating >= MERIT_THRESHOLD Then
    AwardText = "Merit"
  ElseIf rating >= PASS_THRESHOLD Then
    AwardText = "Pass"
  ElseIf rating >= CONDITIONAL_LOW Then
    AwardText = "Conditional Pass"
  ElseIf rating >= FAIL_LOW Then
    AwardText = "Fail"
  Else
    AwardText = "Severe Fail"
  End If
End Function
```

Code: Button handlers

vb

```
Private Sub cmdReconcile_Click()
  Dim r As StudentRecord
  r = ReadForm()
  ComputeScores r

  Dim status As ReleaseStatus
  status = EvaluateStatus(r)
  lblReleaseStatus.Caption = StatusToText(status) & " | Rating: " & Format(r.rating, "0.00") & "% | Award: " &
AwardText(r.rating)
End Sub

Private Sub cmdEvaluate_Click()
  Call cmdReconcile_Click
End Sub

Private Sub cmdRelease_Click()
  Dim r As StudentRecord
  r = ReadForm()
  ComputeScores r

  Dim status As ReleaseStatus
  status = EvaluateStatus(r)
  If status <> EligibleForRelease Then
    MsgBox "Cannot release. Status: " & StatusToText(status), vbExclamation
    Exit Sub
  End If

  SaveRecord r, "Released"
  MsgBox "Final result released and certificate queued.", vbInformation
End Sub

Private Sub cmdReprint_Click()
  Dim r As StudentRecord
  r = ReadForm()
  PrintStatement r, True
End Sub
```

```
Private Sub cmdRecertify_Click()
    Dim r As StudentRecord
    r = ReadForm()
    SaveRecord r, "Recertification Required"
    MsgBox "Recertification case opened. QA Ref: " & r.QARef, vbInformation
End Sub

Private Sub cmdSave_Click()
    Dim r As StudentRecord
    r = ReadForm()
    ComputeScores r
    SaveRecord r, "Saved"
    MsgBox "Record saved.", vbInformation
End Sub



    Select Case st
        Case EligibleForRelease: StatusToText = "Eligible for Release"
        Case HoldIrregularity:   StatusToText = "Hold - Irregularity"
        Case HoldBacklog:        StatusToText = "Hold - Backlog"
        Case HoldQANotCleared:   StatusToText = "Hold - QA/Insurance Not Cleared"
        Case ReviewVariance:     StatusToText = "Hold - Internal/External Variance Review"
        Case RecertificationRequired: StatusToText = "Recertification Required"
        Case ReprintAllowed:     StatusToText = "Reprint Allowed"
        Case Else:               StatusToText = "Unknown"
    End Select
End Function
```

Code: saving, printing, and export

```
    ' Example: save to worksheet "Results"
    Dim ws As Worksheet, nextRow As Long
    Set ws = ThisWorkbook.Worksheets("Results")
    nextRow = ws.Cells(ws.Rows.count, "A").End(xlUp).row + 1

    ws.Cells(nextRow, 1).Value = r.StudentID
    ws.Cells(nextRow, 2).Value = r.Username
    ws.Cells(nextRow, 3).Value = r.Surname
    ws.Cells(nextRow, 4).Value = r.YOB
    ws.Cells(nextRow, 5).Value = r.AdminYear
    ws.Cells(nextRow, 6).Value = r.programme
    ws.Cells(nextRow, 7).Value = r.Level
    ws.Cells(nextRow, 8).Value = r.Trade
    ws.Cells(nextRow, 9).Value = r.internalScore
    ws.Cells(nextRow, 10).Value = r.externalScore
    ws.Cells(nextRow, 11).Value = r.finalScore
    ws.Cells(nextRow, 12).Value = AwardText(r.rating)
    ws.Cells(nextRow, 13).Value = r.ExamType
    ws.Cells(nextRow, 14).Value = r.Attempt
    ws.Cells(nextRow, 15).Value = r.BacklogCredits
    ws.Cells(nextRow, 16).Value = r.CombineWithID
```

```
    ws.Cells(nextRow, 17).Value = IIf(r.Irregularity, "Yes", "No")
    ws.Cells(nextRow, 18).Value = r.IrregularityNote
    ws.Cells(nextRow, 19).Value = IIf(r.QACleared, "Yes", "No")
    ws.Cells(nextRow, 20).Value = r.QARef
    ws.Cells(nextRow, 21).Value = IIf(r.ProctorIssue, "Yes", "No")
    ws.Cells(nextRow, 22).Value = r.MaterialBatch
    ws.Cells(nextRow, 23).Value = stateText
    ws.Cells(nextRow, 24).Value = Now
End Sub


    Dim txt As String, hdr As String
    hdr = IIf(isReprint, "REPRINTED RESULT STATEMENT", "RESULT STATEMENT")
    txt = hdr & vbCrLf & String(40, "-") & vbCrLf & _
        "Student: " & r.Username & " " & r.Surname & " | ID: " & r.StudentID & vbCrLf & _
        "YOB: " & r.YOB & " | Admin Year: " & r.AdminYear & vbCrLf & _
        "Programme: " & r.programme & " (L" & r.Level & ") | Trade: " & r.Trade & vbCrLf & vbCrLf & _
        "Internal: " & Format(r.internalScore, "0.0") & "/100" & vbCrLf & _
        "External: " & Format(r.externalScore, "0.0") & "/100" & vbCrLf & _
        "Final Rating: " & Format(r.rating, "0.0") & "% | Award: " & AwardText(r.rating) & vbCrLf & _
        "Exam: " & r.ExamType & " | Attempt: " & r.Attempt & vbCrLf & _
        "Backlog Credits: " & r.BacklogCredits & vbCrLf & _
        "QA Cleared: " & IIf(r.QACleared, "Yes", "No") & " | QA Ref: " & r.QARef & vbCrLf & _
        "Irregularity: " & IIf(r.Irregularity, "Yes", "No") & _
        IIf(r.Irregularity, " (" & r.IrregularityNote & ")", "") & vbCrLf & _
        "Material/Proctor Issue: " & IIf(r.ProctorIssue, "Yes", "No") & _
        IIf(r.ProctorIssue, " (" & r.MaterialBatch & ")", "")

    ' Simple preview
    MsgBox txt, vbInformation, "Print Preview"
    ' Replace with: export to a formatted sheet and print
End Sub
Optional: variance review and quality notes
Private Sub FlagVarianceNote(ByVal internalScore As Double, ByVal externalScore As Double)
    Dim variance As Double
    variance = Abs(internalScore - externalScore)
    If variance > VARIANCE_THRESHOLD Then
        txtIrregularityNote.text = "Variance " & Format(variance, "0.0") & "pp exceeds threshold; send to
moderation."
    End If
End Sub
Visual Basic framework for student portfolio clearance, attendance, finance, and printouts
Below is a practical VBA/VB6-style scaffold to manage student records, portfolio availability by prior years,
attendance, bursary and fee allocation, payroll-like study stipends, and printable statements. It also
includes a simple logigram flow.
UserForm Structure
'   Form name: frmClearance
'   Tabs: Identity | Portfolio | Attendance | Finance | Academics | Actions
Identity tab
'   TextBox: txtStudentID, txtUsername, txtSurname, txtFirstName, txtPassword
'   ComboBox: cboProgramme (Engineering courses), cboCourseID, cboExamYear
```

" Labels: lblStatus
Portfolio t(https://www.elektormagazine.com)
" CheckBox: chkPortfolioAvailable
" TextBox: txtPortfolioYears (comma-separated years, e.g., 2022,2023)
" ListBox: lstArtifacts (research papers, lab reports, workshop models)
" CommandButton: cmdAddArtifact, cmdRemoveArtifact
Attendance tab
" TextBox: txtDaysPresent4W, txtDaysPresent30D, txtDaysPresent360D
" TextBox: txtDaysOff, txtSchoolDaysAvailable
" Labels: lblAttendanceRate4W, lblAttendanceRate30D, lblAttendanceRate360D
Finance tab
" Group: Stipend/Salary-like items
o TextBox: txtDailyRate (default 100) 'rand/day
o TextBox: txtShiftDays , txtOffDays
o Labels: lblGrossPay
" Group: Deductions
o TextBox: txtDeduction (generic), txtInsuranceLevy, txtPortalFee
" Group: Benefits/Allocations
o TextBox: txtBonus , txtAccommodation, txtLibraryFee, txtClassFee, txtAllocationPay, txtLearningGrant
" Labels: lblNetPay
Academics tab
" TextBox: txtHomework, txtClasswork, txtPractical, txtExam, txtWorkshopModel, txtTradeLab, txtManufactureClaim, txtTenderValue, txtBudget
" Labels: lblTotal100, lblRatingPct, lblAward
Actions tab
" Buttons: cmdCalculate, cmdPrintIdentity, cmdPrintAttendance, cmdPrintFinance, cmdPrintAcademics, cmdSave, cmdClear, cmdClose
Core data model and utilities

```
Option Explicit

Private Type Student
    StudentID As String
    Username As String
    FirstName As String
    Surname As String
    Password As String
    programme As String
    CourseID As String
    ExamYear As Integer
End Type

Private Type Attendance
    DaysPresent4W As Double
    DaysPresent30D As Double
    DaysPresent360D As Double
    SchoolDaysAvailable As Double
    DaysOff As Double
End Type

Private Type Finance
    DailyRate As Double
```

```
    ShiftDays As Double
    OffDays As Double
    Deduction As Double
    InsuranceLevy As Double
    PortalFee As Double
    Bonus As Double
    Accommodation As Double
    LibraryFee As Double
    ClassFee As Double
    AllocationPay As Double
    LearningGrant As Double
    Gross As Double
    Net As Double
End Type

Private Type Academics
    Homework As Double
    Classwork As Double
    practical As Double
    Exam As Double
    WorkshopModel As Double
    TradeLab As Double
    ManufactureClaim As Double
    TenderValue As Double
    Budget As Double
    Total100 As Double
    RatingPct As Double
    Award As String
End Type

Private Const PASS50 As Double = 50#
Private Const COND40 As Double = 40#
Private Const FAIL20 As Double = 20#
Form readers And calculators

    Dim s As Student
    s.StudentID = Trim$(txtStudentID.text)
    s.Username = Trim$(txtUsername.text)
    s.FirstName = Trim$(txtFirstName.text)
    s.Surname = Trim$(txtSurname.text)
    s.Password = Trim$(txtPassword.text)
    s.programme = cboProgramme.text
    s.CourseID = cboCourseID.text
    s.ExamYear = Val(cboExamYear.text)
    ReadStudent = s
End Function



    Dim a As Attendance
    a.DaysPresent4W = Val(txtDaysPresent4W.text)
    a.DaysPresent30D = Val(txtDaysPresent30D.text)
```

```
    a.DaysPresent360D = Val(txtDaysPresent360D.text)
    a.SchoolDaysAvailable = Val(txtSchoolDaysAvailable.text)
    a.DaysOff = Val(txtDaysOff.text)
    ReadAttendance = a
End Function


    Dim f As Finance
    f.DailyRate = Val(txtDailyRate.text)
    f.ShiftDays = Val(txtShiftDays.text)
    f.OffDays = Val(txtOffDays.text)
    f.Deduction = Val(txtDeduction.text)
    f.InsuranceLevy = Val(txtInsuranceLevy.text)
    f.PortalFee = Val(txtPortalFee.text)
    f.Bonus = Val(txtBonus.text)
    f.Accommodation = Val(txtAccommodation.text)
    f.LibraryFee = Val(txtLibraryFee.text)
    f.ClassFee = Val(txtClassFee.text)
    f.AllocationPay = Val(txtAllocationPay.text)
    f.LearningGrant = Val(txtLearningGrant.text)
    ReadFinance = f
End Function


    Dim ac As Academics
    ac.Homework = Val(txtHomework.text)
    ac.Classwork = Val(txtClasswork.text)
    ac.practical = Val(txtPractical.text)
    ac.Exam = Val(txtExam.text)
    ac.WorkshopModel = Val(txtWorkshopModel.text)
    ac.TradeLab = Val(txtTradeLab.text)
    ac.ManufactureClaim = Val(txtManufactureClaim.text)
    ac.TenderValue = Val(txtTenderValue.text)
    ac.Budget = Val(txtBudget.text)
    ReadAcademics = ac
End Function


    If a.SchoolDaysAvailable <= 0 Then a.SchoolDaysAvailable = 360
    lblAttendanceRate4W.Caption = Format(100 * a.DaysPresent4W / 20, "0.0") & "%"
    lblAttendanceRate30D.Caption = Format(100 * a.DaysPresent30D / 30, "0.0") & "%"
    lblAttendanceRate360D.Caption = Format(100 * a.DaysPresent360D / a.SchoolDaysAvailable, "0.0") &
"%"
End Sub


    f.Gross = f.DailyRate * f.ShiftDays
    Dim totalDeductions As Double
    totalDeductions = f.Deduction + f.InsuranceLevy + f.PortalFee + f.LibraryFee + f.ClassFee
    Dim totalBenefits As Double
    totalBenefits = f.Bonus + f.Accommodation + f.AllocationPay + f.LearningGrant
```

```
    f.Net = f.Gross - totalDeductions + totalBenefits
    lblGross.Caption = "R " & Format(f.Gross, "0,0.00")
    lblNetPay.Caption = "R " & Format(f.Net, "0,0.00")
End Sub


    ' Normalize to 100: Homework(15) + Classwork(15) + Practical(20) + Exam(50)
    Dim total As Double
    total = ac.Homework + ac.Classwork + ac.practical + ac.Exam
    ac.Total100 = total
    ac.RatingPct = total ' already out of 100 if inputs constrained
    ac.Award = AwardFromPct(ac.RatingPct)
    lblTotal100.Caption = Format(ac.Total100, "0.0") & " / 100"
    lblRatingPct.Caption = Format(ac.RatingPct, "0.0") & "%"
    lblAward.Caption = ac.Award
End Sub

Private Function AwardFromPct(ByVal pct As Double) As String
    If pct >= 85 Then
        AwardFromPct = "Distinction"
    ElseIf pct >= 70 Then
        AwardFromPct = "Merit"
    ElseIf pct >= PASS50 Then
        AwardFromPct = "Pass"
    ElseIf pct >= COND40 Then
        AwardFromPct = "Borderline"
    ElseIf pct >= FAIL20 Then
        AwardFromPct = "Fail"
    Else
        AwardFromPct = "Severe Fail"
    End If
End Function

    Dim a As Attendance, f As Finance, ac As Academics
    a = ReadAttendance(): Call CalcAttendance(a)
    f = ReadFinance():    Call CalcFinance(f)
    ac = ReadAcademics(): Call CalcAcademics(ac)
    lblStatus.Caption = "Calculated at " & Format(Now, "yyyy-mm-dd hh:nn")
End Sub

    Dim ctl As Control
    For Each ctl In Me.Controls
        Select Case TypeName(ctl)
            Case "TextBox": ctl.text = ""
            Case "Label"
                If ctl.Name Like "lbl*" Then ctl.Caption = ""
        End Select
    Next ctl
    chkPortfolioAvailable.Value = False
    lstArtifacts.Clear
    lblStatus.Caption = "Cleared"
```

End Sub


```
    Dim s As Student, a As Attendance, f As Finance, ac As Academics
    s = ReadStudent(): a = ReadAttendance(): f = ReadFinance(): ac = ReadAcademics()
    SaveToSheet s, a, f, ac
    lblStatus.Caption = "Saved at " & Format(Now, "yyyy-mm-dd hh:nn")
End Sub


    Dim s As Student: s = ReadStudent()
    Dim txt As String
    txt = "STUDENT IDENTITY" & vbCrLf & String(40, "-") & vbCrLf & _
          "ID: " & s.StudentID & vbCrLf & _
          "Name: " & s.FirstName & " " & s.Surname & vbCrLf & _
          "Username: " & s.Username & vbCrLf & _
          "Programme: " & s.programme & " | Course ID: " & s.CourseID & vbCrLf & _
          "Exam Year: " & s.ExamYear
    MsgBox txt, vbInformation, "Print Preview"
End Sub

    Dim a As Attendance: a = ReadAttendance()
    Dim txt As String
    txt = "ATTENDANCE SUMMARY" & vbCrLf & String(40, "-") & vbCrLf & _
          "4 Weeks Present: " & a.DaysPresent4W & " (" & lblAttendanceRate4W.Caption & ")" & vbCrLf & _
          "30 Days Present: " & a.DaysPresent30D & " (" & lblAttendanceRate30D.Caption & ")" & vbCrLf & _
          "360 Days Present: " & a.DaysPresent360D & " (" & lblAttendanceRate360D.Caption & ")" & vbCrLf & _
          "Days Off: " & a.DaysOff & " | School Days: " & a.SchoolDaysAvailable
    MsgBox txt, vbInformation, "Print Preview"
End Sub


    Dim f As Finance: f = ReadFinance(): Call CalcFinance(f)
    Dim txt As String
    txt = "FINANCE SUMMARY" & vbCrLf & String(40, "-") & vbCrLf & _
          "Daily Rate: R " & Format(f.DailyRate, "0,0.00") & vbCrLf & _
          "Shift Days: " & f.ShiftDays & " | Off Days: " & f.OffDays & vbCrLf & _
          "Gross: " & lblGrossPay.Caption & vbCrLf & _
          "Deductions (incl. insurance/portal/library/class): R " & _
          Format(f.Deduction + f.InsuranceLevy + f.PortalFee + Val(txtLibraryFee.text) + Val(txtClassFee.text),
"0,0.00") & vbCrLf & _
          "Benefits (bonus/accommodation/allocation/grant): R " & _
          Format(f.Bonus + f.Accommodation + f.AllocationPay + f.LearningGrant, "0,0.00") & vbCrLf & _
          "Net: " & lblNetPay.Caption
    MsgBox txt, vbInformation, "Print Preview"
End Sub


    Dim ac As Academics: ac = ReadAcademics(): Call CalcAcademics(ac)
    Dim txt As String
    txt = "ACADEMIC MARKSHEET" & vbCrLf & String(40, "-") & vbCrLf & _
```

```
        "Homework: " & ac.Homework & "/15" & vbCrLf & _
        "Classwork: " & ac.Classwork & "/15" & vbCrLf & _
        "Practical/Lab: " & ac.practical & "/20" & vbCrLf & _
        "Exam: " & ac.Exam & "/50" & vbCrLf & _
        "Total: " & lblTotal100.Caption & " | Rating: " & lblRatingPct.Caption & vbCrLf & _
        "Award: " & lblAward.Caption & vbCrLf & _
        "Workshop Model: " & ac.WorkshopModel & " | Trade Lab: " & ac.TradeLab & vbCrLf & _
        "Manufacture Claim: " & ac.ManufactureClaim & " | Tender Value: " & ac.TenderValue & vbCrLf & _
        "Budget: " & ac.Budget
    MsgBox txt, vbInformation, "Print Preview"
End Sub
)
    Dim ws As Worksheet, r As Long
    Set ws = ThisWorkbook.Worksheets("ClearanceRecords")
    r = ws.Cells(ws.Rows.count, "A").End(xlUp).row + 1

    ws.Cells(r, 1).Value = s.StudentID
    ws.Cells(r, 2).Value = s.Username
    ws.Cells(r, 3).Value = s.FirstName
    ws.Cells(r, 4).Value = s.Surname
    ws.Cells(r, 5).Value = s.programme
    ws.Cells(r, 6).Value = s.CourseID
    ws.Cells(r, 7).Value = s.ExamYear
    ws.Cells(r, 8).Value = a.DaysPresent4W
    ws.Cells(r, 9).Value = a.DaysPresent30D
    ws.Cells(r, 10).Value = a.DaysPresent360D
    ws.Cells(r, 11).Value = a.SchoolDaysAvailable
    ws.Cells(r, 12).Value = a.DaysOff
    ws.Cells(r, 13).Value = f.DailyRate
    ws.Cells(r, 14).Value = f.ShiftDays
    ws.Cells(r, 15).Value = f.OffDays
    ws.Cells(r, 16).Value = f.Gross
    ws.Cells(r, 17).Value = f.Deduction
    ws.Cells(r, 18).Value = f.InsuranceLevy
    ws.Cells(r, 19).Value = f.PortalFee
    ws.Cells(r, 20).Value = f.Bonus
    ws.Cells(r, 21).Value = f.Accommodation
    ws.Cells(r, 22).Value = f.LibraryFee
    ws.Cells(r, 23).Value = f.ClassFee
    ws.Cells(r, 24).Value = f.AllocationPay
    ws.Cells(r, 25).Value = f.LearningGrant
    ws.Cells(r, 26).Value = f.Net
    ws.Cells(r, 27).Value = ac.Homework
    ws.Cells(r, 28).Value = ac.Classwork
    ws.Cells(r, 29).Value = ac.practical
    ws.Cells(r, 30).Value = ac.Exam
    ws.Cells(r, 31).Value = ac.Total100
    ws.Cells(r, 32).Value = ac.RatingPct
    ws.Cells(r, 33).Value = ac.Award
    ws.Cells(r, 34).Value = Now
End Sub
```

Minimal module to show the form
frmClearance.Show
End Sub
Logigram Overview
" Start
" Identity validation
o If missing fields: Halt ? Prompt to complete
" Portfolio check
o If available: Load artifacts ? Proceed
o If not: Flag "Pending Portfolio"
" Attendance calculation
o If < 75% (360-day rate): Flag "Attendance Risk"
" Finance calculation
o Compute gross ? apply deductions ? add allocations ? net
" Academics aggregation
o Compute total/100 ? rating ? award
" Clearance gate
o If award Pass and attendance ? 75%: Clearance Approved
o Else: Clearance Conditional/Pending
" Save + Print statements
o Identity | Attendance | Finance | Academics
" End
Notes and next steps
" Field caps: For academic inputs, constrain to their maximums (e.g., Homework ? 15) to preserve 100 total.
" Attendance presets: 4 weeks = 20 days; 30 days; year = 360 days; adjust if your calendar differs.
" Currency: Values shown in R (rand). Adapt rates and fee items as needed.
" Data protection: Avoid storing plaintext passwords; use masked fields or exclude from saving.
" Enhancements: Export to a formatted sheet/PDF with institutional branding and a QR for verification.
If you share your exact environment (Excel VBA vs. VB6 vs. VB.NET (http://VB.NET)), I'll tailor control properties, add input validation masks, and provide a printable marks
Absolutely , Tshingombe.Here 's a structured Visual Basic (VBA) framework tailored to your advanced certification and qualification logic, aligned with SAQA, UCPD/UCD, and trade-based assessment systems. This model supports:
" Certificate granting based on test/class performance
" Replacement or exemption of marks
" SAQA qualification mapping
" Final job evaluation and employment scoring
" Print-ready certificate and diploma logic
" Multi-phase award logic (1st-4th phase)
" Degree, diploma, and postgraduate recognition
?? UserForm Structure: frmCertification
?? Identity & Qualification Tab
Control Name Purpose
TextBox txtStudentID Unique learner ID
TextBox txtStudentName Full name
TextBox txtSurname Surname
TextBox txtLogin System login
TextBox txtPassword Masked password
ComboBox cboTrade Trade (e.g., Electrical, Mechanical)

ComboBox cboQualificationType NDip, BTech, UCPD, UCD, Master, Doctoral
TextBox txtQualificationTitle Qualification title
TextBox txtQualificationID Internal Qualification ID
ComboBox cboAssessor Assigned assessor
ComboBox cboPhase Final Phase (1st-4th)
?? Assessment & Exemption Tab
Field Controls
Subject Name txtSubjectName
Course ID txtCourseID
Test Score txtTestScore
Exam Score txtExamScore
exempted chkExempted
Replacement Score txtReplacementScore
Minimum Required txtMinMark
Maximum Allowed txtMaxMark
Meets Requirement lblMeetsRequirement (calculated)
Award Status lblAwardStatus (calculated)
?? Employment & Job Evaluation Tab
Field Controls
Job Function txtJobFunction
Log Activity lstActivityLog
Employment Duration txtYearsWorked (e.g., 2 years)
Working Days txtDaysWorked (e.g., 30 days)
Final Score lblFinalScore (calculated)
Employment Award lblEmploymentAward (calculated)
?? Certificate & Diploma Tab
Button Function
cmdPrintCertificate Print SAQA Certificate
cmdPrintDiploma Print SAQA Diploma
cmdEvaluateAward Evaluate qualification and award
cmdSaveRecord Save to sheet
cmdClearForm Reset form
cmdCloseForm Exit
?? Core Logic: Award Evaluation
vb

```
Dim testScore As Double, examScore As Double, replacementScore As Double
Dim exempted As Boolean, minMark As Double, maxMark As Double
Dim finalScore As Double, meetsRequirement As Boolean

testScore = Val(txtTestScore.text)
examScore = Val(txtExamScore.text)
replacementScore = Val(txtReplacementScore.text)
exempted = chkExempted.Value
minMark = Val(txtMinMark.text)
maxMark = Val(txtMaxMark.text)

If exempted Then
    finalScore = replacementScore
Else
    finalScore = (testScore + examScore) / 2
```

```
    End If
    lblFinalScore.Caption = Format(finalScore, "0.0")

    If finalScore >= minMark And finalScore <= maxMark Then
        lblMeetsRequirement.Caption = "Yes"
        lblAwardStatus.Caption = "Eligible for Certificate"
    Else
        lblMeetsRequirement.Caption = "No"
        lblAwardStatus.Caption = "Not Eligible"
    End If
End Sub
??? Certificate & Diploma Print Logic
vb

    Dim txt As String
    txt = "SAQA CERTIFICATE OF COMPETENCE" & vbCrLf & String(40, "-") & vbCrLf & _
          "Student: " & txtStudentName.text & " " & txtSurname.text & vbCrLf & _
          "Trade: " & cboTrade.text & vbCrLf & _
          "Qualification: " & cboQualificationType.text & vbCrLf & _
          "SAQA ID: " & txtSAQAID.text & " | Internal ID: " & txtQualificationID.text & vbCrLf & _
          "Assessor: " & cboAssessor.text & " | Phase: " & cboPhase.text & vbCrLf & _
          "Final Score: " & lblFinalScore.Caption & " | Award Status: " & lblAwardStatus.Caption
    MsgBox txt, vbInformation, "Certificate Preview"
End Sub



    Dim txt As String
    txt = "SAQA DIPLOMA STATEMENT" & vbCrLf & String(40, "-") & vbCrLf & _
          "Course: " & txtSubjectName.text & " | Course ID: " & txtCourseID.text & vbCrLf & _
          "Test: " & txtTestScore.text & " | Exam: " & txtExamScore.text & vbCrLf & _
          "Exempted: " & IIf(chkExempted.Value, "Yes", "No") & _
          IIf(chkExempted.Value, " | Replacement: " & txtReplacementScore.text, "") & vbCrLf & _
          "Final Score: " & lblFinalScore.Caption & " | Meets Requirement: " & lblMeetsRequirement.Caption
    MsgBox txt, vbInformation, "Diploma Preview"
End Sub
?? Employment Score Logic
vb

    Dim yearsWorked As Double, daysWorked As Double
    Dim employmentScore As Double

    yearsWorked = Val(txtYearsWorked.text)
    daysWorked = Val(txtDaysWorked.text)

    employmentScore = (yearsWorked * 360 + daysWorked) / 720 ' Normalize to 2-year scale
    lblEmploymentAward.Caption = IIf(employmentScore >= 1, "Eligible for Job Certification", "Insufficient
Duration")
End Sub
[Start]
  Print
```

[Identity Verified]
  Print
[Assessment Complete?] ? No ? [Hold]
  Print Yes
[Exempted or Replaced?] ? Yes ? [Use Replacement Score]
  Print No
[Calculate Final Score]
  Print
[Meets Min/Max?] ? No ? [Reject]
  Print Yes
[Employment Duration ? 2 years?] ? No ? [Issue Statement Only]
  Print Yes
[Print Certificate + Diploma]
  Print
[Save Record]
  Print
End
?? Optional Enhancements
" SAQA XML export: For qualification registry submission
" QR code on certificate: For verification
" Assessor dashboard: Track phase completions and exemptions
" UCPD/UCD logic: Map development capacity to trade replacement pathways
" Project-based override: If no exam but project completed, allow award with assessor validation
Would you like this scaffolded into a full Excel VBA workbook with certificate templates, logigram visuals, and SAQA-aligned qualification sheets? I can also help you build exemption logic for trade replacement and automate assessor workflows.

Absolutely , Tshingombe.Here 's a structured Visual Basic (VBA) framework tailored to managing trade license certification, test results, work permit conditions, and council-aligned engineering records. This model supports:
" Trade theory + practical test tracking
" License certificate issuance
" Work permit and conduct conditions
" Dismissal, reward, and amendment logic
" Multi-year record logging
" Council registration and qualification mapping
?? UserForm Structure: frmTradeLicense
?? Identity & Trade Info Tab
Control Name Purpose
TextBox txtStudentID Unique learner ID
TextBox txtFullName Full name
ComboBox cboTradeType Trade (Electrical, Mechanical, etc.)
ComboBox cboCouncil Engineering Council (e.g., ECSA)
TextBox txtCouncilRegID Council registration number
ComboBox cboLicenseType Theory, Practical, Combined
TextBox txtLicenseID Generated license number
?? Test & Certification Tab
Field Controls
Theory Score txtTheoryScore
Practical Score txtPracticalScore
Total Score lblTotalScore (calculated)

Testify Status chkTestify (passed witness verification)
Certificate Issued chkCertificateIssued
Certificate Date txtCertificateDate
License Validity (Years) txtLicenseYears
License Expiry lblLicenseExpiry (calculated)
?? Work Permit & Conduct Tab
Field Controls
Work Permit ID txtWorkPermitID
Permit Conditions txtPermitConditions
Conduct Status cboConduct (Good, Warning, Dismissed)
Dismissal Reason txtDismissalReason
Reward Points txtRewardPoints
Amendment Notes txtAmendmentNotes
?? Record & Council Tab
Field Controls
Record Year txtRecordYear
Amendment Year txtAmendmentYear
Qualification Level cboQualificationLevel (NDip, BTech, Trade Cert)
Council Status lblCouncilStatus (calculated)
Final Status lblFinalStatus (calculated)
?? Buttons
Button Function
cmdCalculate Compute scores, expiry, council status
cmdPrintLicense Print license certificate
cmdSaveRecord Save to sheet
cmdClearForm Reset form
cmdCloseForm Exit
?? Core Logic: License Evaluation

```
    Dim theory As Double, practical As Double, total As Double
    Dim licenseYears As Integer, expiryDate As Date

    theory = Val(txtTheoryScore.text)
    practical = Val(txtPracticalScore.text)
    total = (theory + practical) / 2
    lblTotalScore.Caption = Format(total, "0.0")

    If total >= 50 And chkTestify.Value = True Then
        chkCertificateIssued.Value = True
        txtCertificateDate.text = Format(Date, "yyyy-mm-dd")
        licenseYears = Val(txtLicenseYears.text)
        expiryDate = DateAdd("yyyy", licenseYears, Date)
        lblLicenseExpiry.Caption = Format(expiryDate, "yyyy-mm-dd")
        lblFinalStatus.Caption = "License Granted"
    Else
        lblFinalStatus.Caption = "License Denied"
    End If

    ' Council logic
    If cboCouncil.text <> "" And txtCouncilRegID.text <> "" Then
        lblCouncilStatus.Caption = "Registered with " & cboCouncil.text
```

```
    Else
        lblCouncilRegStatus.Caption = "Not Registered"
    End If
End Sub
??? License Certificate Print Logic
vb

    Dim txt As String
    txt = "TRADE LICENSE CERTIFICATE" & vbCrLf & String(40, "-") & vbCrLf & _
          "Name: " & txtFullName.text & vbCrLf & _
          "Trade: " & cboTradeType.text & vbCrLf & _
          "License Type: " & cboLicenseType.text & vbCrLf & _
          "License ID: " & txtLicenseID.text & vbCrLf & _
          "Theory Score: " & txtTheoryScore.text & " | Practical Score: " & txtPracticalScore.text & vbCrLf & _
          "Total: " & lblTotalScore.Caption & vbCrLf & _
          "Certificate Issued: " & IIf(chkCertificateIssued.Value, "Yes", "No") & _
          " | Expiry: " & lblLicenseExpiry.Caption & vbCrLf & _
          "Council: " & cboCouncil.text & " | Reg ID: " & txtCouncilRegID.text & vbCrLf & _
          "Final Status: " & lblFinalStatus.Caption
    MsgBox txt, vbInformation, "License Preview"
End Sub
Conduct & Work Permit Logic

    If cboConduct.text = "Dismissed" Then
        lblFinalStatus.Caption = "License Revoked"
    ElseIf cboConduct.text = "Warning" Then
        lblFinalStatus.Caption = "License Under Review"
    Else
        lblFinalStatus.Caption = "License Active"
    End If
End Sub
?? Logigram Flow
Code
[Start]
  Print
[Identity Verified]
  Print
[Test Scores Entered]
  Print
[Testify Passed?] ? No ? [Hold License]
  Print Yes
[Total Score ? 50?] ? No ? [Reject]
  Print Yes
[Certificate Issued]
  Print
[Work Permit Conditions Met?] ? No ? [Flag]
  Print Yes
[Conduct Status?]
  ? Dismissed ? [Revoke License]
  ? Warning ? [Review]
  ? Good ? [License Active]
```

Print
[Council R[illegible]] Elektor MAG (https://www.elektormagazine.com)
Print Yes
[Print Certificate]
Print
[Save Record]
Print
End
Optional Enhancements
" QR code on certificate for verification
" Council API integration for registration validation
" Amendment tracking with version history
" Reward system for high performers (e.g., ?85%)
" Dismissal audit trail with timestamp and assessor notes
Would you like this scaffolded into a full Excel VBA workbook with certificate templates,

'
' engitraderegistrmark Macro
'
'

End Sub

Absolutely , Tshingombe.Here 's a structured Visual Basic (VBA) framework tailored to managing trade license certification, test results, work permit conditions, and council-aligned engineering records. This model supports:
" Trade theory + practical test tracking
" License certificate issuance
" Work permit and conduct conditions
" Dismissal, reward, and amendment logic
" Multi-year record logging
" Council registration and qualification mapping
?? UserForm Structure: frmTradeLicense
?? Identity & Trade Info Tab
Control Name Purpose
TextBox txtStudentID Unique learner ID
TextBox txtFullName Full name
ComboBox cboTradeType Trade (Electrical, Mechanical, etc.)
ComboBox cboCouncil Engineering Council (e.g., ECSA)
TextBox txtCouncilRegID Council registration number
ComboBox cboLicenseType Theory, Practical, Combined
TextBox txtLicenseID Generated license number
?? Test & Certification Tab
Field Controls
Theory Score txtTheoryScore
Practical Score txtPracticalScore
Total Score lblTotalScore (calculated)
Testify Status chkTestify (passed witness verification)
Certificate Issued chkCertificateIssued

Certificate Date txtCertificateDate
License Valid (Yes/No) chkLicenseValid
License Expiry lblLicenseExpiry (calculated)
?? Work Permit & Conduct Tab
Field Controls
Work Permit ID txtWorkPermitID
Permit Conditions txtPermitConditions
Conduct Status cboConduct (Good, Warning, Dismissed)
Dismissal Reason txtDismissalReason
Reward Points txtRewardPoints
Amendment Notes txtAmendmentNotes
?? Record & Council Tab
Field Controls
Record Year txtRecordYear
Amendment Year txtAmendmentYear
Qualification Level cboQualificationLevel (NDip, BTech, Trade Cert)
Council Status lblCouncilStatus (calculated)
Final Status lblFinalStatus (calculated)
?? Buttons
Button Function
cmdCalculate Compute scores, expiry, council status
cmdPrintLicense Print license certificate
cmdSaveRecord Save to sheet
cmdClearForm Reset form
cmdCloseForm Exit
?? Core Logic: License Evaluation

```
  Dim theory As Double, practical As Double, total As Double
  Dim licenseYears As Integer, expiryDate As Date

  theory = Val(txtTheoryScore.text)
  practical = Val(txtPracticalScore.text)
  total = (theory + practical) / 2
  lblTotalScore.Caption = Format(total, "0.0")

  If total >= 50 And chkTestify.Value = True Then
    chkCertificateIssued.Value = True
    txtCertificateDate.text = Format(Date, "yyyy-mm-dd")
    licenseYears = Val(txtLicenseYears.text)
    expiryDate = DateAdd("yyyy", licenseYears, Date)
    lblLicenseExpiry.Caption = Format(expiryDate, "yyyy-mm-dd")
    lblFinalStatus.Caption = "License Granted"
  Else
    lblFinalStatus.Caption = "License Denied"
  End If

  ' Council logic
  If cboCouncil.text <> "" And txtCouncilRegID.text <> "" Then
    lblCouncilStatus.Caption = "Registered with " & cboCouncil.text
  Else
    lblCouncilStatus.Caption = "Not Registered"
```

```
    End If
End Sub
```

??? License Certificate Print Logic

```
vb
Private Sub cmdPrintLicense_Click()
    Dim txt As String
    txt = "TRADE LICENSE CERTIFICATE" & vbCrLf & String(40, "-") & vbCrLf & _
          "Name: " & txtFullName.text & vbCrLf & _
          "Trade: " & cboTradeType.text & vbCrLf & _
          "License Type: " & cboLicenseType.text & vbCrLf & _
          "License ID: " & txtLicenseID.text & vbCrLf & _
          "Theory Score: " & txtTheoryScore.text & " | Practical Score: " & txtPracticalScore.text & vbCrLf & _
          "Total: " & lblTotalScore.Caption & vbCrLf & _
          "Certificate Issued: " & IIf(chkCertificateIssued.Value, "Yes", "No") & _
          " | Expiry: " & lblLicenseExpiry.Caption & vbCrLf & _
          "Council: " & cboCouncil.text & " | Reg ID: " & txtCouncilRegID.text & vbCrLf & _
          "Final Status: " & lblFinalStatus.Caption
    MsgBox txt, vbInformation, "License Preview"
End Sub
```

Conduct & Work Permit Logic

```
Private Sub EvaluateConduct()
    If cboConduct.text = "Dismissed" Then
        lblFinalStatus.Caption = "License Revoked"
    ElseIf cboConduct.text = "Warning" Then
        lblFinalStatus.Caption = "License Under Review"
    Else
        lblFinalStatus.Caption = "License Active"
    End If
End Sub
```

?? Logigram Flow

Code

```
[Start]
  Print
[Identity Verified]
  Print
[Test Scores Entered]
  Print
[Testify Passed?] ? No ? [Hold License]
  Print Yes
[Total Score ? 50?] ? No ? [Reject]
  Print Yes
[Certificate Issued]
  Print
[Work Permit Conditions Met?] ? No ? [Flag]
  Print Yes
[Conduct Status?]
  ? Dismissed ? [Revoke License]
  ? Warning ? [Review]
  ? Good ? [License Active]
  Print
[Council Registered?] ? No ? [Flag]
```

Print Yes
[Print Cert elektor MAG (https://www.elektormagazine.com)
Print
[Save Record]
Print
End
Optional Enhancements
" QR code on certificate for verification
" Council API integration for registration validation
" Amendment tracking with version history
" Reward system for high performers (e.g., ?85%)
" Dismissal audit trail with timestamp and assessor notes
Would you like this scaffolded into a full Excel VBA workbook with certificate templates, council registry sheets, and logigram visuals? I can also help you automate license expiry
Thanks for sharing this extensive curriculum breakdown from NICHRIST Institute of Management and Engineering Technology (NIMET). Based on your request, it sounds like you're looking to generate a Visual Basic (VB) program that can produce a release result statement for a candidate enrolled in one of these N-level engineering programs. Let's break this down into a modular VB concept that aligns with your curriculum architecture and portfolio logic.
?? Objective
Create a Visual Basic application that:
" Inputs candidate details and subject marks
" Maps subjects to modules (N1-N3)
" Calculates results based on a mark scale
" Outputs a formatted Result Statement for printing or digital archiving
?? Core Modules in the VB Program
1. Data Structures
Use Class or Structure to define:
vb
```
Public Class Subject
    Public Property Code As String
    Public Property Name As String
    Public Property Mark As Integer
End Class

Public Class ModuleRecord
    Public Property CandidateName As String
    Public Property IDNumber As String
    Public Property CourseName As String
    Public Property Level As String ' N1, N2, N3
    Public Property Subjects As List(Of Subject)
End Class
roperty Subjects As List(Of Subject)
End Class
```
2. Mark Scale Logic
You can define a function to convert marks to result statu
```
Public Function GetResultStatus(mark As Integer) As String
    If mark >= 75 Then
        Return "Distinction"
    ElseIf mark >= 50 Then
        Return "Pass"
```

```
    Else
        Return
    End If
End Function
```

3. Result Statement Generator

```
vb

    Console.WriteLine ("NICHRIST INSTITUTE OF MANAGEMENT AND ENGINEERING TECHNOLOGY
(NIMET)")
    Console.WriteLine ("Candidate Name: " & Record.CandidateName)
    Console.WriteLine ("ID Number: " & Record.IDNumber)
    Console.WriteLine ("Course: " & Record.CourseName & " " & Record.Level)
    Console.WriteLine ("----------------------------------------------")
    Console.WriteLine ("Subject Code | Subject Name | Mark | Result")
    For Each subj In Record.subjects
        Console.WriteLine($"{subj.Code} | {subj.Name} | {subj.Mark} | {GetResultStatus(subj.Mark)}")
    Next
    Console.WriteLine ("----------------------------------------------")
End Sub
Dim record As New ModuleRecord With {
    .CandidateName = "Tshingombe Tshitadi Fiston",
    .IDNumber = "1234567890",
    .CourseName = "Mechanical Engineering",
    .Level = "N2",
    .Subjects = New List(Of Subject) From {
        New Subject With {.Code = "16030192", .Name = "Mathematics", .Mark = 78},
        New Subject With {.Code = "15070402", .Name = "Engineering Science", .Mark = 65},
        New Subject With {.Code = "8090272", .Name = "Engineering Drawing", .Mark = 55},
        New Subject With {.Code = "11022032", .Name = "Fitting & Machining", .Mark = 48}
    }
}

GenerateResultStatement (Record)
NICHRIST INSTITUTE OF MANAGEMENT AND ENGINEERING TECHNOLOGY (NIMET)
Candidate Name: Tshingombe Tshitadi Fiston
ID Number: 1234567890
Course: Mechanical Engineering N2/n3/n4/n5/n6
----------------------------------------------
Subject Code | Subject Name          | Mark | Result
16030192     | Mathematics           | 78   | Distinction
15070402     | Engineering Science   | 65   | Pass
8090272      | Engineering Drawing   | 55   | Pass
11022032     | Fitting & Machining   | 60   | pass
----------------------------------------------
Would you like me to extend this into a GUI version using Windows Forms or integrate it with a database
for long-term record keeping
Visual Basic result statement and certificate engine for N certificates, N diplomas, and trade theory
equivalences
you 're building something bigger than a marksheet-you're building an auditable pathway from subject-level
evidence to certification and trade test readiness. Below is a compact, production-ready VB.NET (http://
VB.NET) core you can drop into a console or WinForms app. It covers result statements, N-certificate
```

combination checks (N1-N3), N-diploma pathway scaffolding (N4-N6), and theory "replacement/ equivalence" handling across Diesel/Electrical/Mechanical/Engineering/Plant Operations/ Industrial Orientation.

Scope and outputs

" Result statement: Candidate + program + subjects + marks + pass band + remarks.
" Certificate check: N1-N3 4-subject combination validation per level/stream.
" N diploma tracker: N4-N6 accumulation, configurable practice/work-experience requirement.
" Equivalence handling: Declare acceptable replacements (e.g., Engineering Drawing OR Industrial Electronics) and cross-stream theory equivalence (e.g., Diesel Trade Theory vs Electrical Trade Theory where policy allows).
" Replacement letter: Formal letter for theory replacement/credit transfer decisions.

Data model And rules

" Pass bands: Configurable (default: Pass ? 50; Distinction ? 75).
" N certificates (N1-N3): Require 4 passed subjects in the same stream and level.
" N diploma (N4-N6): Track subjects across N4-N6; experience requirement configurable; emit "Progress," "Eligible," or "Pending Experience."
" Equivalence: Declare canonical subject with a set of acceptable alternates; use when validating combinations.

Tip: Keep policy strings external (JSON/DB) so you can align wording with SAQA/QCTO/UMALUSI communications without code changes.

VB.NET (http://VB.NET) core (drop-in)

```
Imports System.text

' ---------- Domain ----------
Public Class Subject
    Public Property Code As String
    Public Property Name As String
    Public Property Level As String  ' N1..N6
    Public Property Stream As String ' e.g., Diesel Trade Engineering, Motor Trade Engineering
    Public Property Mark As Integer  ' -1 for not-yet-assessed (optional)
End Class

Public Class Enrollment
    Public Property CandidateName As String
    Public Property CandidateID As String
    Public Property Provider As String
    Public Property Stream As String       ' e.g., Diesel Trade Engineering
    Public Property Level As String        ' e.g., N3
    Public Property Subjects As List(Of Subject) = New List(Of Subject)
    Public Property Session As String      ' e.g., 2025 Trimester 1
End Class

Public Class EquivalenceRule
    Public Property CanonicalCode As String
    Public Property CanonicalName As String
    Public Property AcceptableCodes As HashSet(Of String) = New HashSet(Of String)
End Class

Public Class ProgramRules
    Public Property PassThreshold As Integer = 50
    Public Property DistinctionThreshold As Integer = 75
```

```
    Public Property RequiredSubjectsPerLevel As Integer = 4
    Public Property EquivalenceRules As List(Of EquivalenceRule) = New List(Of EquivalenceRule)

    ' N4-N6 diploma tracking (configurable)
    Public Property DiplomaRequiredSubjects As Integer = 12   ' typical N4-N6 total
    Public Property DiplomaExperienceMonths As Integer = 18   ' configurable per policy
End Class

' ---------- Catalog (partial; extend as needed) ----------
Public Module Catalog
    Public Function DieselN3(stream As String) As List(Of Subject)
        Return New List(Of Subject) From {
            New Subject With {.Code = "16030143", .Name = "Mathematics", .Level = "N3", .Stream = stream},
            New Subject With {.Code = "15070413", .Name = "Engineering Science", .Level = "N3", .Stream =
stream},
            New Subject With {.Code = "8090283",  .Name = "Engineering Drawing", .Level = "N3", .Stream =
stream},
            New Subject With {.Code = "8080613",  .Name = "Industrial Electronics", .Level = "N3", .Stream =
stream},
            New Subject With {.Code = "11041823", .Name = "Diesel Trade Theory", .Level = "N3", .Stream =
stream}
        }
    End Function

    Public Function MotorN1(stream As String) As List(Of Subject)
        Return New List(Of Subject) From {
            New Subject With {.Code = "16030121", .Name = "Mathematics", .Level = "N1", .Stream = stream},
            New Subject With {.Code = "15070391", .Name = "Engineering Science", .Level = "N1", .Stream =
stream},
            New Subject With {.Code = "8090261",  .Name = "Engineering Drawing", .Level = "N1", .Stream =
stream},
            New Subject With {.Code = "8080641",  .Name = "Industrial Electronics", .Level = "N1", .Stream =
stream},
            New Subject With {.Code = "11040651", .Name = "Motor Trade Theory", .Level = "N1", .Stream =
stream}
        }
    End Function

    Public Function MotorN2(stream As String) As List(Of Subject)
        Return New List(Of Subject) From {
            New Subject With {.Code = "16030192", .Name = "Mathematics", .Level = "N2", .Stream = stream},
            New Subject With {.Code = "15070402", .Name = "Engineering Science", .Level = "N2", .Stream =
stream},
            New Subject With {.Code = "8090272",  .Name = "Engineering Drawing", .Level = "N2", .Stream =
stream},
            New Subject With {.Code = "8080602",  .Name = "Industrial Electronics", .Level = "N2", .Stream =
stream},
            New Subject With {.Code = "11040662", .Name = "Motor Trade Theory", .Level = "N2", .Stream =
stream}
        }
    End Function
```

```
    Public Function GetSubjects(stream As String) As List(Of Subject)
        Return New List(Of Subject) From {
            New Subject With {.Code = "16030143", .Name = "Mathematics", .Level = "N3", .Stream = stream},
            New Subject With {.Code = "15070413", .Name = "Engineering Science", .Level = "N3", .Stream =
stream},
            New Subject With {.Code = "8090283",  .Name = "Engineering Drawing", .Level = "N3", .Stream =
stream},
            New Subject With {.Code = "8080613",  .Name = "Industrial Electronics", .Level = "N3", .Stream =
stream},
            New Subject With {.Code = "11040673", .Name = "Motor Trade Theory", .Level = "N3", .Stream =
stream}
        }
    End Function

    Public Function IndustrialOrientation() As List(Of Subject)
        Return New List(Of Subject) From {
            New Subject With {.Code = "4110011",  .Name = "Industrial Orientation", .Level = "N1", .Stream =
"Cross-Stream"},
            New Subject With {.Code = "4110022",  .Name = "Industrial Orientation", .Level = "N2", .Stream =
"Cross-Stream"},
            New Subject With {.Code = "04110033", .Name = "Industrial Orientation", .Level = "N3", .Stream =
"Cross-Stream"}
        }
    End Function
End Module

' ---------- Result logic ----------
Public Module ResultEngine

    If mark >= rules.DistinctionThreshold Then Return "Distinction"
    If mark >= rules.PassThreshold Then Return "Pass"
    If mark >= 0 Then Return "Fail"
    Return "N/A"
End Function

Public Function SubjectsPassed(subjects As IEnumerable(Of Subject), rules As ProgramRules) As
Integer
    Return subjects.Count(Function(s) s.Mark >= rules.PassThreshold)
End Function

' Apply equivalences: if canonical not present but an acceptable alternate passed, count it once.
Public Function ApplyEquivalence(subjects As IEnumerable(Of Subject), rules As ProgramRules) As
List(Of Subject)
    Dim takenCodes = New HashSet(Of String)(subjects.Select(Function(s) s.Code))
    Dim normalized As New List(Of Subject)(subjects)

    For Each eq In rules.Equivalences
        Dim hasCanonical = subjects.Any(Function(s) s.Code = eq.CanonicalCode AndAlso s.Mark >=
rules.PassThreshold)
        If Not hasCanonical Then
```

```
            Dim alt = subjects.FirstOrDefault(Function(s) eq.AcceptableCodes.Contains(s.Code) AndAlso
s.Mark >= rules.PassThreshold)
            If alt IsNot Nothing Then
               ' Replace alternate with canonical proxy (keeps the achieved mark)
               normalized.Remove (alt)
               normalized.Add(New Subject With {
                  .Code = eq.CanonicalCode,
                  .Name = eq.CanonicalName & " (via equivalence: " & alt.Name & ")",
                  .Level = alt.Level,
                  .Stream = alt.Stream,
                  .mark = alt.mark
               })
            End If
         End If
      Next

      Return normalized
   End Function


      Dim sameLevel = enrol.Subjects.Where(Function(s) s.Level = enrol.Level AndAlso s.Stream =
enrol.Stream)
      Dim normalized = ApplyEquivalence(sameLevel, rules)
      Return SubjectsPassed(normalized, rules) >= rules.RequiredSubjectsPerLevel
   End Function

   ' Diplomas (N4-N6): Provide counts; the policy decision text remains configurable in your UI/DB.
   Public Function DiplomaProgress(allN4ToN6 As IEnumerable(Of Subject), rules As ProgramRules,
completedMonthsExperience As Integer) As String
      Dim passed = allN4ToN6.Count(Function(s) s.Mark >= rules.PassThreshold AndAlso (s.Level = "N4"
OrElse s.Level = "N5" OrElse s.Level = "N6"))
      Dim subs = $"Subjects passed: {passed}/{rules.DiplomaRequiredSubjects}"
      Dim exp = $"Experience: {completedMonthsExperience}/{rules.DiplomaExperienceMonths} months"
      If passed >= rules.DiplomaRequiredSubjects AndAlso completedMonthsExperience >=
rules.DiplomaExperienceMonths Then
         Return $"Eligible for N Diploma - {subs}; {exp}"
      ElseIf Passed >= rules.DiplomaRequiredSubjects Then
         Return $"Pending workplace experience - {subs}; {exp}"
      Else
         Return $"In progress - {subs}; {exp}"
      End If
   End Function
End Module

' --------- Document generators ---------
Public Module Documents

      Dim sb As New StringBuilder()
      sb.AppendLine ("NICHRIST INSTITUTE OF MANAGEMENT AND ENGINEERING TECHNOLOGY
(NIMET)")
      sb.AppendLine ("10 Top Road, Anderbolt, Boksburg 1459. Tel: 067 154 8507 | www.nimet.co.za (http://
```

```
www.nimet.co.za)")
      sb.AppendLine("Elektor MAG (https://www.elektormagazine.com)")
      sb.AppendLine($"Statement of Results  |  Session: {enrol.Session}")
      sb.AppendLine($"Candidate: {enrol.CandidateName}  |  ID: {enrol.CandidateID}")
      sb.AppendLine($"Programme: {enrol.Stream} {enrol.Level}  |  Provider: {enrol.Provider}")
      sb.AppendLine ("-------------------------------------------------------------------")
      sb.AppendLine ("Code      Subject                          Mark   Result")
      sb.AppendLine ("--------  -----------------------------------  -----  ----------")

      Dim normalized = ResultEngine.ApplyEquivalence(enrol.Subjects, rules)
      For Each s In normalized.OrderBy(Function(x) x.Code)
         Dim band = ResultEngine.ResultBand(s.Mark, rules)
         sb.AppendLine($"{s.Code.PadRight(9)} {s.Name.PadRight(40)} {s.Mark.ToString().PadLeft(5)}
{band.PadRight(10)}")
      Next

      sb.AppendLine ("-------------------------------------------------------------------")
      Dim eligible = ResultEngine.EligibleForNCertificate(enrol, rules)
      sb.AppendLine($"N{enrol.Level.Substring(1)} Certificate Eligibility: {(If(eligible, "Meets minimum
combination", "Does not meet minimum combination"))}")
      sb.AppendLine($"Pass ? {rules.PassThreshold}; Distinction ? {rules.DistinctionThreshold}")
      sb.AppendLine ("-------------------------------------------------------------------")
      sb.AppendLine ("This statement is issued subject to DHET/QCTO/UMALUSI verification and
institutional records.")
      Return sb.ToString()
   End Function



      Dim eligible = ResultEngine.EligibleForNCertificate(enrol, rules)
      Dim sb As New StringBuilder()
      sb.AppendLine ("CERTIFICATE COMBINATION OUTCOME")
      sb.AppendLine($"Candidate: {enrol.CandidateName}  |  Programme: {enrol.Stream} {enrol.Level}")
      If eligible Then
         sb.AppendLine ("Outcome: Eligible for DHET N-level Certificate (subject to external verification).")
      Else
         sb.AppendLine ("Outcome: Not yet eligible - minimum four passed subjects at this level/stream not
met.")
      End If
      Return sb.ToString()
   End Function

   Public Function ReplacementEquivalenceLetter(candidate As String, idNo As String, canonical As String,
alternate As String, policyRef As String) As String
      Dim sb As New StringBuilder()
      sb.AppendLine ("THEORY REPLACEMENT / EQUIVALENCE CONFIRMATION")
      sb.AppendLine($"Candidate: {candidate}   ID: {idNo}")
      sb.AppendLine($"Approved Equivalence: {alternate} accepted in lieu of {canonical}")
      sb.AppendLine($"Basis: {policyRef}")
      sb.AppendLine ("Note: This decision is recorded for audit purposes and remains subject to awarding
body policies.")
      Return sb.ToString()
```

End Function
End Module


```
' ---------- Example wiring ----------
Module Demo

    Dim rules As New ProgramRules With {
      .PassThreshold = 50,
      .DistinctionThreshold = 75,
      .RequiredSubjectsPerLevel = 4,
      .DiplomaRequiredSubjects = 12,
      .DiplomaExperienceMonths = 18,
      .Equivalences = New List(Of EquivalenceRule) From {
        New EquivalenceRule With {
          .CanonicalCode = "8090283",
          .CanonicalName = "Engineering Drawing (N3)",
          .AcceptableCodes = New HashSet(Of String) From {"8080613"} ' Industrial Electronics (N3) as
declared option
        },
        New EquivalenceRule With {
          .CanonicalCode = "11041823",
          .CanonicalName = "Diesel Trade Theory (N3)",
          .AcceptableCodes = New HashSet(Of String) From {
            "11040343",  ' Electro-Technology (contextual acceptance, configure per policy)
            "11041861",  ' Electrical Trade Theory (N1) - example: only map when policy allows cross-
level or RPL
            "11041872"   ' Electrical Trade Theory (N2)
          }
        }
      }
    }

    ' Example: Diesel Trade Engineering N3 enrollment
    Dim dieselN3 = Catalog.DieselN3("Diesel Trade Engineering")
    ' Assign marks (demo values)
    dieselN3.First(Function(s) s.Code = "16030143").Mark = 68
    dieselN3.First(Function(s) s.Code = "15070413").Mark = 52
    dieselN3.First(Function(s) s.Code = "8090283").Mark = 47 ' failed Drawing
    dieselN3.First(Function(s) s.Code = "8080613").Mark = 65 ' passed Industrial Electronics
(equivalence)
    dieselN3.First(Function(s) s.Code = "11041823").Mark = 74

    Dim enrol As New Enrollment With {
      .CandidateName = "Tshingombe Tshitadi Fiston",
      .CandidateID = "9001015800082",
      .Provider = "NIMET",
      .Stream = "Diesel Trade Engineering",
      .Level = "N3",
      .Session = "2025 Trimester 1",
      .subjects = DieselN3
    }
```

```
    Console.WriteLine (Documents.ResultStatement(enrol, rules))
    Console.WriteLine()
    Console.WriteLine (Documents.CertificateNotice(enrol, rules))
    Console.WriteLine()

    ' Equivalence letter example (Diesel ? Electrical where policy-approved)
    Console.WriteLine(Documents.ReplacementEquivalenceLetter(
      enrol.CandidateName,
      enrol.CandidateID,
      "Diesel Trade Theory (N3)",
      "Electrical Trade Theory (per declared mapping)",
      "Institutional Equivalence Matrix v1.2; subject to DHET/QCTO policy and trade test route evidence"
    ))

    ' Diploma tracking example (N4-N6 scaffold)
    Dim n4to6 As New List(Of Subject) From {
      New Subject With {.Code = "ELECTROTECH", .Name = "Electrotechnics", .Level = "N4", .Stream =
"Electrical", .Mark = 62},
      New Subject With {.Code = "ELECTROTECH5", .Name = "Electrotechnics", .Level = "N5", .Stream =
"Electrical", .Mark = 55},
      New Subject With {.Code = "PLANTOPS4", .Name = "Plant Operations", .Level = "N4", .Stream =
"Plant", .Mark = 71}
      ' ... extend to total passed subjects
    }
    Console.WriteLine()
    Console.WriteLine (ResultEngine.DiplomaProgress(n4to6, rules, completedMonthsExperience:=12))
  End Sub
End Module
```

Ready-to-print templates

Statement of results

```
NICHRIST INSTITUTE OF MANAGEMENT AND ENGINEERING TECHNOLOGY (NIMET)
Statement of Results - [Session]
Candidate: [Name]  |  ID: [ID]
Programme: [Stream] [Level]  |  Provider: NIMET
------------------------------------------------------
Code     Subject                          Mark   Result
--------  ------------------------------------  -----  ---------
[code]    [name]                           [##]  [Pass/Distinction/Fail]
...
------------------------------------------------------
N[Level] Certificate Eligibility: [Meets/Does not meet]
Pass ? 50; Distinction ? 75
Issued subject to DHET/QCTO/UMALUSI verification.
```

Certificate combination outcome

```
CERTIFICATE COMBINATION OUTCOME
Candidate: [Name]  |  Programme: [Stream] [Level]
Outcome: [Eligible/Not yet eligible]
Note: Subject to external verification and certification body policy.
```

Theory replacement/equivalence confirmation

Code

THEORY REPLACEMENT / EQUIVALENCE CONFIRMATION
Candidate [illegible]
Approved Equivalence: [Alternate] accepted in lieu of [Canonical]
Basis: [Institutional policy reference; DHET/QCTO rule where applicable]
This decision is recorded for audit purposes.
What I need from you to finalize
" Pass bands: Confirm final thresholds (e.g., 40/50/75 or other).
" Equivalence matrix: Approve exact mappings for:
o Diesel Trade Theory vs Electrical Trade Theory/Electrotechnology/Electrotechnics
o Engineering Drawing vs Industrial Electronics options per level
o Plant Operations inclusion for N4-N6 routes
" N4-N6 list: Provide your exact subject catalog for Electrotechnics/Electrotechnologie/Plant Operations/ Orientation Industrial.
" Wording: Your preferred DHET/QCTO/UMALUSI phrasing for certificates and replacement letters.
" Trade test link: Which routes must explicitly state "pre-trade" or "trade test readiness" in the statement.
If you share your equivalence table and final pass policy, I'll lock the code to your compliance matrix and deliver a polished WinForms front end with PDF export and an audit log of equivalence decisions.
Visual Basic result release, printer, diploma N4-N6 application, and bid activity log
You want a single, auditable pipeline: generate a compliant result statement, print it, capture N4-N6 diploma application data, and log bid-related activities with timestamps and tender metadata. Below is a compact VB.NET (http://VB.NET) foundation for a console or WinForms app that you can extend into your existing portfolio and logigram stack.
Overview
" Result release: Generates and prints N-level statements with pass bands and eligibility notes.
" Diploma applications (N4-N6): Captures candidate + subjects + workplace experience; produces a printable application form.
" Printer: Uses PrintDocument to send finalized statements/forms to a printer.
" Activity log: Tamper-resistant log of actions (drafted, printed, submitted) with SHA-256 hash chaining.
" Bid/tender log: Records tender details (opening in public, PPPFA 80/20, SBD4/conditions) and links artifacts to candidate actions.
Data model
vb

```
Imports System.Security.Cryptography
Imports System.text
Imports System.Drawing.Printing

' ---------- Core domain ----------
Public Class Subject
    Public Property Code As String
    Public Property Name As String
    Public Property Level As String   ' N1..N6
    Public Property Stream As String  ' Diesel, Motor, Electrical, etc.
    Public Property Mark As Integer
End Class

Public Class Enrollment
    Public Property CandidateName As String
    Public Property CandidateID As String
    Public Property Provider As String
    Public Property Stream As String
    Public Property Level As String
```

```
    Public Property Session As String
    Public Property Subjects As List(Of Subject) = New List(Of Subject)
End Class

Public Class DiplomaApplication
    Public Property CandidateName As String
    Public Property CandidateID As String
    Public Property ContactEmail As String
    Public Property ContactPhone As String
    Public Property Provider As String
    Public Property Streams As List(Of String) = New List(Of String) ' e.g., Electrotechnics, Plant Ops
    Public Property N4N6Subjects As List(Of Subject) = New List(Of Subject)
    Public Property WorkplaceMonths As Integer
    Public Property DeclarationSigned As Boolean
    Public Property DateSubmitted As Date?
End Class

' ---------- Bid / Tender ----------
Public Class BidRecord
    Public Property BidNumber As String          ' e.g., HO5-22/23-0073
    Public Property Description As String
    Public Property Department As String         ' EC DPWI
    Public Property ClosingDate As Date
    Public Property OpeningPublic As Boolean
    Public Property OpeningVenue As String
    Public Property ContactSCM As String
    Public Property ContactTechnical As String
    Public Property PPPFAMaxPricePoints As Integer = 80
    Public Property PPPFAMaxBBBEEPoints As Integer = 20
    Public Property TotalPoints As Integer = 100
End Class
Result statement and diploma application generators
vb
Public Class ProgramRules
    Public Property PassThreshold As Integer = 50
    Public Property DistinctionThreshold As Integer = 75
    Public Property RequiredSubjectsPerLevel As Integer = 4
    Public Property DiplomaRequiredSubjects As Integer = 12
    Public Property DiplomaExperienceMonths As Integer = 18
End Class

Public Module ResultEngine

        If mark >= rules.DistinctionThreshold Then Return "Distinction"
        If mark >= rules.PassThreshold Then Return "Pass"
        Return "Fail"
    End Function



        Dim lvl = enrol.Subjects.Where(Function(s) s.Level = enrol.Level)
        Dim passed = lvl.Count(Function(s) s.Mark >= rules.PassThreshold)
```

```
    Return passed >= rules.RequiredSubjectsPerLevel
  End Function

    Dim passed = app.N4N6Subjects.Count(Function(s) s.Mark >= rules.PassThreshold AndAlso (s.Level
= "N4" OrElse s.Level = "N5" OrElse s.Level = "N6"))
    If passed >= rules.DiplomaRequiredSubjects AndAlso app.WorkplaceMonths >=
rules.DiplomaExperienceMonths Then
      Return "Eligible for N Diploma"
    ElseIf Passed >= rules.DiplomaRequiredSubjects Then
      Return "Pending workplace experience"
    Else
      Return "In progress"
    End If
  End Function
End Module

Public Module Documents

    Dim sb As New StringBuilder()
    sb.AppendLine ("NICHRIST INSTITUTE OF MANAGEMENT AND ENGINEERING TECHNOLOGY
(NIMET)")
    sb.AppendLine ("10 Top Road, Anderbolt, Boksburg 1459 | Tel: 067 154 8507 | www.nimet.co.za
(http://www.nimet.co.za)")
    sb.AppendLine($"Statement of Results - {enrol.Session}")
    sb.AppendLine($"Candidate: {enrol.CandidateName} | ID: {enrol.CandidateID}")
    sb.AppendLine($"Programme: {enrol.Stream} {enrol.Level} | Provider: {enrol.Provider}")
    sb.AppendLine(New String("-"c, 74))
    sb.AppendLine ("Code      Subject                              Mark   Result")
    sb.AppendLine ("--------  -----------------------------------  -----  ---------")
    For Each s In enrol.Subjects.OrderBy(Function(x) x.Code)
      sb.AppendLine($"{s.Code.PadRight(9)} {s.Name.PadRight(40)} {s.Mark.ToString().PadLeft(5)}
{ResultEngine.Band(s.Mark, rules).PadRight(10)}")
    Next
    sb.AppendLine(New String("-"c, 74))
    Dim elig = If(ResultEngine.EligibleNCertificate(enrol, rules), "Meets minimum combination", "Does not
meet minimum combination")
    sb.AppendLine($"N Certificate Eligibility: {elig}")
    sb.AppendLine($"Pass ? {rules.PassThreshold}; Distinction ? {rules.DistinctionThreshold}")
    sb.AppendLine ("Issued subject to DHET/QCTO/UMALUSI verification.")
    Return sb.ToString()
  End Function


    Dim sb As New StringBuilder()
    sb.AppendLine ("N4-N6 DIPLOMA APPLICATION FORM")
    sb.AppendLine($"Candidate: {app.CandidateName} | ID: {app.CandidateID}")
    sb.AppendLine($"Provider: {app.Provider} | Email: {app.ContactEmail} | Phone: {app.ContactPhone}")
    sb.AppendLine($"Streams: {String.Join(", ", app.Streams)}")
    sb.AppendLine(New String("-"c, 74))
    sb.AppendLine ("N4-N6 Subjects")
```

```
        sb.AppendLine ("Level  Code      Subject                          Mark   Result")
        sb.AppendLine ("-----  --------  -------------------------------  -----  ------")
        For Each s In app.N4N6Subjects.OrderBy(Function(x) x.Level).ThenBy(Function(x) x.Code)
            sb.AppendLine($"{s.Level.PadRight(5)} {s.Code.PadRight(9)} {s.Name.PadRight(40)}
{s.Mark.ToString().PadLeft(5)}  {ResultEngine.Band(s.Mark, rules)}")
        Next
        sb.AppendLine(New String("-"c, 74))
        sb.AppendLine($"Workplace Experience: {app.WorkplaceMonths}/{rules.DiplomaExperienceMonths}
months")
        sb.AppendLine($"Status: {ResultEngine.DiplomaStatus(app, rules)}")
        sb.AppendLine($"Declaration signed: {If(app.DeclarationSigned, "Yes", "No")} | Date submitted:
{If(app.DateSubmitted.HasValue, app.DateSubmitted.Value.ToShortDateString(), "-")}")
        sb.AppendLine ("Note: Attach certified ID, statements of results, and workplace logbook.")
        Return sb.ToString()
    End Function
End Module
Printing engine
Public Class PrintJob
    Private ReadOnly _content As String
    Private _lines() As String
    Private _lineIndex As Integer

    Public Sub New(content As String)
        _content = content
        _lines = _content.Replace(vbCrLf, vbLf).Split(ControlChars.Lf)
        _lineIndex = 0
    End Sub

    Public Sub Print(title As String)
        Dim pd As New PrintDocument()
        pd.DocumentName = Title
        AddHandler pd.PrintPage, AddressOf OnPrintPage
        pd.Print()
    End Sub


        Dim font = New Font("Consolas", 9.0F)
        Dim lineHeight = font.GetHeight(e.Graphics)
        Dim left = e.MarginBounds.Left
        Dim top = e.MarginBounds.Top
        Dim y = top
        Dim linesPerPage = CInt(Math.Floor(e.MarginBounds.Height / lineHeight))

        Dim count As Integer = 0
        While count < linesPerPage AndAlso _lineIndex < _lines.Length
            e.Graphics.DrawString(_lines(_lineIndex), font, Brushes.Black, left, y)
            y += lineHeight
            count += 1
            _lineIndex += 1
        End While
```

```
        e.HasMorePages = (_lineIndex < _lines.Length)
    End Sub
End Class
```

Activity and bid logging with hash chaining

vb

```
Public Class ActivityLogEntry
    Public Property Timestamp As Date
    Public Property Actor As String             ' user or system
    Public Property Action As String            ' Drafted, Printed, Submitted
    Public Property Entity As String            ' ResultStatement, DiplomaApplication, Bid
    Public Property EntityId As String          ' e.g., CandidateID, BidNumber
    Public Property Details As String           ' e.g., printer name, pages, PPPFA calc
    Public Property PreviousHash As String
    Public Property Hash As String
End Class

Public Class ActivityLogger
    Private ReadOnly _path As String
    Private _lastHash As String = ""

    Public Sub New(filePath As String)
        _path = filePath
        If Not IO.File.Exists(_path) Then
            IO.File.WriteAllText(_path, "")
        Else
            _lastHash = GetLastHash()
        End If
    End Sub


        Dim entry As New ActivityLogEntry With {
            .Timestamp = Date.UtcNow,
            .Actor = actor,
            .Action = action,
            .Entity = entity,
            .EntityId = entityId,
            .Details = details,
            .PreviousHash = _lastHash
        }
        entry.hash = ComputeHash(entry)
        IO.File.AppendAllText(_path, Serialize(entry) & Environment.NewLine)
        _lastHash = entry.Hash
        Return entry
    End Function


        ' Simple pipe-delimited; swap to JSON if preferred
        Return $"{e.Timestamp:o}|{e.Actor}|{e.Action}|{e.Entity}|{e.EntityId}|{e.Details}|{e.PreviousHash}|
{e.Hash}"
    End Function
```

```
    Private Function GetLastHash() As String
        Dim lines = File.ReadAllLines(_path)
        If lines.Length = 0 Then Return ""
        Return lines.Last().Split("|"c).Last()
    End Function


        Dim raw = $"{e.Timestamp:o}|{e.Actor}|{e.Action}|{e.Entity}|{e.EntityId}|{e.Details}|{e.PreviousHash}"
        Using sha = SHA256.Create()
            Dim bytes = sha.ComputeHash(Encoding.UTF8.GetBytes(raw))
            Return BitConverter.ToString(bytes).Replace("-", "").ToLowerInvariant()
        End Using
    End Function
End Class

' --------- Bid logger ---------
Public Class BidLogger
    Private ReadOnly _path As String
    Public Sub New(filePath As String)
        _path = filePath
        If Not IO.File.Exists(_path) Then IO.File.WriteAllText(_path, "BidNumber|Description|Dept|Closing|
OpeningPublic|Venue|SCM|Tech|PPPFA(Price/BBBEE/Total)" & Environment.NewLine)
    End Sub


        Dim line = $"{b.BidNumber}|{b.Description}|{b.Department}|{b.ClosingDate:yyyy-MM-dd}|
{b.OpeningPublic}|{b.OpeningVenue}|{b.ContactSCM}|{b.ContactTechnical}|{b.PPPFAMaxPricePoints}/
{b.PPPFAMaxBBBEEPoints}/{b.TotalPoints}"
        IO.File.AppendAllText(_path, line & Environment.NewLine)
    End Sub
End Class
Example wiring And usage
vb
odule Demo

        Dim rules As New ProgramRules With {
            .PassThreshold = 50,
            .DistinctionThreshold = 75,
            .RequiredSubjectsPerLevel = 4,
            .DiplomaRequiredSubjects = 12,
            .DiplomaExperienceMonths = 18
        }

        ' 1) Result statement (e.g., Diesel N3)
        Dim enrol As New Enrollment With {
            .CandidateName = "Tshingombe Tshitadi Fiston",
            .CandidateID = "9001015800082",
            .Provider = "NIMET",
            .Stream = "Diesel Trade Engineering",
            .Level = "N3",
            .Session = "2025 Trimester 1",
```

```
        .Subjects = New List(Of Subject) From {
            New Subject With {.Code = "15030143", .Name = "Mathematics", .Level = "N3", .Stream = "Diesel",
.Mark = 68},
            New Subject With {.Code = "15070413", .Name = "Engineering Science", .Level = "N3", .Stream =
"Diesel", .Mark = 52},
            New Subject With {.Code = "8090283",  .Name = "Engineering Drawing", .Level = "N3", .Stream =
"Diesel", .Mark = 55},
            New Subject With {.Code = "8080613",  .Name = "Industrial Electronics", .Level = "N3", .Stream =
"Diesel", .Mark = 62},
            New Subject With {.Code = "11041823", .Name = "Diesel Trade Theory", .Level = "N3", .Stream =
"Diesel", .Mark = 74}
        }
    }
    Dim statement = Documents.RenderResultStatement(enrol, rules)
    Console.WriteLine (Statement)

    ' Print
    Dim printer = New PrintJob(statement)
    ' printer.Print("NIMET - Statement of Results") ' uncomment to print

    ' 2) Diploma application (N4-N6)
    Dim app As New DiplomaApplication With {
        .CandidateName = enrol.CandidateName,
        .CandidateID = enrol.CandidateID,
        .ContactEmail = "candidate@example.com (mailto:candidate@example.com)",
        .ContactPhone = "+27 72 000 0000",
        .Provider = "NIMET",
        .Streams = New List(Of String) From {"Electrotechnics", "Plant Operations"},
        .WorkplaceMonths = 12,
        .DeclarationSigned = False
    }
    app.N4N6Subjects.AddRange(New List(Of Subject) From {
        New Subject With {.Code = "ELECTRO4", .Name = "Electrotechnics", .Level = "N4", .Stream =
"Electrical", .Mark = 65},
        New Subject With {.Code = "PLANTOP4", .Name = "Plant Operations", .Level = "N4", .Stream = "Plant",
.Mark = 71},
        New Subject With {.Code = "ELECTRO5", .Name = "Electrotechnologie", .Level = "N5", .Stream =
"Electrical", .Mark = 58}
    })
    Dim appDoc = Documents.RenderDiplomaApplication(app, rules)
    Console.WriteLine()
    Console.WriteLine (appDoc)
    ' Dim appPrinter = New PrintJob(appDoc) : appPrinter.Print("NIMET - N4-N6 Diploma Application")

    ' 3) Activity log (hash-chained)
    Dim audit = New ActivityLogger("audit_log.txt")
    audit.Append("system", "Drafted", "ResultStatement", enrol.CandidateID, $"Session={enrol.Session};
Pages=1")
    audit.Append("system", "Printed", "ResultStatement", enrol.CandidateID, "Printer=Default")
    audit.Append("system", "Drafted", "DiplomaApplication", enrol.CandidateID, "N4-N6 subjects=3")
    audit.Append("candidate", "SignedDeclaration", "DiplomaApplication", enrol.CandidateID,
```

```
"DigitalSign=True")
        ' 4) Bid/tender log (PPPFA 80/20; opening in public)
        Dim bid As New BidRecord With {
            .BidNumber = "HO5-22/23-0073",
            .Description = "Electrical pre-trade and trade test training for APTCOD learners",
            .Department = "Eastern Cape Department of Public Works & Infrastructure",
            .ClosingDate = New Date(2022, 8, 17, 11, 0, 0),
            .OpeningPublic = True,
            .OpeningVenue = "Qhasana Building, Ground Floor, Bhisho",
            .ContactSCM = "Ms N. Matinise, nelisa.matinise@ecdpw.gov.za
(mailto:nelisa.matinise@ecdpw.gov.za), 040 602 4345",
            .ContactTechnical = "Ms A. Bangani, Abulele.Bangani@ecdpw.gov.za
(mailto:Abulele.Bangani@ecdpw.gov.za), 040 602 4312 / 079 269 8675"
        }
        Dim bidLog = New BidLogger("bids_log.txt")
        bidLog.Record (bid)
        audit.Append("scm", "RecordedBid", "Bid", bid.BidNumber, "PPPFA=80/20; Total=100")

        ' 5) Example PPPFA scoring helper (optional)
        Dim pricePoints = PPPFAPricePoints(bidLowest:=1000000D, tendererPrice:=1100000D,
maxPoints:=80)
        Dim bbbeePoints = 20 ' from valid certificate/affidavit level mapping
        audit.Append("scm", "ScoredBid", "Bid", bid.BidNumber, $"PricePts={pricePoints:F2};
BBBEEPts={bbbeePoints}")
    End Sub

    Private Function PPPFAPricePoints(bidLowest As Decimal, tendererPrice As Decimal, maxPoints As
Integer) As Decimal
        If tendererPrice <= 0 OrElse bidLowest <= 0 Then Return 0
        Dim score = (bidLowest / tendererPrice) * maxPoints
        If Score < 0 Then Score = 0
        If Score > maxPoints Then Score = maxPoints
        Return score
    End Function
End Module
Implementation Notes
"   Lead-in labels to configure:
o   Pass bands: If your policy uses different thresholds, update ProgramRules.
o   Diploma requirement: Finalize total subjects and experience months per stream.
o   Streams mapping: Electrotechnics vs Electrotechnologie naming; ensure consistency across N4-N6
codes.
"   Printing: The PrintJob class is monospace-friendly; if you need logos/branding, add header drawing with
images in OnPrintPage.
"   Evidence trail: The ActivityLogger's hash-chaining creates a tamper-evident trail for audits; rotate logs by
term to keep files small.
"   Bid compliance: Store SBD4 completion status, declarations, and supporting files alongside BidRecord
(add fields or a file index).
Want me to wrap this into a WinForms front end?
"   Tabs: Results | Diploma N4-N6 | Print | Activity Log | Bids.
"   Buttons: Generate, Validate, Print, Export PDF, Log Action.
```

" Exports: CSV/JSON for logs; PDF for statements/applications via a PDF printer driver.
If you share your Subject Code list (from N4–N6 (https://www.elektormagazine.com/labs/Electrotechnologie/Plant Operations/ Industrial Orientation) and your exact pass/equivalence policy, I'll lock in the catalogs and deliver the WinForms UI scaffolding with ready-to-use forms and print previews.
Visual Basic certificate of results and recertification generator for SAQA N diplomas (NQF 6-7)
You want audit-ready, printable certificates that align to SAQA fields and support re-certification. Below is a clean VB.NET (http://VB.NET) core you can drop into a console or WinForms app. It renders:
" Certificate of Results
" Qualification Attestation (SAQA-aligned for N Diploma at NQF 6 or 7)
" Re-certification notice (replacement/duplicate with reason and chain-of-custody)
" Optional verification hash and QR payload text
" Print pipeline via PrintDocument
Core models
vb

```
Imports System.text
Imports System.Drawing.Printing
Imports System.Security.Cryptography

' ---------- Domain ----------
Public Class SubjectResult
    Public Property Code As String
    Public Property Name As String
    Public Property Level As String     ' N1..N6 or module level
    Public Property Credits As Integer  ' optional
    Public Property Mark As Integer
    Public Property Result As String    ' Pass/Distinction/Fail (optional override)
End Class

Public Class Candidate
    Public Property FullName As String
    Public Property IDNumber As String
    Public Property StudentNumber As String
End Class

Public Class Provider
    Public Property Name As String
    Public Property ExamCentre As String
    Public Property Address As String
    Public Property Contact As String
    Public Property AccredRefs As String ' e.g., QCTO/DHET/UMALUSI numbers
End Class

Public Class Qualification
    Public Property Title As String            ' e.g., National N Diploma: Engineering Studies
    Public Property SAQAID As String           ' SAQA qualification ID
    Public Property NQFLevel As String         ' "6" or "7"
    Public Property TotalCredits As Integer
    Public Property FieldOfStudy As String      ' optional: sub-field
    Public Property ExitLevelOutcomes As String  ' optional summary
End Class
```

```
Public Class CertificateMeta
    Public Property DocumentType As String    ' "Certificate of Results", "Qualification Attestation, Re-Certification
    Public Property SerialNumber As String
    Public Property IssueDate As Date
    Public Property ReissueReason As String      ' for re-certification
    Public Property ReplacesSerial As String     ' for re-certification
    Public Property SignatoryName As String
    Public Property SignatoryTitle As String
    Public Property VerificationURL As String    ' optional
End Class
Rendering engines
Public Module Renderers

    Public Function RenderCertificateOfResults(candidate As Candidate,
                                provider As Provider,
                                programme As String,     ' e.g., Electrical Engineering
                                sessionLabel As String,  ' e.g., 2025 Trimester 1
                                subjects As IEnumerable(Of SubjectResult),
                                meta As CertificateMeta) As String
        Dim sb As New StringBuilder()
        sb.AppendLine (provider.Name)
        sb.AppendLine (provider.Address)
        sb.AppendLine (provider.Contact)
        sb.AppendLine (provider.AccredRefs)
        sb.AppendLine(New String("-"c, 78))
        sb.AppendLine ("CERTIFICATE OF RESULTS")
        sb.AppendLine($"Programme: {programme}    Session: {sessionLabel}")
        sb.AppendLine($"Candidate: {candidate.FullName}    ID: {candidate.IDNumber}    Student No.:
{candidate.StudentNumber}")
        sb.AppendLine($"Doc Serial: {meta.SerialNumber}    Issue date: {meta.IssueDate:yyyy-MM-dd}")
        sb.AppendLine(New String("-"c, 78))
        sb.AppendLine ("Level  Code      Subject                              Credits  Mark  Result")
        sb.AppendLine ("-----  --------  -----------------------------------  -------  ----  ----------")
        For Each s In subjects.OrderBy(Function(x) x.Level).ThenBy(Function(x) x.Code)
            Dim res = If(String.IsNullOrWhiteSpace(s.Result), BandFromMark(s.Mark), s.Result)
            sb.AppendLine($"{s.Level.PadRight(5)} {s.Code.PadRight(9)} {s.Name.PadRight(40)}
{s.Credits.ToString().PadLeft(7)}  {s.Mark.ToString().PadLeft(4)}  {res}")
        Next
        sb.AppendLine(New String("-"c, 78))
        sb.AppendLine ("Note: This institutional statement is issued subject to verification by the awarding/
certifying authorities.")
        AppendVerification(sb, candidate, meta)
        Return sb.ToString()
    End Function

    Public Function RenderQualificationAttestation(candidate As Candidate,
                                provider As Provider,
                                qual As Qualification,
                                achievedDate As Date,
                                meta As CertificateMeta) As String
```

```
        Dim sb As New StringBuilder()
        sb.AppendLine(provider.Name)
        sb.AppendLine (provider.Address)
        sb.AppendLine (provider.Contact)
        sb.AppendLine (provider.AccredRefs)
        sb.AppendLine(New String("-"c, 78))
        sb.AppendLine ("QUALIFICATION ATTESTATION")
        sb.AppendLine($"Candidate: {candidate.FullName}    ID: {candidate.IDNumber}    Student No.:
{candidate.StudentNumber}")
        sb.AppendLine($"Qualification: {qual.Title}")
        sb.AppendLine($"SAQA ID: {qual.SAQAID}    NQF Level: {qual.NQFLevel}    Total Credits:
{qual.TotalCredits}")
        If Not String.IsNullOrWhiteSpace(qual.FieldOfStudy) Then sb.AppendLine($"Field of Study:
{qual.FieldOfStudy}")
        sb.AppendLine($"Date of Achievement: {achievedDate:yyyy-MM-dd}")
        If Not String.IsNullOrWhiteSpace(qual.ExitLevelOutcomes) Then
            sb.AppendLine(New String("-"c, 78))
            sb.AppendLine ("Exit Level Outcomes (summary)")
            sb.AppendLine (qual.ExitLevelOutcomes)
        End If
        sb.AppendLine(New String("-"c, 78))
        sb.AppendLine($"Doc Serial: {meta.SerialNumber}    Issue date: {meta.IssueDate:yyyy-MM-dd}")
        sb.AppendLine($"Signed: {meta.SignatoryName}, {meta.SignatoryTitle}")
        sb.AppendLine ("Note: This attestation references the registered qualification data; the official
certificate remains the property of the awarding body.")
        AppendVerification(sb, candidate, meta)
        Return sb.ToString()
    End Function

    Public Function RenderRecertification(candidate As Candidate,
                                          provider As Provider,
                                          qual As Qualification,
                                          meta As CertificateMeta) As String
        Dim sb As New StringBuilder()
        sb.AppendLine (provider.Name)
        sb.AppendLine (provider.Address)
        sb.AppendLine (provider.Contact)
        sb.AppendLine (provider.AccredRefs)
        sb.AppendLine(New String("-"c, 78))
        sb.AppendLine ("RE-CERTIFICATION NOTICE")
        sb.AppendLine($"Candidate: {candidate.FullName}    ID: {candidate.IDNumber}    Student No.:
{candidate.StudentNumber}")
        sb.AppendLine($"Qualification: {qual.Title}  (SAQA {qual.SAQAID}, NQF {qual.NQFLevel})")
        sb.AppendLine($"New Serial: {meta.SerialNumber}    Issue date: {meta.IssueDate:yyyy-MM-dd}")
        sb.AppendLine($"Replaces Serial: {meta.ReplacesSerial}")
        sb.AppendLine($"Reason for Re-issue: {meta.ReissueReason}")
        sb.AppendLine($"Signed: {meta.SignatoryName}, {meta.SignatoryTitle}")
        sb.AppendLine ("Note: This re-issue supersedes prior versions. Prior serial is recorded for chain-of-
custody and audit.")
        AppendVerification(sb, candidate, meta)
        Return sb.ToString()
```

```
    End Function
    Private Function BandFromMark(mark As Integer) As String
        If mark >= 75 Then Return "Distinction"
        If mark >= 50 Then Return "Pass"
        Return "Fail"
    End Function


        Dim payload = $"serial={meta.SerialNumber}|id={candidate.IDNumber}|date={meta.IssueDate:yyyy-
MM-dd}"
        Dim hash = Sha256Hex(payload)
        sb.AppendLine($"Verification hash: {hash}")
        If Not String.IsNullOrWhiteSpace(meta.VerificationURL) Then
            sb.AppendLine($"Verify at: {meta.VerificationURL}?serial={meta.SerialNumber}")
            sb.AppendLine($"QR payload: {payload}") ' feed to QR generator if available
        End If
    End Sub

    Private Function Sha256Hex(input As String) As String
        Using sha = SHA256.Create()
            Dim bytes = sha.ComputeHash(Encoding.UTF8.GetBytes(input))
            Return BitConverter.ToString(bytes).Replace("-", "").ToLowerInvariant()
        End Using
    End Function

End Module
Printing helper Public Class TextPrintJob
    Private ReadOnly _content As String
    Private _lines() As String
    Private _index As Integer

    Public Sub New(content As String)
        _content = content
        _lines = _content.Replace(vbCrLf, vbLf).Split(ControlChars.Lf)
    End Sub

    Public Sub Print(documentName As String)
        Dim pd As New PrintDocument()
        pd.DocumentName = DocumentName
        AddHandler pd.PrintPage, AddressOf OnPrintPage
        pd.Print()
    End Sub


        Dim font = New Font("Consolas", 9.5F)
        Dim lh = font.GetHeight(e.Graphics)
        Dim y = e.MarginBounds.Top
        Dim left = e.MarginBounds.Left
        Dim linesPerPage = CInt(Math.Floor(e.MarginBounds.Height / lh))
        Dim count As Integer = 0
```

```
        While count < linesPerPage AndAlso _index < _lines.Length
            e.Graphics.DrawString(_lines(_index), _font, Brushes.Black, x, y)
            y += lh
            count += 1
            _index += 1
        End While
        e.HasMorePages = (_index < _lines.Length)
    End Sub
End Class
Example usage
Module Demo

    Dim candidate = New Candidate With {
        .FullName = "Tshingombe Tshitadi Fiston",
        .IDNumber = "9001015800082",
        .StudentNumber = "NIMET-2025-00123"
    }
    Dim provider = New Provider With {
        .Name = "NICHRIST Institute of Management and Engineering Technology (NIMET)",
        .ExamCentre = "Boksburg Centre",
        .Address = "10 Top Road, Anderbolt, Boksburg 1459, South Africa",
        .Contact = "Tel: 067 154 8507 | info@nimet.co.za (mailto:info@nimet.co.za) | www.nimet.co.za
(http://www.nimet.co.za)",
        .AccredRefs = "QCTO: SDP1220/18/00146 | DHET: 0899992889 | Umalusi: 20 FET02 00191 PA"
    }
    Dim qual = New Qualification With {
        .Title = "National N Diploma: Engineering Studies (Electrical)",
        .SAQAID = "XXXXXX",        ' supply the registered SAQA ID
        .NQFLevel = "6",           ' or "7" if applicable to the attested award
        .TotalCredits = 360,
        .FieldOfStudy = "Manufacturing, Engineering and Technology",
        .ExitLevelOutcomes = "Demonstrate applied competence in electrical systems, fault finding, plant
operations, and safety."
    }
    Dim metaResults = New CertificateMeta With {
        .DocumentType = "Certificate of Results",
        .SerialNumber = "NIMET-SOR-2025-000987",
        .IssueDate = Date.Today,
        .SignatoryName = "Nicholas Phiri",
        .SignatoryTitle = "Director",
        .VerificationURL = "https://verify.nimet.co.za (https://verify.nimet.co.za)"
    }
    Dim metaQual = New CertificateMeta With {
        .DocumentType = "Qualification Attestation",
        .SerialNumber = "NIMET-QA-2025-000321",
        .IssueDate = Date.Today,
        .SignatoryName = "Nicholas Phiri",
        .SignatoryTitle = "Director",
        .VerificationURL = "https://verify.nimet.co.za (https://verify.nimet.co.za)"
    }
    Dim metaReissue = New CertificateMeta With {
```

```
        .DocumentType = "Re-Certification",
        .SerialNumber = "NIMET-RRP-2025-000045",
        .ReplacesSerial = "NIMET-QA-2023-000112",
        .ReissueReason = "Original lost; affidavit provided",
        .IssueDate = Date.Today,
        .SignatoryName = "Nicholas Phiri",
        .SignatoryTitle = "Director",
        .VerificationURL = "https://verify.nimet.co.za (https://verify.nimet.co.za)"
    }

    ' Subjects for Certificate of Results
    Dim subjects = New List(Of SubjectResult) From {
        New SubjectResult With {.Code = "16030143", .Name = "Mathematics", .Level = "N3", .Credits = 0,
.Mark = 78},
        New SubjectResult With {.Code = "15070413", .Name = "Engineering Science", .Level = "N3", .Credits
= 0, .Mark = 66},
        New SubjectResult With {.Code = "8080613",  .Name = "Industrial Electronics", .Level = "N3", .Credits
= 0, .Mark = 59},
        New SubjectResult With {.Code = "11040343", .Name = "Electro-Technology", .Level = "N3", .Credits =
0, .Mark = 72}
    }

    Dim sor = Renderers.RenderCertificateOfResults(candidate, provider, "Electrical Engineering", "2025
Trimester 1", subjects, metaResults)
    Console.WriteLine (sor)
    ' New TextPrintJob(sor).Print("Certificate of Results")

    Dim qa = Renderers.RenderQualificationAttestation(candidate, provider, qual,
achievedDate:=#2025-09-01#, meta:=metaQual)
    Console.WriteLine()
    Console.WriteLine (qa)
    ' New TextPrintJob(qa).Print("Qualification Attestation (SAQA-aligned)")

    Dim rep = Renderers.RenderRecertification(candidate, provider, qual, metaReissue)
    Console.WriteLine()
    Console.WriteLine (rep)
    ' New TextPrintJob(rep).Print("Re-Certification Notice")
  End Sub
End Module
```

Implementation Notes

" NQF level and SAQA ID:

o Set Qualification.SAQAID and NQFLevel to match the registered qualification you're attesting. N Diplomas commonly align at NQF 6; use NQF 7 only when the registered qualification is at level 7.

" Institution vs awarding body:

o Use "Certificate of Results" and "Qualification Attestation" as institutional documents. The official certificate is conferred by the awarding/certifying authority; include your accreditation references and a verification URL.

" Re-certification controls:

o Always store ReplacesSerial and ReissueReason (lost/damaged/name change/correction). Keep scanned affidavit or proof attached in your records.

" Credits:

o If subject level credits are not applicable at N1-N3, leave as 0 or omit. At N4-N6, populate where your curriculum [illegible] supports it.
If you share the exact SAQA qualification IDs and your preferred wording blocks, I can harden the templates, add your logo in the print layout, and export to PDF-ready formatting.
you 're reaching into the upper echelons of academic and professional certification-NQF Levels 8 to 10, postdoctoral awards, and advanced diplomas like UCPD (University Certificate in Professional Development) and UCG (University Certificate in Graduate Studies). To support this in Visual Basic, we'll build a modular certificate generator that can handle:
" ?? Advanced qualification attestation (NQF 8-10)
" ?? Postdoctoral and professional awards
" ?? Diploma and transcript rendering
" ?? Lecture and trade specialization mapping (e.g., Electrical Engineering, N-Trade)
" ?? SAQA-aligned metadata and verification hash
?? Core Structure
1. Qualification Model

```
Public Class AdvancedQualification
    Public Property Title As String
    Public Property SAQAID As String
    Public Property NQFLevel As String ' "8", "9", "10"
    Public Property Credits As Integer
    Public Property AwardType As String ' "Postdoctoral Diploma", "UCPD", "UCG", etc.
    Public Property Specialization As String ' "Electrical Engineering", "Trade Theory", etc.
    Public Property Institution As String
    Public Property Lecturer As String
    Public Property DateAwarded As Date
End Class
```

2. Candidate Model

```
Public Class Candidate
    Public Property FullName As String
    Public Property IDNumber As String
    Public Property StudentNumber As String
    Public Property Email As String
End Class
```

3. Certificate Metad

```
Public Class CertificateMeta
    Public Property SerialNumber As String
    Public Property IssueDate As Date
    Public Property SignatoryName As String
    Public Property SignatoryTitle As String
    Public Property VerificationURL As String
End Class
End Class
```

??? Certificate Generator
Render Advanced Diploma Certificate
vb

```
    Dim sb As New StringBuilder()
    sb.AppendLine (cert.Institution)
    sb.AppendLine ("Advanced Qualification Certificate")
    sb.AppendLine(New String("-"c, 70))
    sb.AppendLine($"Candidate: {candidate.FullName}")
```

```
    sb.AppendLine($"ID Number: {candidate.IDNumber} | Student No.: {candidate.StudentNumber}")
    sb.AppendLine($"Email: {candidate.Email}")
    sb.AppendLine(New String("-"c, 70))
    sb.AppendLine($"Award: {cert.AwardType}")
    sb.AppendLine($"Title: {cert.Title}")
    sb.AppendLine($"Specialization: {cert.Specialization}")
    sb.AppendLine($"SAQA ID: {cert.SAQAID} | NQF Level: {cert.NQFLevel} | Credits: {cert.Credits}")
    sb.AppendLine($"Lecturer/Supervisor: {cert.Lecturer}")
    sb.AppendLine($"Date Awarded: {cert.DateAwarded:yyyy-MM-dd}")
    sb.AppendLine(New String("-"c, 70))
    sb.AppendLine($"Certificate Serial: {meta.SerialNumber}")
    sb.AppendLine($"Issued: {meta.IssueDate:yyyy-MM-dd}")
    sb.AppendLine($"Signed: {meta.SignatoryName}, {meta.SignatoryTitle}")
    sb.AppendLine ("This certificate confirms the successful completion of the above qualification.")
    sb.AppendLine ("Subject to verification by the awarding institution and SAQA registry.")
    sb.AppendLine($"Verify at: {meta.VerificationURL}?serial={meta.SerialNumber}")
    sb.AppendLine($"Verification Hash: {ComputeHash(candidate.IDNumber & meta.SerialNumber)}")
    Return sb.ToString()
End Function

Private Function ComputeHash(input As String) As String
    Using sha = System.Security.Cryptography.SHA256.Create()
        Dim bytes = sha.ComputeHash(System.Text.Encoding.UTF8.GetBytes(input))
        Return BitConverter.ToString(bytes).Replace("-", "").ToLowerInvariant()
    End Using
End Function
?? Example Usage
Dim candidate = New Candidate With {
    .FullName = "Tshingombe Tshitadi Fiston",
    .IDNumber = "9001015800082",
    .StudentNumber = "UCPD-ENG-2025-001",
    .Email = "fiston@nimet.ac.za (mailto:fiston@nimet.ac.za)"
}

Dim cert = New AdvancedQualification With {
    .Title = "Postdoctoral Diploma in Electrical Systems and Trade Innovation",
    .SAQAID = "123456",
    .NQFLevel = "9",
    .Credits = 180,
    .AwardType = "Postdoctoral Diploma",
    .Specialization = "Electrical Engineering and N-Trade",
    .Institution = "NICHRIST Institute of Management and Engineering Technology",
    .Lecturer = "Prof. Nicholas Phiri",
    .DateAwarded = #9/1/2025#
}

Dim meta = New CertificateMeta With {
    .SerialNumber = "NIMET-PD-2025-00045",
    .IssueDate = Date.Today,
    .SignatoryName = "Nicholas Phiri",
    .SignatoryTitle = "Director of Academic Affairs",
```

```
    .VerificationURL = "https://verify.nimet.ac.za (https://verify.nimet.ac.za)"
}
```



```
Dim output = RenderAdvancedDiploma(cert, candidate, meta)
Console.WriteLine (output)
?? Expandable Features
Imports System.text
Imports System.Security.Cryptography
Imports System.Drawing.Printing

Public Enum CertificateType
    QCTO_TradeTest
    QCTO_StatementOfResults
    Council_Engineering
    Council_Education
    NRF_Award
    SARAO_Award
    Postdoctoral_Award
    NationalN_Diploma_Attestation
End Enum

Public Class Person
    Public Property FullName As String
    Public Property IDNumber As String
    Public Property RegNo As String       ' Council/Student/Apprentice No.
    Public Property Email As String
End Class

Public Class ProviderAuthority
    Public Property Name As String
    Public Property Address As String
    Public Property Contact As String
    Public Property Accreditation As String  ' QCTO/DHET/UMALUSI/Council refs
End Class

Public Class QualificationRef
    Public Property Title As String        ' e.g., National N Diploma: Engineering Studies
    Public Property SAQAID As String
    Public Property NQFLevel As String     ' 5..10 depending on award
    Public Property Credits As Integer
    Public Property Specialisation As String
End Class

Public Class CertificateMeta
    Public Property Type As CertificateType
    Public Property Serial As String
    Public Property IssueDate As Date
    Public Property Signatory As String
    Public Property SignatoryTitle As String
    Public Property VerificationURL As String
    ' Reissue fields
```

```
    Public Property IsReissue As Boolean
    Public Property ReissueOriginalAs String
    Public Property ReissueReason As String   ' Lost, Damaged, Correction, Legal name change, etc.
End Class

Public Class OutcomeRecord
    Public Property Code As String       ' Module/Trade Code
    Public Property Name As String
    Public Property Result As String     ' Competent/NYC, Pass/Fail, Awarded
    Public Property DateAchieved As Date
    Public Property Score As String      ' %, Class, or “Competent”
    Public Property Credits As Integer
End Class

RenderersPublic Module CertificateRenderers

    Public Function RenderCertificate(p As Person,
                                      auth As ProviderAuthority,
                                      q As QualificationRef,
                                      outcomes As IEnumerable(Of OutcomeRecord),
                                      m As CertificateMeta) As String
        Dim sb As New StringBuilder()

        ' Header
        sb.AppendLine (auth.Name)
        sb.AppendLine (auth.Address)
        sb.AppendLine (auth.Contact)
        If Not String.IsNullOrWhiteSpace(auth.Accreditation) Then sb.AppendLine(auth.Accreditation)
        sb.AppendLine(New String("-"c, 80))

        ' Title by type
        sb.AppendLine (CertificateTitle(m.Type))
        sb.AppendLine(New String("-"c, 80))

        ' Person + qual
        sb.AppendLine($"Name: {p.FullName}   ID: {p.IDNumber}   Reg/Student No.: {p.RegNo}")
        If q IsNot Nothing Then
            sb.AppendLine($"Qualification: {q.Title}")
            Dim qline = $"SAQA: {q.SAQAID}"
            If Not String.IsNullOrEmpty(q.NQFLevel) Then qline &= $"   NQF: {q.NQFLevel}"
            If q.Credits > 0 Then qline &= $"   Credits: {q.Credits}"
            If Not String.IsNullOrWhiteSpace(q.Specialisation) Then qline &= $"   Specialisation:
{q.Specialisation}"
            sb.AppendLine (qline)
        End If

        ' Trade test specific note
        If m.Type = CertificateType.QCTO_TradeTest Then
            sb.AppendLine ("Trade Test Outcome: Refer to results table below. Certification subject to QCTO
verification.")
        End If
```

```
    ' Outcomes table (optional)
    If outcomes IsNot Nothing AndAlso outcomes.Any() Then
      sb.AppendLine(New String("-"c, 80))
      sb.AppendLine ("Code      Component/Module/Trade Element              Date       Credits  Score
Result")
      sb.AppendLine ("--------  ------------------------------------------  ----------  -------  -----  --------")
      For Each o In outcomes.OrderBy(Function(x) x.DateAchieved).ThenBy(Function(x) x.Code)
        sb.AppendLine($"{o.Code.PadRight(9)} {o.Name.PadRight(45)} {o.DateAchieved:yyyy-MM-dd}
{o.Credits.ToString().PadLeft(7)}  {o.Score.PadLeft(5)}  {o.Result}")
      Next
    End If

    ' Footer and verification
    sb.AppendLine(New String("-"c, 80))
    Dim serialBlock = $"Serial: {m.Serial}    Issued: {m.IssueDate:yyyy-MM-dd}    Signed: {m.Signatory},
{m.SignatoryTitle}"
    sb.AppendLine (serialBlock)
    If m.IsReissue Then
      sb.AppendLine($"Re-issue of: {m.ReplacesSerial}    Reason: {m.ReissueReason}")
    End If
    sb.AppendLine (StandardFootnote(m.Type))
    AppendVerification(sb, p, m)

    Return sb.ToString()
  End Function


    Select Case t
      Case CertificateType.QCTO_TradeTest : Return "QCTO TRADE TEST CERTIFICATE"
      Case CertificateType.QCTO_StatementOfResults : Return "QCTO STATEMENT OF RESULTS"
      Case CertificateType.Council_Engineering : Return "ENGINEERING COUNCIL CERTIFICATE"
      Case CertificateType.Council_Education : Return "EDUCATION COUNCIL CERTIFICATE"
      Case CertificateType.NRF_Award : Return "NRF AWARD CERTIFICATE"
      Case CertificateType.SARAO_Award : Return "SARAO AWARD CERTIFICATE"
      Case CertificateType.Postdoctoral_Award : Return "POSTDOCTORAL AWARD ATTESTATION"
      Case CertificateType.NationalN_Diploma_Attestation : Return "NATIONAL N DIPLOMA
ATTESTATION"
      Case Else : Return "CERTIFICATE"
    End Select
  End Function


    Select Case t
      Case CertificateType.QCTO_TradeTest, CertificateType.QCTO_StatementOfResults
        Return "Issued subject to QCTO verification and national learner records."
      Case CertificateType.Council_Engineering
        Return "Registration/recognition subject to engineering council governance and CPD
requirements."
      Case CertificateType.Council_Education
        Return "Recognition subject to education council regulations and professional standards."
```

```
            Case CertificateType.NRF_Award, CertificateType.SARAO_Award
                Return "Research award acknowledgement; verification as per the awarding body's criteria."
            Case CertificateType.Postdoctoral_Award
                Return "Postdoctoral award attestation; institutional verification applies."
            Case CertificateType.NationalN_Diploma_Attestation
                Return "Institutional attestation of National N Diploma achievements; awarding body remains custodian of the official certificate."
            Case Else
                Return "Verification required with issuing authority."
        End Select
    End Function


        Dim payload = $"serial={m.Serial}|id={p.IDNumber}|date={m.IssueDate:yyyy-MM-dd}|type={m.Type}"
        Dim hash = Sha256Hex(payload)
        sb.AppendLine($"Verification hash: {hash}")
        If Not String.IsNullOrWhiteSpace(m.VerificationURL) Then
            sb.AppendLine($"Verify at: {m.VerificationURL}?serial={m.Serial}")
            sb.AppendLine($"QR payload: {payload}")
        End If
    End Sub

    Private Function Sha256Hex(input As String) As String
        Using sha = SHA256.Create()
            Dim bytes = sha.ComputeHash(Encoding.UTF8.GetBytes(input))
            Return BitConverter.ToString(bytes).Replace("-", "").ToLowerInvariant()
        End Using
    End Function

End Module

Numbering, validation, printing, and auditPublic Module Governance

    ' --------- Serial rules (adjust to your authority prefixes) ----------

        Dim prefix As String = t.ToString().Replace("_", "-")
        Return $"{prefix}-{issueDate:yyyy}-{sequence:000000}"
    End Function


        Return serial.StartsWith(t.ToString().Replace("_", "-") & "-")
    End Function

    ' --------- Minimal PrintDocument wrapper ----------
    Public Class TextPrintJob
        Private ReadOnly _content As String
        Private _lines() As String
        Private _i As Integer

        Public Sub New(content As String)
```

```
        _content = content
        _lines = _content.Replace(vbCrLf, vbLf).Split(vbLf)
        _i = 0
    End Sub

    Public Sub Print(docName As String)
        Dim pd As New PrintDocument()
        pd.DocumentName = docName
        AddHandler pd.PrintPage, AddressOf OnPrintPage
        pd.Print()
    End Sub


        Dim font = New Font("Consolas", 9.0F)
        Dim lh = font.GetHeight(e.Graphics)
        Dim y = e.MarginBounds.Top
        Dim left = e.MarginBounds.Left
        Dim linesPerPage = CInt(Math.Floor(e.MarginBounds.Height / lh))
        Dim count As Integer = 0
        While count < linesPerPage AndAlso _i < _lines.Length
            e.Graphics.DrawString(_lines(_i), font, Brushes.Black, left, y)
            y += lh : count += 1 : _i += 1
        End While
        e.HasMorePages = (_i < _lines.Length)
    End Sub
End Class

' --------- Hash-chained audit log ----------
Public Class AuditLogger
    Private ReadOnly _path As String
    Private _lastHash As String = ""

    Public Sub New(path As String)
        _path = path
        If Not IO.File.Exists(_path) Then IO.File.WriteAllText(_path, "")
        _lastHash = GetLastHash()
    End Sub

    Public Sub Append(actor As String, Action As String, entity As String, entityId As String, details As
String)
        Dim ts = Date.UtcNow.ToString("o")
        Dim raw = $"{ts}|{actor}|{action}|{entity}|{entityId}|{details}|{_lastHash}"
        Dim hash = Sha256Hex(raw)
        IO.File.AppendAllText(_path, raw & "|" & hash & Environment.NewLine)
        _lastHash = hash
    End Sub


        Dim lines = IO.File.ReadAllLines(_path)
        If lines.Length = 0 Then Return ""
        Return lines.Last().Split("|"c).Last()
    End Function
```

```
    Private Function ComputeSha256Hash(s As String) As String
      Using sha = SHA256.Create()
        Dim b = sha.ComputeHash(Encoding.UTF8.GetBytes(s))
        Return BitConverter.ToString(b).Replace("-", "").ToLowerInvariant()
      End Using
    End Function
  End Class

End Module

Example usageModule Demo
    Dim person = New Person With {
      .FullName = "Tshingombe Tshitadi Fiston",
      .IDNumber = "9001015800082",
      .RegNo = "APP-EL-2025-0042",
      .Email = "t.t.fiston@example.org (mailto:t.t.fiston@example.org)"
    }

    Dim qcto = New ProviderAuthority With {
      .Name = "Quality Council for Trades and Occupations (QCTO)",
      .Address = "Pretoria, South Africa",
      .Contact = "info@qcto.org.za (mailto:info@qcto.org.za) | +27 (0)12 000 0000",
      .Accreditation = "Trade Test Centre: XYZ-000123 | SDP: SDP1220/18/00146"
    }

    Dim nimet = New ProviderAuthority With {
      .Name = "NICHRIST Institute of Management and Engineering Technology (NIMET)",
      .Address = "10 Top Road, Anderbolt, Boksburg 1459, South Africa",
      .Contact = "Tel: 067 154 8507 | info@nimet.co.za (mailto:info@nimet.co.za)",
      .Accreditation = "QCTO: SDP1220/18/00146 | DHET: 0899992889 | Umalusi: 20 FET02 00191 PA"
    }

    Dim q_nDipl = New QualificationRef With {
      .Title = "National N Diploma: Electrical Engineering",
      .SAQAID = "SAQA-XXXXXX",
      .NQFLevel = "6",
      .Credits = 360,
      .Specialisation = "Electrotechnics"
    }

    Dim tradeOutcomes = New List(Of OutcomeRecord) From {
      New OutcomeRecord With {.Code = "TT-EL-01", .Name = "Electrical Systems Fault Finding", .Result =
"Competent", .DateAchieved = #2025-08-18#, .Score = "Comp", .Credits = 0},
      New OutcomeRecord With {.Code = "TT-EL-02", .Name = "Schematic Interpretation", .Result =
"Competent", .DateAchieved = #2025-08-18#, .Score = "Comp", .Credits = 0}
    }

    Dim metaTrade = New CertificateMeta With {
      .Type = CertificateType.QCTO_TradeTest,
      .Serial = Governance.GenerateSerial(CertificateType.QCTO_TradeTest, Date.Today, 73),
```

```
        .IssueDate = Date.Today,
        .SignatoryTitle = "Trade Test Centre Manager",
        .VerificationURL = "https://verify.example.org (https://verify.example.org)"
    }

    Dim docTrade = CertificateRenderers.RenderCertificate(person, qcto, Nothing, tradeOutcomes,
metaTrade)
    Console.WriteLine (docTrade)
    ' New Governance.TextPrintJob(docTrade).Print("QCTO Trade Test Certificate")

    Dim outcomesN = New List(Of OutcomeRecord) From {
        New OutcomeRecord With {.Code = "N4-ET-01", .Name = "Electrotechnics", .Result = "Pass",
.DateAchieved = #2024-11-30#, .Score = "68", .Credits = 0},
        New OutcomeRecord With {.Code = "N5-ET-02", .Name = "Electrotechnologie", .Result = "Pass",
.DateAchieved = #2025-03-31#, .Score = "62", .Credits = 0},
        New OutcomeRecord With {.Code = "N6-PO-03", .Name = "Plant Operations", .Result = "Pass",
.DateAchieved = #2025-06-30#, .Score = "71", .Credits = 0}
    }

    Dim metaN = New CertificateMeta With {
        .Type = CertificateType.NationalN_Diploma_Attestation,
        .Serial = Governance.GenerateSerial(CertificateType.NationalN_Diploma_Attestation, Date.Today,
15),
        .IssueDate = Date.Today,
        .Signatory = "Nicholas Phiri",
        .SignatoryTitle = "Director",
        .VerificationURL = "https://verify.nimet.co.za (https://verify.nimet.co.za)"
    }

    Dim docN = CertificateRenderers.RenderCertificate(person, nimet, q_nDipl, outcomesN, metaN)
    Console.WriteLine()
    Console.WriteLine (docN)

    ' Re-certification of council engineering certificate
    Dim engCouncil = New ProviderAuthority With {
        .Name = "Engineering Council Certificate",
        .Address = "South Africa",
        .Contact = "support@engcouncil.org.za (mailto:support@engcouncil.org.za)",
        .Accreditation = "Professional Registration Reference: PR-ENG-2025"
    }
    Dim metaCouncil = New CertificateMeta With {
        .Type = CertificateType.Council_Engineering,
        .Serial = Governance.GenerateSerial(CertificateType.Council_Engineering, Date.Today, 101),
        .IssueDate = Date.Today,
        .Signatory = "Registrar",
        .SignatoryTitle = "Council Registrar",
        .VerificationURL = "https://verify.council.org.za (https://verify.council.org.za)",
        .IsReissue = True,
        .ReplacesSerial = "COUNCIL-ENGINEERING-2023-000045",
        .ReissueReason = "Legal name change"
```

```
        }
        Dim c[illegible]CertificateEnd(https://www.elektormagazine.com)
Enumerable.Empty(Of OutcomeRecord)(), metaCouncil)
        Console.WriteLine()
        Console.WriteLine (councilDoc)

        ' Audit trail
        Dim audit = New Governance.AuditLogger("certificate_audit.log")
        audit.Append("system", "Issued", metaTrade.Type.ToString(), metaTrade.Serial, "QCTO trade test")
        audit.Append("system", "Issued", metaN.Type.ToString(), metaN.Serial, "N Diploma attestation")
        audit.Append("registrar", "Reissued", metaCouncil.Type.ToString(), metaCouncil.Serial,
$"Replaces={metaCouncil.ReplacesSerial}")
    End Sub
End Module

Where to tune next
Imports System.text
Imports System.Security.Cryptography
Imports System.Drawing.Printing

Public Enum DocType
    DHA_ID_Application
    DHA_ID_Statement
    DHA_Citizenship_Certificate_Attestation
    DHA_PermanentResidence_Attestation
    Refugee_Asylum_Permit_Summary
    Refugee_Status_Recognition_Summary
    Appeal_Letter
    Work_Permit_Support_Letter
    Birth_Certificate_Extract_Attestation
    Labour_COID_GoodStanding
    UIF_Compliance_Letter
    Machinery_Regulator_Registration
End Enum

Public Enum CaseStatus
    Draft
    Submitted
    InReview
    approved
    Rejected
    Appealed
    Resolved
End Enum

Public Class Person
    Public Property FullName As String
    Public Property DOB As Date
    Public Property Nationality As String
    Public Property IDNumber As String       ' SA ID or Passport
    Public Property PassportNumber As String
```

```
    Public Property RefugeeFileNumber As String
    Public Property ContactEmail As String
    Public Property ContactPhone As String
End Class

Public Class Organisation
    Public Property Name As String
    Public Property Address As String
    Public Property Contact As String
    Public Property RegNumbers As String      ' e.g., COID Reg, UIF Ref, Company Reg
End Class

Public Class CaseMeta
    Public Property Doc As DocType
    Public Property Serial As String
    Public Property Status As CaseStatus
    Public Property Reason As String          ' e.g., refusal reason, appeal grounds
    Public Property IssueDate As Date
    Public Property Reference As String       ' File/reference number
    Public Property Signatory As String
    Public Property SignatoryTitle As String
    Public Property VerificationURL As String
End Class

Public Class AttachmentRef
    Public Property Name As String            ' e.g., Proof of address
    Public Property Description As String
    Public Property FilePath As String
End ClassImports System.Text
Imports System.Drawing.Printing
Imports System.Security.Cryptography

Public Enum DocType
    Labour_Competence_Certificate             ' Engineering competence/authorisation
    PSIRA_Certificate_Management              ' Security management/grade/registration
    SAPS_Firearm_Competency_Certificate       ' Firearm competency outcome
    Police_Clearance_Certificate_Attestation  ' Clearance summary/attestation
    DOJ_Dossier_Summary                       ' DoJ&CD matter summary
    HighCourt_Transcript_Certificate          ' Transcript cover/attestation
    LabourCourt_Transcript_Certificate
    CCMA_Award_Certificate                    ' Award/outcome cover
    HR_Outcome_Letter                         ' Offer/termination/misconduct/outcome
    SAQA_Statement_Of_Results                 ' Units/modules transcript
    Reissue_Notice
End Enum

Public Enum CaseStatus
Draft:     Submitted: InReview: approved: Rejected: Awarded: Enforced: Reissued
End Enum

Public Class Organisation
```

```
    Public Property Name As String
    Public Property Address As String
    Public Property Contact As String
    Public Property Registrations As String     ' e.g., PSIRA/SAPS vendor/COID/UIF/company refs
End Class

Public Class Person
    Public Property FullName As String
    Public Property IDNumber As String
    Public Property DOB As Date
    Public Property EmployeeNo As String
    Public Property Email As String
    Public Property Phone As String
End Class

Public Class CaseMeta
    Public Property Type As DocType
    Public Property Serial As String
    Public Property Reference As String        ' Case/file/ref (e.g., CCMA GAJBxxxx)
    Public Property IssueDate As Date
    Public Property Status As CaseStatus
    Public Property Reason As String           ' Grounds/notes (refusal/award basis)
    Public Property Signatory As String
    Public Property SignatoryTitle As String
    Public Property VerificationURL As String
    ' Reissue
    Public Property IsReissue As Boolean
    Public Property ReplacesSerial As String
    Public Property ReissueReason As String
End Class

' Tabular lines for transcripts/awards/SoR
Public Class LineItem
    Public Property Code As String             ' Module/section/case item
    Public Property Title As String            ' e.g., Unit Standard, count, exhibit
    Public Property DateEntry As Date
    Public Property Score As String            ' %, “Comp”, count, amount
    Public Property Credits As Integer         ' SAQA credits where relevant
    Public Property Result As String           ' Pass/Fail/Comp/Granted/Dismissed
End Class

Renderers
vbPublic Module Renderers

    Public Function RenderDocument(org As Organisation,
                                   person As Person,
                                   meta As CaseMeta,
                                   Optional headerDetails As Dictionary(Of String, String) = Nothing,
                                   Optional items As IEnumerable(Of LineItem) = Nothing,
                                   Optional footerDetails As Dictionary(Of String, String) = Nothing) As String
        Dim sb As New StringBuilder()
```

```
    ' Header
    sb.AppendLine (org.Name)
    If Not String.IsNullOrWhiteSpace(org.Address) Then sb.AppendLine(org.Address)
    If Not String.IsNullOrWhiteSpace(org.Contact) Then sb.AppendLine(org.Contact)
    If Not String.IsNullOrWhiteSpace(org.Registrations) Then sb.AppendLine(org.Registrations)
    sb.AppendLine(New String("-"c, 88))
    sb.AppendLine (TitleFor(meta.Type))
    sb.AppendLine(New String("-"c, 88))

    ' Person + meta
    sb.AppendLine($"Name: {person.FullName}   ID: {person.IDNumber}   DOB: {person.DOB:yyyy-MM-dd}
Emp#: {person.EmployeeNo}")
    sb.AppendLine($"Contact: {person.Email} | {person.Phone}")
    sb.AppendLine($"Reference: {meta.Reference}   Serial: {meta.Serial}   Status: {meta.Status}   Issued:
{meta.IssueDate:yyyy-MM-dd}")
    If Not String.IsNullOrWhiteSpace(meta.Reason) Then sb.AppendLine($"Notes: {meta.Reason}")

    ' Header details (authority-specific fields)
    If headerDetails IsNot Nothing AndAlso headerDetails.Count > 0 Then
      sb.AppendLine(New String("-"c, 88))
      For Each kv In headerDetails
        sb.AppendLine($" - {kv.Key}: {kv.Value}")
      Next
    End If

    ' Table
    If items IsNot Nothing AndAlso items.Any() Then
      sb.AppendLine(New String("-"c, 88))
      sb.AppendLine ("Code       Title                                    Date       Credits  Score   Result")
      sb.AppendLine ("---------  ---------------------------------------  ---------  -------  ------  ---------")
      For Each it In items.OrderBy(Function(x) x.DateEntry).ThenBy(Function(x) x.Code)
        sb.AppendLine($"{it.Code.PadRight(10)} {it.Title.PadRight(46)} {it.DateEntry:yyyy-MM-dd}
{it.Credits.ToString().PadLeft(7)}  {it.Score.PadLeft(6)}  {it.Result.PadRight(10)}")
      Next
    End If

    ' Footer details
    If footerDetails IsNot Nothing AndAlso footerDetails.Count > 0 Then
      sb.AppendLine(New String("-"c, 88))
      For Each kv In footerDetails
        sb.AppendLine($" - {kv.Key}: {kv.Value}")
      Next
    End If

    ' Footer
    sb.AppendLine(New String("-"c, 88))
    sb.AppendLine($"Signed: {meta.Signatory}, {meta.SignatoryTitle}")
    If meta.IsReissue Then sb.AppendLine($"Re-issue of: {meta.ReplacesSerial} | Reason:
{meta.ReissueReason}")
    sb.AppendLine (FooterFor(meta.Type))
```

```
    AppendVerification(sb, person, meta)
    Return sb.ToString()
  End Function


    Select Case t
      Case DocType.Labour_Competence_Certificate : Return "LABOUR COMPETENCE CERTIFICATE
(ENGINEERING)"
      Case DocType.PSIRA_Certificate_Management : Return "PSIRA CERTIFICATE OF REGISTRATION
(MANAGEMENT)"
      Case DocType.SAPS_Firearm_Competency_Certificate : Return "SAPS FIREARM COMPETENCY
CERTIFICATE (SUMMARY)"
      Case DocType.Police_Clearance_Certificate_Attestation : Return "POLICE CLEARANCE CERTIFICATE
ATTESTATION"
      Case DocType.DOJ_Dossier_Summary : Return "DEPARTMENT OF JUSTICE DOSSIER SUMMARY"
      Case DocType.HighCourt_Transcript_Certificate : Return "HIGH COURT TRANSCRIPT CERTIFICATE"
      Case DocType.LabourCourt_Transcript_Certificate : Return "LABOUR COURT TRANSCRIPT
CERTIFICATE"
      Case DocType.CCMA_Award_Certificate : Return "CCMA AWARD CERTIFICATE"
      Case DocType.HR_Outcome_Letter : Return "HR OUTCOME LETTER"
      Case DocType.SAQA_Statement_Of_Results : Return "SAQA-ALIGNED STATEMENT OF RESULTS"
      Case DocType.Reissue_Notice : Return "CERTIFICATE RE-ISSUE NOTICE"
      Case Else : Return "OFFICIAL CERTIFICATE"
    End Select
  End Function


    Select Case t
      Case DocType.PSIRA_Certificate_Management
        Return "Subject to PSIRA verification and continued compliance with security legislation."
      Case DocType.SAPS_Firearm_Competency_Certificate,
DocType.Police_Clearance_Certificate_Attestation
        Return "Summary for file/HR use. Official record remains with SAPS CRC/CFR."
      Case DocType.HighCourt_Transcript_Certificate, DocType.LabourCourt_Transcript_Certificate
        Return "Transcript issued/attested for the referenced matter; check with court registrar for
certified copies."
      Case DocType.CCMA_Award_Certificate
        Return "Award outcome summary; service and enforcement per LRA and CCMA rules."
      Case DocType.Labour_Competence_Certificate
        Return "Competence attestation; authorisations and licenses remain with the regulator/
appointing authority."
      Case DocType.HR_Outcome_Letter
        Return "Internal HR outcome; subject to company policy and applicable labour law."
      Case DocType.SAQA_Statement_Of_Results
        Return "This statement reflects learning achievements aligned for SAQA/NLRD reporting."
      Case DocType.DOJ_Dossier_Summary
        Return "Dossier summary for DoJ&CD proceedings; consult official records for certified
documents."
      Case DocType.Reissue_Notice
        Return "This re-issue supersedes prior serial; chain-of-custody retained for audit."
```

```
        Case Else
            R"Verification with ass" (https://www.elektormagazine.com)
        End Select
    End Function


    Dim payload = $"serial={meta.Serial}|ref={meta.Reference}|id={person.IDNumber}|
date={meta.IssueDate:yyyy-MM-dd}|type={meta.Type}|status={meta.Status}"
    Dim hash = Sha256Hex(payload)
    sb.AppendLine($"Verification hash: {hash}")
    If Not String.IsNullOrWhiteSpace(meta.VerificationURL) Then
        sb.AppendLine($"Verify at: {meta.VerificationURL}?serial={meta.Serial}")
        sb.AppendLine($"QR payload: {payload}")
    End If
  End Sub

  Private Function Sha256Hex(input As String) As String
    Using sha = SHA256.Create()
      Dim b = sha.ComputeHash(Encoding.UTF8.GetBytes(input))
      Return BitConverter.ToString(b).Replace("-", "").ToLowerInvariant()
    End Using
  End Function
End Module

Printing and audit
vbPublic Class TextPrintJob
  Private ReadOnly _content As String
  Private _lines() As String
  Private _idx As Integer

  Public Sub New(content As String)
    _content = content
    _lines = _content.Replace(vbCrLf, vbLf).Split(ControlChars.Lf)
  End Sub

  Public Sub Print(docName As String)
    Dim pd As New PrintDocument()
    pd.DocumentName = docName
    AddHandler pd.PrintPage, AddressOf OnPrintPage
    pd.Print()
  End Sub


    Dim font = New Font("Consolas", 9.0F)
    Dim lh = font.GetHeight(e.Graphics)
    Dim y = e.MarginBounds.Top
    Dim left = e.MarginBounds.Left
    Dim perPage = CInt(Math.Floor(e.MarginBounds.Height / lh))
    Dim count As Integer = 0
    While count < perPage AndAlso _idx < _lines.Length
      e.Graphics.DrawString(_lines(_idx), font, Brushes.Black, left, y)
```

```
        y += lh : count += 1 : _idx += 1
    End W
    e.HasMorePages = (_idx < _lines.Length)
  End Sub
End Class

Public Class AuditLogger
  Private ReadOnly _path As String
  Private _lastHash As String = ""

  Public Sub New(path As String)
    _path = path
    If Not IO.File.Exists(_path) Then IO.File.WriteAllText(_path, "")
    _lastHash = GetLastHash()
  End Sub



    Dim ts = Date.UtcNow.ToString("o")
    Dim raw = $"{ts}|{actor}|{action}|{entity}|{entityId}|{details}|{_lastHash}"
    Dim hash = Sha256Hex(raw)
    IO.File.AppendAllText(_path, raw & "|" & hash & Environment.NewLine)
    _lastHash = hash
  End Sub



    Dim lines = IO.File.ReadAllLines(_path)
    If lines.Length = 0 Then Return ""
    Return lines.Last().Split("|"c).Last()
  End Function



    Using sha = SHA256.Create()
      Dim b = sha.ComputeHash(Encoding.UTF8.GetBytes(s))
      Return BitConverter.ToString(b).Replace("-", "").ToLowerInvariant()
    End Using
  End Function
End Class

Example usageModule Demo

    Dim org = New Organisation With {
      .Name = "NICHRIST Institute of Management and Engineering Technology (NIMET)",
      .Address = "10 Top Road, Anderbolt, Boksburg 1459, South Africa",
      .Contact = "Tel: 067 154 8507 | info@nimet.co.za (mailto:info@nimet.co.za)",
      .Registrations = "PSIRA: 1234567 | COID: R9876543 | UIF: U1234567 | Company: 2017/067113/07"
    }
    Dim person = New Person With {
      .FullName = "Tshingombe Tshitadi Fiston",
      .IDNumber = "9001015800082",
      .DOB = #1990-01-01#,
      .EmployeeNo = "ENG-042",
```

```
    .Email = "fiston@example.org (mailto:fiston@example.org)",
    .Ph[illegible]
}

' 1) Labour competence (engineering)
Dim metaLab As New CaseMeta With {
    .Type = DocType.Labour_Competence_Certificate,
    .Serial = "LAB-COMP-2025-000121",
    .Reference = "OHS-APPT-18.1/2025/121",
    .IssueDate = Date.Today,
    .Status = CaseStatus.Approved,
    .Reason = "Competent for LV/MV switchgear isolation and lockout per SOP-OHS-12.",
    .Signatory = "Safety Manager",
    .SignatoryTitle = "OHS 16(2) Appointee",
    .VerificationURL = "https://verify.nimet.co.za (https://verify.nimet.co.za)"
}
Dim hdrLab = New Dictionary(Of String, String) From {
    {"Scope", "Isolation, lockout, testing, permit-to-work"},
    {"Plant", "MV Switchgear, MCCs, VSDs"},
    {"Validity", "24 months, with 6-month refresher"}
}
Dim itemsLab = New List(Of LineItem) From {
    New LineItem With {.Code = "PTW-01", .Title = "Permit-to-Work Competency", .DateEntry =
#2025-09-01#, .Credits = 0, .Score = "Comp", .Result = "Pass"},
    New LineItem With {.Code = "LOTO-02", .Title = "Lockout/Tagout Practical", .DateEntry =
#2025-09-01#, .Credits = 0, .Score = "Comp", .Result = "Pass"}
}
Console.WriteLine (Renderers.RenderDocument(org, person, metaLab, hdrLab, itemsLab))

' 2) PSIRA management certificate
Dim metaPsira As New CaseMeta With {
    .Type = DocType.PSIRA_Certificate_Management,
    .Serial = "PSIRA-MGMT-2025-000077",
    .Reference = "PSIRA-REG-EMP-001122",
    .IssueDate = Date.Today,
    .Status = CaseStatus.Approved,
    .Reason = "Compliant with PSIRA Grade A management requirements.",
    .Signatory = "Security Compliance Officer",
    .SignatoryTitle = "PSIRA Liaison"
}
Dim hdrPsira = New Dictionary(Of String, String) From {
    {"PSIRA Grade", "A"},
    {"Category", "Security Management"},
    {"Registration Validity", "2025-09-01 to 2026-08-31"}
}
Console.WriteLine()
Console.WriteLine (Renderers.RenderDocument(org, person, metaPsira, hdrPsira))

' 3) SAPS firearm competency summary
Dim metaFirearm As New CaseMeta With {
    .Type = DocType.SAPS_Firearm_Competency_Certificate,
```

```
        .Serial = "SAPS-FC-2025-000233",
        .Reference = "FC-ENDS-45/2025",
        .IssueDate = Date.Today,
        .Status = CaseStatus.Approved,
        .Reason = "Competent for handgun, rifle (business purposes).",
        .Signatory = "Firearm Compliance Officer",
        .SignatoryTitle = "Designated Official"
    }
    Dim hdrFirearm = New Dictionary(Of String, String) From {
        {"Proficiency", "U/S 119649–119651 (Demonstrate knowledge of Firearms Control Act), SASSETA credits"},
        {"Competency", "Handgun, Rifle (Business)"},
        {"CFR Status", "Approved, card pending"}
    }
    Console.WriteLine()
    Console.WriteLine (Renderers.RenderDocument(org, person, metaFirearm, hdrFirearm))

    ' 4) Police clearance attestation
    Dim metaPCC As New CaseMeta With {
        .Type = DocType.Police_Clearance_Certificate_Attestation,
        .Serial = "PCC-ATT-2025-000089",
        .Reference = "SAPS-CRC-2025/09/0089",
        .IssueDate = Date.Today,
        .Status = CaseStatus.Approved,
        .Reason = "No adverse record found (as per CRC feedback).",
        .Signatory = "Records Officer",
        .SignatoryTitle = "Compliance Registry"
    }
    Dim hdrPCC = New Dictionary(Of String, String) From {
        {"Submission", "2025-08-20"},
        {"CRC Ref", "CRC-09-2025-7765"},
        {"Result", "Clear"}
    }
    Console.WriteLine()
    Console.WriteLine (Renderers.RenderDocument(org, person, metaPCC, hdrPCC))

    ' 5) High Court transcript certificate
    Dim metaHC As New CaseMeta With {
        .Type = DocType.HighCourt_Transcript_Certificate,
        .Serial = "HC-TR-CERT-2025-000041",
        .Reference = "2025/HC/GAUT/012345",
        .IssueDate = Date.Today,
        .Status = CaseStatus.Awarded,
        .Signatory = "Court Liaison",
        .SignatoryTitle = "Registrar Interface"
    }
    Dim itemsHC = New List(Of LineItem) From {
        New LineItem With {.Code = "VOL1", .Title = "Proceedings Volume 1", .DateEntry = #2025-07-01#, .Score = "178p", .Credits = 0, .Result = "Filed"},
        New LineItem With {.Code = "JDG", .Title = "Judgment", .DateEntry = #2025-08-15#, .Score = "15p", .Credits = 0, .Result = "Delivered"}
```

```
    }
    Console.WriteLine()
    Console.WriteLine (Renderers.RenderDocument(org, person, metaHC, Nothing, itemsHC))

    ' 6) Labour Court transcript certificate
    Dim metaLC As New CaseMeta With {
        .Type = DocType.LabourCourt_Transcript_Certificate,
        .Serial = "LC-TR-CERT-2025-000022",
        .Reference = "JR 1234/25",
        .IssueDate = Date.Today,
        .Status = CaseStatus.Awarded,
        .Signatory = "Court Liaison",
        .SignatoryTitle = "Registrar Interface"
    }
    Console.WriteLine()
    Console.WriteLine (Renderers.RenderDocument(org, person, metaLC, Nothing, itemsHC))

    ' 7) CCMA award certificate
    Dim metaCCMA As New CaseMeta With {
        .Type = DocType.CCMA_Award_Certificate,
        .Serial = "CCMA-AWD-2025-000311",
        .Reference = "GAJB 12345-25",
        .IssueDate = Date.Today,
        .Status = CaseStatus.Awarded,
        .Reason = "Reinstatement with back pay (30 days).",
        .Signatory = "Commissioner",
        .SignatoryTitle = "CCMA"
    }
    Dim itemsCCMA = New List(Of LineItem) From {
        New LineItem With {.Code = "AWD", .Title = "Award Outcome", .DateEntry = #2025-09-10#, .Score =
"R 45,000", .Credits = 0, .Result = "Granted"},
        New LineItem With {.Code = "CST", .Title = "Costs", .DateEntry = #2025-09-10#, .Score = "Each own",
.Credits = 0, .Result = "Set"}
    }
    Console.WriteLine()
    Console.WriteLine (Renderers.RenderDocument(org, person, metaCCMA, Nothing, itemsCCMA))

    ' 8) HR outcome letter (disciplinary/appointment)
    Dim metaHR As New CaseMeta With {
        .Type = DocType.HR_Outcome_Letter,
        .Serial = "HR-OUT-2025-000141",
        .Reference = "HR-DISC-2025/09/0141",
        .IssueDate = Date.Today,
        .Status = CaseStatus.Resolved,
        .Reason = "Final written warning; performance plan agreed.",
        .Signatory = "HR Director",
        .SignatoryTitle = "Human Resources"
    }
    Dim hdrHR = New Dictionary(Of String, String) From {
        {"Effective", "2025-09-30"},
        {"Review", "2026-03-31"},
```

```
        {"Appeal Window", "5 working days"}
    }
    Console.WriteLine()
    Console.WriteLine (Renderers.RenderDocument(org, person, metaHR, hdrHR))

    ' 9) SAQA Statement of Results (modules/US)
    Dim metaSoR As New CaseMeta With {
        .Type = DocType.SAQA_Statement_Of_Results,
        .Serial = "SAQA-SOR-2025-000219",
        .Reference = "NLRD-REF-2025/09/0219",
        .IssueDate = Date.Today,
        .Status = CaseStatus.Approved,
        .Signatory = "Registry Officer",
        .SignatoryTitle = "Learning Records"
    }
    Dim itemsSoR = New List(Of LineItem) From {
        New LineItem With {.Code = "244288", .Title = "Apply SHE principles (Engineering Safety)",
.DateEntry = #2025-06-30#, .Credits = 10, .Score = "Comp", .Result = "Pass"},
        New LineItem With {.Code = "119649", .Title = "Handle firearms safely (knowledge)", .DateEntry =
#2025-05-20#, .Credits = 3, .Score = "Comp", .Result = "Pass"}
    }
    Console.WriteLine()
    Console.WriteLine (Renderers.RenderDocument(org, person, metaSoR, Nothing, itemsSoR))

    ' Audit
    Dim audit = New AuditLogger("labour_security_audit.log")
    audit.Append("system", "Issued", "LabourCompetence", metaLab.Serial, metaLab.Reference)
    audit.Append("system", "Issued", "PSIRA", metaPsira.Serial, metaPsira.Reference)
    audit.Append("system", "Issued", "SAPSFirearm", metaFirearm.Serial, metaFirearm.Reference)
    audit.Append("system", "Issued", "PCC", metaPCC.Serial, metaPCC.Reference)
    audit.Append("system", "Issued", "HighCourtTranscript", metaHC.Serial, metaHC.Reference)
    audit.Append("system", "Issued", "LabourCourtTranscript", metaLC.Serial, metaLC.Reference)
    audit.Append("system", "Issued", "CCMAAward", metaCCMA.Serial, metaCCMA.Reference)
    audit.Append("system", "Issued", "HROutcome", metaHR.Serial, metaHR.Reference)
    audit.Append("system", "Issued", "SAQA-SoR", metaSoR.Serial, metaSoR.Reference)
  End Sub
End ModulePublic Module Renderers

  Public Function RenderDocument(org As Organisation,
                                 person As Person,
                                 meta As CaseMeta,
                                 Optional headerDetails As Dictionary(Of String, String) = Nothing,
                                 Optional items As IEnumerable(Of LineItem) = Nothing,
                                 Optional footerDetails As Dictionary(Of String, String) = Nothing) As String
    Dim sb As New StringBuilder()

    ' Header
    sb.AppendLine (org.Name)
    If Not String.IsNullOrWhiteSpace(org.Address) Then sb.AppendLine(org.Address)
    If Not String.IsNullOrWhiteSpace(org.Contact) Then sb.AppendLine(org.Contact)
    If Not String.IsNullOrWhiteSpace(org.Registrations) Then sb.AppendLine(org.Registrations)
```

```
        sb.AppendLine(New String("-"c, 90))
        sb.AppendLine($"Certificate Type: {meta.Type}")
        sb.AppendLine(New String("-"c, 90))

        ' Person + meta
        sb.AppendLine($"Name: {person.FullName}   ID: {person.IDNumber}   DOB: {person.DOB:yyyy-MM-dd}
Emp#: {person.EmployeeNo}")
        sb.AppendLine($"Contact: {person.Email} | {person.Phone}")
        sb.AppendLine($"Reference: {meta.Reference}   Serial: {meta.Serial}   Status: {meta.Status}   Issued:
{meta.IssueDate:yyyy-MM-dd}")
        If Not String.IsNullOrWhiteSpace(meta.Notes) Then sb.AppendLine($"Notes: {meta.Notes}")

        ' Authority-specific header details
        If headerDetails IsNot Nothing AndAlso headerDetails.Count > 0 Then
            sb.AppendLine(New String("-"c, 90))
            For Each kv In headerDetails
                sb.AppendLine($" - {kv.Key}: {kv.Value}")
            Next
        End If

        ' Tabular content
        If items IsNot Nothing AndAlso items.Any() Then
            sb.AppendLine(New String("-"c, 90))
            sb.AppendLine ("Code       Title                                    Date       Credits  Score    Result")
            sb.AppendLine ("---------- ---------------------------------------- ---------- ------- ------- ----------")
            For Each it In items.OrderBy(Function(x) x.DateEntry).ThenBy(Function(x) x.Code)
                sb.AppendLine($"{it.Code.PadRight(10)} {it.Title.PadRight(53)} {it.DateEntry:yyyy-MM-dd}
{it.Credits.ToString().PadLeft(7)}  {it.Score.PadLeft(7)}  {it.Result.PadRight(10)}")
            Next
        End If

        ' Footer details
        If footerDetails IsNot Nothing AndAlso footerDetails.Count > 0 Then
            sb.AppendLine(New String("-"c, 90))
            For Each kv In footerDetails
                sb.AppendLine($" - {kv.Key}: {kv.Value}")
            Next
        End If

        ' Footer + verification
        sb.AppendLine(New String("-"c, 90))
        sb.AppendLine($"Signed: {meta.Signatory}, {meta.SignatoryTitle}")
        If meta.IsReissue Then sb.AppendLine($"Re-issue of: {meta.ReplacesSerial} | Reason:
{meta.ReissueReason}")
        sb.AppendLine (FooterFor(meta.Type))
        AppendVerification(sb, person, meta)

        Return sb.ToString()
    End Function
```

```
    Select Case t
        Case DocType.Labour_Competence_Certificate : Return "LABOUR COMPETENCE CERTIFICATE
(ENGINEERING)"
        Case DocType.PSIRA_Management_Certificate : Return "PSIRA MANAGEMENT CERTIFICATE"
        Case DocType.SAPS_Firearm_Competency_Summary : Return "SAPS FIREARM COMPETENCY
SUMMARY"
        Case DocType.Police_Clearance_Attestation : Return "POLICE CLEARANCE CERTIFICATE
ATTESTATION"
        Case DocType.DOJ_Matter_Summary : Return "DEPARTMENT OF JUSTICE MATTER SUMMARY"
        Case DocType.HighCourt_Transcript_Certificate : Return "HIGH COURT TRANSCRIPT CERTIFICATE"
        Case DocType.LabourCourt_Transcript_Certificate : Return "LABOUR COURT TRANSCRIPT
CERTIFICATE"
        Case DocType.CCMA_Award_Certificate : Return "CCMA AWARD CERTIFICATE"
        Case DocType.HR_Outcome_Letter : Return "HR OUTCOME LETTER"
        Case DocType.SAQA_Statement_Of_Results : Return "SAQA-ALIGNED STATEMENT OF RESULTS"
        Case DocType.Reissue_Notice : Return "CERTIFICATE RE-ISSUE NOTICE"
        Case Else : Return "OFFICIAL CERTIFICATE"
    End Select
  End Function


    Select Case t
        Case DocType.PSIRA_Management_Certificate
            Return "Subject to PSIRA verification and continued compliance with security legislation."
        Case DocType.SAPS_Firearm_Competency_Summary, DocType.Police_Clearance_Attestation
            Return "Summary for HR/file use. Official record remains with SAPS CRC/CFR."
        Case DocType.HighCourt_Transcript_Certificate, DocType.LabourCourt_Transcript_Certificate
            Return "Transcript attestation; consult the registrar for certified copies."
        Case DocType.CCMA_Award_Certificate
            Return "Award summary; service and enforcement per LRA and CCMA rules."
        Case DocType.Labour_Competence_Certificate
            Return "Competence attestation; legal authorisations remain with the regulator/appointing
authority."
        Case DocType.HR_Outcome_Letter
            Return "Internal HR outcome; subject to policy and applicable labour law."
        Case DocType.SAQA_Statement_Of_Results
            Return "Learning achievements aligned for SAQA/NLRD reporting."
        Case DocType.DOJ_Matter_Summary
            Return "Dossier summary for proceedings; official records held by the court/DoJ."
        Case DocType.Reissue_Notice
            Return "This re-issue supersedes prior serial; chain-of-custody retained for audit."
        Case Else
            Return "Verification with issuing authority may be required."
    End Select
  End Function


    Dim payload = $"serial={meta.Serial}|ref={meta.Reference}|id={person.IDNumber}|
date={meta.IssueDate:yyyy-MM-dd}|type={meta.Type}|status={meta.Status}"
    Dim hash = Sha256Hex(payload)
    sb.AppendLine($"Verification hash: {hash}")
```

```
        If Not String.IsNullOrWhiteSpace(meta.VerificationURL) Then
            sb.AppendLine($"Verify at: {meta.VerificationURL}?s={serial}")
            sb.AppendLine($"QR payload: {payload}")
        End If
    End Sub



        Using sha = SHA256.Create()
            Dim b = sha.ComputeHash(Encoding.UTF8.GetBytes(s))
            Return BitConverter.ToString(b).Replace("-", "").ToLowerInvariant()
        End Using
    End Function

End Module

Public Class TextPrintJob
    Private ReadOnly _content As String
    Private _lines() As String
    Private _idx As Integer

    Public Sub New(content As String)
        _content = content
        _lines = _content.Replace(vbCrLf, vbLf).Split(ControlChars.Lf)
    End Sub

    Public Sub Print(docName As String)
        Dim pd As New PrintDocument()
        pd.DocumentName = docName
        AddHandler pd.PrintPage, AddressOf OnPrintPage
        pd.Print()
    End Sub



        Dim font = New Font("Consolas", 9.0F)
        Dim lh = font.GetHeight(e.Graphics)
        Dim y = e.MarginBounds.Top
        Dim left = e.MarginBounds.Left
        Dim perPage = CInt(Math.Floor(e.MarginBounds.Height / lh))
        Dim count As Integer = 0
        While count < perPage AndAlso _idx < _lines.Length
            e.Graphics.DrawString(_lines(_idx), font, Brushes.Black, left, y)
            y += lh : count += 1 : _idx += 1
        End While
        e.HasMorePages = (_idx < _lines.Length)
    End Sub
End Class

serial Governance And audit
vbPublic Module Governance
    Public Function GenerateSerial(prefix As String, issueDate As Date, sequence As Integer) As String
        Return $"{prefix}-{issueDate:yyyy}-{sequence:000000}"
```

```
    End Function
End Module

Public Class AuditLogger
    Private ReadOnly _path As String
    Private _lastHash As String = ""

    Public Sub New(path As String)
        _path = path
        If Not IO.File.Exists(_path) Then IO.File.WriteAllText(_path, "")
        _lastHash = GetLastHash()
    End Sub


        Dim ts = Date.UtcNow.ToString("o")
        Dim raw = $"{ts}|{actor}|{action}|{entity}|{entityId}|{details}|{_lastHash}"
        Dim hash = Sha256Hex(raw)
        IO.File.AppendAllText(_path, raw & "|" & hash & Environment.NewLine)
        _lastHash = hash
    End Sub


        Dim lines = IO.File.ReadAllLines(_path)
        If lines.Length = 0 Then Return ""
        Return lines.Last().Split("|"c).Last()
    End Function


        Using sha = SHA256.Create()
            Dim b = sha.ComputeHash(Encoding.UTF8.GetBytes(s))
            Return BitConverter.ToString(b).Replace("-", "").ToLowerInvariant()
        End Using
    End Function
End Class

Example usageModule Demo

        Dim org = New Organisation With {
            .Name = "NICHRIST Institute of Management and Engineering Technology (NIMET)",
            .Address = "10 Top Road, Anderbolt, Boksburg 1459, South Africa",
            .Contact = "Tel: 067 154 8507 | info@nimet.co.za (mailto:info@nimet.co.za)",
            .Registrations = "PSIRA: 1234567 | COID: R9876543 | UIF: U1234567 | Co. Reg: 2017/067113/07"
        }
        Dim person = New Person With {
            .FullName = "Tshingombe Tshitadi Fiston",
            .IDNumber = "9001015800082",
            .DOB = #1990-01-01#,
            .EmployeeNo = "ENG-042",
            .Email = "fiston@example.org (mailto:fiston@example.org)",
            .Phone = "+27 72 000 0000"
        }
```

```
' Labour competence certificate
Dim metaLab As New CaseMeta With {
    .Type = DocType.Labour_Competence_Certificate,
    .Serial = Governance.GenerateSerial("LAB-COMP", Date.Today, 121),
    .Reference = "OHS-18.1/2025/0121",
    .IssueDate = Date.Today,
    .Status = CaseStatus.Approved,
    .Notes = "Competent for LV/MV isolation and lockout per SOP-OHS-12.",
    .Signatory = "Safety Manager",
    .SignatoryTitle = "OHS 16(2) Appointee",
    .VerificationURL = "https://verify.nimet.co.za (https://verify.nimet.co.za)"
}
Dim hdrLab = New Dictionary(Of String, String) From {
    {"Scope", "Isolation, lockout, testing, permit-to-work"},
    {"Plant", "MV Switchgear, MCCs, VSDs"},
    {"Validity", "24 months (6-month refresher)"}
}
Dim itemsLab = New List(Of LineItem) From {
    New LineItem With {.Code = "PTW-01", .Title = "Permit-to-Work Competency", .DateEntry =
#2025-09-01#, .Score = "Comp", .Result = "Pass"},
    New LineItem With {.Code = "LOTO-02", .Title = "Lockout/Tagout Practical", .DateEntry =
#2025-09-01#, .Score = "Comp", .Result = "Pass"}
}
Dim docLab = Renderers.RenderDocument(org, person, metaLab, hdrLab, itemsLab)
Console.WriteLine (docLab)

' PSIRA management certificate
Dim metaPsira As New CaseMeta With {
    .Type = DocType.PSIRA_Management_Certificate,
    .Serial = Governance.GenerateSerial("PSIRA-MGMT", Date.Today, 77),
    .Reference = "PSIRA-REG-EMP-001122",
    .IssueDate = Date.Today,
    .Status = CaseStatus.Approved,
    .Notes = "Compliant with PSIRA Grade A management requirements.",
    .Signatory = "Security Compliance Officer",
    .SignatoryTitle = "PSIRA Liaison"
}
Dim hdrPsira = New Dictionary(Of String, String) From {
    {"PSIRA Grade", "A"},
    {"Category", "Security Management"},
    {"Validity", "2025-09-01 to 2026-08-31"}
}
Console.WriteLine()
Console.WriteLine (Renderers.RenderDocument(org, person, metaPsira, hdrPsira))

' SAPS firearm competency summary
Dim metaFire As New CaseMeta With {
    .Type = DocType.SAPS_Firearm_Competency_Summary,
    .Serial = Governance.GenerateSerial("SAPS-FC", Date.Today, 233),
    .Reference = "CFR-APP-45/2025",
```

```
        .IssueDate = Date.Today,
        .Status = CaseStatus.Approved,
        .Notes = "Competent for handgun, rifle (business purposes).",
        .Signatory = "Firearm Compliance Officer",
        .SignatoryTitle = "Designated Official"
    }
    Dim hdrFire = New Dictionary(Of String, String) From {
        {"Proficiency", "US 119649–119651 (SASSETA)"},
        {"Competency", "Handgun, Rifle (Business)"},
        {"CFR Status", "Approved — card pending"}
    }
    Console.WriteLine()
    Console.WriteLine (Renderers.RenderDocument(org, person, metaFire, hdrFire))

    ' Police clearance attestation
    Dim metaPCC As New CaseMeta With {
        .Type = DocType.Police_Clearance_Attestation,
        .Serial = Governance.GenerateSerial("PCC-ATT", Date.Today, 89),
        .Reference = "SAPS-CRC-2025/09/0089",
        .IssueDate = Date.Today,
        .Status = CaseStatus.Approved,
        .Notes = "No adverse record found (CRC feedback).",
        .Signatory = "Records Officer",
        .SignatoryTitle = "Compliance Registry"
    }
    Dim hdrPCC = New Dictionary(Of String, String) From {
        {"Submission", "2025-08-20"},
        {"CRC Ref", "CRC-09-2025-7765"},
        {"Result", "Clear"}
    }
    Console.WriteLine()
    Console.WriteLine (Renderers.RenderDocument(org, person, metaPCC, hdrPCC))

    ' High Court transcript certificate
    Dim metaHC As New CaseMeta With {
        .Type = DocType.HighCourt_Transcript_Certificate,
        .Serial = Governance.GenerateSerial("HC-TR-CERT", Date.Today, 41),
        .Reference = "2025/HC/GAUT/012345",
        .IssueDate = Date.Today,
        .Status = CaseStatus.Awarded,
        .Signatory = "Court Liaison",
        .SignatoryTitle = "Registrar Interface"
    }
    Dim itemsHC = New List(Of LineItem) From {
        New LineItem With {.Code = "VOL1", .Title = "Proceedings Volume 1", .DateEntry = #2025-07-01#,
.Score = "178p", .Result = "Filed"},
        New LineItem With {.Code = "JDG", .Title = "Judgment", .DateEntry = #2025-08-15#, .Score = "15p",
.Result = "Delivered"}
    }
    Console.WriteLine()
    Console.WriteLine (Renderers.RenderDocument(org, person, metaHC, Nothing, itemsHC))
```

```
' Labour Court transcript certificate
Dim metaLC As New CaseMeta With {
    .Type = DocType.LabourCourt_Transcript_Certificate,
    .Serial = Governance.GenerateSerial("LC-TR-CERT", Date.Today, 22),
    .Reference = "JR 1234/25",
    .IssueDate = Date.Today,
    .Status = CaseStatus.Awarded,
    .Signatory = "Court Liaison",
    .SignatoryTitle = "Registrar Interface"
}
Console.WriteLine()
Console.WriteLine (Renderers.RenderDocument(org, person, metaLC, Nothing, itemsHC))

' CCMA award certificate
Dim metaCCMA As New CaseMeta With {
    .Type = DocType.CCMA_Award_Certificate,
    .Serial = Governance.GenerateSerial("CCMA-AWD", Date.Today, 311),
    .Reference = "GAJB 12345-25",
    .IssueDate = Date.Today,
    .Status = CaseStatus.Awarded,
    .Notes = "Reinstatement with back pay (30 days).",
    .Signatory = "Commissioner",
    .SignatoryTitle = "CCMA"
}
Dim itemsCCMA = New List(Of LineItem) From {
    New LineItem With {.Code = "AWD", .Title = "Award Outcome", .DateEntry = #2025-09-10#, .Score =
"R 45,000", .Result = "Granted"},
    New LineItem With {.Code = "CST", .Title = "Costs", .DateEntry = #2025-09-10#, .Score = "Each own",
.Result = "Set"}
}
Console.WriteLine()
Console.WriteLine (Renderers.RenderDocument(org, person, metaCCMA, Nothing, itemsCCMA))

' HR outcome letter
Dim metaHR As New CaseMeta With {
    .Type = DocType.HR_Outcome_Letter,
    .Serial = Governance.GenerateSerial("HR-OUT", Date.Today, 141),
    .Reference = "HR-DISC-2025/09/0141",
    .IssueDate = Date.Today,
    .Status = CaseStatus.Resolved,
    .Notes = "Final written warning; performance plan agreed.",
    .Signatory = "HR Director",
    .SignatoryTitle = "Human Resources"
}
Dim hdrHR = New Dictionary(Of String, String) From {
    {"Effective", "2025-09-30"},
    {"Review", "2026-03-31"},
    {"Appeal Window", "5 working days"}
}
Console.WriteLine()
```

```
        Console.WriteLine (Renderers.RenderDocument(org, person, metaHR, hdrHR))
        ' SAQA Statement of Results
        Dim metaSoR As New CaseMeta With {
            .Type = DocType.SAQA_Statement_Of_Results,
            .Serial = Governance.GenerateSerial("SAQA-SOR", Date.Today, 219),
            .Reference = "NLRD-REF-2025/09/0219",
            .IssueDate = Date.Today,
            .Status = CaseStatus.Approved,
            .Signatory = "Registry Officer",
            .SignatoryTitle = "Learning Records"
        }
        Dim itemsSoR = New List(Of LineItem) From {
            New LineItem With {.Code = "244288", .Title = "Apply SHE principles (Engineering Safety)",
.DateEntry = #2025-06-30#, .Credits = 10, .Score = "Comp", .Result = "Pass"},
            New LineItem With {.Code = "119649", .Title = "Demonstrate knowledge of FCA", .DateEntry =
#2025-05-20#, .Credits = 3, .Score = "Comp", .Result = "Pass"}
        }
        Console.WriteLine()
        Console.WriteLine (Renderers.RenderDocument(org, person, metaSoR, Nothing, itemsSoR))

        ' Audit trail
        Dim audit = New AuditLogger("justice_security_audit.log")
        audit.Append("system", "Issued", "LabourCompetence", metaLab.Serial, metaLab.Reference)
        audit.Append("system", "Issued", "PSIRA", metaPsira.Serial, metaPsira.Reference)
        audit.Append("system", "Issued", "SAPSFirearm", metaFire.Serial, metaFire.Reference)
        audit.Append("system", "Issued", "PCC", metaPCC.Serial, metaPCC.Reference)
        audit.Append("system", "Issued", "HighCourtTranscript", metaHC.Serial, metaHC.Reference)
        audit.Append("system", "Issued", "LabourCourtTranscript", metaLC.Serial, metaLC.Reference)
        audit.Append("system", "Issued", "CCMAAward", metaCCMA.Serial, metaCCMA.Reference)
        audit.Append("system", "Issued", "HROutcome", metaHR.Serial, metaHR.Reference)
        audit.Append("system", "Issued", "SAQA-SoR", metaSoR.Serial, metaSoR.Reference)
    End Sub
End Module
```

Final tweaks I can wire in next

Serial formats: Define your fixed prefixes per authority for uniform numbering.

Branding: Add logos/seals in PrintPage for court/PSIRA/CCMA variants.

Data bindings: Pull items from your case/HR/learning database; keep phrasing in external templates for easy policy updates.

Reissue controls: Lock reasons (lost, damaged, correction, name change) and require prior-serial reference.

If you share your exact authority wording blocks and serial patterns, I'll deliver a WinForms UI with dropdowns (document type, authority fields), print preview, and PDF export.

Imports System.text

```
Imports System.Drawing.Printing
Imports System.Security.Cryptography

Public Enum AuthorityType
    DTIC              ' Department of Trade, Industry and Competition
    dst               ' Department of Science and Technology (DSI)
    dmre              ' Department of Mineral Resources and Energy
    Eskom_Training
    CityPower_Subcontractor_Training
    Eaton_Power
    Schneider_Electric
    Microsoft
End Enum

Public Enum DocKind
    Training_Certificate
    Compliance_Certificate
    Participation_Certificate
    Partner_Certificate
    Accreditation_Notice
    Statement_Of_Completion
    Reissue_Notice
End Enum

Public Class Organisation
    Public Property Name As String
    Public Property Address As String
    Public Property Contact As String
    Public Property References As String   ' Accreditations, Vendor IDs, Partner IDs
End Class

Public Class Person
    Public Property FullName As String
    Public Property IDNumber As String
    Public Property Email As String
    Public Property EmployeeNo As String
End Class

Public Class ProgrammeRef
    Public Property Title As String
    Public Property Track As String         ' Stream/track e.g., Safety, Grid Ops, Azure Admin
    Public Property Level As String         ' Foundation/Intermediate/Advanced
    Public Property Hours As Integer
    Public Property Credits As Integer      ' Optional
End Class

Public Class CertificateMeta
    Public Property Authority As AuthorityType
    Public Property Kind As DocKind
    Public Property Serial As String
    Public Property IssueDate As Date
```

```
    Public Property ValidTo As Date?
    Public Property SignatoryName As String
    Public Property SignatoryTitle As String
    Public Property VerificationURL As String
    ' Reissue
    Public Property IsReissue As Boolean
    Public Property ReplacesSerial As String
    Public Property ReissueReason As String
End Class

Public Class LineItem
    Public Property Code As String
    Public Property Title As String
    Public Property DateEntry As Date
    Public Property Score As String        ' %, Comp, Pass, Badge, etc.
    Public Property Result As String       ' Pass/Fail/Comp/Badge/Partner
    Public Property Credits As Integer
End Class

RenderersPublic Module Renderers

    Public Function RenderCertificate(issuer As Organisation,
                                      person As Person,
                                      prog As ProgrammeRef,
                                      meta As CertificateMeta,
                                      Optional items As IEnumerable(Of LineItem) = Nothing,
                                      Optional extra As Dictionary(Of String, String) = Nothing) As String
        Dim sb As New StringBuilder()

        ' Header
        sb.AppendLine (HeaderTitle(meta.authority))
        sb.AppendLine (issuer.Name)
        If Not String.IsNullOrWhiteSpace(issuer.Address) Then sb.AppendLine(issuer.Address)
        If Not String.IsNullOrWhiteSpace(issuer.Contact) Then sb.AppendLine(issuer.Contact)
        If Not String.IsNullOrWhiteSpace(issuer.References) Then sb.AppendLine(issuer.References)
        sb.AppendLine(New String("-"c, 90))

        ' Document title
        sb.AppendLine (TitleFor(meta.authority, meta.kind))
        sb.AppendLine(New String("-"c, 90))

        ' Person and programme
        sb.AppendLine($"Candidate: {person.FullName}   ID: {person.IDNumber}   Emp#:
{person.EmployeeNo}")
        sb.AppendLine($"Email: {person.Email}")
        If prog IsNot Nothing Then
            Dim pl = $"Programme: {prog.Title}"
            If Not String.IsNullOrWhiteSpace(prog.Track) Then pl &= $"  |  Track: {prog.Track}"
            If Not String.IsNullOrWhiteSpace(prog.Level) Then pl &= $"  |  Level: {prog.Level}"
            If prog.Hours > 0 Then pl &= $"  |  Hours: {prog.Hours}"
            If prog.Credits > 0 Then pl &= $"  |  Credits: {prog.Credits}"
```

```
            sb.AppendLine (pl)
        End If

        ' Extras
        If extra IsNot Nothing AndAlso extra.Count > 0 Then
            sb.AppendLine(New String("-"c, 90))
            For Each kv In extra
                sb.AppendLine($" - {kv.Key}: {kv.Value}")
            Next
        End If

        ' Items/table
        If items IsNot Nothing AndAlso items.Any() Then
            sb.AppendLine(New String("-"c, 90))
            sb.AppendLine ("Code       Component/Module                              Date       Credits  Score   Result")
            sb.AppendLine ("---------  --------------------------------------------  ---------- -------  ------  --------")
            For Each it In items.OrderBy(Function(x) x.DateEntry).ThenBy(Function(x) x.Code)
                sb.AppendLine($"{it.Code.PadRight(10)} {it.Title.PadRight(49)} {it.DateEntry:yyyy-MM-dd}
{it.Credits.ToString().PadLeft(7)}  {it.Score.PadLeft(6)}  {it.Result}")
            Next
        End If

        ' Footer and verification
        sb.AppendLine(New String("-"c, 90))
        Dim valTo = If(meta.ValidTo.HasValue, $"  Valid to: {meta.ValidTo.Value:yyyy-MM-dd}", "")
        sb.AppendLine($"Serial: {meta.Serial}  Issued: {meta.IssueDate:yyyy-MM-dd}{valTo}")
        sb.AppendLine($"Signed: {meta.Signatory}, {meta.SignatoryTitle}")
        If meta.IsReissue Then sb.AppendLine($"Re-issue of: {meta.ReplacesSerial}  Reason:
{meta.ReissueReason}")
        sb.AppendLine (FooterFor(meta.authority, meta.kind))
        AppendVerification(sb, person, meta)

        Return sb.ToString()
    End Function


    Select Case a
        Case AuthorityType.DTIC : Return "Department of Trade, Industry and Competition (the dtic)"
        Case AuthorityType.DST : Return "Department of Science and Innovation (DSI/DST)"
        Case AuthorityType.DMRE : Return "Department of Mineral Resources and Energy (DMRE)"
        Case AuthorityType.Eskom_Training : Return "Eskom Training and Development"
        Case AuthorityType.CityPower_Subcontractor_Training : Return "City Power Johannesburg —
Subcontractor Training"
        Case AuthorityType.Eaton_Power : Return "Eaton Power Quality and Energy Management"
        Case AuthorityType.Schneider_Electric : Return "Schneider Electric Training Services"
        Case AuthorityType.Microsoft : Return "Microsoft Learning and Certifications"
        Case Else : Return "Authority"
    End Select
End Function
```

```
    Dim baseTitle As String
    Select Case k
      Case DocKind.Training_Certificate: baseTitle = "TRAINING CERTIFICATE"
      Case DocKind.Compliance_Certificate: baseTitle = "COMPLIANCE CERTIFICATE"
      Case DocKind.Participation_Certificate: baseTitle = "CERTIFICATE OF PARTICIPATION"
      Case DocKind.Partner_Certificate: baseTitle = "PARTNER/ASSOCIATE CERTIFICATE"
      Case DocKind.Accreditation_Notice: baseTitle = "ACCREDITATION NOTICE"
      Case DocKind.Statement_Of_Completion: baseTitle = "STATEMENT OF COMPLETION"
      Case DocKind.Reissue_Notice: baseTitle = "RE-ISSUE NOTICE"
      Case Else: baseTitle = "CERTIFICATE"
    End Select
    Return $"{baseTitle} — {a}"
  End Function


    Select Case a
      Case AuthorityType.DTIC
        Return "Issued for programme/compliance acknowledgement. Official records remain with the
dtic."
      Case AuthorityType.dst
        Return "Issued for science and innovation programme recognition."
      Case AuthorityType.dmre
        Return "Subject to DMRE regulations and verification where applicable."
      Case AuthorityType.Eskom_Training
        Return "Training recognition for internal/contractor compliance; site authorisations remain
separate."
      Case AuthorityType.CityPower_Subcontractor_Training
        Return "Subcontractor training certificate; work permits and site access are governed by City
Power policies."
      Case AuthorityType.Eaton_Power, AuthorityType.Schneider_Electric
        Return "OEM training/partner recognition; product warranties and authorisations per OEM policy."
      Case AuthorityType.Microsoft
        Return "Training/achievement record; official Microsoft Certifications are issued via the credential
platform."
      Case Else
        Return "Subject to verification with the issuing authority."
    End Select
  End Function


    Dim payload = $"serial={m.Serial}|id={p.IDNumber}|date={m.IssueDate:yyyy-MM-dd}|
auth={m.Authority}|kind={m.Kind}"
    Dim hash = Sha256Hex(payload)
    sb.AppendLine($"Verification hash: {hash}")
    If Not String.IsNullOrWhiteSpace(m.VerificationURL) Then
      sb.AppendLine($"Verify at: {m.VerificationURL}?serial={m.Serial}")
      sb.AppendLine($"QR payload: {payload}")
    End If
  End Sub
```

```
    Using sha = SHA256.Create()
      Dim b = sha.ComputeHash(Encoding.UTF8.GetBytes(s))
      Return BitConverter.ToString(b).Replace("-", "").ToLowerInvariant()
    End Using
  End Function

End ModulePublic Class TextPrintJob
  Private ReadOnly _content As String
  Private _lines() As String
  Private _i As Integer

  Public Sub New(content As String)
    _content = content
    _lines = _content.Replace(vbCrLf, vbLf).Split(ControlChars.Lf)
  End Sub

  Public Sub Print(docName As String)
    Dim pd As New PrintDocument()
    pd.DocumentName = docName
    AddHandler pd.PrintPage, AddressOf OnPrintPage
    pd.Print()
  End Sub



    Dim font = New Font("Consolas", 9.0F)
    Dim lh = font.GetHeight(e.Graphics)
    Dim y = e.MarginBounds.Top
    Dim left = e.MarginBounds.Left
    Dim perPage = CInt(Math.Floor(e.MarginBounds.Height / lh))
    Dim count As Integer = 0
    While count < perPage AndAlso _i < _lines.Length
      e.Graphics.DrawString(_lines(_i), font, Brushes.Black, left, y)
      y += lh : count += 1 : _i += 1
    End While
    e.HasMorePages = (_i < _lines.Length)
  End Sub
End Class

Public Module SerialRules


    Dim ap = authority.ToString().Replace("_", "-")
    Dim kp = kind.ToString().Replace("_", "-")
    Return $"{ap}-{kp}-{issueDate:yyyy}-{sequence:000000}"
  End Function
End Module

Example usage with each authorityModule Demo

    Dim candidate = New Person With {
      .FullName = "Tshingombe Tshitadi Fiston",
      .IDNumber = "9001015800082",
```

```
        .Email = "tshingombe@example.org (mailto:tshingombe@example.org)",
        .Em[illegible]42"
    }

    ' dtic compliance/participation
    Dim dtic = New Organisation With {
        .Name = "the dtic",
        .Address = "77 Meintjies Street, Sunnyside, Pretoria",
        .Contact = "Tel: 012 394 0000 | www.thedtic.gov.za (http://www.thedtic.gov.za)",
        .References = "Bid/Programme Ref: DTIC-IND-LOCAL-2025"
    }
    Dim dticMeta = New CertificateMeta With {
        .Authority = AuthorityType.DTIC,
        .Kind = DocKind.Compliance_Certificate,
        .Serial = SerialRules.NextSerial(AuthorityType.DTIC, DocKind.Compliance_Certificate, Date.Today,
73),
        .IssueDate = Date.Today,
        .Signatory = "Director: Industrial Development",
        .SignatoryTitle = "the dtic",
        .VerificationURL = "https://verify.example.org (https://verify.example.org)"
    }
    Dim dticDoc = Renderers.RenderCertificate(dtic, candidate,
        New ProgrammeRef With {.Title = "Local Content and Designation Workshop", .Track =
"Electrotechnical", .Level = "Advanced", .Hours = 8},
        dticMeta,
        extra:=New Dictionary(Of String, String) From {{"Local Content Threshold", "90% (info cable
designation)"}, {"Venue", "Pretoria"}})
    Console.WriteLine (dticDoc)

    ' dst/dsi participation
    Dim dst = New Organisation With {.Name = "Department of Science and Innovation (DSI)", .Address =
"Pretoria", .Contact = "www.dst.gov.za (http://www.dst.gov.za)"}
    Dim dstMeta = New CertificateMeta With {
        .Authority = AuthorityType.DST, .Kind = DocKind.Participation_Certificate,
        .Serial = SerialRules.NextSerial(AuthorityType.DST, DocKind.Participation_Certificate, Date.Today,
12),
        .IssueDate = Date.Today, .Signatory = "Programme Manager", .SignatoryTitle = "DSI"
    }
    Console.WriteLine()
    Console.WriteLine(Renderers.RenderCertificate(dst, candidate,
        New ProgrammeRef With {.Title = "Research and Innovation Policy Colloquium", .Track = "Energy
Systems"},
        dstMeta))

    ' dmre compliance
    Dim dmre = New Organisation With {.Name = "DMRE", .Address = "Pretoria", .Contact =
"www.dmre.gov.za (http://www.dmre.gov.za)", .References = "Regulatory Ref: ELEC-OPS-2025"}
    Dim dmreMeta = New CertificateMeta With {
        .Authority = AuthorityType.DMRE, .Kind = DocKind.Compliance_Certificate,
        .Serial = SerialRules.NextSerial(AuthorityType.DMRE, DocKind.Compliance_Certificate, Date.Today,
31),
```

```
        .IssueDate = Date.Today, .ValidTo = Date.Today.AddYears(1),
        .Signatory = "Chief Inspector", .SignatoryTitle = "Electrical Safety"
    }
    Console.WriteLine()
    Console.WriteLine(Renderers.RenderCertificate(dmre, candidate,
        New ProgrammeRef With {.Title = "Electrical Safety Regulatory Briefing", .Track = "Compliance"},
        dmreMeta, extra:=New Dictionary(Of String, String) From {{"Scope", "OHS Act & Grid Code
alignment"}}))

    ' eskom training
    Dim eskom = New Organisation With {.Name = "Eskom Academy of Learning", .Address = "Gauteng",
.Contact = "www.eskom.co.za (http://www.eskom.co.za)", .References = "Vendor: VND-ESK-00123"}
    Dim eskMeta = New CertificateMeta With {
        .Authority = AuthorityType.Eskom_Training, .Kind = DocKind.Training_Certificate,
        .Serial = SerialRules.NextSerial(AuthorityType.Eskom_Training, DocKind.Training_Certificate,
Date.Today, 145),
        .IssueDate = Date.Today, .Signatory = "Training Manager", .SignatoryTitle = "EAL"
    }
    Dim eskItems = New List(Of LineItem) From {
        New LineItem With {.Code = "ESK-SC-01", .Title = "Substation Safety Induction", .DateEntry =
Date.Today, .Score = "Comp", .Result = "Pass"},
        New LineItem With {.Code = "ESK-LOTO", .Title = "Lockout/Tagout", .DateEntry = Date.Today, .Score =
"Comp", .Result = "Pass"}
    }
    Console.WriteLine()
    Console.WriteLine(Renderers.RenderCertificate(eskom, candidate,
        New ProgrammeRef With {.Title = "Grid Operations Safety", .Level = "Intermediate", .Hours = 12},
        eskMeta, eskItems, extra:=New Dictionary(Of String, String) From {{"Site", "Simmerpan"}}))

    ' city power subcontractor training
    Dim cityPower = New Organisation With {.Name = "City Power Johannesburg", .Address = "40
Heronmere Rd, Reuven", .Contact = "www.citypower.co.za (http://www.citypower.co.za)"}
    Dim cpMeta = New CertificateMeta With {
        .Authority = AuthorityType.CityPower_Subcontractor_Training, .Kind = DocKind.Training_Certificate,
        .Serial = SerialRules.NextSerial(AuthorityType.CityPower_Subcontractor_Training,
DocKind.Training_Certificate, Date.Today, 19),
        .IssueDate = Date.Today, .ValidTo = Date.Today.AddYears(1),
        .Signatory = "Learning & Development", .SignatoryTitle = "City Power"
    }
    Console.WriteLine()
    Console.WriteLine(Renderers.RenderCertificate(cityPower, candidate,
        New ProgrammeRef With {.Title = "Subcontractor Induction", .Track = "Electrical Safety", .Hours = 6},
        cpMeta, extra:=New Dictionary(Of String, String) From {{"Badge ID", "CP-IND-2025-0042"}, {"Access",
"Depot/Live sites"}}))

    ' eaton power certificate
    Dim eaton = New Organisation With {.Name = "Eaton", .Address = "Power Quality Division", .Contact =
"www.eaton.com (http://www.eaton.com)", .References = "Partner ID: EAT-PT-7788"}
    Dim eatMeta = New CertificateMeta With {
        .Authority = AuthorityType.Eaton_Power, .Kind = DocKind.Training_Certificate,
        .Serial = SerialRules.NextSerial(AuthorityType.Eaton_Power, DocKind.Training_Certificate,
```

```
Date.Today, 7)
        .IssueDate = Date.Today, .Signatory = "Training Director", .SignatoryTitle = "Eaton Academy"
    }
    Dim eatItems = New List(Of LineItem) From {
        New LineItem With {.Code = "UPS-PRV", .Title = "UPS Installation & Commissioning", .DateEntry =
Date.Today, .Score = "92%", .Result = "Pass", .Credits = 0}
    }
    Console.WriteLine()
    Console.WriteLine(Renderers.RenderCertificate(eaton, candidate,
        New ProgrammeRef With {.Title = "Power Quality Professional", .Level = "Advanced", .Hours = 10},
        eatMeta, eatItems))

    ' schneider electric
    Dim sch = New Organisation With {.Name = "Schneider Electric", .Address = "Training Services",
.Contact = "www.se.com (http://www.se.com)", .References = "EcoXpert Candidate"}
    Dim schMeta = New CertificateMeta With {
        .Authority = AuthorityType.Schneider_Electric, .Kind = DocKind.Training_Certificate,
        .Serial = SerialRules.NextSerial(AuthorityType.Schneider_Electric, DocKind.Training_Certificate,
Date.Today, 28),
        .IssueDate = Date.Today, .Signatory = "Academy Lead", .SignatoryTitle = "Schneider Electric
University"
    }
    Dim schItems = New List(Of LineItem) From {
        New LineItem With {.Code = "SE-LV-01", .Title = "LV Switchgear Commissioning", .DateEntry =
Date.Today, .Score = "Comp", .Result = "Pass"}
    }
    Console.WriteLine()
    Console.WriteLine(Renderers.RenderCertificate(sch, candidate,
        New ProgrammeRef With {.Title = "EcoStruxure Power Commission", .Level = "Intermediate", .Hours
= 8},
        schMeta, schItems))

    ' microsoft training/partner
    Dim ms = New Organisation With {.Name = "Microsoft", .Address = "Microsoft Learning", .Contact =
"learn.microsoft.com (http://learn.microsoft.com)", .References = "Org ID: MS-ORG-4421"}
    Dim msMeta = New CertificateMeta With {
        .Authority = AuthorityType.Microsoft, .Kind = DocKind.Statement_Of_Completion,
        .Serial = SerialRules.NextSerial(AuthorityType.Microsoft, DocKind.Statement_Of_Completion,
Date.Today, 64),
        .IssueDate = Date.Today, .Signatory = "Learning Admin", .SignatoryTitle = "Microsoft Learning"
    }
    Dim msItems = New List(Of LineItem) From {
        New LineItem With {.Code = "AZ-104", .Title = "Azure Administrator (course completion)", .DateEntry
= Date.Today, .Score = "Completed", .Result = "Badge"}
    }
    Console.WriteLine()
    Console.WriteLine(Renderers.RenderCertificate(ms, candidate,
        New ProgrammeRef With {.Title = "Azure Administrator Associate", .Track = "Cloud", .Hours = 24},
        msMeta, msItems, extra:=New Dictionary(Of String, String) From {{"Exam", "AZ-104 (separate
certification)"}}))
  End Sub
```

End Module

What to configure next

Serial patterns per authority (e.g., DTIC-COMP-YYYY-######, ESK-TRN-YYYY-######).

Authority wording blocks and signatory roles you want printed.

Programme catalogs per issuer (Eskom safety tracks, City Power induction modules, OEM course codes, Microsoft course/exam mappings).

Branding assets: logos and seals to draw in PrintPage for each authority variant.

Verification endpoint: host the serial lookup and return status for the hash payload.

If you share your serial formats and preferred text blocks per autho

```
End Sub
Public Enum AuthorityType
    CSD_CentralSupplierDatabase
    SciBono_DiscoveryCentre
    Engineering_CareerDiscovery
    ICT_Libraries
    SME_Mentoring
    Research_Publication
End Enum

Public Enum DocKind
    Supplier_Registration_Certificate
    Supplier_Validation_Summary
    Facilitator_Assessment_Certificate
    Participation_Certificate
    Skills_Statement
    Technical_Documentation_Endorsement
    Mentoring_Certificate
    Research_Publication_Certificate
    Portfolio_Summary
    Reissue_Notice
End Enum

Public Class Organisation
    Public Property Name As String
    Public Property Address As String
    Public Property Contact As String
    Public Property References As String      ' CSD No., Vendor IDs, Repository IDs, etc.
End Class

Public Class Person
    Public Property FullName As String
    Public Property IDNumber As String
    Public Property Email As String
```

```
    Public Property Phone As String
    Public Property Email As String   ' (https://www.elektormagazine.com/)
End Class

Public Class ProgrammeRef
    Public Property Title As String
    Public Property Track As String        ' Engineering Science, Physical Skills, etc.
    Public Property Level As String
    Public Property Hours As Integer
    Public Property Credits As Integer
End Class

Public Class CertificateMeta
    Public Property Authority As AuthorityType
    Public Property Kind As DocKind
    Public Property Serial As String
    Public Property IssueDate As Date
    Public Property ValidTo As Date?
    Public Property Signatory As String
    Public Property SignatoryTitle As String
    Public Property VerificationURL As String
    Public Property IsReissue As Boolean
    Public Property ReplacesSerial As String
    Public Property ReissueReason As String
End Class

Public Class LineItem
    Public Property Code As String
    Public Property Title As String
    Public Property DateEntry As Date
    Public Property Score As String        ' %, Comp, DOI, Status, Pages, Hours
    Public Property Result As String       ' Valid/Pass/Comp/Accepted/Indexed
    Public Property Credits As Integer
End Class

Renderers and printingPublic Module Certificates

    Public Function Render(issuer As Organisation,
                           person As Person,
                           prog As ProgrammeRef,
                           meta As CertificateMeta,
                           Optional items As IEnumerable(Of LineItem) = Nothing,
                           Optional extra As Dictionary(Of String, String) = Nothing) As String
        Dim sb As New StringBuilder()

        ' Header
        sb.AppendLine (HeaderTitle(meta.authority))
        sb.AppendLine (issuer.Name)
        If Not String.IsNullOrWhiteSpace(issuer.Address) Then sb.AppendLine(issuer.Address)
        If Not String.IsNullOrWhiteSpace(issuer.Contact) Then sb.AppendLine(issuer.Contact)
        If Not String.IsNullOrWhiteSpace(issuer.References) Then sb.AppendLine(issuer.References)
```

```
        sb.AppendLine(New String("-"c, 92))
        ' Title
        sb.AppendLine (TitleFor(meta.authority, meta.kind))
        sb.AppendLine(New String("-"c, 92))

        ' Person + programme
        sb.AppendLine($"Candidate: {person.FullName}   ID: {person.IDNumber}   Profile: {person.ProfileId}")
        sb.AppendLine($"Email: {person.Email} | {person.Phone}")
        If prog IsNot Nothing Then
            Dim ln = $"Programme: {prog.Title}"
            If Not String.IsNullOrWhiteSpace(prog.Track) Then ln &= $"  |  Track: {prog.Track}"
            If Not String.IsNullOrWhiteSpace(prog.Level) Then ln &= $"  |  Level: {prog.Level}"
            If prog.Hours > 0 Then ln &= $"  |  Hours: {prog.Hours}"
            If prog.Credits > 0 Then ln &= $"  |  Credits: {prog.Credits}"
            sb.AppendLine (ln)
        End If

        ' Extra fields
        If extra IsNot Nothing AndAlso extra.Count > 0 Then
            sb.AppendLine(New String("-"c, 92))
            For Each kv In extra
                sb.AppendLine($" - {kv.Key}: {kv.Value}")
            Next
        End If

        ' Items table
        If items IsNot Nothing AndAlso items.Any() Then
            sb.AppendLine(New String("-"c, 92))
            sb.AppendLine ("Code       Component/Module/Record                    Date       Credits  Score
 Result")
            sb.AppendLine ("---------  ----------------------------------------  ----------  -------  --------  --------")
            For Each it In items.OrderBy(Function(x) x.DateEntry).ThenBy(Function(x) x.Code)
                sb.AppendLine($"{it.Code.PadRight(10)} {it.Title.PadRight(49)} {it.DateEntry:yyyy-MM-dd}
{it.Credits.ToString().PadLeft(7)}  {it.Score.PadLeft(8)}  {it.Result}")
            Next
        End If

        ' Footer & verification
        sb.AppendLine(New String("-"c, 92))
        Dim valTo = If(meta.ValidTo.HasValue, $"  Valid to: {meta.ValidTo.Value:yyyy-MM-dd}", "")
        sb.AppendLine($"Serial: {meta.Serial}  Issued: {meta.IssueDate:yyyy-MM-dd}{valTo}")
        sb.AppendLine($"Signed: {meta.Signatory}, {meta.SignatoryTitle}")
        If meta.IsReissue Then sb.AppendLine($"Re-issue of: {meta.ReplacesSerial}  Reason:
{meta.ReissueReason}")
        sb.AppendLine (FooterFor(meta.authority))
        AppendVerification(sb, person, meta)

        Return sb.ToString()
    End Function
```

```
    Select Case a
      Case AuthorityType.CSD_CentralSupplierDatabase : Return "National Treasury — Central Supplier
Database (CSD)"
      Case AuthorityType.SciBono_DiscoveryCentre : Return "Sci-Bono Discovery Centre"
      Case AuthorityType.Engineering_CareerDiscovery : Return "Engineering Career Discovery"
      Case AuthorityType.ICT_Libraries : Return "ICT Libraries & Technical Documentation"
      Case AuthorityType.SME_Mentoring : Return "Subject-Matter Expert Mentoring"
      Case AuthorityType.Research_Publication : Return "Research Publication Record"
      Case Else : Return "Authority"
    End Select
  End Function


    Dim baseTitle As String
    Select Case k
      Case DocKind.Supplier_Registration_Certificate: baseTitle = "SUPPLIER REGISTRATION
CERTIFICATE"
      Case DocKind.Supplier_Validation_Summary: baseTitle = "SUPPLIER VALIDATION SUMMARY"
      Case DocKind.Facilitator_Assessment_Certificate: baseTitle = "FACILITATOR ASSESSMENT
CERTIFICATE"
      Case DocKind.Participation_Certificate: baseTitle = "CERTIFICATE OF PARTICIPATION"
      Case DocKind.Skills_Statement: baseTitle = "SKILLS STATEMENT"
      Case DocKind.Technical_Documentation_Endorsement: baseTitle = "TECHNICAL DOCUMENTATION
ENDORSEMENT"
      Case DocKind.Mentoring_Certificate: baseTitle = "MENTORING CERTIFICATE"
      Case DocKind.Research_Publication_Certificate: baseTitle = "RESEARCH PUBLICATION
CERTIFICATE"
      Case DocKind.Portfolio_Summary: baseTitle = "PORTFOLIO SUMMARY"
      Case DocKind.Reissue_Notice: baseTitle = "RE-ISSUE NOTICE"
      Case Else: baseTitle = "CERTIFICATE"
    End Select
    Return $"{baseTitle} — {a}"
  End Function


    Select Case a
      Case AuthorityType.CSD_CentralSupplierDatabase
        Return "Supplier status subject to CSD verification and tax compliance."
      Case AuthorityType.SciBono_DiscoveryCentre
        Return "Facilitation outcomes recorded; institutional verification applies."
      Case AuthorityType.Engineering_CareerDiscovery
        Return "Career discovery achievements reflect assessed competence and exposure."
      Case AuthorityType.ICT_Libraries
        Return "Endorsement acknowledges technical documentation quality and accessibility."
      Case AuthorityType.SME_Mentoring
        Return "Mentoring record reflects supervised practice hours and competencies."
      Case AuthorityType.Research_Publication
        Return "Publication status based on submitted evidence; verify via DOI/indexes."
      Case Else
        Return "Subject to verification with the issuing authority."
```

```
        End Select
    End Function

    Dim payload = $"serial={m.Serial}|id={p.IDNumber}|date={m.IssueDate:yyyy-MM-dd}|
auth={m.Authority}|kind={m.Kind}"
    Dim hash = Sha256Hex(payload)
    sb.AppendLine($"Verification hash: {hash}")
    If Not String.IsNullOrWhiteSpace(m.VerificationURL) Then
      sb.AppendLine($"Verify at: {m.VerificationURL}?serial={m.Serial}")
      sb.AppendLine($"QR payload: {payload}")
    End If
  End Sub


    Using sha = SHA256.Create()
      Dim b = sha.ComputeHash(Encoding.UTF8.GetBytes(s))
      Return BitConverter.ToString(b).Replace("-", "").ToLowerInvariant()
    End Using
  End Function

End Module

Public Class TextPrintJob
  Private ReadOnly _content As String
  Private _lines() As String
  Private _i As Integer
  Public Sub New(content As String)
    _content = content
    _lines = _content.Replace(vbCrLf, vbLf).Split(ControlChars.Lf)
  End Sub
  Public Sub Print(docName As String)
    Dim pd As New PrintDocument()
    pd.DocumentName = docName
    AddHandler pd.PrintPage, AddressOf OnPrintPage
    pd.Print()
  End Sub

    Dim font = New Font("Consolas", 9.0F)
    Dim lh = font.GetHeight(e.Graphics)
    Dim y = e.MarginBounds.Top
    Dim left = e.MarginBounds.Left
    Dim perPage = CInt(Math.Floor(e.MarginBounds.Height / lh))
    Dim count As Integer = 0
    While count < perPage AndAlso _i < _lines.Length
      e.Graphics.DrawString(_lines(_i), font, Brushes.Black, left, y)
      y += lh : count += 1 : _i += 1
    End While
    e.HasMorePages = (_i < _lines.Length)
  End Sub
End Class
```

```
    Dim ap = auth.ToString().Replace("_", "-")
    Dim kp = kind.ToString().Replace("_", "-")
    Return $"{ap}-{kp}-{issueDate:yyyy}-{sequence:000000}"
  End Function
End Module

Ready-to-run examplesModule Demo


      .FullName = "Tshingombe Tshitadi Fiston",
      .IDNumber = "9001015800082",
      .Email = "tshingombe@example.org (mailto:tshingombe@example.org)",
      .Phone = "+27 72 000 0000",
      .ProfileId = "CSD-SUP-204512"
    }

    ' 1) CSD supplier certificate
    Dim csd = New Organisation With {
      .Name = "National Treasury — CSD",
      .Address = "240 Madiba St, Pretoria",
      .Contact = "csd@treasury.gov.za (mailto:csd@treasury.gov.za) | www.csd.gov.za (http://
www.csd.gov.za)",
      .References = "CSD Supplier No.: CSD-SUP-204512"
    }
    Dim csdMeta = New CertificateMeta With {
      .Authority = AuthorityType.CSD_CentralSupplierDatabase,
      .Kind = DocKind.Supplier_Registration_Certificate,
      .Serial = SerialRules.NextSerial(AuthorityType.CSD_CentralSupplierDatabase,
DocKind.Supplier_Registration_Certificate, Date.Today, 45),
      .IssueDate = Date.Today,
      .ValidTo = Date.Today.AddYears(1),
      .Signatory = "Chief Director: SCM",
      .SignatoryTitle = "National Treasury",
      .VerificationURL = "https://verify.example.org (https://verify.example.org)"
    }
    Dim csdItems = New List(Of LineItem) From {
      New LineItem With {.Code = "TAX", .Title = "Tax Compliance Status (TCS)", .DateEntry = Date.Today,
.Score = "Compliant", .Result = "Valid"},
      New LineItem With {.Code = "BBBEE", .Title = "B-BBEE Status", .DateEntry = Date.Today, .Score =
"Level 2", .Result = "Valid"},
      New LineItem With {.Code = "BANK", .Title = "Bank Account Verification", .DateEntry = Date.Today,
.Score = "Confirmed", .Result = "Valid"}
    }
    Dim csdDoc = Certificates.Render(csd, person, New ProgrammeRef With {.Title = "Supplier
Registration"}, csdMeta, csdItems)
    Console.WriteLine (csdDoc)

    ' 2) Sci-Bono facilitator assessment (engineering career discovery)
```

```
    Dim sci = New Organisation With {.Name = "Sci-Bono Discovery Centre", .Address = "Newtown,
Johannesburg", .Contact = "www.sci-bono.co.za (https://www.elektormagazine.com)"}
    Dim sciMeta = New CertificateMeta With {
      .Authority = AuthorityType.SciBono_DiscoveryCentre,
      .Kind = DocKind.Facilitator_Assessment_Certificate,
      .Serial = SerialRules.NextSerial(AuthorityType.SciBono_DiscoveryCentre,
DocKind.Facilitator_Assessment_Certificate, Date.Today, 12),
      .IssueDate = Date.Today,
      .Signatory = "Programme Manager",
      .SignatoryTitle = "Sci-Bono"
    }
    Dim sciItems = New List(Of LineItem) From {
      New LineItem With {.Code = "ENG-SCI", .Title = "Engineering Science Facilitation", .DateEntry =
Date.Today, .Score = "Comp", .Result = "Pass"},
      New LineItem With {.Code = "PHYS-ENG", .Title = "Physical Engineering Skills Lab", .DateEntry =
Date.Today, .Score = "Comp", .Result = "Pass"}
    }
    Console.WriteLine()
    Console.WriteLine(Certificates.Render(sci, person,
      New ProgrammeRef With {.Title = "Career Discovery Facilitator", .Track = "Engineering", .Level =
"Advanced", .Hours = 12},
      sciMeta, sciItems, extra:=New Dictionary(Of String, String) From {{"Assessment", "Observed
facilitation + portfolio"}}))

    ' 3) Engineering career skills statement (modular)
    Dim eng = New Organisation With {.Name = "Engineering Career Discovery", .Address =
"Johannesburg", .Contact = "eng-discovery@example.org (mailto:eng-discovery@example.org)"}
    Dim engMeta = New CertificateMeta With {
      .Authority = AuthorityType.Engineering_CareerDiscovery,
      .Kind = DocKind.Skills_Statement,
      .Serial = SerialRules.NextSerial(AuthorityType.Engineering_CareerDiscovery,
DocKind.Skills_Statement, Date.Today, 8),
      .IssueDate = Date.Today,
      .Signatory = "Technical Lead",
      .SignatoryTitle = "Engineering Discovery"
    }
    Dim engItems = New List(Of LineItem) From {
      New LineItem With {.Code = "ES-101", .Title = "Vectors & Statics (Engineering Science)", .DateEntry =
Date.Today, .Credits = 0, .Score = "85%", .Result = "Pass"},
      New LineItem With {.Code = "PE-201", .Title = "Bench-fitting & Measurement (Physical Skill)",
.DateEntry = Date.Today, .Score = "Comp", .Result = "Pass"}
    }
    Console.WriteLine()
    Console.WriteLine(Certificates.Render(eng, person,
      New ProgrammeRef With {.Title = "Engineering Science & Physical Skills", .Level = "Intermediate",
.Hours = 16},
      engMeta, engItems))

    ' 4) ICT libraries technical documentation endorsement
    Dim ict = New Organisation With {.Name = "ICT Libraries", .Address = "Gauteng", .Contact = "ict-
libraries@example.org (mailto:ict-libraries@example.org)", .References = "Repository: TECHDOC-ARCH"}
```

```
    Dim ictMeta = New CertificateMeta With {
        .Authority = AuthorityType.ICT_Libraries,
        .Kind = DocKind.Technical_Documentation_Endorsement,
        .Serial = SerialRules.NextSerial(AuthorityType.ICT_Libraries,
DocKind.Technical_Documentation_Endorsement, Date.Today, 5),
        .IssueDate = Date.Today,
        .Signatory = "Chief Librarian",
        .SignatoryTitle = "Technical Documentation"
    }
    Dim ictItems = New List(Of LineItem) From {
        New LineItem With {.Code = "DOC-001", .Title = "Electrical Safety SOP (LOTO)", .DateEntry =
Date.Today, .Score = "Peer-reviewed", .Result = "Endorsed"},
        New LineItem With {.Code = "DOC-002", .Title = "Plant Maintenance Checklist", .DateEntry =
Date.Today, .Score = "QA Passed", .Result = "Endorsed"}
    }
    Console.WriteLine()
    Console.WriteLine(Certificates.Render(ict, person,
        New ProgrammeRef With {.Title = "Technical Documentation Set", .Track = "ICT & Engineering",
.Hours = 6},
        ictMeta, ictItems, extra:=New Dictionary(Of String, String) From {{"Access", "Repository with version
control"}}))

    ' 5) SME mentoring certificate
    Dim sme = New Organisation With {.Name = "Expert Mentoring Council", .Address = "Johannesburg",
.Contact = "mentoring@example.org (mailto:mentoring@example.org)"}
    Dim smeMeta = New CertificateMeta With {
        .Authority = AuthorityType.SME_Mentoring,
        .Kind = DocKind.Mentoring_Certificate,
        .Serial = SerialRules.NextSerial(AuthorityType.SME_Mentoring, DocKind.Mentoring_Certificate,
Date.Today, 21),
        .IssueDate = Date.Today,
        .Signatory = "Lead Mentor",
        .SignatoryTitle = "Subject-Matter Expert"
    }
    Dim smeItems = New List(Of LineItem) From {
        New LineItem With {.Code = "MENT-ENG", .Title = "Mentored: Engineering Science Facilitation",
.DateEntry = Date.Today, .Score = "12h", .Result = "Completed"},
        New LineItem With {.Code = "MENT-ICT", .Title = "Mentored: ICT Library Curation", .DateEntry =
Date.Today, .Score = "8h", .Result = "Completed"}
    }
    Console.WriteLine()
    Console.WriteLine(Certificates.Render(sme, person,
        New ProgrammeRef With {.Title = "SME Guided Practice", .Track = "Engineering & ICT", .Hours = 20},
        smeMeta, smeItems, extra:=New Dictionary(Of String, String) From {{"Outcome", "Ready for
independent facilitation"}}))

    ' 6) Research publication certificate
    Dim res = New Organisation With {.Name = "Research Registry", .Address = "South Africa", .Contact =
"registry@example.org (mailto:registry@example.org)"}
    Dim resMeta = New CertificateMeta With {
        .Authority = AuthorityType.Research_Publication,
```

```
        .Kind = DocKind.Research_Publication_Certificate,
        .Serial = Certificates.NextSerial(DocKind.Research_Publication_Certificate, Date.Today, 9),
        .IssueDate = Date.Today,
        .Signatory = "Editorial Board",
        .SignatoryTitle = "Publications"
      }
      Dim resItems = New List(Of LineItem) From {
        New LineItem With {.Code = "DOI:10.1234/abc123", .Title = "Audit-Ready Certificate Engines for Multi-Authority Compliance", .DateEntry = Date.Today, .Score = "Indexed", .Result = "Accepted"},
        New LineItem With {.Code = "ARX:2509.001", .Title = "Chain-of-Custody in Educational Records", .DateEntry = Date.Today, .Score = "Preprint", .Result = "Available"}
      }
      Console.WriteLine()
      Console.WriteLine(Certificates.Render(res, person,
        New ProgrammeRef With {.Title = "Research Portfolio", .Track = "Compliance & Education Tech"},
        resMeta, resItems, extra:=New Dictionary(Of String, String) From {{"ORCID", "0000-0002-XXXX-XXXX"}}))
    End Sub
End Module
Imports System.text
Imports System.Drawing.Printing
Imports System.Security.Cryptography

Public Enum AuthorityType
    sarb              ' South African Reserve Bank (engineering/electronic training/permits)
    sars              ' South African Revenue Service (graduate/learner tax/compliance attestations)
    Graduate_DataScience    ' Graduate info/data science programme
    Alison_LMS            ' Alison LMS diploma/certificate
    AUI_Postdoctoral       ' Postdoctoral award (AUI or similar institute)
    MetPolice_IP_License    ' Metropolitan Police IP license/cert
    MetPolice_Training      ' Metropolitan Police training completion
    Tableau_Trailblazer     ' Tableau Trailblazer badges/certs
End Enum

Public Enum DocKind
    Training_Certificate
    Compliance_Certificate
    Participation_Certificate
    Diploma_Certificate
    Award_Certificate
    Statement_Of_Completion
    License_Certificate
    Badge_Certificate
    Reissue_Notice
End Enum

Public Class Organisation
    Public Property Name As String
    Public Property Address As String
    Public Property Contact As String
```

```
    Public Property References As String     ' Vendor IDs, partner IDs, programme codes
End Class
```



```
Public Class Person
    Public Property FullName As String
    Public Property IDNumber As String
    Public Property Email As String
    Public Property ProfileId As String      ' Student/Vendor/Badge/License ID
End Class

Public Class ProgrammeRef
    Public Property Title As String
    Public Property Track As String          ' Stream e.g., Electronics, Data Science, IP Law
    Public Property Level As String          ' Intro/Intermediate/Advanced/Postdoctoral
    Public Property Hours As Integer
    Public Property Credits As Integer
End Class

Public Class CertificateMeta
    Public Property Authority As AuthorityType
    Public Property Kind As DocKind
    Public Property Serial As String
    Public Property IssueDate As Date
    Public Property ValidTo As Date?
    Public Property Signatory As String
    Public Property SignatoryTitle As String
    Public Property VerificationURL As String
    ' Reissue
    Public Property IsReissue As Boolean
    Public Property ReplacesSerial As String
    Public Property ReissueReason As String
End Class

' Line items (modules, badges, units, compliance checks)
Public Class LineItem
    Public Property Code As String
    Public Property Title As String
    Public Property DateEntry As Date
    Public Property Score As String          ' %, Comp, Badge, DOI, License No.
    Public Property Result As String         ' Pass/Comp/Awarded/Issued/Valid
    Public Property Credits As Integer
End Class

RenderersPublic Module Renderers

    Public Function RenderCertificate(issuer As Organisation,
                                      person As Person,
                                      prog As ProgrammeRef,
                                      meta As CertificateMeta,
                                      Optional items As IEnumerable(Of LineItem) = Nothing,
                                      Optional extra As Dictionary(Of String, String) = Nothing) As String
```

```
    Dim sb As New StringBuilder()
    ' Header
    sb.AppendLine (HeaderTitle(meta.authority))
    sb.AppendLine (issuer.Name)
    If Not String.IsNullOrWhiteSpace(issuer.Address) Then sb.AppendLine(issuer.Address)
    If Not String.IsNullOrWhiteSpace(issuer.Contact) Then sb.AppendLine(issuer.Contact)
    If Not String.IsNullOrWhiteSpace(issuer.References) Then sb.AppendLine(issuer.References)
    sb.AppendLine(New String("-"c, 94))

    ' Title
    sb.AppendLine (TitleFor(meta.authority, meta.kind))
    sb.AppendLine(New String("-"c, 94))

    ' Person + programme
    sb.AppendLine($"Candidate: {person.FullName}   ID: {person.IDNumber}   Profile: {person.ProfileId}")
    sb.AppendLine($"Email: {person.Email}")
    If prog IsNot Nothing Then
       Dim ln = $"Programme: {prog.Title}"
       If Not String.IsNullOrWhiteSpace(prog.Track) Then ln &= $"  |  Track: {prog.Track}"
       If Not String.IsNullOrWhiteSpace(prog.Level) Then ln &= $"  |  Level: {prog.Level}"
       If prog.Hours > 0 Then ln &= $"  |  Hours: {prog.Hours}"
       If prog.Credits > 0 Then ln &= $"  |  Credits: {prog.Credits}"
       sb.AppendLine (ln)
    End If

    ' Extra fields (key-value)
    If extra IsNot Nothing AndAlso extra.Count > 0 Then
       sb.AppendLine(New String("-"c, 94))
       For Each kvp In extra
          sb.AppendLine($" - {kvp.Key}: {kvp.Value}")
       Next
    End If

    ' Items table
    If items IsNot Nothing AndAlso items.Any() Then
       sb.AppendLine(New String("-"c, 94))
       sb.AppendLine ("Code       Component/Module/Record                    Date       Credits  Score
Result")
       sb.AppendLine ("---------  ----------------------------------------  ----------  -------  ---------- --------")
       For Each it In items.OrderBy(Function(x) x.DateEntry).ThenBy(Function(x) x.Code)
          sb.AppendLine($"{it.Code.PadRight(10)} {it.Title.PadRight(49)} {it.DateEntry:yyyy-MM-dd}
{it.Credits.ToString().PadLeft(7)}  {it.Score.PadLeft(10)} {it.Result}")
       Next
    End If

    ' Footer & verification
    sb.AppendLine(New String("-"c, 94))
    Dim valid = If(meta.ValidTo.HasValue, $"  Valid to: {meta.ValidTo.Value:yyyy-MM-dd}", "")
    sb.AppendLine($"Serial: {meta.Serial}  Issued: {meta.IssueDate:yyyy-MM-dd}{valid}")
    sb.AppendLine($"Signed: {meta.Signatory}, {meta.SignatoryTitle}")
```

```
    If meta.IsReissue Then sb.AppendLine($"Re-issue of: {meta.ReplacesSerial}  Reason:
{meta.ReissueReason}")
    sb.AppendLine(FooterFor(meta.authority, meta.kind))
    AppendVerification(sb, person, meta)

    Return sb.ToString()
  End Function



    Select Case a
      Case AuthorityType.SARB : Return "South African Reserve Bank (SARB)"
      Case AuthorityType.SARS : Return "South African Revenue Service (SARS)"
      Case AuthorityType.Graduate_DataScience : Return "Graduate Programme — Information & Data
Science"
      Case AuthorityType.Alison_LMS : Return "Alison LMS — Diploma/Certificate"
      Case AuthorityType.AUI_Postdoctoral : Return "AUI — Postdoctoral Award"
      Case AuthorityType.MetPolice_IP_License : Return "Metropolitan Police — IP License"
      Case AuthorityType.MetPolice_Training : Return "Metropolitan Police — Training"
      Case AuthorityType.Tableau_Trailblazer : Return "Tableau — Trailblazer"
      Case Else : Return "Issuing Authority"
    End Select
  End Function



    Dim baseTitle As String
    Select Case k
      Case DocKind.Training_Certificate: baseTitle = "TRAINING CERTIFICATE"
      Case DocKind.Compliance_Certificate: baseTitle = "COMPLIANCE CERTIFICATE"
      Case DocKind.Participation_Certificate: baseTitle = "CERTIFICATE OF PARTICIPATION"
      Case DocKind.Diploma_Certificate: baseTitle = "DIPLOMA CERTIFICATE"
      Case DocKind.Award_Certificate: baseTitle = "AWARD CERTIFICATE"
      Case DocKind.Statement_Of_Completion: baseTitle = "STATEMENT OF COMPLETION"
      Case DocKind.License_Certificate: baseTitle = "LICENSE CERTIFICATE"
      Case DocKind.Badge_Certificate: baseTitle = "BADGE CERTIFICATE"
      Case DocKind.Reissue_Notice: baseTitle = "RE-ISSUE NOTICE"
      Case Else: baseTitle = "CERTIFICATE"
    End Select
    Return $"{baseTitle} — {a}"
  End Function



    Select Case a
      Case AuthorityType.sarb
        Return "Training/compliance acknowledgement; official SARB records remain authoritative."
      Case AuthorityType.sars
        Return "Issued for graduate tax/compliance support; official status remains with SARS."
      Case AuthorityType.Graduate_DataScience
        Return "Graduate learning outcomes acknowledgement; align with programme handbook."
      Case AuthorityType.Alison_LMS
        Return "Alison LMS achievement; verify via Alison credential link if applicable."
      Case AuthorityType.AUI_Postdoctoral
```

```
            Return "Postdoctoral award attestation; institutional verification applies."
        Case AuthorityType.MetPolice_License
            Return "License summary; validity subject to Metropolitan Police licensing conditions."
        Case AuthorityType.MetPolice_Training
            Return "Training completion; operational authorisations governed by service policy."
        Case AuthorityType.Tableau_Trailblazer
            Return "Trailblazer badge/certificate; official record via Trailhead/Trailblazer profile."
        Case Else
            Return "Subject to verification with issuing authority."
    End Select
  End Function


    Dim payload = $"serial={m.Serial}|id={p.IDNumber}|date={m.IssueDate:yyyy-MM-dd}|
auth={m.Authority}|kind={m.Kind}"
    Dim hash = Sha256Hex(payload)
    sb.AppendLine($"Verification hash: {hash}")
    If Not String.IsNullOrWhiteSpace(m.VerificationURL) Then
      sb.AppendLine($"Verify at: {m.VerificationURL}?serial={m.Serial}")
      sb.AppendLine($"QR payload: {payload}")
    End If
  End Sub


    Using sha = SHA256.Create()
      Dim b = sha.ComputeHash(Encoding.UTF8.GetBytes(s))
      Return BitConverter.ToString(b).Replace("-", "").ToLowerInvariant()
    End Using
  End Function

End Module

Printing and serialsModule Demo

    Dim person = New Person With {
      .FullName = "Tshingombe Tshitadi Fiston",
      .IDNumber = "9001015800082",
      .Email = "tshingombe@example.org (mailto:tshingombe@example.org)",
      .ProfileId = "GRAD-DS-2025-0042"
    }

    ' 1) SARB certificate — Engineering Electronics compliance/training
    Dim sarb = New Organisation With {.Name = "South African Reserve Bank (SARB)", .Address =
"Pretoria", .Contact = "www.resbank.co.za (http://www.resbank.co.za)", .References = "Vendor: SARB-
VND-0099"}
    Dim sarbMeta = New CertificateMeta With {
      .Authority = AuthorityType.SARB, .Kind = DocKind.Compliance_Certificate,
      .Serial = SerialRules.NextSerial(AuthorityType.SARB, DocKind.Compliance_Certificate, Date.Today,
11),
      .IssueDate = Date.Today, .ValidTo = Date.Today.AddYears(1),
      .Signatory = "Engineering Compliance Lead", .SignatoryTitle = "SARB Facilities & Ops",
```

```
        .VerificationURL = "https://verify.example.org (https://verify.example.org)"
    }
    Dim sarbItems = New List(Of LineItem) From {
        New LineItem With {.Code = "ELEC-01", .Title = "Electronic Systems Safety (Bank Sites)", .DateEntry
= Date.Today, .Score = "Comp", .Result = "Valid"},
        New LineItem With {.Code = "PTW", .Title = "Permit-to-Work & Isolation", .DateEntry = Date.Today,
.Score = "Comp", .Result = "Valid"}
    }
    Console.WriteLine(Renderers.RenderCertificate(sarb, person,
        New ProgrammeRef With {.Title = "Engineering Electronics Compliance", .Track = "Bank
Infrastructure", .Level = "Advanced", .Hours = 8},
        sarbMeta, sarbItems))

    ' 2) Graduate information & data science certificate
    Dim grad = New Organisation With {.Name = "Graduate Data Science Programme", .Address =
"Johannesburg", .Contact = "grad-ds@example.org (mailto:grad-ds@example.org)"}
    Dim gradMeta = New CertificateMeta With {
        .Authority = AuthorityType.Graduate_DataScience, .Kind = DocKind.Training_Certificate,
        .Serial = SerialRules.NextSerial(AuthorityType.Graduate_DataScience, DocKind.Training_Certificate,
Date.Today, 37),
        .IssueDate = Date.Today, .Signatory = "Programme Director", .SignatoryTitle = "Graduate School"
    }
    Dim gradItems = New List(Of LineItem) From {
        New LineItem With {.Code = "DS-101", .Title = "Python for Data Science", .DateEntry = Date.Today,
.Score = "86%", .Result = "Pass", .Credits = 0},
        New LineItem With {.Code = "DS-201", .Title = "ML Foundations", .DateEntry = Date.Today, .Score =
"82%", .Result = "Pass"}
    }
    Console.WriteLine()
    Console.WriteLine(Renderers.RenderCertificate(grad, person,
        New ProgrammeRef With {.Title = "Information & Data Science", .Level = "Graduate", .Hours = 60},
        gradMeta, gradItems))

    ' 3) SARS certificate — Engineering graduate compliance/registration support
    Dim sars = New Organisation With {.Name = "South African Revenue Service (SARS)", .Address =
"Pretoria", .Contact = "www.sars.gov.za (http://www.sars.gov.za)"}
    Dim sarsMeta = New CertificateMeta With {
        .Authority = AuthorityType.SARS, .Kind = DocKind.Compliance_Certificate,
        .Serial = SerialRules.NextSerial(AuthorityType.SARS, DocKind.Compliance_Certificate, Date.Today,
18),
        .IssueDate = Date.Today, .Signatory = "Tax Compliance Officer", .SignatoryTitle = "SARS"
    }
    Dim sarsItems = New List(Of LineItem) From {
        New LineItem With {.Code = "TCS", .Title = "Tax Compliance Status", .DateEntry = Date.Today, .Score
= "Compliant", .Result = "Valid"},
        New LineItem With {.Code = "PAYE", .Title = "PAYE/Student Stipend Registration", .DateEntry =
Date.Today, .Score = "Registered", .Result = "Valid"}
    }
    Console.WriteLine()
    Console.WriteLine(Renderers.RenderCertificate(sars, person,
        New ProgrammeRef With {.Title = "Engineering Graduate Compliance Summary"},
```

```
        sarsMeta, sarsItems))

    ' 4) Alison LMS diploma
    Dim alison = New Organisation With {.Name = "Alison LMS", .Address = "Online", .Contact =
"alison.com (http://alison.com)", .References = "Learner ID: ALN-778899"}
    Dim alisonMeta = New CertificateMeta With {
        .Authority = AuthorityType.Alison_LMS, .Kind = DocKind.Diploma_Certificate,
        .Serial = SerialRules.NextSerial(AuthorityType.Alison_LMS, DocKind.Diploma_Certificate,
Date.Today, 5),
        .IssueDate = Date.Today, .Signatory = "LMS Registrar", .SignatoryTitle = "Alison"
    }
    Dim alisonItems = New List(Of LineItem) From {
        New LineItem With {.Code = "ALS-DS", .Title = "Diploma in Data Science", .DateEntry = Date.Today,
.Score = "Completed", .Result = "Awarded"}
    }
    Console.WriteLine()
    Console.WriteLine(Renderers.RenderCertificate(alison, person,
        New ProgrammeRef With {.Title = "Alison Diploma", .Track = "Data Science", .Level = "Diploma",
.Hours = 25},
        alisonMeta, alisonItems))

    ' 5) AUI postdoctoral award
    Dim aui = New Organisation With {.Name = "AUI", .Address = "Research Institute", .Contact =
"aui.example.org (http://aui.example.org)"}
    Dim auiMeta = New CertificateMeta With {
        .Authority = AuthorityType.AUI_Postdoctoral, .Kind = DocKind.Award_Certificate,
        .Serial = SerialRules.NextSerial(AuthorityType.AUI_Postdoctoral, DocKind.Award_Certificate,
Date.Today, 2),
        .IssueDate = Date.Today, .Signatory = "Dean of Research", .SignatoryTitle = "AUI"
    }
    Dim auiItems = New List(Of LineItem) From {
        New LineItem With {.Code = "PD-ELX", .Title = "Postdoctoral Fellowship — Electronics & Systems",
.DateEntry = Date.Today, .Score = "Awarded", .Result = "Active"}
    }
    Console.WriteLine()
    Console.WriteLine(Renderers.RenderCertificate(aui, person,
        New ProgrammeRef With {.Title = "Postdoctoral Award", .Track = "Electronics", .Level =
"Postdoctoral"},
        auiMeta, auiItems))

    ' 6) Met Police IP license certificate
    Dim metIP = New Organisation With {.Name = "Metropolitan Police", .Address = "London", .Contact =
"www.met.police.uk (http://www.met.police.uk)", .References = "Licensing Unit"}
    Dim metIpMeta = New CertificateMeta With {
        .Authority = AuthorityType.MetPolice_IP_License, .Kind = DocKind.License_Certificate,
        .Serial = SerialRules.NextSerial(AuthorityType.MetPolice_IP_License, DocKind.License_Certificate,
Date.Today, 14),
        .IssueDate = Date.Today, .ValidTo = Date.Today.AddYears(1),
        .Signatory = "Licensing Officer", .SignatoryTitle = "MET"
    }
    Dim metIpItems = New List(Of LineItem) From {
```

```
        New LineItem With {.Code = "IP-001", .Title = "Intellectual Property Handling License", .DateEntry =
Date.Today, .Score = "Comp", .Result = "Issued"}
      }
      Console.WriteLine()
      Console.WriteLine(Renderers.RenderCertificate(metIP, person,
        New ProgrammeRef With {.Title = "IP License", .Track = "Evidence/IP", .Level = "Operational"},
        metIpMeta, metIpItems))

      ' 7) Met Police training certificate
      Dim metTr = New Organisation With {.Name = "Metropolitan Police — Training", .Address = "London",
.Contact = "training@met.police.uk (mailto:training@met.police.uk)"}
      Dim metTrMeta = New CertificateMeta With {
        .Authority = AuthorityType.MetPolice_Training, .Kind = DocKind.Training_Certificate,
        .Serial = SerialRules.NextSerial(AuthorityType.MetPolice_Training, DocKind.Training_Certificate,
Date.Today, 29),
        .IssueDate = Date.Today, .Signatory = "Training Commander", .SignatoryTitle = "MET Academy"
      }
      Dim metTrItems = New List(Of LineItem) From {
        New LineItem With {.Code = "MET-DFIR", .Title = "Digital Forensics & IP Handling", .DateEntry =
Date.Today, .Score = "Comp", .Result = "Pass"}
      }
      Console.WriteLine()
      Console.WriteLine(Renderers.RenderCertificate(metTr, person,
        New ProgrammeRef With {.Title = "MET Training Block", .Track = "Digital Forensics", .Hours = 12},
        metTrMeta, metTrItems))

      ' 8) Tableau Trailblazer badge/certificate
      Dim tbl = New Organisation With {.Name = "Tableau", .Address = "Tableau/Salesforce", .Contact =
"trailhead.salesforce.com (http://trailhead.salesforce.com)", .References = "Trailblazer Profile: TB-009921"}
      Dim tblMeta = New CertificateMeta With {
        .Authority = AuthorityType.Tableau_Trailblazer, .Kind = DocKind.Badge_Certificate,
        .Serial = SerialRules.NextSerial(AuthorityType.Tableau_Trailblazer, DocKind.Badge_Certificate,
Date.Today, 61),
        .IssueDate = Date.Today, .Signatory = "Trailhead Team", .SignatoryTitle = "Salesforce"
      }
      Dim tblItems = New List(Of LineItem) From {
        New LineItem With {.Code = "TB-VA", .Title = "Tableau Visual Analytics Badge", .DateEntry =
Date.Today, .Score = "Badge", .Result = "Earned"}
      }
      Console.WriteLine()
      Console.WriteLine(Renderers.RenderCertificate(tbl, person,
        New ProgrammeRef With {.Title = "Trailblazer Achievement", .Track = "Analytics", .Level = "Badge"},
        tblMeta, tblItems))
    End Sub
End Module
```

Tailor and harden

Serial schemes per authority (e.g., SARB-COMP-YYYY-######; MET-LIC-YYYY-######; TB-BADGE-YYYY-######).

Exact field blocks (license numbers, badge URLs, diploma links, tax TCS pins).

Branding (logos/seals) via PrintPage drawing for each authority variant.

Verification endpoint to validate serial/hash payloads.

If you share your preferred serial prefixes and authority text blocks, I'll wrap this into a WinForms UI with dropdowns, autoserials, print preview, and PDF export.

```
Imports System.text
Imports System.Drawing.Printing
Imports System.Security.Cryptography

' ---------- Enumerations ----------
Public Enum DocType
    Registration_Certificate
    Statement_Of_Results
    Academic_Transcript
    NDiploma_Attestation
    Skill_Endorsement
    Graduation_Outcome
    Experimental_Certificate_PowerElectrician
    Policing_Security_Certificate
    Role_Badge
    Reissue_Notice
End Enum

Public Enum Stream
    Electrical_NATED
    Electrical_Construction
    Refrigeration_Aircon
    Power_Generation
    Power_Transmission
    Panel_Control_Wiring
    ICT
    Policing
    Traffic
    Private_Security
    Detective
    Teaching
    Assessor
    Other
End Enum

Public Enum RoleLevel
    Learner
    Junior_Trade
    Senior_Trade
    Lecturer
    Teacher
    Assessor
    ICT_Practitioner
```

```
End Enum
Public Enum CaseStatus
Draft:      Submitted: InReview: approved: Awarded: Reissued: Archived
End Enum

' ---------- Parties ----------
Public Class Institution
    Public Property Name As String
    Public Property Address As String
    Public Property Contact As String
    Public Property Registrations As String ' e.g., DHET/QCTO/UMALUSI/PSIRA/Company refs
End Class

Public Class Person
    Public Property FullName As String
    Public Property IDNumber As String
    Public Property DOB As Date
    Public Property StudentNo As String
    Public Property Email As String
    Public Property Phone As String
End Class

' ---------- Academic + Skills ----------
Public Class SubjectResult
    Public Property Code As String
    Public Property Name As String
    Public Property Level As String       ' N1..N6 or module level
    Public Property Stream As Stream
    Public Property Term As String        ' e.g., 2025 T1
    Public Property Mark As Integer       ' 0..100
    Public Property Credits As Integer    ' optional for N4–N6/skills
End Class

Public Class SkillItem
    Public Property Code As String
    Public Property Title As String
    Public Property Stream As Stream
    Public Property Hours As Integer
    Public Property Result As String      ' Comp/Pass/Fail
    Public Property DateAchieved As Date
End Class

' ---------- Certificates ----------
Public Class CertificateMeta
    Public Property Type As DocType
    Public Property Serial As String
    Public Property IssueDate As Date
    Public Property Status As CaseStatus
    Public Property Signatory As String
    Public Property SignatoryTitle As String
```

```
    Public Property VerificationURL As String
    ' Reissu
    Public Property IsReissue As Boolean
    Public Property ReplacesSerial As String
    Public Property ReissueReason As String
End Class

' ---------- Rules ----------
Public Class Rules
    Public Property PassThreshold As Integer = 50
    Public Property DistinctionThreshold As Integer = 75
    Public Property RequiredSubjectsPerLevel As Integer = 4   ' N1..N3 typical
    Public Property DiplomaSubjectsTotal As Integer = 12      ' N4–N6
    Public Property DiplomaExperienceMonths As Integer = 18   ' configurable
End Class

Catalogs and helpers (NATED electrical + skills)
vbPublic Module Catalog
    ' Minimal NATED Electrical subject maps (extend as needed)
    Public Function ElectricalN1() As List(Of SubjectResult)
        Return New List(Of SubjectResult) From {
            New SubjectResult With {.Code = "16030121", .Name = "Mathematics", .Level = "N1", .Stream =
Stream.Electrical_NATED},
            New SubjectResult With {.Code = "15070391", .Name = "Engineering Science", .Level = "N1", .Stream
= Stream.Electrical_NATED},
            New SubjectResult With {.Code = "8080641", .Name = "Industrial Electronics", .Level = "N1", .Stream
= Stream.Electrical_NATED},
            New SubjectResult With {.Code = "11041861", .Name = "Electrical Trade Theory", .Level = "N1",
.Stream = Stream.Electrical_NATED}
        }
    End Function

    Public Function ElectricalN2() As List(Of SubjectResult)
        Return New List(Of SubjectResult) From {
            New SubjectResult With {.Code = "16030192", .Name = "Mathematics", .Level = "N2", .Stream =
Stream.Electrical_NATED},
            New SubjectResult With {.Code = "15070402", .Name = "Engineering Science", .Level = "N2", .Stream
= Stream.Electrical_NATED},
            New SubjectResult With {.Code = "8080602", .Name = "Industrial Electronics", .Level = "N2", .Stream
= Stream.Electrical_NATED},
            New SubjectResult With {.Code = "11041872", .Name = "Electrical Trade Theory", .Level = "N2",
.Stream = Stream.Electrical_NATED}
        }
    End Function

    Public Function ElectricalN3() As List(Of SubjectResult)
        Return New List(Of SubjectResult) From {
            New SubjectResult With {.Code = "16030143", .Name = "Mathematics", .Level = "N3", .Stream =
Stream.Electrical_NATED},
            New SubjectResult With {.Code = "15070413", .Name = "Engineering Science", .Level = "N3", .Stream
= Stream.Electrical_NATED},
```

```
        New SubjectResult With {.Code = "8080613", .Name = "Industrial Electronics", .Level = "N3", .Stream
= Stream.Electrical_NATED},
        New SubjectResult With {.Code = "11040343", .Name = "Electro-Technology", .Level = "N3", .Stream =
Stream.Electrical_NATED}
      }
  End Function

  ' Example skills catalog (extend per college offer)
  Public Function SkillCatalog() As List(Of SkillItem)
    Return New List(Of SkillItem) From {
      New SkillItem With {.Code = "PCW-101", .Title = "Panel Control Wiring", .Stream =
Stream.Panel_Control_Wiring, .Hours = 16},
      New SkillItem With {.Code = "REF-AC-201", .Title = "Refrigeration & Air Conditioning", .Stream =
Stream.Refrigeration_Aircon, .Hours = 24},
      New SkillItem With {.Code = "CON-EL-150", .Title = "Construction Electrical Basics", .Stream =
Stream.Electrical_Construction, .Hours = 20},
      New SkillItem With {.Code = "GEN-310", .Title = "Power Generation Fundamentals", .Stream =
Stream.Power_Generation, .Hours = 18},
      New SkillItem With {.Code = "TX-320", .Title = "Transmission Safety & Switching", .Stream =
Stream.Power_Transmission, .Hours = 18},
      New SkillItem With {.Code = "POL-401", .Title = "Resolved Crime (Applied Policing)", .Stream =
Stream.Policing, .Hours = 16},
      New SkillItem With {.Code = "SEC-221", .Title = "Private Security Operations", .Stream =
Stream.Private_Security, .Hours = 12},
      New SkillItem With {.Code = "ICT-110", .Title = "ICT Fundamentals & Tech Docs", .Stream =
Stream.ICT, .Hours = 12}
    }
  End Function
End Module

Public Module AcademicRules

    If mark >= r.DistinctionThreshold Then Return "Distinction"
    If mark >= r.PassThreshold Then Return "Pass"
    Return "Fail"
  End Function

  Public Function EligibleNCertificate(results As IEnumerable(Of SubjectResult), level As String, r As
Rules) As Boolean
    Dim lvl = results.Where(Function(s) s.Level = level)
    Dim passed = lvl.Count(Function(s) s.Mark >= r.PassThreshold)
    Return passed >= r.RequiredSubjectsPerLevel
  End Function

  Public Function DiplomaProgress(results As IEnumerable(Of SubjectResult), r As Rules, expMonths As
Integer) As String
    Dim passedN4N6 = results.Count(Function(s) (s.Level = "N4" Or s.Level = "N5" Or s.Level = "N6")
AndAlso s.Mark >= r.PassThreshold)
    If passedN4N6 >= r.DiplomaSubjectsTotal AndAlso expMonths >= r.DiplomaExperienceMonths Then
      Return "Eligible for National N Diploma"
    ElseIf passedN4N6 >= r.DiplomaSubjectsTotal Then
```

```
        Return "Pending workplace experience"
      Else
        Return $"In progress ({passedN4N6}/{r.DiplomaSubjectsTotal})"
      End If
    End Function
End ModulePublic Module Documents



      Dim sb As New StringBuilder()
      sb.AppendLine (inst.Name)
      sb.AppendLine (inst.Address)
      sb.AppendLine (inst.Contact)
      If Not String.IsNullOrWhiteSpace(inst.Registrations) Then sb.AppendLine(inst.Registrations)
      sb.AppendLine(New String("-"c, 86))
      sb.AppendLine ("REGISTRATION CERTIFICATE")
      sb.AppendLine(New String("-"c, 86))
      sb.AppendLine($"Student: {p.FullName}   ID: {p.IDNumber}   Student No.: {p.StudentNo}   DOB:
{p.DOB:yyyy-MM-dd}")
      sb.AppendLine($"Programme: {programme}   Session: {session}")
      sb.AppendLine($"Serial: {meta.Serial}   Issued: {meta.IssueDate:yyyy-MM-dd}   Status: {meta.Status}")
      sb.AppendLine ("Note: Registration subject to institutional policies and fee clearance.")
      AppendVerification(sb, p, meta)
      Return sb.ToString()
    End Function

    Public Function RenderStatementOfResults(inst As Institution, p As Person, meta As CertificateMeta,
results As IEnumerable(Of SubjectResult), rules As Rules, session As String, level As String) As String
      Dim sb As New StringBuilder()
      sb.AppendLine (inst.Name)
      sb.AppendLine (inst.Address)
      sb.AppendLine (inst.Contact)
      sb.AppendLine(New String("-"c, 86))
      sb.AppendLine($"STATEMENT OF RESULTS — {session} (Level {level})")
      sb.AppendLine(New String("-"c, 86))
      sb.AppendLine($"Student: {p.FullName}   ID: {p.IDNumber}   Student No.: {p.StudentNo}")
      sb.AppendLine ("Code      Subject                                  Mark  Result")
      sb.AppendLine ("--------  ---------------------------------------  ----  ----------")
      For Each s In results.Where(Function(x) x.Term = session AndAlso x.Level = level).OrderBy(Function(x)
x.Code)
        sb.AppendLine($"{s.Code.PadRight(9)} {s.Name.PadRight(40)} {s.Mark.ToString().PadLeft(4)}
{AcademicRules.Band(s.Mark, rules)}")
      Next
      Dim eligible = AcademicRules.EligibleNCertificate(results.Where(Function(x) x.Term = session), level,
rules)
      sb.AppendLine(New String("-"c, 86))
      sb.AppendLine($"N{level.Substring(1)} Certificate Eligibility: {If(eligible, "Meets combination", "Not
met")}")
      sb.AppendLine($"Pass = {rules.PassThreshold}; Distinction = {rules.DistinctionThreshold}")
      sb.AppendLine($"Serial: {meta.Serial}   Issued: {meta.IssueDate:yyyy-MM-dd}")
      AppendVerification(sb, p, meta)
      Return sb.ToString()
```

```
    End Function
```



```
    Public Function RenderAcademicTranscript(inst As Institution, p As Person, meta As CertificateMeta,
results As IEnumerable(Of SubjectResult), rules As Rules) As String
        Dim sb As New StringBuilder()
        sb.AppendLine (inst.Name)
        sb.AppendLine (inst.Address)
        sb.AppendLine (inst.Contact)
        sb.AppendLine(New String("-"c, 86))
        sb.AppendLine ("ACADEMIC TRANSCRIPT — NATED (Electrical)")
        sb.AppendLine(New String("-"c, 86))
        sb.AppendLine($"Student: {p.FullName}   ID: {p.IDNumber}   Student No.: {p.StudentNo}")
        sb.AppendLine ("Term    Level  Code      Subject                          Mark  Result")
        sb.AppendLine ("------  -----  --------  -------------------------------  ----  ----------")
        For Each s In results.OrderBy(Function(x) x.Term).ThenBy(Function(x) x.Level).ThenBy(Function(x)
x.Code)
            sb.AppendLine($"{s.Term.PadRight(6)} {s.Level.PadRight(5)} {s.Code.PadRight(9)}
{s.Name.PadRight(40)} {s.Mark.ToString().PadLeft(4)}  {AcademicRules.Band(s.Mark, rules)}")
        Next
        sb.AppendLine(New String("-"c, 86))
        sb.AppendLine($"Serial: {meta.Serial}   Issued: {meta.IssueDate:yyyy-MM-dd}   Status: {meta.Status}")
        sb.AppendLine ("Note: This transcript is subject to awarding body verification (DHET/UMALUSI/QCTO
where applicable).")
        AppendVerification(sb, p, meta)
        Return sb.ToString()
    End Function

    Public Function RenderNDiplomaAttestation(inst As Institution, p As Person, meta As CertificateMeta,
results As IEnumerable(Of SubjectResult), rules As Rules, expMonths As Integer) As String
        Dim sb As New StringBuilder()
        sb.AppendLine (inst.Name)
        sb.AppendLine (inst.Address)
        sb.AppendLine (inst.Contact)
        sb.AppendLine(New String("-"c, 86))
        sb.AppendLine ("NATIONAL N DIPLOMA ATTESTATION — Electrical Engineering")
        sb.AppendLine(New String("-"c, 86))
        sb.AppendLine($"Student: {p.FullName}   ID: {p.IDNumber}   Student No.: {p.StudentNo}")
        Dim status = AcademicRules.DiplomaProgress(results, rules, expMonths)
        sb.AppendLine($"Status: {status} (Experience: {expMonths}/{rules.DiplomaExperienceMonths}
months)")
        sb.AppendLine ("Summary (N4–N6 passed subjects count used for eligibility; detailed transcript
attached).")
        sb.AppendLine($"Serial: {meta.Serial}   Issued: {meta.IssueDate:yyyy-MM-dd}   Signatory:
{meta.Signatory}, {meta.SignatoryTitle}")
        AppendVerification(sb, p, meta)
        Return sb.ToString()
    End Function

    Public Function RenderSkillEndorsement(inst As Institution, p As Person, meta As CertificateMeta, skills
As IEnumerable(Of SkillItem), title As String) As String
        Dim sb As New StringBuilder()
```

```
        sb.AppendLine (inst.Name)
        sb.AppendLine (inst.Address)
        sb.AppendLine (inst.Contact)
        sb.AppendLine(New String("-"c, 86))
        sb.AppendLine($"SKILL ENDORSEMENT — {title}")
        sb.AppendLine(New String("-"c, 86))
        sb.AppendLine($"Candidate: {p.FullName}   ID: {p.IDNumber}   Student No.: {p.StudentNo}")
        sb.AppendLine ("Code     Skill/Module                          Hours  Date       Result")
        sb.AppendLine ("-------  ------------------------------------  -----  ---------  -------")
        For Each k In skills.OrderBy(Function(x) x.DateAchieved).ThenBy(Function(x) x.Code)
            sb.AppendLine($"{k.Code.PadRight(8)} {k.Title.PadRight(40)} {k.Hours.ToString().PadLeft(5)}
{k.DateAchieved:yyyy-MM-dd}  {k.Result}")
        Next
        sb.AppendLine(New String("-"c, 86))
        sb.AppendLine($"Serial: {meta.Serial}   Issued: {meta.IssueDate:yyyy-MM-dd}   Signatory:
{meta.Signatory}, {meta.SignatoryTitle}")
        AppendVerification(sb, p, meta)
        Return sb.ToString()
    End Function

    Public Function RenderPowerExperimental(inst As Institution, p As Person, meta As CertificateMeta,
components As IEnumerable(Of SkillItem), track As String) As String
        Dim sb As New StringBuilder()
        sb.AppendLine (inst.Name)
        sb.AppendLine (inst.Address)
        sb.AppendLine (inst.Contact)
        sb.AppendLine(New String("-"c, 86))
        sb.AppendLine($"EXPERIMENTAL CERTIFICATE — Power Electrician ({track})")
        sb.AppendLine(New String("-"c, 86))
        sb.AppendLine ("Code     Component                             Hours  Date       Result")
        sb.AppendLine ("-------  ------------------------------------  -----  ---------  -------")
        For Each c In components.OrderBy(Function(x) x.DateAchieved)
            sb.AppendLine($"{c.Code.PadRight(8)} {c.Title.PadRight(42)} {c.Hours.ToString().PadLeft(5)}
{c.DateAchieved:yyyy-MM-dd}  {c.Result}")
        Next
        sb.AppendLine(New String("-"c, 86))
        sb.AppendLine($"Serial: {meta.Serial}   Issued: {meta.IssueDate:yyyy-MM-dd}")
        AppendVerification(sb, p, meta)
        Return sb.ToString()
    End Function

    Public Function RenderPolicingSecurity(inst As Institution, p As Person, meta As CertificateMeta, items
As IEnumerable(Of SkillItem), title As String) As String
        Dim sb As New StringBuilder()
        sb.AppendLine (inst.Name)
        sb.AppendLine (inst.Address)
        sb.AppendLine (inst.Contact)
        sb.AppendLine(New String("-"c, 86))
        sb.AppendLine($"{title.ToUpper()} — Policing/Security")
        sb.AppendLine(New String("-"c, 86))
        sb.AppendLine ("Code     Outcome/Module                        Hours  Date       Result")
```

```
        sb.AppendLine ("-------- ------------------------------------ ----- ---------- --------")
        For Each it In ordered
            sb.AppendLine($"{it.Code.PadRight(8)} {it.Title.PadRight(42)} {it.Hours.ToString().PadLeft(5)}
{it.DateAchieved:yyyy-MM-dd}  {it.Result}")
        Next
        sb.AppendLine(New String("-"c, 86))
        sb.AppendLine($"Serial: {meta.Serial}   Issued: {meta.IssueDate:yyyy-MM-dd}   Status: {meta.Status}")
        AppendVerification(sb, p, meta)
        Return sb.ToString()
    End Function

    Public Function RenderRoleBadge(inst As Institution, p As Person, meta As CertificateMeta, role As
RoleLevel, tracks As IEnumerable(Of Stream), validTo As Date?) As String
        Dim sb As New StringBuilder()
        sb.AppendLine (inst.Name)
        sb.AppendLine (inst.Address)
        sb.AppendLine (inst.Contact)
        sb.AppendLine(New String("-"c, 86))
        sb.AppendLine ("ROLE BADGE")
        sb.AppendLine(New String("-"c, 86))
        sb.AppendLine($"Holder: {p.FullName}   ID: {p.IDNumber}   Role: {role}")
        sb.AppendLine($"Tracks: {String.Join(", ", tracks.Select(Function(t) t.ToString()))}")
        If validTo.HasValue Then sb.AppendLine($"Valid To: {validTo.Value:yyyy-MM-dd}")
        sb.AppendLine($"Serial: {meta.Serial}   Issued: {meta.IssueDate:yyyy-MM-dd}   Signatory:
{meta.Signatory}, {meta.SignatoryTitle}")
        AppendVerification(sb, p, meta)
        Return sb.ToString()
    End Function

    )
        Dim payload = $"serial={meta.Serial}|id={p.IDNumber}|date={meta.IssueDate:yyyy-MM-dd}|
type={meta.Type}|status={meta.Status}"
        Dim hash = Sha256Hex(payload)
        sb.AppendLine($"Verification hash: {hash}")
        If Not String.IsNullOrWhiteSpace(meta.VerificationURL) Then
            sb.AppendLine($"Verify: {meta.VerificationURL}?serial={meta.Serial}")
            sb.AppendLine($"QR payload: {payload}")
        End If
        If meta.IsReissue Then
            sb.AppendLine($"Re-issue of: {meta.ReplacesSerial}   Reason: {meta.ReissueReason}")
        End If
    End Sub

    Private Function Sha256Hex(input As String) As String
        Using sha = SHA256.Create()
            Dim b = sha.ComputeHash(Encoding.UTF8.GetBytes(input))
            Return BitConverter.ToString(b).Replace("-", "").ToLowerInvariant()
        End Using
    End Function
End Module
```

Printing and auditModule Demo


```
    Dim inst As New Institution With {
        .Name = "St. Peace College Africa Institute (Police & Engineering)",
        .Address = "Johannesburg, South Africa",
        .Contact = "Tel: +27 11 000 0000 | admin@stpeace.africa",
        .Registrations = "DHET/QCTO/UMALUSI Refs; PSIRA/Training Approvals"
    }

    Dim p As New Person With {
        .FullName = "Tshingombe Tshitadi Fiston",
        .IDNumber = "9001015800082",
        .DOB = #1990-01-01#,
        .StudentNo = "SPC-EL-2025-0042",
        .Email = "tshingombe@example.org (mailto:tshingombe@example.org)",
        .Phone = "+27 72 000 0000"
    }

    Dim rules As New Rules With {.PassThreshold = 50, .DistinctionThreshold = 75}

    ' 1) Registration
    Dim regMeta As New CertificateMeta With {.Type = DocType.Registration_Certificate, .Serial = "SPC-
REG-2025-000121", .IssueDate = Date.Today, .Status = CaseStatus.Approved, .Signatory = "Registrar",
.SignatoryTitle = "Academic Registry", .VerificationURL = "https://verify.stpeace.africa (https://
verify.stpeace.africa)"}
    Console.WriteLine (Documents.RenderRegistration(inst, p, regMeta, "Electrical Engineering (NATED)",
"2025 Trimester 1"))

    ' 2) Statement of results (N2)
    Dim rN2 = Catalog.ElectricalN2()
    For Each s In rN2 : s.Term = "2025T1" : s.Mark = If(s.Code = "16030192", 78, If(s.Code = "15070402",
66, If(s.Code = "8080602", 61, 54))) : Next
    Dim sorMeta As New CertificateMeta With {.Type = DocType.Statement_Of_Results, .Serial = "SPC-
SOR-2025-000441", .IssueDate = Date.Today, .Status = CaseStatus.Approved, .Signatory = "Exams Officer",
.SignatoryTitle = "Assessment"}
    Console.WriteLine()
    Console.WriteLine (Documents.RenderStatementOfResults(inst, p, sorMeta, rN2, rules, "2025T1",
"N2"))

    ' 3) Academic transcript (N1–N3)
    Dim all = New List(Of SubjectResult)()
    Dim rN1 = Catalog.ElectricalN1() : For Each s In rN1 : s.Term = "2024T3" : s.Mark = 71 : Next :
all.AddRange(rN1)
    Dim rN3 = Catalog.ElectricalN3() : For Each s In rN3 : s.Term = "2025T2" : s.Mark = 62 : Next :
all.AddRange(rN2) : all.AddRange(rN3)
    Dim trMeta As New CertificateMeta With {.Type = DocType.Academic_Transcript, .Serial = "SPC-
TR-2025-000199", .IssueDate = Date.Today, .Status = CaseStatus.Approved, .Signatory = "Registrar",
.SignatoryTitle = "Academic Registry"}
    Console.WriteLine()
    Console.WriteLine (Documents.RenderAcademicTranscript(inst, p, trMeta, all, rules))
```

```
' 4) N Diploma attestation
Dim ndMeta As New CertificateMeta With {.Type = DocType.NDiploma_Attestation, .Serial = "SPC-ND-2026-000031", .IssueDate = Date.Today, .Status = CaseStatus.InReview, .Signatory = "Dean", .SignatoryTitle = "School of Engineering"}
Console.WriteLine()
Console.WriteLine (Documents.RenderNDiplomaAttestation(inst, p, ndMeta, all, rules, expMonths:=12))

' 5) Skills endorsements (panel control, refrigeration/aircon, construction electrical)
Dim skillBase = Catalog.SkillCatalog()
' Simulate completion
For i = 0 To skillBase.count - 1
  skillBase(i).Result = "Comp" : skillBase(i).DateAchieved = Date.Today.AddDays(-i)
Next
Dim skillMeta As New CertificateMeta With {.Type = DocType.Skill_Endorsement, .Serial = "SPC-SKL-2025-000088", .IssueDate = Date.Today, .Status = CaseStatus.Approved, .Signatory = "Head of Training", .SignatoryTitle = "Technical"}
Console.WriteLine()
Console.WriteLine(Documents.RenderSkillEndorsement(inst, p, skillMeta, skillBase.Where(Function(k) k.Stream = Stream.Panel_Control_Wiring Or k.Stream = Stream.Refrigeration_Aircon Or k.Stream = Stream.Electrical_Construction), "Electrical Skills Suite"))

' 6) Experimental certificate — Power Electrician (Generation/Transmission)
Dim powerMeta As New CertificateMeta With {.Type = DocType.Experimental_Certificate_PowerElectrician, .Serial = "SPC-PWR-2025-000057", .IssueDate = Date.Today, .Status = CaseStatus.Approved, .Signatory = "Power Faculty Lead", .SignatoryTitle = "Power Systems"}
Console.WriteLine()
Console.WriteLine(Documents.RenderPowerExperimental(inst, p, powerMeta, skillBase.Where(Function(k) k.Stream = Stream.Power_Generation), "Generation"))
Console.WriteLine()
Console.WriteLine(Documents.RenderPowerExperimental(inst, p, powerMeta, skillBase.Where(Function(k) k.Stream = Stream.Power_Transmission), "Transmission"))

' 7) Policing/Security certificate — Resolved Crime / Detective / Private Security
Dim polMeta As New CertificateMeta With {.Type = DocType.Policing_Security_Certificate, .Serial = "SPC-POL-2025-000133", .IssueDate = Date.Today, .Status = CaseStatus.Awarded, .Signatory = "Programme Director", .SignatoryTitle = "Justice & Security"}
Console.WriteLine()
Console.WriteLine(Documents.RenderPolicingSecurity(inst, p, polMeta, skillBase.Where(Function(k) k.Stream = Stream.Policing Or k.Stream = Stream.Private_Security), "Resolved Crime & Private Security"))

' 8) Role badges — learner, junior/senior trade, lecturer/teacher, assessor, ICT
Dim badgeMeta As New CertificateMeta With {.Type = DocType.Role_Badge, .Serial = "SPC-RB-2025-000311", .IssueDate = Date.Today, .Status = CaseStatus.Approved, .Signatory = "HR Director", .SignatoryTitle = "People & Capability"}
Console.WriteLine()
Console.WriteLine(Documents.RenderRoleBadge(inst, p, badgeMeta, RoleLevel.Learner, {Stream.Electrical_NATED, Stream.Policing}, validTo:=Date.Today.AddYears(1)))
Console.WriteLine()
Console.WriteLine(Documents.RenderRoleBadge(inst, p, badgeMeta, RoleLevel.Assessor,
```

{Stream.Electrical_NATED, Stream.Panel_Control_Wiring, Stream.ICT}, validTo:=Date.Today.AddYears(2)))


```
    ' Audit
    Dim audit = New AuditLogger("stpeace_audit.log")
    audit.Append("system", "Issued", "Registration", regMeta.Serial, "Electrical NATED 2025T1")
    audit.Append("system", "Issued", "StatementOfResults", sorMeta.Serial, "N2 2025T1")
    audit.Append("system", "Issued", "Transcript", trMeta.Serial, "N1–N3")
    audit.Append("system", "Drafted", "NDiploma", ndMeta.Serial, "N4–N6 tracking pending")
    audit.Append("system", "Issued", "Skills", skillMeta.Serial, "Electrical Skills Suite")
    audit.Append("system", "Issued", "PowerExperimental", powerMeta.Serial, "Gen/Tx")
    audit.Append("system", "Issued", "PolicingSecurity", polMeta.Serial, "Resolved Crime & Security")
    audit.Append("system", "Issued", "RoleBadge", badgeMeta.Serial, "Learner/Assessor")
  End Sub
End Module
```

What I need to finalize for St. Peace College Africa Institute
Sub engdatacertifica()

End Sub

On Fri, Sep 19, 2025 at 3:29 PM TSHINGOMBEKB TSHITADI <tshingombekb@gmail.com (mailto:tshingombekb@gmail.com)> wrote:

On Thu, Sep 18, 2025 at 3:52 PM TSHINGOMBEKB TSHITADI <tshingombekb@gmail.com (mailto:tshingombekb@gmail.com)> wrote:
Absolutely, Tshingombe. Here's a structured **Visual Basic (VBA)** framework tailored to managing **trade license certification**, **test results**, **work permit conditions**, and **council-aligned engineering records**. This model supports: Trade theory + practical test trackingLicense certificate issuanceWork permit and conduct conditionsDismissal, reward, and amendment logicMulti-year record loggingCouncil registration and qualification mapping 📄 **UserForm Structure: frmTradeLicense** 🔷 **Identity & Trade Info Tab**
| **Control** | **Name** | **Purpose** | TextBox | txtStudentID | Unique learner ID
| TextBox | txtFullName | Full name
| ComboBox | cboTradeType | Trade (Electrical, Mechanical, etc.)
| ComboBox | cboCouncil | Engineering Council (e.g., ECSA)
| TextBox | txtCouncilRegID | Council registration number
| ComboBox | cboLicenseType | Theory, Practical, Combined
| TextBox | txtLicenseID | Generated license number🔷 **Test & Certification Tab**
| **Field** | **Controls** | Theory Score | txtTheoryScore
| Practical Score | txtPracticalScore
| Total Score | lblTotalScore (calculated)
| Testify Status | chkTestify (passed witness verification)
| Certificate Issued | chkCertificateIssued
| Certificate Date | txtCertificateDate
| License Validity (Years) | txtLicenseYears
| License Expiry | lblLicenseExpiry (calculated) 🔷 **Work Permit & Conduct Tab**
| **Field** | **Controls** | Work Permit ID | txtWorkPermitID
| Permit Conditions | txtPermitConditions
| Conduct Status | cboConduct (Good, Warning, Dismissed)
| Dismissal Reason | txtDismissalReason

| Reward Points | txtRewardPoints
| Amendm | txtAmendmentNotes | Reward & Council Tab
| **Field** | **Controls** | Record Year | txtRecordYear
| Amendment Year | txtAmendmentYear
| Qualification Level | cboQualificationLevel (NDip, BTech, Trade Cert)
| Council Status | lblCouncilStatus (calculated)
| Final Status | lblFinalStatus (calculated) 🔷 **Buttons**
| **Button** | **Function** | cmdCalculate | Compute scores, expiry, council status
| cmdPrintLicense | Print license certificate
| cmdSaveRecord | Save to sheet
| cmdClearForm | Reset form
| cmdCloseForm | Exit 🧮 **Core Logic: License Evaluation** Private Sub cmdCalculate_Click() Dim theory As Double, practical As Double, total As Double Dim licenseYears As Integer, expiryDate As Date theory = Val(txtTheoryScore.Text) practical = Val(txtPracticalScore.Text) total = (theory + practical) / 2 lblTotalScore.Caption = Format(total, "0.0") If total >= 50 And chkTestify.Value = True Then chkCertificateIssued.Value = True txtCertificateDate.Text = Format(Date, "yyyy-mm-dd") licenseYears = Val(txtLicenseYears.Text) expiryDate = DateAdd("yyyy", licenseYears, Date) lblLicenseExpiry.Caption = Format(expiryDate, "yyyy-mm-dd") lblFinalStatus.Caption = "License Granted" Else lblFinalStatus.Caption = "License Denied" End If ' Council logic If cboCouncil.Text <> "" And txtCouncilRegID.Text <> "" Then lblCouncilStatus.Caption = "Registered with " & cboCouncil.Text Else lblCouncilStatus.Caption = "Not Registered" End IfEnd Sub 🖨️ **License Certificate Print Logic** vb Private Sub cmdPrintLicense_Click() Dim txt As String txt = "TRADE LICENSE CERTIFICATE" & vbCrLf & String(40, "-") & vbCrLf & _ "Name: " & txtFullName.Text & vbCrLf & _ "Trade: " & cboTradeType.Text & vbCrLf & _ "License Type: " & cboLicenseType.Text & vbCrLf & _ "License ID: " & txtLicenseID.Text & vbCrLf & _ "Theory Score: " & txtTheoryScore.Text & " | Practical Score: " & txtPracticalScore.Text & vbCrLf & _ "Total: " & lblTotalScore.Caption & vbCrLf & _ "Certificate Issued: " & IIf(chkCertificateIssued.Value, "Yes", "No") & _ " | Expiry: " & lblLicenseExpiry.Caption & vbCrLf & _ "Council: " & cboCouncil.Text & " | Reg ID: " & txtCouncilRegID.Text & vbCrLf & _ "Final Status: " & lblFinalStatus.Caption MsgBox txt, vbInformation, "License Preview" End Sub Conduct & Work Permit Logic Private Sub EvaluateConduct() If cboConduct.Text = "Dismissed" Then lblFinalStatus.Caption = "License Revoked" ElseIf cboConduct.Text = "Warning" Then lblFinalStatus.Caption = "License Under Review" Else lblFinalStatus.Caption = "License Active" End IfEnd Sub 📊 **Logigram Flow** Code [Start] ↓[Identity Verified] ↓[Test Scores Entered] ↓[Testify Passed?] → No → [Hold License] ↓ Yes[Total Score ≥ 50?] → No → [Reject] ↓ Yes[Certificate Issued] ↓[Work Permit Conditions Met?] → No → [Flag] ↓ Yes[Conduct Status?] → Dismissed → [Revoke License] → Warning → [Review] → Good → [License Active] ↓[Council Registered?] → No → [Flag] ↓ Yes[Print Certificate] ↓[Save Record] ↓[End] **Optional Enhancements** · **QR code on certificate for verification** · **Council API integration for registration validation** · **Amendment tracking with version history** · **Reward system for high performers (e.g., ≥85%)** · **Dismissal audit trail with timestamp and assessor notes** Would you like this scaffolded into a full Excel VBA workbook with certificate templates,

On Thu, Sep 18, 2025 at 3:52 PM TSHINGOMBEKB TSHITADI <tshingombekb@gmail.com (mailto:tshingombekb@gmail.com)> wrote:

On Thu, Sep 18, 2025 at 3:21 PM tshingombe fiston <tshingombefiston@gmail.com (mailto:tshingombefiston@gmail.com)> wrote:
Visual Basic userform for a registration page with tabs Below is a complete, ready-to-use VBA/VB6-style UserForm pattern with labels, a frame, textboxes, gender options, password field, and OK/Cancel/Next buttons. It uses a MultiPage (tabs) to organize “Personal Info” and “Account” pages. **Layout and control**

naming Form name: frmRegistration**MultiPage:** MultiPage1Page(0) Caption: Personal InfoPage(1) Caption: Account**Label:** lblTitle (Caption: Personal details)**Label:** lblFirstName (Caption: First name)**TextBox:** txtFirstName**Label:** lblSurname (Caption: Surname)**TextBox:** txtSurname**Label:** lblDOB (Caption: Birth date)**TextBox:** txtDOB (e.g., YYYY-MM-DD)**Label:** lblGender (Caption: Gender)**OptionButton:** optMale (Caption: Male)**OptionButton:** optFemale (Caption: Female)Page(1) controls:**Label:** lblUsername (Caption: Username)**TextBox:** txtUsername**Label:** lblPassword (Caption: Password)**TextBox:** txtPassword (PasswordChar = •)Buttons (place at bottom of the form, visible on all tabs):**CommandButton:** cmdOK (Caption: OK)**CommandButton:** cmdCancel (Caption: Cancel)**CommandButton:** cmdNext (Caption: Next ») Tip: If available in your environment, you can use a DatePicker instead of txtDOB (e.g., DTPicker). Otherwise, keep a textbox and validate the date. **Suggested tab order Personal Info page:** txtFirstName → txtSurname → txtDOB → optMale → optFemale**Account page:** txtUsername → txtPassword**Form-level:** cmdNext → cmdOK → cmdCancel **UserForm code** Copy this into the frmRegistration code-behind. vb Option Explicit Private Sub UserForm_Initialize() ' Initialize defaults Me.MultiPage1.Value = 0 ' First tab Me.optMale.Value = False Me.optFemale.Value = False Me.txtPassword.PasswordChar = "•" End Sub Private Sub cmdNext_Click() ' Toggle between tabs If Me.MultiPage1.Value < Me.MultiPage1.Pages.Count - 1 Then Me.MultiPage1.Value = Me.MultiPage1.Value + 1 Else Me.MultiPage1.Value = 0 End If End Sub Private Sub cmdCancel_Click() If MsgBox("Cancel registration?", vbQuestion + vbYesNo, "Confirm") = vbYes Then Unload Me End If End Sub Private Sub cmdOK_Click() Dim errMsg As String If Not ValidateInputs(errMsg) Then MsgBox errMsg, vbExclamation, "Validation" Exit Sub End If ' Simulated save; replace with your persistence logic ' e.g., write to worksheet/database/API ' Example (Excel): WriteToSheet MsgBox "Registration successful.", vbInformation, "Success" Unload Me End Sub Private Function ValidateInputs(ByRef errMsg As String) As Boolean Dim dt As Date Dim genderSelected As Boolean ' Basic required fields If Trim$(Me.txtFirstName.Text) = "" Then errMsg = "First name is required." ValidateInputs = False Exit Function End If If Trim$(Me.txtSurname.Text) = "" Then errMsg = "Surname is required." ValidateInputs = False Exit Function End If If Trim$(Me.txtDOB.Text) = "" Then errMsg = "Birth date is required (YYYY-MM-DD)." ValidateInputs = False Exit Function End If ' Date validation (expects a valid date; adjust to your locale/format) On Error GoTo BadDate dt = CDate(Me.txtDOB.Text) On Error GoTo 0 If dt > Date Then errMsg = "Birth date cannot be in the future." ValidateInputs = False Exit Function End If ' Gender genderSelected = (Me.optMale.Value Or Me.optFemale.Value) If Not genderSelected Then errMsg = "Please select a gender." ValidateInputs = False Exit Function End If ' Account page checks If Trim$(Me.txtUsername.Text) = "" Then errMsg = "Username is required." ValidateInputs = False Exit Function End If If Len(Me.txtPassword.Text) < 6 Then errMsg = "Password must be at least 6 characters." ValidateInputs = False Exit Function End If ValidateInputs = True Exit Function BadDate: errMsg = "Invalid birth date. Use a valid date (e.g., 2001-05-17)." ValidateInputs = False End Function ' Optional: Excel example of saving to a sheet Private Sub WriteToSheet() Dim ws As Worksheet Dim nextRow As Long Dim gender As String Set ws = ThisWorkbook.Worksheets("Registrations") If Me.optMale.Value Then gender = "Male" ElseIf Me.optFemale.Value Then gender = "Female" Else gender = "" End If nextRow = ws.Cells(ws.Rows.Count, "A").End(xlUp).Row + 1 With ws .Cells(nextRow, 1).Value = Me.txtFirstName.Text .Cells(nextRow, 2).Value = Me.txtSurname.Text .Cells(nextRow, 3).Value = Me.txtDOB.Text .Cells(nextRow, 4).Value = gender .Cells(nextRow, 5).Value = Me.txtUsername.Text .Cells(nextRow, 6).Value = MaskPassword(Me.txtPassword.Text) ' Store hashed/hidden if possible .Cells(nextRow, 7).Value = Now End With End Sub Private Function MaskPassword(ByVal pwd As String) As String ' Placeholder for hashing/encryption MaskPassword = String$(Len(pwd), "*") End Function **Minimal module to show the form (Excel/VBA)** Place this in a standard module to open the form. vb Sub ShowRegistration() frmRegistration.Show End Sub **Quick build steps** · **Add a UserForm:** Insert → UserForm → Rename to frmRegistration. · **Add a MultiPage:** Insert MultiPage → Name: MultiPage1 → Set Captions to “Personal Info” and “Account”. · **Add a Frame to Page(0):** Name: fraPersonal → Caption: Personal details. · **Add labels/textboxes:** o

lblFirstName + txtFirstName o lblSurname + txtSurname o lblDOB + txtDOB o lblGender + optMale + optFemale + optOther + lblPassword + txtPassword (set PasswordChar = "•"). · **Add buttons:** cmdOK, cmdCancel, cmdNext; set their captions. · **Paste code:** Into frmRegistration code-behind and optional module. · **Run:** Call ShowRegistration. **Optional enhancements** · **Input masks:** Enforce date format with a simple mask or a calendar control. · **Gender options:** Add "Other/Prefer not to say." · **Validation UI:** Highlight invalid fields by changing BackColor. · **Persistence:** Replace WriteToSheet with a database call or JSON file write. · **Navigation:** Disable cmdOK until all validations pass. If you want, tell me your target environment (Excel VBA, Access, VB6, or VB.NET (http://VB.NET) WinForms), and I'll tailor the exact control names and code syntax accordingly. Absolutely, Tshingombe. Here's a structured **Visual Basic (VBA)** framework for a **National Diploma Examination Result Statement Generator** tailored to engineering studies. It includes: · Student info fields · Course modules across terms and semesters · Assessment breakdown (homework, classwork, exams) · Score scaling and qualification levels · Print-ready marksheet logic · Award and pass/fail classification 📄 **UserForm Layout Overview** 🔷 **Student Info Section**

Control Type	**Name**	**Caption**
Label	lblStudentName	Student Name
TextBox	txtStudentName	—
Label	lblSurname	Surname
TextBox	txtSurname	—
Label	lblAchievementYear	Achievement Year
TextBox	txtAchievementYear	—
Label	lblExperienceYears	Work Experience (Years)
TextBox	txtExperienceYears	—

🔷 **Academic Record Section** Use a **MultiPage** or **TabStrip** to organize:

📘 *Page 1: Terms & Semesters*

Term	**Controls**
Term 1–4	txtTerm1, txtTerm2, txtTerm3, txtTerm4
Semester 1–2	txtSem1, txtSem2

📘 *Page 2: Course Modules & Assessment*

Field	**Controls**
Course Topics	lstCourseTopics (ListBox or ComboBox)
Homework	txtHomeworkScore
Classwork	txtClassworkScore
Exams	txtExamScore
Total Score	lblTotalScore (calculated)
Rating (%)	lblRating (calculated)

🔷 **Qualification & Scaling**

Field	**Controls**
Final Qualification	cboQualification (e.g., 1st, 2nd, 3rd, 4th)
Level	cboLevel (1–9)
Course Weight	txtCourseWeight
Scaling Factor	txtScalingFactor
Final Score	lblFinalScore (calculated)

🔷 **Buttons**

Button	**Function**
cmdCalculate	Compute total, rating, qualification
cmdPrint	Print formatted marksheet
cmdClear	Reset form
cmdExit	Close form

🧮 **Calculation Logic (VBA)** Private Sub cmdCalculate_Click() Dim homework As Double, classwork As Double, exam As Double Dim total As Double, rating As Double, scaledScore As Double Dim weight As Double, scaleFactor As Double ' Get scores homework = Val(txtHomeworkScore.Text) classwork = Val(txtClassworkScore.Text) exam = Val(txtExamScore.Text) ' Compute total total = homework + classwork + exam lblTotalScore.Caption = total & " / 100" ' Rating rating = (total / 100) * 100 lblRating.Caption = Format(rating, "0.00") & "%" ' Scaling weight = Val(txtCourseWeight.Text) scaleFactor = Val(txtScalingFactor.Text) scaledScore = total * (weight / 100) * scaleFactor lblFinalScore.Caption = Format(scaledScore, "0.00") ' Qualification logic Select Case rating Case Is >= 50 cboQualification.Text = "Pass" Case 40 To 49 cboQualification.Text = "Conditional Pass" Case 20 To 39 cboQualification.Text = "Fail" Case Is < 20 cboQualification.Text = "Severe Fail" End SelectEnd Sub 🖨️ **Print Statement Logic** Private Sub cmdPrint_Click() Dim msg As String msg = "National Diploma Result Statement" & vbCrLf &

vbCrLf msg = msg & "Student: " & txtStudentName.Text & " " & txtSurname.Text & vbCrLf msg = msg & "Achievem [illegible]" & txtAchievementPts.Text & vbCrLf msg = msg & "Experience: " & txtExperienceYears.Text & " years" & vbCrLf & vbCrLf msg = msg & "Course Topics: " & lstCourseTopics.Text & vbCrLf msg = msg & "Homework: " & txtHomeworkScore.Text & vbCrLf msg = msg & "Classwork: " & txtClassworkScore.Text & vbCrLf msg = msg & "Exam: " & txtExamScore.Text & vbCrLf msg = msg & "Total Score: " & lblTotalScore.Caption & vbCrLf msg = msg & "Rating: " & lblRating.Caption & vbCrLf msg = msg & "Final Qualification: " & cboQualification.Text & vbCrLf msg = msg & "Level: " & cboLevel.Text & vbCrLf msg = msg & "Final Scaled Score: " & lblFinalScore.Caption & vbCrLf MsgBox msg, vbInformation, "Print Preview" ' Replace with actual print logic if needed End Sub 🏅 **Award Logic (Optional)** vb If rating >= 85 Then MsgBox "Student qualifies for Distinction Award!", vbInformation ElseIf rating >= 70 Then MsgBox "Student qualifies for Merit Award.", vbInformation ElseIf rating >= 50 Then MsgBox "Student passed successfully.", vbInformation Else MsgBox "Student did not meet pass criteria.", vbExclamation End If Visual Basic framework for reprint, release, and recertification of result statements Below is a practical Visual Basic/VBA scaffold to manage reprint and release workflows for electrical trade theory result statements, including backlog checks, irregularity flags, insurance/quality-body attestations, and reconciliation of internal vs external assessment. It covers student identity, term/semester records, combination/replace results, and recertification. **Userform structure and fields** · **Form name:** frmResultRelease · **Pages:** MultiPage1 with tabs: Identity, Assessments, Quality, Actions *Identity page* · **Student ID:** txtStudentID · **Username:** txtUsername · **Surname:** txtSurname · **Year of birth:** txtYOB · **Admin year:** txtAdminYear · **Programme:** cboProgramme (NDip, Advanced Dip, BEngTech, Postgrad, etc.) · **Level:** cboLevel (1–8) · **Trade:** cboTrade (Electrical, Instrumentation, etc.) *Assessments page* · **Internal assessment total (0–100):** txtInternal · **External assessment total (0–100):** txtExternal · **Exam type:** cboExamType (Main, Rewrite, Supplementary) · **Attempt count:** txtAttempt · **Backlog credits outstanding:** txtBacklogCredits · **Combination/replace source ID:** txtCombineWithResultID *Quality page* · **Irregularity flag:** chkIrregularity · **Irregularity note:** txtIrregularityNote · **Insurance/QA body clearance:** chkQACleared · **QA reference number:** txtQARef · **Material/proctor issue flag:** chkProctorIssue · **Material batch ref:** txtMaterialBatch *Actions page* · **Status label:** lblReleaseStatus · **Buttons:** cmdReconcile, cmdEvaluate, cmdRelease, cmdReprint, cmdRecertify, cmdSave, cmdExportPDF, cmdClose **Business rules** · **Pass thresholds:** o Pass ≥ 50%; Conditional pass 40–49%; Fail 20–39%; Severe fail < 20. · **Variance check internal vs external:** o If absolute difference > 20 percentage points, set ReviewRequired. · **Irregularity or QA not cleared:** o Hold release until cleared. · **Backlog credits > 0:** o Hold certificate; allow statement with “Provisional” if enabled. · **Rewrite attempt logic:** o If cboExamType = “Rewrite”, mark AttemptedRewrite = True; allow combination/replace if improved. · **Combination and replace result:** o If txtCombineWithResultID not empty and new score higher, replace; else keep best. **Status model** · EligibleForRelease · HoldIrregularity · HoldBacklog · HoldQANotCleared · ReviewVariance · RecertificationRequired · ReprintAllowed **Code: core types and utilities** Option Explicit Private Enum ReleaseStatus EligibleForRelease = 0 HoldIrregularity = 1 HoldBacklog = 2 HoldQANotCleared = 3 ReviewVariance = 4 RecertificationRequired = 5 ReprintAllowed = 6End Enum Private Type StudentRecord StudentID As String Username As String Surname As String YOB As Integer AdminYear As Integer Programme As String Level As Integer Trade As String InternalScore As Double ExternalScore As Double ExamType As String Attempt As Integer BacklogCredits As Integer CombineWithID As String Irregularity As Boolean IrregularityNote As String QACleared As Boolean QARef As String ProctorIssue As Boolean MaterialBatch As String FinalScore As Double Rating As DoubleEnd Type Private Const PASS_THRESHOLD As Double = 50#Private Const CONDITIONAL_LOW As Double = 40#Private Const FAIL_LOW As Double = 20#Private Const VARIANCE_THRESHOLD As Double = 20# 'percentage points **Code: data capture and reconciliation** Private Function ReadForm() As StudentRecord Dim r As StudentRecord r.StudentID = Trim$(txtStudentID.Text) r.Username = Trim$(txtUsername.Text) r.Surname = Trim$(txtSurname.Text) r.YOB = Val(txtYOB.Text) r.AdminYear = Val(txtAdminYear.Text) r.Programme = cboProgramme.Text r.Level = Val(cboLevel.Text) r.Trade =

cboTrade.Text r.InternalScore = Val(txtInternal.Text) r.ExternalScore = Val(txtExternal.Text) r.ExamType = cboExamType.Text r.Attempt = Val(txtAttempt.Text) r.BacklogCredits = Val(txtBacklogCredits.Text) r.CombineWithID = Trim$(txtCombineWithResultID.Text) r.Irregularity = chkIrregularity.Value r.IrregularityNote = Trim$(txtIrregularityNote.Text) r.QACleared = chkQACleared.Value r.QARef = Trim$(txtQARef.Text) r.ProctorIssue = chkProctorIssue.Value r.MaterialBatch = Trim$(txtMaterialBatch.Text) ReadForm = rEnd Function Private Sub ComputeScores(ByRef r As StudentRecord) ' Weighted blend: external prioritized; adjust as needed Dim blended As Double blended = (0.4 * r.InternalScore) + (0.6 * r.ExternalScore) r.FinalScore = blended r.Rating = blended ' out of 100End Sub Private Function EvaluateStatus(ByRef r As StudentRecord) As ReleaseStatus Dim variance As Double variance = Abs(r.InternalScore - r.ExternalScore) If r.Irregularity Then EvaluateStatus = HoldIrregularity: Exit Function End If If Not r.QACleared Then EvaluateStatus = HoldQANotCleared: Exit Function End If If r.BacklogCredits > 0 Then EvaluateStatus = HoldBacklog: Exit Function End If If variance > VARIANCE_THRESHOLD Then EvaluateStatus = ReviewVariance: Exit Function End If ' Recertification if severe fail on external or repeated attempts If r.ExternalScore < FAIL_LOW Or r.Attempt >= 3 Then EvaluateStatus = RecertificationRequired: Exit Function End If EvaluateStatus = EligibleForReleaseEnd Function **Code: combination/replace and award logic** vb Private Function BestOf(oldScore As Double, newScore As Double) As Double If newScore > oldScore Then BestOf = newScore Else BestOf = oldScore End IfEnd Function Private Function AwardText(ByVal rating As Double) As String If rating >= 85 Then AwardText = "Distinction" ElseIf rating >= 70 Then AwardText = "Merit" ElseIf rating >= PASS_THRESHOLD Then AwardText = "Pass" ElseIf rating >= CONDITIONAL_LOW Then AwardText = "Conditional Pass" ElseIf rating >= FAIL_LOW Then AwardText = "Fail" Else AwardText = "Severe Fail" End IfEnd Function **Code: button handlers** vb Private Sub cmdReconcile_Click() Dim r As StudentRecord r = ReadForm() ComputeScores r Dim status As ReleaseStatus status = EvaluateStatus(r) lblReleaseStatus.Caption = StatusToText(status) & " | Rating: " & Format(r.Rating, "0.00") & "% | Award: " & AwardText(r.Rating) End Sub Private Sub cmdEvaluate_Click() Call cmdReconcile_Click End Sub Private Sub cmdRelease_Click() Dim r As StudentRecord r = ReadForm() ComputeScores r Dim status As ReleaseStatus status = EvaluateStatus(r) If status <> EligibleForRelease Then MsgBox "Cannot release. Status: " & StatusToText(status), vbExclamation Exit Sub End If SaveRecord r, "Released" MsgBox "Final result released and certificate queued.", vbInformation End Sub Private Sub cmdReprint_Click() Dim r As StudentRecord r = ReadForm() PrintStatement r, True End Sub Private Sub cmdRecertify_Click() Dim r As StudentRecord r = ReadForm() SaveRecord r, "Recertification Required" MsgBox "Recertification case opened. QA Ref: " & r.QARef, vbInformation End Sub Private Sub cmdSave_Click() Dim r As StudentRecord r = ReadForm() ComputeScores r SaveRecord r, "Saved" MsgBox "Record saved.", vbInformation End Sub Private Function StatusToText(ByVal st As ReleaseStatus) As String Select Case st Case EligibleForRelease: StatusToText = "Eligible for Release" Case HoldIrregularity: StatusToText = "Hold - Irregularity" Case HoldBacklog: StatusToText = "Hold - Backlog" Case HoldQANotCleared: StatusToText = "Hold - QA/Insurance Not Cleared" Case ReviewVariance: StatusToText = "Hold - Internal/External Variance Review" Case RecertificationRequired: StatusToText = "Recertification Required" Case ReprintAllowed: StatusToText = "Reprint Allowed" Case Else: StatusToText = "Unknown" End Select End Function Code: saving, printing, and export Private Sub SaveRecord(ByRef r As StudentRecord, ByVal stateText As String) ' Example: save to worksheet "Results" Dim ws As Worksheet, nextRow As Long Set ws = ThisWorkbook.Worksheets("Results") nextRow = ws.Cells(ws.Rows.Count, "A").End(xlUp).Row + 1 ws.Cells(nextRow, 1).Value = r.StudentID ws.Cells(nextRow, 2).Value = r.Username ws.Cells(nextRow, 3).Value = r.Surname ws.Cells(nextRow, 4).Value = r.YOB ws.Cells(nextRow, 5).Value = r.AdminYear ws.Cells(nextRow, 6).Value = r.Programme ws.Cells(nextRow, 7).Value = r.Level ws.Cells(nextRow, 8).Value = r.Trade ws.Cells(nextRow, 9).Value = r.InternalScore ws.Cells(nextRow, 10).Value = r.ExternalScore ws.Cells(nextRow, 11).Value = r.FinalScore ws.Cells(nextRow, 12).Value = AwardText(r.Rating) ws.Cells(nextRow, 13).Value = r.ExamType ws.Cells(nextRow, 14).Value = r.Attempt ws.Cells(nextRow, 15).Value = r.BacklogCredits

ws.Cells(nextRow, 16).Value = r.CombineWithID ws.Cells(nextRow, 17).Value = IIf(r.Irregularity, "Yes", "No") ws.Cells(nextRow, 18).Value = IIf(r.QACleared, "Yes", "No") ws.Cells(nextRow, 19).Value = IIf(r.QACleared, "Yes", "No") ws.Cells(nextRow, 20).Value = r.QARef ws.Cells(nextRow, 21).Value = IIf(r.ProctorIssue, "Yes", "No") ws.Cells(nextRow, 22).Value = r.MaterialBatch ws.Cells(nextRow, 23).Value = stateText ws.Cells(nextRow, 24).Value = NowEnd Sub Private Sub PrintStatement(ByRef r As StudentRecord, Optional ByVal isReprint As Boolean = False) Dim txt As String, hdr As String hdr = IIf(isReprint, "REPRINTED RESULT STATEMENT", "RESULT STATEMENT") txt = hdr & vbCrLf & String(40, "-") & vbCrLf & _ "Student: " & r.Username & " " & r.Surname & " | ID: " & r.StudentID & vbCrLf & _ "YOB: " & r.YOB & " | Admin Year: " & r.AdminYear & vbCrLf & _ "Programme: " & r.Programme & " (L" & r.Level & ") | Trade: " & r.Trade & vbCrLf & vbCrLf & _ "Internal: " & Format(r.InternalScore, "0.0") & "/100" & vbCrLf & _ "External: " & Format(r.ExternalScore, "0.0") & "/100" & vbCrLf & _ "Final Rating: " & Format(r.Rating, "0.0") & "% | Award: " & AwardText(r.Rating) & vbCrLf & _ "Exam: " & r.ExamType & " | Attempt: " & r.Attempt & vbCrLf & _ "Backlog Credits: " & r.BacklogCredits & vbCrLf & _ "QA Cleared: " & IIf(r.QACleared, "Yes", "No") & " | QA Ref: " & r.QARef & vbCrLf & _ "Irregularity: " & IIf(r.Irregularity, "Yes", "No") & _ IIf(r.Irregularity, " (" & r.IrregularityNote & ")", "") & vbCrLf & _ "Material/Proctor Issue: " & IIf(r.ProctorIssue, "Yes", "No") & _ IIf(r.ProctorIssue, " (" & r.MaterialBatch & ")", "") ' Simple preview MsgBox txt, vbInformation, "Print Preview" ' Replace with: export to a formatted sheet and printEnd Sub **Optional: variance review and quality notes** Private Sub FlagVarianceNote(ByVal internalScore As Double, ByVal externalScore As Double) Dim variance As Double variance = Abs(internalScore - externalScore) If variance > VARIANCE_THRESHOLD Then txtIrregularityNote.Text = "Variance " & Format(variance, "0.0") & "pp exceeds threshold; send to moderation." End If End Sub Visual Basic framework for student portfolio clearance, attendance, finance, and printouts Below is a practical VBA/VB6-style scaffold to manage student records, portfolio availability by prior years, attendance, bursary and fee allocation, payroll-like study stipends, and printable statements. It also includes a simple logigram flow. **Userform structure** · **Form name:** frmClearance · **Tabs:** Identity | Portfolio | Attendance | Finance | Academics | Actions **Identity tab** · **TextBox:** txtStudentID, txtUsername, txtSurname, txtFirstName, txtPassword · **ComboBox:** cboProgramme (Engineering courses), cboCourseID, cboExamYear · **Labels:** lblStatus **Portfolio tab** · **CheckBox:** chkPortfolioAvailable · **TextBox:** txtPortfolioYears (comma-separated years, e.g., 2022,2023) · **ListBox:** lstArtifacts (research papers, lab reports, workshop models) · **CommandButton:** cmdAddArtifact, cmdRemoveArtifact **Attendance tab** · **TextBox:** txtDaysPresent4W, txtDaysPresent30D, txtDaysPresent360D · **TextBox:** txtDaysOff, txtSchoolDaysAvailable · **Labels:** lblAttendanceRate4W, lblAttendanceRate30D, lblAttendanceRate360D **Finance tab** · **Group: Stipend/Salary-like items** o **TextBox:** txtDailyRate (default 100) 'rand/day o **TextBox:** txtShiftDays, txtOffDays o **Labels:** lblGrossPay · **Group: Deductions** o **TextBox:** txtDeduction (generic), txtInsuranceLevy, txtPortalFee · **Group: Benefits/Allocations** o **TextBox:** txtBonus, txtAccommodation, txtLibraryFee, txtClassFee, txtAllocationPay, txtLearningGrant · **Labels:** lblNetPay **Academics tab** · **TextBox:** txtHomework, txtClasswork, txtPractical, txtExam, txtWorkshopModel, txtTradeLab, txtManufactureClaim, txtTenderValue, txtBudget · **Labels:** lblTotal100, lblRatingPct, lblAward **Actions tab** · **Buttons:** cmdCalculate, cmdPrintIdentity, cmdPrintAttendance, cmdPrintFinance, cmdPrintAcademics, cmdSave, cmdClear, cmdClose **Core data model and utilities** Option Explicit Private Type Student StudentID As String Username As String FirstName As String Surname As String Password As String Programme As String CourseID As String ExamYear As IntegerEnd Type Private Type Attendance DaysPresent4W As Double DaysPresent30D As Double DaysPresent360D As Double SchoolDaysAvailable As Double DaysOff As DoubleEnd Type Private Type Finance DailyRate As Double ShiftDays As Double OffDays As Double Deduction As Double InsuranceLevy As Double PortalFee As Double Bonus As Double Accommodation As Double LibraryFee As Double ClassFee As Double AllocationPay As Double LearningGrant As Double Gross As Double Net As DoubleEnd Type Private Type Academics Homework As Double Classwork As Double Practical As Double Exam As Double WorkshopModel As Double TradeLab As Double ManufactureClaim As Double TenderValue As Double Budget As Double Total100 As Double RatingPct As Double Award As StringEnd Type Private Const PASS50 As Double = 50#Private Const

COND40 As Double = 40#Private Const FAIL20 As Double = 20# **Form readers and calculators** Private Function ReadStudent() As Student (https://www.elektormagazine.com/) s.StudentID = Trim$(txtStudentID.Text) s.Username = Trim$(txtUsername.Text) s.FirstName = Trim$(txtFirstName.Text) s.Surname = Trim$(txtSurname.Text) s.Password = Trim$(txtPassword.Text) s.Programme = cboProgramme.Text s.CourseID = cboCourseID.Text s.ExamYear = Val(cboExamYear.Text) ReadStudent = s End Function Private Function ReadAttendance() As Attendance Dim a As Attendance a.DaysPresent4W = Val(txtDaysPresent4W.Text) a.DaysPresent30D = Val(txtDaysPresent30D.Text) a.DaysPresent360D = Val(txtDaysPresent360D.Text) a.SchoolDaysAvailable = Val(txtSchoolDaysAvailable.Text) a.DaysOff = Val(txtDaysOff.Text) ReadAttendance = a End Function Private Function ReadFinance() As Finance Dim f As Finance f.DailyRate = Val(txtDailyRate.Text) f.ShiftDays = Val(txtShiftDays.Text) f.OffDays = Val(txtOffDays.Text) f.Deduction = Val(txtDeduction.Text) f.InsuranceLevy = Val(txtInsuranceLevy.Text) f.PortalFee = Val(txtPortalFee.Text) f.Bonus = Val(txtBonus.Text) f.Accommodation = Val(txtAccommodation.Text) f.LibraryFee = Val(txtLibraryFee.Text) f.ClassFee = Val(txtClassFee.Text) f.AllocationPay = Val(txtAllocationPay.Text) f.LearningGrant = Val(txtLearningGrant.Text) ReadFinance = f End Function Private Function ReadAcademics() As Academics Dim ac As Academics ac.Homework = Val(txtHomework.Text) ac.Classwork = Val(txtClasswork.Text) ac.Practical = Val(txtPractical.Text) ac.Exam = Val(txtExam.Text) ac.WorkshopModel = Val(txtWorkshopModel.Text) ac.TradeLab = Val(txtTradeLab.Text) ac.ManufactureClaim = Val(txtManufactureClaim.Text) ac.TenderValue = Val(txtTenderValue.Text) ac.Budget = Val(txtBudget.Text) ReadAcademics = ac End Function Private Sub CalcAttendance(ByRef a As Attendance) If a.SchoolDaysAvailable <= 0 Then a.SchoolDaysAvailable = 360 lblAttendanceRate4W.Caption = Format(100 * a.DaysPresent4W / 20, "0.0") & "%" lblAttendanceRate30D.Caption = Format(100 * a.DaysPresent30D / 30, "0.0") & "%" lblAttendanceRate360D.Caption = Format(100 * a.DaysPresent360D / a.SchoolDaysAvailable, "0.0") & "%" End Sub Private Sub CalcFinance(ByRef f As Finance) f.Gross = f.DailyRate * f.ShiftDays Dim totalDeductions As Double totalDeductions = f.Deduction + f.InsuranceLevy + f.PortalFee + f.LibraryFee + f.ClassFee Dim totalBenefits As Double totalBenefits = f.Bonus + f.Accommodation + f.AllocationPay + f.LearningGrant f.Net = f.Gross - totalDeductions + totalBenefits lblGrossPay.Caption = "R " & Format(f.Gross, "0,0.00") lblNetPay.Caption = "R " & Format(f.Net, "0,0.00") End Sub Private Sub CalcAcademics(ByRef ac As Academics) ' Normalize to 100: Homework(15) + Classwork(15) + Practical(20) + Exam(50) Dim total As Double total = ac.Homework + ac.Classwork + ac.Practical + ac.Exam ac.Total100 = total ac.RatingPct = total ' already out of 100 if inputs constrained ac.Award = AwardFromPct(ac.RatingPct) lblTotal100.Caption = Format(ac.Total100, "0.0") & " / 100" lblRatingPct.Caption = Format(ac.RatingPct, "0.0") & "%" lblAward.Caption = ac.Award End Sub Private Function AwardFromPct(ByVal pct As Double) As String If pct >= 85 Then AwardFromPct = "Distinction" ElseIf pct >= 70 Then AwardFromPct = "Merit" ElseIf pct >= PASS50 Then AwardFromPct = "Pass" ElseIf pct >= COND40 Then AwardFromPct = "Borderline" ElseIf pct >= FAIL20 Then AwardFromPct = "Fail" Else AwardFromPct = "Severe Fail" End If End Function Private Sub cmdCalculate_Click() Dim a As Attendance, f As Finance, ac As Academics a = ReadAttendance(): Call CalcAttendance(a) f = ReadFinance(): Call CalcFinance(f) ac = ReadAcademics(): Call CalcAcademics(ac) lblStatus.Caption = "Calculated at " & Format(Now, "yyyy-mm-dd hh:nn") End Sub Private Sub cmdClear_Click() Dim ctl As Control For Each ctl In Me.Controls Select Case TypeName(ctl) Case "TextBox": ctl.Text = "" Case "Label" If ctl.Name Like "lbl*" Then ctl.Caption = "" End Select Next ctl chkPortfolioAvailable.Value = False lstArtifacts.Clear lblStatus.Caption = "Cleared" End Sub Private Sub cmdSave_Click() Dim s As Student, a As Attendance, f As Finance, ac As Academics s = ReadStudent(): a = ReadAttendance(): f = ReadFinance(): ac = ReadAcademics() SaveToSheet s, a, f, ac lblStatus.Caption = "Saved at " & Format(Now, "yyyy-mm-dd hh:nn") End Sub Private Sub cmdPrintIdentity_Click() Dim s As Student: s = ReadStudent() Dim txt As String txt = "STUDENT IDENTITY" & vbCrLf & String(40, "-") & vbCrLf & _ "ID: " & s.StudentID & vbCrLf & _ "Name: " & s.FirstName & " " & s.Surname & vbCrLf & _ "Username: " & s.Username & vbCrLf & _ "Programme: " & s.Programme & " | Course ID: " &

s.CourseID & vbCrLf & _ "Exam Year: " & s.ExamYear MsgBox txt, vbInformation, "Print Preview" End Sub Private Sub cmdPrintAttendance_Click() Dim a As Attendance: a = ReadAttendance() Dim txt As String txt = "ATTENDANCE SUMMARY" & vbCrLf & String(40, "-") & vbCrLf & _ "4 Weeks Present: " & a.DaysPresent4W & " (" & lblAttendanceRate4W.Caption & ")" & vbCrLf & _ "30 Days Present: " & a.DaysPresent30D & " (" & lblAttendanceRate30D.Caption & ")" & vbCrLf & _ "360 Days Present: " & a.DaysPresent360D & " (" & lblAttendanceRate360D.Caption & ")" & vbCrLf & _ "Days Off: " & a.DaysOff & " | School Days: " & a.SchoolDaysAvailable MsgBox txt, vbInformation, "Print Preview" End Sub Private Sub cmdPrintFinance_Click() Dim f As Finance: f = ReadFinance(): Call CalcFinance(f) Dim txt As String txt = "FINANCE SUMMARY" & vbCrLf & String(40, "-") & vbCrLf & _ "Daily Rate: R " & Format(f.DailyRate, "0,0.00") & vbCrLf & _ "Shift Days: " & f.ShiftDays & " | Off Days: " & f.OffDays & vbCrLf & _ "Gross: " & lblGrossPay.Caption & vbCrLf & _ "Deductions (incl. insurance/portal/library/class): R " & _ Format(f.Deduction + f.InsuranceLevy + f.PortalFee + Val(txtLibraryFee.Text) + Val(txtClassFee.Text), "0,0.00") & vbCrLf & _ "Benefits (bonus/accommodation/allocation/grant): R " & _ Format(f.Bonus + f.Accommodation + f.AllocationPay + f.LearningGrant, "0,0.00") & vbCrLf & _ "Net: " & lblNetPay.Caption MsgBox txt, vbInformation, "Print Preview" End Sub Private Sub cmdPrintAcademics_Click() Dim ac As Academics: ac = ReadAcademics(): Call CalcAcademics(ac) Dim txt As String txt = "ACADEMIC MARKSHEET" & vbCrLf & String(40, "-") & vbCrLf & _ "Homework: " & ac.Homework & "/15" & vbCrLf & _ "Classwork: " & ac.Classwork & "/15" & vbCrLf & _ "Practical/Lab: " & ac.Practical & "/20" & vbCrLf & _ "Exam: " & ac.Exam & "/50" & vbCrLf & _ "Total: " & lblTotal100.Caption & " | Rating: " & lblRatingPct.Caption & vbCrLf & _ "Award: " & lblAward.Caption & vbCrLf & _ "Workshop Model: " & ac.WorkshopModel & " | Trade Lab: " & ac.TradeLab & vbCrLf & _ "Manufacture Claim: " & ac.ManufactureClaim & " | Tender Value: " & ac.TenderValue & vbCrLf & _ "Budget: " & ac.Budget MsgBox txt, vbInformation, "Print Preview" End Sub Private Sub SaveToSheet(ByVal s As Student, ByVal a As Attendance, ByVal f As Finance, ByVal ac As Academics) Dim ws As Worksheet, r As Long Set ws = ThisWorkbook.Worksheets("ClearanceRecords") r = ws.Cells(ws.Rows.Count, "A").End(xlUp).Row + 1 ws.Cells(r, 1).Value = s.StudentID ws.Cells(r, 2).Value = s.Username ws.Cells(r, 3).Value = s.FirstName ws.Cells(r, 4).Value = s.Surname ws.Cells(r, 5).Value = s.Programme ws.Cells(r, 6).Value = s.CourseID ws.Cells(r, 7).Value = s.ExamYear ws.Cells(r, 8).Value = a.DaysPresent4W ws.Cells(r, 9).Value = a.DaysPresent30D ws.Cells(r, 10).Value = a.DaysPresent360D ws.Cells(r, 11).Value = a.SchoolDaysAvailable ws.Cells(r, 12).Value = a.DaysOff ws.Cells(r, 13).Value = f.DailyRate ws.Cells(r, 14).Value = f.ShiftDays ws.Cells(r, 15).Value = f.OffDays ws.Cells(r, 16).Value = f.Gross ws.Cells(r, 17).Value = f.Deduction ws.Cells(r, 18).Value = f.InsuranceLevy ws.Cells(r, 19).Value = f.PortalFee ws.Cells(r, 20).Value = f.Bonus ws.Cells(r, 21).Value = f.Accommodation ws.Cells(r, 22).Value = f.LibraryFee ws.Cells(r, 23).Value = f.ClassFee ws.Cells(r, 24).Value = f.AllocationPay ws.Cells(r, 25).Value = f.LearningGrant ws.Cells(r, 26).Value = f.Net ws.Cells(r, 27).Value = ac.Homework ws.Cells(r, 28).Value = ac.Classwork ws.Cells(r, 29).Value = ac.Practical ws.Cells(r, 30).Value = ac.Exam ws.Cells(r, 31).Value = ac.Total100 ws.Cells(r, 32).Value = ac.RatingPct ws.Cells(r, 33).Value = ac.Award ws.Cells(r, 34).Value = NowEnd Sub **Minimal module to show the form** Sub ShowClearance() frmClearance.Show End Sub **Logigram overview** · **Start** · **Identity validation** o **If missing fields:** Halt → Prompt to complete · **Portfolio check** o **If available:** Load artifacts → Proceed o **If not:** Flag “Pending Portfolio” · **Attendance calculation** o **If < 75% (360-day rate):** Flag “Attendance Risk” · **Finance calculation** o **Compute gross → apply deductions → add allocations → net** · **Academics aggregation** o **Compute total/100 → rating → award** · **Clearance gate** o **If award Pass and attendance ≥ 75%:** Clearance Approved o **Else:** Clearance Conditional/Pending · **Save + Print statements** o Identity | Attendance | Finance | Academics · **End Notes and next steps** · **Field caps:** For academic inputs, constrain to their maximums (e.g., Homework ≤ 15) to preserve 100 total. · **Attendance presets:** 4 weeks = 20 days; 30 days; year = 360 days; adjust if your calendar differs. · **Currency:** Values shown in R (rand). Adapt rates and fee items as needed. · **Data protection:** Avoid storing plaintext passwords; use masked fields or exclude from saving. · **Enhancements:** Export to a formatted sheet/PDF with institutional branding and a QR for verification. If you share your exact

environment (Excel VBA vs. VB6 vs. VB.NET (http://VB.NET)), I'll tailor control properties, add input validation [illegible] print-ready certificate. Here's a structured **Visual Basic (VBA)** framework tailored to your advanced certification and qualification logic, aligned with **SAQA**, **UCPD/UCD**, and **trade-based assessment systems**. This model supports: Certificate granting based on test/class performanceReplacement or exemption of marksSAQA qualification mappingFinal job evaluation and employment scoringPrint-ready certificate and diploma logicMulti-phase award logic (1st–4th phase)Degree, diploma, and postgraduate recognition 📜 **UserForm Structure: frmCertification** 🔷 **Identity & Qualification Tab**

| **Control** | **Name** | **Purpose** | TextBox | txtStudentID | Unique learner ID
| TextBox | txtStudentName | Full name
| TextBox | txtSurname | Surname
| TextBox | txtLogin | System login
| TextBox | txtPassword | Masked password
| ComboBox | cboTrade | Trade (e.g., Electrical, Mechanical)
| ComboBox | cboQualificationType | NDip, BTech, UCPD, UCD, Master, Doctoral
| TextBox | txtSAQAID | SAQA Qualification ID
| TextBox | txtQualificationID | Internal Qualification ID
| ComboBox | cboAssessor | Assigned assessor
| ComboBox | cboPhase | Final Phase (1st–4th) 🔷 **Assessment & Exemption Tab**
| **Field** | **Controls** | Subject Name | txtSubjectName
| Course ID | txtCourseID
| Test Score | txtTestScore
| Exam Score | txtExamScore
| Exempted | chkExempted
| Replacement Score | txtReplacementScore
| Minimum Required | txtMinMark
| Maximum Allowed | txtMaxMark
| Meets Requirement | lblMeetsRequirement (calculated)
| Award Status | lblAwardStatus (calculated) 🔷 **Employment & Job Evaluation Tab**
| **Field** | **Controls** | Job Function | txtJobFunction
| Log Activity | lstActivityLog
| Employment Duration | txtYearsWorked (e.g., 2 years)
| Working Days | txtDaysWorked (e.g., 30 days)
| Final Score | lblFinalScore (calculated)
| Employment Award | lblEmploymentAward (calculated) 🔷 **Certificate & Diploma Tab**
| **Button** | **Function** | cmdPrintCertificate | Print SAQA Certificate
| cmdPrintDiploma | Print SAQA Diploma
| cmdEvaluateAward | Evaluate qualification and award
| cmdSaveRecord | Save to sheet
| cmdClearForm | Reset form
| cmdCloseForm | Exit 📊 **Core Logic: Award Evaluation** vb Private Sub cmdEvaluateAward_Click() Dim testScore As Double, examScore As Double, replacementScore As Double Dim exempted As Boolean, minMark As Double, maxMark As Double Dim finalScore As Double, meetsRequirement As Boolean testScore = Val(txtTestScore.Text) examScore = Val(txtExamScore.Text) replacementScore = Val(txtReplacementScore.Text) exempted = chkExempted.Value minMark = Val(txtMinMark.Text) maxMark = Val(txtMaxMark.Text) If exempted Then finalScore = replacementScore Else finalScore = (testScore + examScore) / 2 End If lblFinalScore.Caption = Format(finalScore, "0.0") If finalScore >= minMark And finalScore <= maxMark Then lblMeetsRequirement.Caption = "Yes" lblAwardStatus.Caption = "Eligible for Certificate" Else lblMeetsRequirement.Caption = "No" lblAwardStatus.Caption = "Not Eligible" End If End Sub 🖨️ **Certificate & Diploma Print Logic** vb Private Sub cmdPrintCertificate_Click() Dim txt As String txt = "SAQA CERTIFICATE OF COMPETENCE" & vbCrLf

& String(40, "-") & vbCrLf & _ "Student: " & txtStudentName.Text & " " & txtSurname.Text & vbCrLf & _ "Trade: " & cboTrade.Text & vbCrLf & _ "Qualification Type: " & cboQualificationType.Text & vbCrLf & _ "SAQA ID: " & txtSAQAID.Text & " | Internal ID: " & txtQualificationID.Text & vbCrLf & _ "Assessor: " & cboAssessor.Text & " | Phase: " & cboPhase.Text & vbCrLf & _ "Final Score: " & lblFinalScore.Caption & " | Award Status: " & lblAwardStatus.Caption MsgBox txt, vbInformation, "Certificate Preview"End Sub Private Sub cmdPrintDiploma_Click() Dim txt As String txt = "SAQA DIPLOMA STATEMENT" & vbCrLf & String(40, "-") & vbCrLf & _ "Course: " & txtSubjectName.Text & " | Course ID: " & txtCourseID.Text & vbCrLf & _ "Test: " & txtTestScore.Text & " | Exam: " & txtExamScore.Text & vbCrLf & _ "Exempted: " & IIf(chkExempted.Value, "Yes", "No") & _ IIf(chkExempted.Value, " | Replacement: " & txtReplacementScore.Text, "") & vbCrLf & _ "Final Score: " & lblFinalScore.Caption & " | Meets Requirement: " & lblMeetsRequirement.Caption MsgBox txt, vbInformation, "Diploma Preview"End Sub 🧠 **Employment Score Logic** vb Private Sub EvaluateEmploymentScore() Dim yearsWorked As Double, daysWorked As Double Dim employmentScore As Double yearsWorked = Val(txtYearsWorked.Text) daysWorked = Val(txtDaysWorked.Text) employmentScore = (yearsWorked * 360 + daysWorked) / 720 ' Normalize to 2-year scale lblEmploymentAward.Caption = IIf(employmentScore >= 1, "Eligible for Job Certification", "Insufficient Duration") End Sub [Start] ↓[Identity Verified] ↓[Assessment Complete?] → No → [Hold] ↓ Yes[Exempted or Replaced?] → Yes → [Use Replacement Score] ↓ No[Calculate Final Score] ↓[Meets Min/Max?] → No → [Reject] ↓ Yes[Employment Duration ≥ 2 years?] → No → [Issue Statement Only] ↓ Yes[Print Certificate + Diploma] ↓[Save Record] ↓[End] 🧩 **Optional Enhancements** · **SAQA XML export:** For qualification registry submission · **QR code on certificate:** For verification · **Assessor dashboard:** Track phase completions and exemptions · **UCPD/UCD logic:** Map development capacity to trade replacement pathways · **Project-based override:** If no exam but project completed, allow award with assessor validation Would you like this scaffolded into a full Excel VBA workbook with certificate templates, logigram visuals, and SAQA-aligned qualification sheets? I can also help you build exemption logic for trade replacement and automate assessor workflows.

On Thu, Sep 18, 2025 at 2:36 PM tshingombe fiston <tshingombefiston@gmail.com (mailto:tshingombefiston@gmail.com)> wrote:
Absolutely, Tshingombe. Here's a structured **Visual Basic (VBA)** framework for a **National Diploma Examination Result Statement Generator** tailored to engineering studies. It includes:Student info fieldsCourse modules across terms and semestersAssessment breakdown (homework, classwork, exams)Score scaling and qualification levelsPrint-ready marksheet logicAward and pass/fail classification 📝 UserForm Layout Overview🔷 Student Info Section**Control TypeNameCaption**Label | lblStudentName | Student Name
TextBox | txtStudentName | —
Label | lblSurname | Surname
TextBox | txtSurname | —
Label | lblAchievementYear | Achievement Year
TextBox | txtAchievementYear | —
Label | lblExperienceYears | Work Experience (Years)
TextBox | txtExperienceYears | —🔷 Academic Record SectionUse a **MultiPage** or **TabStrip** to organize:📘 Page 1: Terms & Semesters**TermControls**Term 1–4 | txtTerm1, txtTerm2, txtTerm3, txtTerm4
Semester 1–2 | txtSem1, txtSem2📘 Page 2: Course Modules & Assessment**FieldControls**Course Topics | lstCourseTopics (ListBox or ComboBox)
Homework | txtHomeworkScore
Classwork | txtClassworkScore
Exams | txtExamScore
Total Score | lblTotalScore (calculated)
Rating (%) | lblRating (calculated)🔷 Qualification & Scaling**FieldControls**Final Qualification | cboQualification (e.g., 1st, 2nd, 3rd, 4th)
Level | cboLevel (1–9)

Course Weight | txtCourseWeight
Scaling Factor | txtScalingFactor
Final Score | lblFinalScore (calculated) 🔷 Buttons**ButtonFunction**cmdCalculate | Compute total, rating, qualification
cmdPrint | Print formatted marksheet
cmdClear | Reset form
cmdExit | Close form 🟥 Calculation Logic (VBA) Private Sub cmdCalculate_Click()

```
    Dim homework As Double, classwork As Double, exam As Double
    Dim total As Double, rating As Double, scaledScore As Double
    Dim weight As Double, scaleFactor As Double

    ' Get scores
    homework = Val(txtHomeworkScore.Text)
    classwork = Val(txtClassworkScore.Text)
    exam = Val(txtExamScore.Text)

    ' Compute total
    total = homework + classwork + exam
    lblTotalScore.Caption = total & " / 100"

    ' Rating
    rating = (total / 100) * 100
    lblRating.Caption = Format(rating, "0.00") & "%"

    ' Scaling
    weight = Val(txtCourseWeight.Text)
    scaleFactor = Val(txtScalingFactor.Text)
    scaledScore = total * (weight / 100) * scaleFactor
    lblFinalScore.Caption = Format(scaledScore, "0.00")

    ' Qualification logic
    Select Case rating
        Case Is >= 50
            cboQualification.Text = "Pass"
        Case 40 To 49
            cboQualification.Text = "Conditional Pass"
        Case 20 To 39
            cboQualification.Text = "Fail"
        Case Is < 20
            cboQualification.Text = "Severe Fail"
    End Select
```

🖨️ Print Statement Logic

```
Private Sub cmdPrint_Click()
    Dim msg As String
    msg = "National Diploma Result Statement" & vbCrLf & vbCrLf
    msg = msg & "Student: " & txtStudentName.Text & " " & txtSurname.Text & vbCrLf
    msg = msg & "Achievement Year: " & txtAchievementYear.Text & vbCrLf
    msg = msg & "Experience: " & txtExperienceYears.Text & " years" & vbCrLf & vbCrLf

    msg = msg & "Course Topics: " & lstCourseTopics.Text & vbCrLf
```

```
    msg = msg & "Homework: " & txtHomeworkScore.Text & vbCrLf
    msg = msg & "Classwork: " & txtClassworkScore.Text & vbCrLf
    msg = msg & "Exam: " & txtExamScore.Text & vbCrLf
    msg = msg & "Total Score: " & lblTotalScore.Caption & vbCrLf
    msg = msg & "Rating: " & lblRating.Caption & vbCrLf
    msg = msg & "Final Qualification: " & cboQualification.Text & vbCrLf
    msg = msg & "Level: " & cboLevel.Text & vbCrLf
    msg = msg & "Final Scaled Score: " & lblFinalScore.Caption & vbCrLf

    MsgBox msg, vbInformation, "Print Preview"
    ' Replace with actual print logic if needed
End Sub
🏅 Award Logic (Optional)
If rating >= 85 Then
    MsgBox "Student qualifies for Distinction Award!", vbInformation
ElseIf rating >= 70 Then
    MsgBox "Student qualifies for Merit Award.", vbInformation
ElseIf rating >= 50 Then
    MsgBox "Student passed successfully.", vbInformation
Else
    MsgBox "Student did not meet pass criteria.", vbExclamation
End If
Would you like this scaffolded into a full UserForm design with tabbed layout and export-ready formatting?
I can also help you build logigram logic for qualification flow or automate Excel-based marksheet
generation.

End Sub

On Thu, Sep 18, 2025 at 2:23 PM TSHINGOMBEKB TSHITADI <tshingombekb@gmail.com
(mailto:tshingombekb@gmail.com)> wrote:
Visual Basic userform for a registration page with tabsBelow is a complete, ready-to-use VBA/VB6-style
UserForm pattern with labels, a frame, textboxes, gender options, password field, and OK/Cancel/Next
buttons. It uses a MultiPage (tabs) to organize “Personal Info” and “Account” pages.Layout and control
namingForm name: frmRegistrationMultiPage: MultiPage1Page(0) Caption: Personal InfoPage(1) Caption:
AccountPage(0) controls (inside a Frame):Frame: fraPersonal (Caption: Personal details)Label:
lblFirstName (Caption: First name)TextBox: txtFirstNameLabel: lblSurname (Caption: Surname)TextBox:
txtSurnameLabel: lblDOB (Caption: Birth date)TextBox: txtDOB (e.g., YYYY-MM-DD)Label: lblGender
(Caption: Gender)OptionButton: optMale (Caption: Male)OptionButton: optFemale (Caption:
Female)Page(1) controls:Label: lblUsername (Caption: Username)TextBox: txtUsernameLabel:
lblPassword (Caption: Password)TextBox: txtPassword (PasswordChar = •)Buttons (place at bottom of the
form, visible on all tabs):CommandButton: cmdOK (Caption: OK)CommandButton: cmdCancel (Caption:
Cancel)CommandButton: cmdNext (Caption: Next »)Tip: If available in your environment, you can use a
DatePicker instead of txtDOB (e.g., DTPicker). Otherwise, keep a textbox and validate the date.Suggested
tab orderPersonal Info page: txtFirstName → txtSurname → txtDOB → optMale → optFemaleAccount
page: txtUsername → txtPasswordForm-level: cmdNext → cmdOK → cmdCancelUserForm codeCopy this
into the frmRegistration code-behind.vb Option Explicit

Private Sub UserForm_Initialize()
    ' Initialize defaults
    Me.MultiPage1.Value = 0 ' First tab
    Me.optMale.Value = False
```

```
    Me.optFemale.Value = False
    Me.txtPasswordVal = ""
End Sub

Private Sub cmdNext_Click()
    ' Toggle between tabs
    If Me.MultiPage1.Value < Me.MultiPage1.Pages.Count - 1 Then
        Me.MultiPage1.Value = Me.MultiPage1.Value + 1
    Else
        Me.MultiPage1.Value = 0
    End If
End Sub

Private Sub cmdCancel_Click()
    If MsgBox("Cancel registration?", vbQuestion + vbYesNo, "Confirm") = vbYes Then
        Unload Me
    End If
End Sub

Private Sub cmdOK_Click()
    Dim errMsg As String
    If Not ValidateInputs(errMsg) Then
        MsgBox errMsg, vbExclamation, "Validation"
        Exit Sub
    End If

    ' Simulated save; replace with your persistence logic
    ' e.g., write to worksheet/database/API
    ' Example (Excel): WriteToSheet

    MsgBox "Registration successful.", vbInformation, "Success"
    Unload Me
End Sub

Private Function ValidateInputs(ByRef errMsg As String) As Boolean
    Dim dt As Date
    Dim genderSelected As Boolean

    ' Basic required fields
    If Trim$(Me.txtFirstName.Text) = "" Then
        errMsg = "First name is required."
        ValidateInputs = False
        Exit Function
    End If

    If Trim$(Me.txtSurname.Text) = "" Then
        errMsg = "Surname is required."
        ValidateInputs = False
        Exit Function
    End If
```

```
    If Trim$(Me.txtDOB.Text) = "" Then
        errMsg = "Birth date is required (YYYY-MM-DD)."
        ValidateInputs = False
        Exit Function
    End If

    ' Date validation (expects a valid date; adjust to your locale/format)
    On Error GoTo BadDate
    dt = CDate(Me.txtDOB.Text)
    On Error GoTo 0

    If dt > Date Then
        errMsg = "Birth date cannot be in the future."
        ValidateInputs = False
        Exit Function
    End If

    ' Gender
    genderSelected = (Me.optMale.Value Or Me.optFemale.Value)
    If Not genderSelected Then
        errMsg = "Please select a gender."
        ValidateInputs = False
        Exit Function
    End If

    ' Account page checks
    If Trim$(Me.txtUsername.Text) = "" Then
        errMsg = "Username is required."
        ValidateInputs = False
        Exit Function
    End If

    If Len(Me.txtPassword.Text) < 6 Then
        errMsg = "Password must be at least 6 characters."
        ValidateInputs = False
        Exit Function
    End If

    ValidateInputs = True
    Exit Function

BadDate:
    errMsg = "Invalid birth date. Use a valid date (e.g., 2001-05-17)."
    ValidateInputs = False
End Function

' Optional: Excel example of saving to a sheet
Private Sub WriteToSheet()
    Dim ws As Worksheet
    Dim nextRow As Long
    Dim gender As String
```

```
    Set ws = ThisWorkbook.Worksheets("Registrations")
    If Me.optMale.Value Then
        gender = "Male"
    ElseIf Me.optFemale.Value Then
        gender = "Female"
    Else
        gender = ""
    End If

    nextRow = ws.Cells(ws.Rows.Count, "A").End(xlUp).Row + 1
    With ws
        .Cells(nextRow, 1).Value = Me.txtFirstName.Text
        .Cells(nextRow, 2).Value = Me.txtSurname.Text
        .Cells(nextRow, 3).Value = Me.txtDOB.Text
        .Cells(nextRow, 4).Value = gender
        .Cells(nextRow, 5).Value = Me.txtUsername.Text
        .Cells(nextRow, 6).Value = MaskPassword(Me.txtPassword.Text) ' Store hashed/hidden if possible
        .Cells(nextRow, 7).Value = Now
    End With
End Sub

Private Function MaskPassword(ByVal pwd As String) As String
    ' Placeholder for hashing/encryption
    MaskPassword = String$(Len(pwd), "*")
End FunctionSub ShowRegistration()
    frmRegistration.Show
End Sub Quick build stepsAdd a UserForm: Insert → UserForm → Rename to frmRegistration.Add a MultiPage: Insert MultiPage → Name: MultiPage1 → Set Captions to “Personal Info” and “Account”.Add a Frame to Page(0): Name: fraPersonal → Caption: Personal details.Add labels/textboxes:lblFirstName + txtFirstNamelblSurname + txtSurnamelblDOB + txtDOBlblGender + optMale + optFemaleAdd Account page fields: lblUsername + txtUsername, lblPassword + txtPassword (set PasswordChar = •).Add buttons: cmdOK, cmdCancel, cmdNext; set their captions.Paste code: Into frmRegistration code-behind and optional module.Run: Call ShowRegistration.Optional enhancementsInput masks: Enforce date format with a simple mask or a calendar control.Gender options: Add “Other/Prefer not to say.”Validation UI: Highlight invalid fields by changing BackColor.Persistence: Replace WriteToSheet with a database call or JSON file write.Navigation: Disable cmdOK until all validations pass.If you want, tell me your target environment (Excel VBA, Access, VB6, or VB.NET (http://VB.NET) WinForms), and I’ll tailor the exact control names and code syntax accordingly.
```

On Thu, Sep 18, 2025 at 12:49 PM TSHINGOMBEKB TSHITADI <tshingombekb@gmail.com (mailto:tshingombekb@gmail.com)> wrote:

On Wed, Sep 17, 2025 at 3:13 PM TSHINGOMBEKB TSHITADI <tshingombekb@gmail.com (mailto:tshingombekb@gmail.com)> wrote:

On Tue, Sep 16, 2025 at 4:16 PM TSHINGOMBEKB TSHITADI <tshingombekb@gmail.com (mailto:tshingombekb@gmail.com)> wrote:

On Tue, Sep 16, 2025 at 4:13 PM TSHINGOMBEKB TSHITADI <tshingombekb@gmail.com (mailto:tshingombekb@gmail.com)> wrote:

On Tue, Sep 16, 2025 at 1:20 PM TSHINGOMBEKB TSHITADI <tshingombekb@gmail.com (mailto:tshingombekb@gmail.com)> wrote:

On Mon, Sep 15, 2025 at 1:42 PM TSHINGOMBEKB TSHITADI <tshingombekb@gmail.com (mailto:tshingombekb@gmail.com)> wrote:

On Fri, Sep 12, 2025 at 4:05 PM TSHINGOMBEKB TSHITADI <tshingombekb@gmail.com (mailto:tshingombekb@gmail.com)> wrote:

On Fri, Sep 12, 2025 at 3:00 PM TSHINGOMBEKB TSHITADI <tshingombekb@gmail.com (mailto:tshingombekb@gmail.com)> wrote:

On Fri, Sep 12, 2025 at 10:43 AM TSHINGOMBEKB TSHITADI <tshingombekb@gmail.com (mailto:tshingombekb@gmail.com)> wrote:

(https://mail.google.com/mail/u/1/?ui=2&ik=ab45b3cb6f&view=att&th=199768ab05f4f78e&attid=0.1&disp=attd&realattid=f_mfwj4z510&safe=1&z
| | **Career10f discovery job trade memo lecture learn note.docx**
202K View as HTML (https://mail.google.com/mail/u/1/?ui=2&ik=ab45b3cb6f&view=att&th=199768ab05f4f78e&attid=0.1&disp=vah&realattid=f_mfwj4z510&safe=1&z
Scan and download (https://mail.google.com/mail/u/1/?ui=2&ik=ab45b3cb6f&view=att&th=199768ab05f4f78e&attid=0.1&disp=attd&realattid=f_mfwj4z510&safe=1&z

Contest

Elektor Helps: Projects

Badge

SOS

Elektor Jumpstarter allows testing the viability of a project before producing it and putting it as a product in the Elektor Store.

If you want to try it, mark it here as Elektor Jumpstarter. The project will then be evaluated by Elektor. If it is accepted, we will determine a price and a minimum amount that must be sold for the project to be viable, and a backing campaign will be launched. If the project attracts enough supporters, the project will move into production.

First time? Our staff approves every first project before publication.

labs.project-elements

Please upload your attachments (additional photos, schematics, PCB's, software, etc.) here. Allowed file types include PNG, JPG, GIF, ZIP, RAR, TAR and PDF. Maximum file size is 5 MB.

form INTELL 90000 , FORM.pdf2.pdf (PDF2, 176kb)

form INTELL 90000 , FORM.pdf (PDF, 164kb)

Save Project