PROJECT 1 OF 5 | START HERE LEARNER GUIDE

Identity Lifecycle & Access Governance

Engineer a control

Harborline Health is heading into SOC 2 testing. Leadership wants to know whether identity lifecycle and access governance actually work, not whether a policy PDF exists. You are the GRC practitioner asked to assess and improve that environment: you will build a JML Control Monitor, investigate an incident, and recommend what to fund.

> **Companion video**
> Watch the walkthrough before you begin: https://youtu.be/NpNsjtArHZg

Your project folder

Item	What it is for
START_HERE_Learner_Guide.pdf	This guide. Read it stage by stage.
Master_Workbook.xlsx	Your main workspace. Every stage has a tab.
1_Assess_Evidence	Current-state evidence for Stage 1.
3_Build_Source_Data	HR and IdP extracts, the stated control brief, and the data dictionary for Stage 3.
4_Break_Incident_Evidence_Open_After_Build	Incident evidence for Stage 4. Do not open until Build is complete.
Deliverable_Templates	Two Word templates: the incident analysis (Stage 4) and the management summary (Stage 6).
Optional_Python_Path	Optional. The Python version of the Build stage.

Contents

Welcome to Project 1 . . . 3
Stage 1: Assess . . . 6
Stage 2: Design . . . 8
Stage 3: Build . . . 10
Stage 4: Break . . . 14
Stage 5: Improve . . . 16
Stage 6: Defend . . . 17
Stage 7: Career Translation . . . 18
Before you call this finished . . . 19
Optional: the Python Build path . . . 21

Welcome to Project 1

Read this once before Stage 1. It takes about ten minutes.

The scenario

Harborline Health is a fictional mid-size digital health company. It runs a clinic-operations platform and a patient portal and is preparing for a SOC 2 Type II examination. Its identity lifecycle looks documented. Leadership is not sure it operates, and is not sure the current design is even the right design.

Your job is to **assess** the environment, **design** an improved control, **build** a JML Control Monitor, **break** it by investigating a live incident, **improve** the design, **defend** a recommendation a CISO or CIO can decide on, and then translate the work into an honest career story.

> **This is a GRC engineering project**
> You engineer the governance and control solution: process, design, monitoring, exception handling, and recommendation. You do not build Harborline's identity platform, connect to a real Okta tenant, or deploy anything to the cloud.

Words you need first

Term	Meaning in this project
JML	Joiner, mover, leaver: hire or contract start, role change, and termination or contract end.
IdP	Identity provider. The central login system (an Okta-style export in this project) that holds accounts and entitlements.
Entitlement	A specific permission inside an application, for example `github_admin`.
SLA	Service level agreement: the maximum time the control is allowed. 24 hours to revoke a leaver; 7 days to remove a mover's leftover access.
Exception	A case where the control did **not** operate as designed. This is a finding.
Watch item	Something that looks unfinished but is still inside the allowed window, for example a leaver still active 8 hours after termination. A watch item is **not** a failure.
True negative	A case that should not be flagged, such as a leaver whose account is already disabled. Getting these right matters as much as catching failures.
RCM	Risk and control matrix: a table mapping risks to the controls that address them.
Review freeze	The fixed timestamp all math uses: 2026-08-15 12:00 UTC. Never use your computer clock.

Choose your depth

Option	Time	What it includes
Standard	About 4 to 6 hours	The full seven-stage loop, a working monitor, and every portfolio-critical artifact. This is complete, legitimate completion.
Deep Dive	About 8 to 12 hours	Everything in Standard plus extra Assess rows, richer design notes, the Python challenge prompts or a second Build path, and full interview defense practice.

Deep Dive is more practice, not a higher grade. Items marked **Deep Dive (optional)** in this guide can be skipped on the Standard path.

Choose your Build path

Path	Tool	Completes Build?
Spreadsheet	The Build tabs in **Master_Workbook.xlsx** (Excel or Google Sheets)	Yes. Complete, not a beginner-only option.
Python	Scripts in **Optional_Python_Path** (Python 3.10 or newer, no installs)	Yes. Recommended if you want coding practice.

You need one of the two, not both. Your choice does not change the maximum quality of your work. Either way, the rest of the project (Assess, Design, Break, Improve, Defend, and Career Translation) happens in the workbook and the two Word templates.

A five-day plan (optional)

Day	Stages	Time
1	Assess	45 to 75 minutes
2	Design	60 to 90 minutes
3	Build (the long day)	90 to 150 minutes
4	Break and Improve	45 to 75 minutes
5	Defend and Career Translation	60 to 90 minutes

If a day overruns, finish that stage before starting the next. Never open the Break evidence before you have a working monitor.

What you will produce

#	Artifact	Where it lives
1	Current-state assessment (at least eight labeled rows plus a two-sentence summary)	**1 Assess**
2	JML process map	**2A Process Map**
3	Risk and control matrix (RCM)	**2B Risk Control Matrix**
4	Proposed control design	**2C Control Design**, Original design column
5	Working JML Control Monitor	Build tabs or **Optional_Python_Path**
6	Exception analysis	**3 Exception Analysis** (or the Python report)
7	Incident and root-cause analysis	**4 Break Timeline** and **Deliverable_Templates/4_Break_Incident_Analysis.docx**
8	Revised control design	**5 Improve** and the Revised design column on **2C Control Design**
9	Exactly three funded improvements	**6 Priorities**
10	One-page management summary	**Deliverable_Templates/6_Defend_Management_Summary.docx**
11	Career Translation (Standard set)	**7 Career Translation**

Stage 1: Assess

Find out how identity lifecycle actually runs, not how the binder describes it.

What you are trying to accomplish

Read Harborline's current-state evidence, record what you notice, and label each item correctly without over-calling. You are building a factual starting point for your design.

Why this matters

Auditors and leaders lose trust in GRC when every messy sentence becomes a critical finding. Real assessment separates facts from judgments and names what you still cannot know. Your labels here shape everything you design later.

Pay attention to SSO (single sign-on) claims in the inventory. Treat "signs in through Okta" as a claim in the evidence, not proof that every system works that way.

Open these

What	Where to find it
The six current-state evidence files	**1_Assess_Evidence**
Your assessment table	**Master_Workbook.xlsx**, tab **1 Assess**
Optional skim of the raw extracts	**3_Build_Source_Data/employees.csv** and **3_Build_Source_Data/application_access.json**

The evidence files

File	What it is
1_Assess_Evidence/Stated_JML_Policy.pdf	What Harborline says the control is
1_Assess_Evidence/Stakeholder_Notes.pdf	Notes from HR, IT, GRC, and Engineering
1_Assess_Evidence/Application_Inventory.pdf	Systems that grant access, and where identity comes from
1_Assess_Evidence/Access_Review_Sample.pdf	The last user-access review
1_Assess_Evidence/Ticket_Sample.pdf	Recent joiner, mover, and leaver tickets
1_Assess_Evidence/Evidence_Catalog.pdf	What evidence GRC can actually produce today

The five labels (pick one per row)

Label	Use it when
Observation	A fact. Not yet a judgment.
Control gap	A rule is missing, unclear, or does not cover the risk.
Risk	Something bad could happen because of a gap or weakness.
Exception	A specific case where a stated control did not operate. You may have few of these until Build.
Missing evidence / unknown	You cannot tell yet. Say what would let you tell.

Do this

1. Open **Master_Workbook.xlsx** and read the **Start Here** tab. Then go to **1 Assess**.
2. Read every file in **1_Assess_Evidence**. Take notes as you go. Suggested order: **Stated_JML_Policy.pdf**, **Stakeholder_Notes.pdf**, **Application_Inventory.pdf**, **Access_Review_Sample.pdf**, **Ticket_Sample.pdf**, **Evidence_Catalog.pdf**.
3. For each issue, add a row on **1 Assess**: the source file (and a short quote), what you noticed, one label from the dropdown, why that label fits, and what would confirm or close it.
4. Aim for at least eight rows that use more than one label. You are not graded against an official list. You are judged on labeling correctly and not over-calling.
5. Fill the two-sentence summary at the bottom of the tab: what worries you most, and what you still cannot know.
6. **Deep Dive (optional):** add more rows and list the evidence you would request next.

What you produce	You are done when
• Tab **1 Assess** with at least eight labeled rows. • The two-sentence summary on the same tab.	☐ You have at least eight rows. ☐ You used more than one label. ☐ You can say in two sentences what worries you most and what you still do not know. ☐ You have not opened the Break folder.

Next: Move to Stage 2: Design. Keep your Assess rows open; you will cite them when you choose scope and owners.

Stage 2: Design

Design how JML should work at Harborline.

What you are trying to accomplish

Create three linked design artifacts: a JML process map, a risk and control matrix, and a proposed control design specific enough that someone could implement it in a sheet or in Python.

Why this matters

A monitor only tests what a design says should happen. If the design has no owner, no scope statement, and no exception path, the monitor produces a list nobody acts on. Design is where you decide authoritative sources, SLAs, scope, owners, and evidence.

The role catalog is Harborline's RBAC (role-based access control) input: entitlements should follow a defined role. Your RCM maps each risk to the control that treats it.

Open these

What	Where to find it
JML process map	**Master_Workbook.xlsx**, tab **2A Process Map**
Risk and control matrix	**Master_Workbook.xlsx**, tab **2B Risk Control Matrix**
Your proposed control design	**Master_Workbook.xlsx**, tab **2C Control Design**, column **Original design (Stage 2)**
Harborline's current stated control (input, not your design)	**3_Build_Source_Data/Harborline_Stated_Control_Brief.pdf**
Your Assess notes	**1 Assess**

Do not confuse the two documents

3_Build_Source_Data/Harborline_Stated_Control_Brief.pdf is Harborline's **current** stated control. It is evidence and the rule set the Build stage tests. **Your** proposed design lives only on **2C Control Design**. Do not copy the brief and call it your design.

Do this

1. Read **3_Build_Source_Data/Harborline_Stated_Control_Brief.pdf**. This is what Harborline says it does today. You may keep the 24-hour leaver SLA, change scope, add owners, or add steps, but you must be able to explain why.
2. On **2A Process Map**, fill every stage of the target flow: HR event, identity event, access determination, approval, provisioning or change or deprovisioning, verification, evidence, and exception or escalation. For each, say what happens, which systems, who does it, when, what evidence it produces, and what happens if the step fails.
3. Still on **2A Process Map**, answer the five notes prompts below the map: joiners, movers, leavers, non-human and shared accounts, and contractors.
4. On **2B Risk Control Matrix**, add at least four risks. Suggested starters (rewrite them in your own words): late leaver revocation, mover privilege creep, privileged access without approval, and access that never enters the central IdP. Map frameworks only where you mean it (SOC 2 CC6.2 and CC6.3, NIST AC-2, ISO 27001 A.5.18).
5. On **2C Control Design**, write your proposed control in the **Original design (Stage 2)** column, one short paragraph per row: control name, authoritative source, trigger, control logic (pass, fail, watch), SLA, scope, owner, evidence, exception process, escalation, monitoring, dependencies, and known limitations.
6. Leave the **Revised design (Stage 5)** column empty for now. You will fill it after the incident.

What you produce	You are done when
• Completed **2A Process Map** including the five notes. • At least four RCM rows on **2B Risk Control Matrix**. • The Original design column on **2C Control Design**.	☐ A colleague who did not see your Assess work could provision, change, and remove access using your map. ☐ They could tell who owns exceptions. ☐ Your known limitations are written down, not hidden.

Next: Move to Stage 3: Build. Your monitor will test Harborline's **stated** rules so you can compare them with your design.

Stage 3: Build

Build a working JML Control Monitor that tests the whole population.

What you are trying to accomplish

Encode Harborline's stated rules as working logic that compares HR records with the IdP export, separates exceptions from watch items, and changes its output when the source data changes.

Why this matters

Sampling five tickets misses leavers who never got a ticket, movers who kept old access, accounts HR does not know, contractors past their end date, and shared accounts with no owner. A monitor tests the **full population** the same way every time.

The skill is not the 25 rows in this file. It is encoding the rules so that 2,500 rows would not change the logic: load, classify, compare, calculate SLA, flag, report.

Open these

What	Where to find it
Stated rules the monitor tests	**3_Build_Source_Data/Harborline_Stated_Control_Brief.pdf** and tab **3 Build Guide**
Column meanings for every source file	**3_Build_Source_Data/Data_Dictionary.pdf**
Source data (spreadsheet path)	Tabs **Employees**, **Application_Access**, **Role_Catalog**, **Privileged_Approvals**
Your working sheets (spreadsheet path)	Tabs **Control_Results**, **Exceptions**, **Watch_Items**
Your exception analysis (both paths)	Tab **3 Exception Analysis**
Python path	**Optional_Python_Path/README.md** (setup is also at the end of this guide)

The rules your monitor must apply

Review freeze	2026-08-15 12:00 UTC (`2026-08-15T12:00:00Z`)
Leaver SLA	24 hours from `termination_date` (or contract end for contractors)
Mover SLA	7 days from `role_change_date`
Join key	Email, compared in lowercase
Privileged list	`okta_super_admin`, `github_admin`, `ehr_admin`, `phi_full`, `netsuite_admin`, `netsuite_ap_approve`, `finance_admin`, `hr_admin`

Exception type	When it applies	Typical severity
`sla_breach`	A leaver (terminated, or a contractor whose contract has ended) still has an **active** IdP account and more than 24 hours have passed since the termination time or contract end.	medium (high if privileged access is live)
`contractor_past_end_date`	Contractor with `contract_end_date` before the freeze and an active IdP account, even if HR still says `active`.	medium
`mover_excessive_access`	Person has both `previous_role_code` and `role_change_date`, more than 7 days have passed since the change, and a live entitlement is not allowed for the **current** role in **Role_Catalog**.	medium
`privileged_without_approval`	Active account holds an entitlement on the privileged list, and there is no approval row for that employee_id plus entitlement whose `expires_on` is blank or on or after 2026-08-15.	high
`loa_with_privileged_access`	Leave-of-absence rule. See the box below.	medium
`orphaned_account`	IdP account of type `user` whose email is not in the HR file.	high if it has write or admin access, otherwise medium
`unowned_nonhuman_account`	Account of type `shared` or `service` with an empty `owner_employee_id`. A service account that has an owner is **not** an exception.	high if it holds admin access

The leave-of-absence rule, explained

A person whose `employment_status` is `leave_of_absence` is **not** a leaver, so no 24-hour clock runs. Harborline's stated policy is to reduce privileged access while someone is on leave.

So: if the person is on leave, their IdP account is active, and it still holds at least one entitlement from the privileged list, flag `loa_with_privileged_access` (severity medium). This applies **even if a valid approval exists**, because the approval covers the job, not the period of leave. Non-privileged access such as email is not flagged.

Watch items are not failures

A leaver whose account is still active but within 24 hours of termination is a **watch item** with exception type `open_within_sla`. The control still has time to operate. It gets a re-check date, not a severity. Calling it a failure trains the control owner to ignore the report.

Things that look interesting but should not be flagged

- A CFO with `finance_admin` and a valid approval.
- A leaver whose IdP account is already `disabled`.
- A service account (`deploy-bot`) that has a named owner.
- Entitlements allowed by the role catalog. Birthright in the catalog does not replace an approval for privileged entitlements, but it does mean the access is not mover leftover.

> **Sanity check (not the answer key)**
> A complete monitor on this sample usually finds **more than five and fewer than twelve** exceptions and **at least one** watch item. Zero exceptions means a rule is not firing. Twenty usually means true negatives are being flagged.

Do this

1. Open **3 Build Guide**. It holds the freeze timestamp, the SLAs, the privileged list, and every rule in plain language. The freeze is stored in a named cell called `REVIEW_AS_OF`, the privileged list in a range called `PRIVILEGED_LIST`.
2. Choose one path: spreadsheet (continue below) or Python (follow the Python section at the end of this guide, then return to step 7).
3. **Spreadsheet path:** on **Control_Results**, build one row per person or account you need to test. Join people to accounts on **lowercase email**. Add helper columns such as lifecycle status, elapsed hours since termination, leftover entitlements, and one yes/no flag per rule. Suggested column headers are already there; rename or add columns freely.
4. Use lookups and counts against the source tabs rather than typing results. Useful functions: `LOWER`, `INDEX/MATCH`, `COUNTIFS`, `IF`, `AND`, `OR`. Excel stores dates as days, so the difference between two date-times multiplied by 24 gives hours.
5. Copy every **failure** to **Exceptions** using only the seven allowed exception types in the dropdown. Put leavers still inside the 24-hour window on **Watch_Items** with a re-check date. Never put `open_within_sla` on the Exceptions tab.
6. Test that it is a real monitor: change one termination date on **Employees** and confirm your results change. Then undo the change.

7 On **3 Exception Analysis**, fill the counts and write the short exception analysis: what failed, what passed (true negatives), and what you would tell the control owner first. Python users base this on **Optional_Python_Path/outputs/exception_report.md**.

8 **Deep Dive (optional):** answer the three Challenge prompts at the bottom of **3 Exception Analysis**.

What you produce	You are done when
• A working monitor (Build tabs or Python output). • Exception and watch lists that follow the allowed types. • The exception analysis on **3 Exception Analysis**.	☐ The monitor runs and you have looked at every exception. ☐ Leavers inside 24 hours are watch items, not exceptions. ☐ Disabled leavers are not current-access exceptions. ☐ Approved privileged users are not flagged as privileged_without_approval. ☐ Owned service accounts are not flagged as unowned. ☐ Changing a source value changes the output.

Next: Only now open the Break evidence folder and move to Stage 4.

Stage 4: Break

Investigate an incident your first monitor may not have been built to see.

What you are trying to accomplish

Read the incident evidence, reconstruct what happened, classify the failure, and write a root-cause analysis from the evidence alone.

Why this matters

Controls fail in ways their designers did not expect. The test of a GRC practitioner is whether they can tell a design failure from an operating failure from a scope failure, and say so without rewriting history or inventing evidence.

Open these

What	Where to find it
Incident evidence (four files)	**4_Break_Incident_Evidence_Open_After_Build**
Your evidence timeline	**Master_Workbook.xlsx**, tab **4 Break Timeline**
Your written incident analysis	**Deliverable_Templates/4_Break_Incident_Analysis.docx**
Your Stage 3 results, for comparison	**Exceptions** or **Optional_Python_Path/outputs/exception_report.md**

Open this evidence only after Build

Do not open **4_Break_Incident_Evidence_Open_After_Build** until you have a working monitor and have reviewed its exceptions. The folder is included in your download so you have everything in one place. It is not a hint for Stage 3.

The evidence files

File	What it is
4_Break_Incident_Evidence_Open_After_Build/Incident_Summary.pdf	What was reported
4_Break_Incident_Evidence_Open_After_Build/IdP_Disable_Evidence.pdf	Central IdP (Okta-style) event log
4_Break_Incident_Evidence_Open_After_Build/SaaS_Removal_Evidence.pdf	Google Workspace and GitHub removal events
4_Break_Incident_Evidence_Open_After_Build/AWS_Auth_Log.pdf	Cloud sign-in log excerpt and IAM user details

Where Break findings go

Excel	Word
4 Break Timeline: the structured timeline of events, sources, and what each one proves.	**Deliverable_Templates/4_Break_Incident_Analysis.docx**: your written analysis and conclusions. This is the deliverable a manager or auditor would read.

The incident account is **not** part of the IdP export your monitor tested, so do not add it to the **Exceptions** tab. Treat it as a separate case.

Three classification terms

Term	Meaning
Control design	The rule as written cannot cover the risk.
Operating effectiveness	A designed rule did not run as specified: wrong time, wrong population, or missing evidence.
Scope	The test or control never included that system.

Do this

1. Confirm Build is complete. If not, stop and finish Stage 3 first.
2. Read the four files in **4_Break_Incident_Evidence_Open_After_Build**, starting with **4_Break_Incident_Evidence_Open_After_Build/Incident_Summary.pdf**.
3. On **4 Break Timeline**, log every timestamped event in time order: the time (UTC), the source file, the system, what happened, what it proves, and what it does not prove.
4. Open **Deliverable_Templates/4_Break_Incident_Analysis.docx**, save a copy with your name, and answer every section in short paragraphs: facts you accept and doubt, what failed, classification, root cause, risk exposure, evidence still required, immediate containment, systemic remediation, whether the original design must change, and how monitoring should change.
5. For classification, decide whether this is a **control design** failure, an **operating effectiveness** failure, a **scope** failure, or a combination, and say which part is which.

What you produce	You are done when
• The evidence timeline on **4 Break Timeline**. • Your completed copy of **Deliverable_Templates/4_Break_Incident_Analysis.docx**.	☐ You can say what failed without rewriting history. ☐ Your classification names design, operating effectiveness, scope, or a mix, with reasons. ☐ You did not invent log entries, claim a dollar loss, or claim Harborline passed or failed SOC 2.

Next: Move to Stage 5: Improve and revise your design in light of what you found.

Stage 5: Improve

Show that your design can learn from the incident.

What you are trying to accomplish

Revise your process map, control design, monitoring scope, and escalation logic, and record what changed and why.

Why this matters

Interviewers ask "what changed after the incident?" because a design that cannot learn is not a control. Keeping the original visible next to the revision shows your reasoning.

Open these

What	Where to find it
Your improve record	**Master_Workbook.xlsx**, tab **5 Improve**
Revised design and process map	**2C Control Design** and **2A Process Map** (Stage 5 columns)

Do this

1. On **2C Control Design**, fill the **Revised design (Stage 5)** column wherever your thinking changed. Leave the original column untouched so the before and after are both visible. Write "No change" where nothing changed.
2. On **2A Process Map**, use the **Revised after Break (Stage 5)** column to update verification and systems outside the IdP.
3. On **5 Improve**, complete every prompt: what the incident taught you (three bullets maximum), what changed in the process map, control design, monitoring scope, and exception or escalation logic.
4. Write the key sentence: **What changed in my design after the incident and why?** This is the line interviewers will ask you to defend.
5. Record what you intentionally did not change, and any residual risk you will ask management to accept.

What you produce	You are done when
• Completed **5 Improve**. • The Revised columns on **2C Control Design** and **2A Process Map**.	☐ Someone reading only your revised design would know that a local cloud account outside the IdP is either in scope, or explicitly accepted as residual risk with an owner and a compensating control. ☐ "We will remember to ping Engineering" does not appear as a control.

Next: Move to Stage 6: Defend.

Stage 6: Defend

Make the case to leadership and choose what to fund.

What you are trying to accomplish

Choose exactly three improvements Harborline can fund this quarter and write a one-page management summary that ends in one clear decision.

Why this matters

Leadership will not read your Python or a many-tab workbook. They will read one page and ask what to fund. Choosing three, and naming the residual risk of what you defer, is the skill. Listing eight "P1" items is a fail.

Open these

What	Where to find it
Your three funded items	**Master_Workbook.xlsx**, tab **6 Priorities**
Your one-page management summary	**Deliverable_Templates/6_Defend_Management_Summary.docx**
Inputs	**3 Exception Analysis**, **5 Improve**, **Deliverable_Templates/4_Break_Incident_Analysis.docx**

Do this

1. On **6 Priorities**, rank exactly three improvements. For each: why this and not something else, the main risk it reduces, effort (Low, Medium, or High), and the dependency. Weigh risk reduction, blast radius, likelihood, privilege, scalability, effort, and dependencies.
2. Below the table, describe what you are deferring and the residual risk of waiting, and write the two sentences you would say if leadership pushed back on item 1.
3. Open **Deliverable_Templates/6_Defend_Management_Summary.docx**, save a copy, and complete every section: what we evaluated, material findings, business and risk exposure, control weaknesses, recommended remediation, priorities (point to your three; do not paste the matrix), residual risk, and the decision required.
4. Keep it to one printed page, about 400 to 500 words. Write for a CISO, CIO, or business executive. If a sentence only makes sense to an Okta administrator, rewrite it.
5. End with one clear ask: fund, accept risk, or assign an owner by a date.

What you produce	You are done when
• Three ranked items on **6 Priorities**. • Your completed one-page copy of **Deliverable_Templates/6_Defend_Management_Summary.docx**.	☐ Exactly three items are funded. ☐ Deferred work has its residual risk named. ☐ The summary fits on one page and ends with one decision.

Next: Move to Stage 7: Career Translation.

Stage 7: Career Translation

Turn the work into an honest professional story.

What you are trying to accomplish

Write the Standard career set: honesty sentence, portfolio description, resume bullets, a 30-second explanation, and a STAR story that covers the whole loop.

Why this matters

Employers hire people who can explain what they built, why, what they discovered, and what they decided. Harborline is fictional: never describe it as an employer or client.

Open these

What	Where to find it
Career prompts and writing areas (your counts come from **3 Exception Analysis**)	**Master_Workbook.xlsx**, tab **7 Career Translation**

Results you may and may not claim

Allowed (only what your files show): X exceptions found, an in-window leaver kept as a watch item, a successful IdP disable that still left cloud access, a revised control, a three-item recommendation. **Not allowed:** "reduced violations by 40%," "we passed SOC 2," "saved $X," or anything implying Harborline was real.

Do this

1. Say the honesty sentence out loud first: *"This was an independent portfolio project where I simulated identity lifecycle and access-governance engineering for a fictional company. I assessed a messy current state, designed controls, built a monitor, investigated an incident, improved the design, and wrote a management recommendation."*
2. On **7 Career Translation**, write a 150 to 200 word portfolio description using the spine: what I built, why, how it works, what I discovered, what decision I made, what I would improve.

3. Write two or three resume bullets (verb, control problem, what you built, a judgment call), a timed 30-second explanation, and a STAR story covering Assess through Defend. Counts must match **your** output.
4. **Deep Dive (optional):** a GitHub README paragraph for a personal re-implementation on your own dummy data, a skills list, the ten interview questions, and the defense drill.

What you produce	You are done when
• The Standard career set on **7 Career Translation**.	☐ You can say the honesty sentence without hesitation. ☐ You can tell the STAR story without reading it. ☐ No invented employment, no fake KPIs, and counts that match your own output.

Next: Complete the final checklist on the next page.

Before you call this finished

A final check across all seven stages.

Standard completion checklist

☐ Current-state assessment (**1 Assess**)

☐ JML process map (**2A Process Map**)

☐ Risk and control matrix (**2B Risk Control Matrix**)

☐ Proposed control design (**2C Control Design**)

☐ Working monitor (spreadsheet or Python)

☐ Exception analysis (**3 Exception Analysis**)

☐ Incident analysis (**Deliverable_Templates/4_Break_Incident_Analysis.docx**)

☐ Revised design (**5 Improve**)

☐ Exactly three funded items (**6 Priorities**)

☐ One-page management summary (**Deliverable_Templates/6_Defend_Management_Summary.docx**)

☐ Career Translation Standard set (**7 Career Translation**)

Reasoning questions

These test your reasoning. They are not an answer key. Answer them in your own notes.

- Can I explain why I designed the JML control this way?
- What evidence did I trust (HR, IdP, approvals) and why? What did I not treat as authoritative?
- What changed during Break? Did the original control cover the scope I assumed?
- Did I reconsider watch items versus exceptions after I saw the output?
- What residual access risk remains after my three funded items?
- What alternative control or SLA did I reject, and what did I deliberately not automate?
- Can I defend the monitor to someone who says "just disable everyone on the termination ticket"?

Optional: the Python Build path

Use this instead of the spreadsheet Build tabs if you want to code.

The Python path uses the **same data and the same rules** as the spreadsheet. It has two styles. **Guided** is seven numbered exercises with TODOs and hints, completed in order. **Challenge** is one starter file with less scaffolding. Pick one.

Step	File	Control skill
1	**guided/01_load_hr_data.py**	Treat HR as the population
2	**guided/02_evaluate_workforce_status.py**	Classify active, terminated, leave, and expired contractors
3	**guided/03_load_application_access.py**	Treat the IdP export as live central access
4	**guided/04_compare_hr_to_access.py**	Join on email; find orphans
5	**guided/05_calculate_revocation_sla.py**	Elapsed time against the 24-hour leaver SLA
6	**guided/06_flag_control_exceptions.py**	Apply every rule in the stated control
7	**guided/07_score_and_report.py**	Write JSON and markdown evidence

All file paths above are inside **Optional_Python_Path**. **guided/validate.py** checks the **structure** of your output after each step. It is not an answer key.

Python setup (one time)

1. Install Python 3.10 or newer from https://www.python.org/downloads/. On Windows, tick **Add Python to PATH** during setup.
2. Open a terminal. On Windows use PowerShell or Command Prompt. On macOS use Terminal.
3. Check the version: `python3 --version` (on Windows the command is often `python --version`). You want 3.10 or higher.
4. Change into the optional Python folder inside this project. The easiest way: type `cd` followed by a space, then drag the **Optional_Python_Path** folder onto the terminal window and press Enter.
5. Nothing needs to be installed. The exercises use the Python standard library only.

Run commands

```
# Guided: run each step, then validate it
python3 guided/01_load_hr_data.py
python3 guided/validate.py --step 1
# ...repeat for steps 2 to 7

# Challenge: one script, then validate the final report
python3 challenge/challenge.py
python3 guided/validate.py --step 7
```

On Windows, type `python` wherever these commands say `python3`. Scripts find the data folder on their own, so you can run them from any location as long as the project folder structure is unchanged. Output is written to the **Optional_Python_Path/outputs** folder, which the scripts create if it is missing.

What you see	What it means
`NotImplementedError`	Expected. You still have a TODO in that function. Read the HINT lines next to it.
`python3: command not found`	Try `python`. If neither works, install Python and reopen the terminal.
`FileNotFoundError ... outputs/0N_...json`	Run the earlier numbered step first. Each step reads the previous step's output.
Wrong SLA or freshness results	Use the review freeze timestamp from this guide, never your computer clock.

Challenge prompts

If you choose Challenge, include answers to the three prompts in **challenge/README.md** inside your markdown report. Spreadsheet users can answer the same prompts on **3 Exception Analysis**.

When you finish

Your evidence is **Optional_Python_Path/outputs/exception_report.json** and **Optional_Python_Path/outputs/exception_report.md**. Use the markdown report for the counts on **3 Exception Analysis**, then continue to Stage 4.