

THE VISUAL EDITION



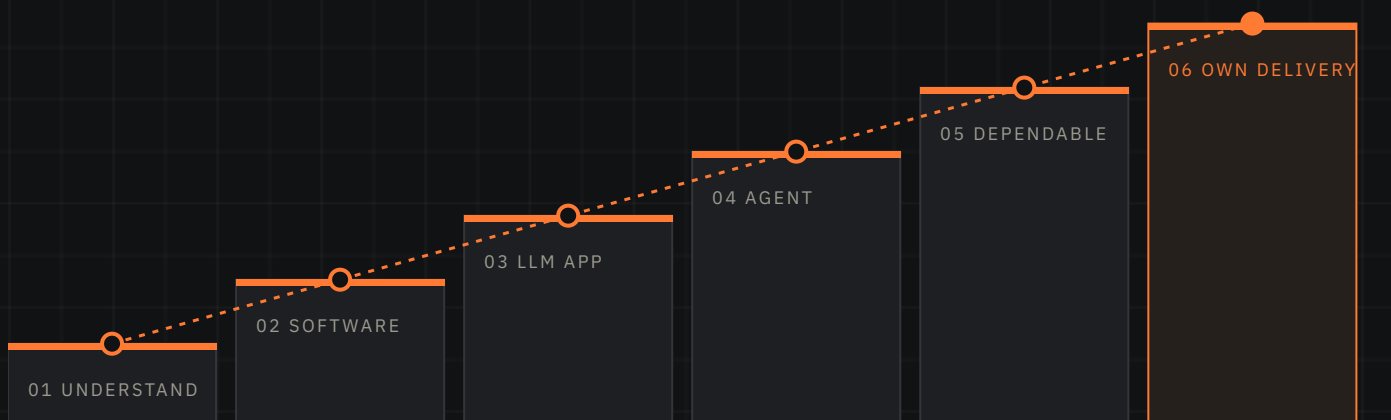
# AGENTIC AI CAREER GUIDE 2026-2027

Choose a target role, find the gap, follow a practical learning sequence and build evidence you can explain in an interview or a workplace review.

## TRANSITION TO AI LEVERAGING YOUR EXPERIENCE

Choose the Starting Point based on your Experience

[manifoldailearning.com](https://manifoldailearning.com) ➤



6

CAPABILITY STAGES

7

ROLE FAMILIES

5

EXPERIENCE PATHWAYS

90

DAY STARTER PLAN



# ONE FOUNDATION. DIFFERENT DEPTHS.

A developer, a testing professional, a systems consultant and an engineering director can share foundational knowledge while needing different depths of implementation, architecture and delivery practice. This guide covers people entering technology as well as experienced professionals moving into AI work.

## THE CENTRAL RECOMMENDATION

01

### Build a shared foundation

Programming, models, tools, data and evaluation. The same base for every role.

&gt;

02

### Complete a small working system

Something narrow you can run, test, break and explain.

&gt;

03

### Deepen your target role

Add the responsibilities that role actually requires.



**Use your experience** to choose problems and ownership scope.



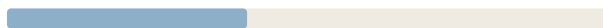
**Use demonstrated capability** to judge readiness.

## EXPERIENCE VS. READINESS

YEARS OF EXPERIENCE

### PROVIDE CONTEXT

They do **not** automatically qualify someone for a senior AI title.



CONTEXT ≠ CAPABILITY

MOVING INTO AI

### ERASES NOTHING VALUABLE

Software delivery

Business processes

Operations

Leadership

CARRY IT FORWARD

## HOW TO READ THIS GUIDE

STEP 1

**Role map & self assessment**

→

STEP 2

**Pathway** closest to your experience

→

STEP 3

**Adjust** for your current background

→

STEP 4

**Foundation** sequence, then specialise

→

STEP 5

**Project, interview & plan** → visible work



**Already building reliable AI apps?** Use the readiness checks to skip material you can demonstrate.



**New to coding?** Give the programming stage the time it needs before attempting a full agent application.

**Using the guide through 2027.** Recommendations are for preparation during 2026 and 2027. Sources and example role descriptions were reviewed on 22 October 2026. Recheck job descriptions, SDK documentation and certification requirements when you use them. The 2027 pathway is a planning horizon, not a forecast of hiring demand or future product features.

# FIND YOUR ROUTE THROUGH THE GUIDE

Read in order or jump to a decision. Keep one target role and one manageable project in view while you work through it.

|                                                                                                                                                          |                                                                                                                                                                    |                                                                                                                                                 |                                                                                                                                                      |
|----------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------|
|  <b>P4–7</b><br><b>Understand</b> Agentic AI & the stages of capability |  <b>P8–9</b><br><b>Choose a role</b> & assess your preparation                    |  <b>P10–15</b><br><b>Plan</b> around your experience level     |  <b>P16</b><br><b>Adapt</b> the path to your current profession   |
|  <b>P17–19</b><br><b>Build</b> programming & AI foundations             |  <b>P20</b><br><b>Evaluate &amp; operate</b> a reliable system                    |  <b>P21–23</b><br><b>Prepare</b> for a particular role         |  <b>P24–26</b><br><b>Choose &amp; develop</b> a practical project |
|  <b>P27–28</b><br><b>Portfolio</b> , resume & interviews                |  <b>P29</b><br><b>Set</b> a realistic schedule                                    |  <b>P30</b><br><b>Decide</b> on certification & a career break |  <b>P31</b><br><b>Resolve</b> common starting questions           |
|  <b>P32</b><br><b>Record</b> your plan & review readiness             |  <b>P33–35</b><br><b>Keep skills current</b> , review terms and consult sources |                                                                                                                                                 |                                                                                                                                                      |

## A FIRST DECISION YOU CAN MAKE TODAY

Which responsibility interests you most?

|                                                                                                                        |                                                                                                               |                                                                                                                       |                                                                                                                      |                                                                                                                        |                                                                                                               |
|------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------|
|  <b>Implement</b><br>an application |  <b>Run it</b><br>reliably |  <b>Design</b><br>its architecture |  <b>Deliver</b><br>with customers |  <b>Evaluate</b><br>its behaviour |  <b>Lead</b><br>the team |
|------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------|

### THEN READ REAL JOB DESCRIPTIONS

Save a small set for that responsibility and mark:

RECURRING  
RESPONSIBILITIES

EVIDENCE REQUESTED

REQUIRED EXPERIENCE

TECH YOU'D ACTUALLY USE

Titles vary. An AI Engineer role may be product development in one company and model training in another. An AI Architect role may require hands-on implementation.

### CHOOSE A USEFUL FIRST PROJECT

A process you understand and can model with sample data, so you know what a correct outcome looks like.

Read-only  
investigation +  
clear handoff

Explain & test the  
first version

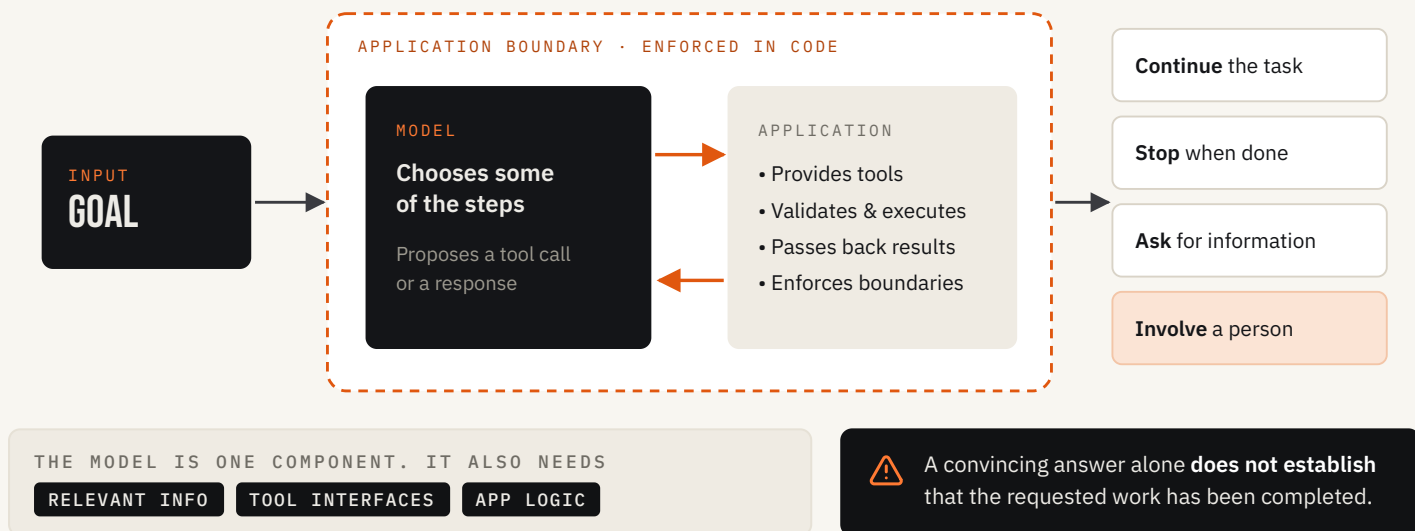
Then controlled  
updates

Good starters: a release investigation, a failed-test assistant, a technical handover.

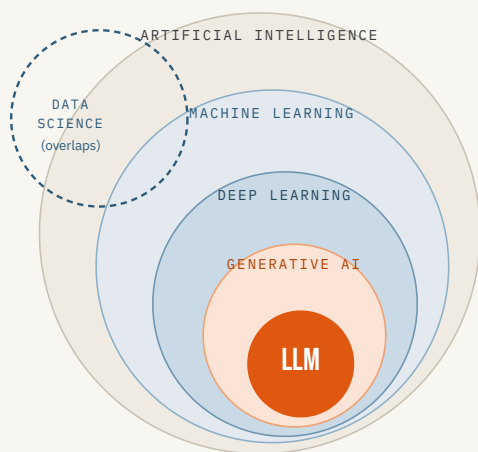
# HOW THE IDEAS FIT TOGETHER

Agentic AI is an approach to building software that uses a model to choose some of the steps needed to achieve a goal. The application provides tools, passes back their results and enforces the boundaries of what can happen.

## ANATOMY OF AN AGENTIC APPLICATION SOURCE [1]



## PLACE THE TERMS ON A SIMPLE MAP

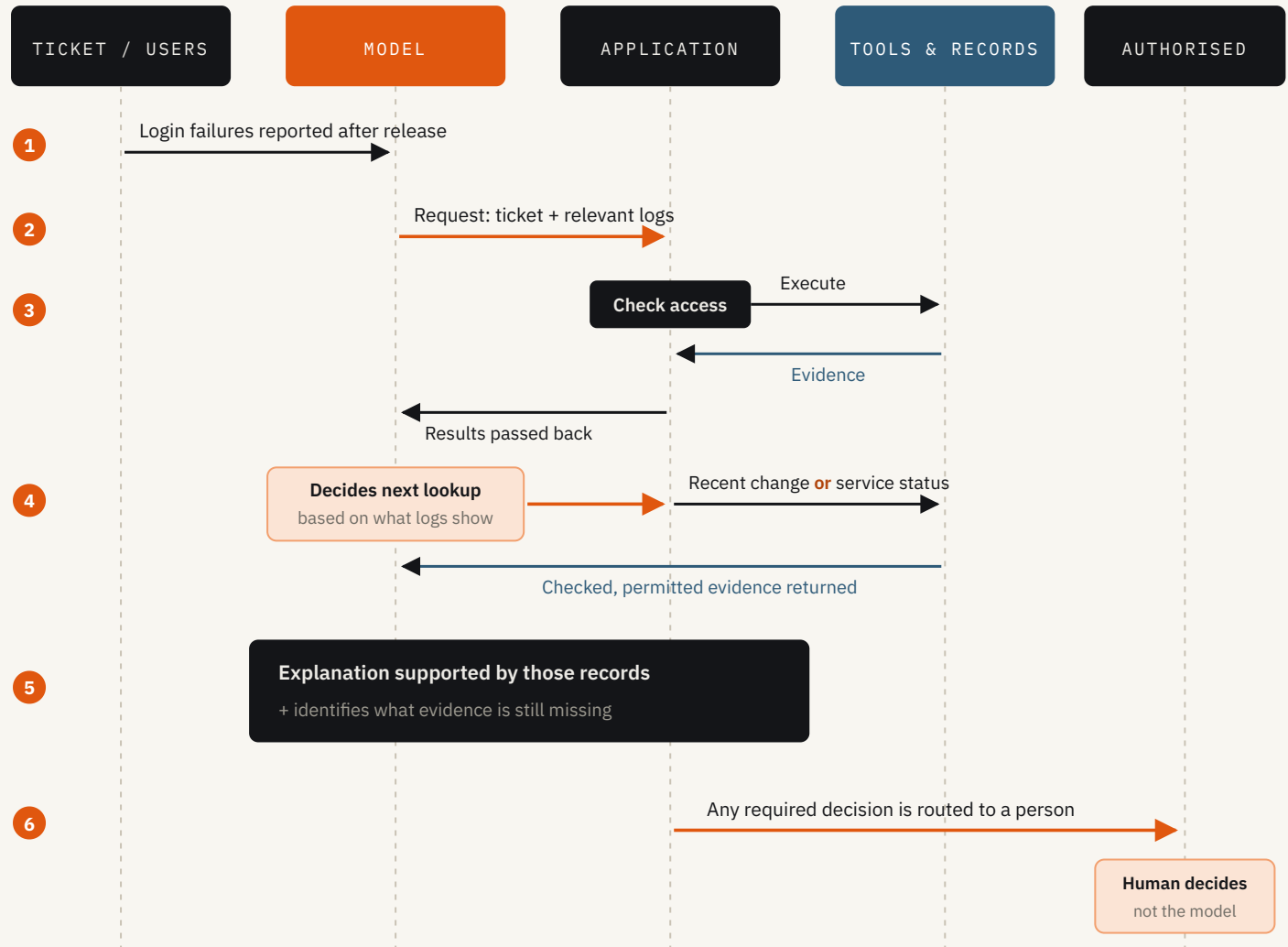


| TERM                                                                                                            | WHAT IT DESCRIBES                                           | EXAMPLE                                            |
|-----------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------|----------------------------------------------------|
|  <b>Data science</b>         | Using data and domain knowledge to investigate questions    | Analyse patterns in recurring application failures |
|  <b>Machine learning</b>     | Methods that learn patterns from examples                   | Predict which support tickets may need escalation  |
|  <b>Deep learning</b>        | ML using layered neural networks                            | Recognise text in scanned technical documents      |
|  <b>Generative AI</b>        | Models that generate content                                | Draft an explanation using supplied facts          |
|  <b>Large language model</b> | Works with language and related tasks                       | Interpret a request or propose a tool call         |
|  <b>Agentic application</b>  | A system that can choose steps and use tools towards a goal | Investigate a reported failure after a release     |

 **Agentic is a way of operating, not a model family.** It describes how a system behaves; it does not make earlier methods obsolete. Deep learning sits within ML, which sits within AI; data science overlaps them. [9, 10]

# FOLLOW ONE REQUEST END TO END

Users report login failures after a release. Here is what actually happens between the model, the application, the tools and the people involved.

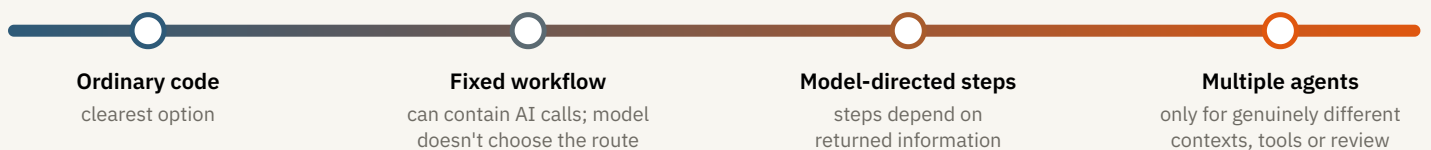


**Reading a tool result does not retrain the model.** The evidence lives in this request's context; the model's parameters do not change.

## KNOW WHEN A SIMPLER DESIGN IS SUFFICIENT

◀ ROUTE IS PREDICTABLE

ROUTE NEEDS VARIATION / AMBIGUITY ▶



Use model-driven choices where they help with variation or ambiguity — and **always compare the result with a simpler baseline.** [1]



# PROGRESS THROUGH CAPABILITY, NOT TOPIC LISTS

Each stage should produce something you can demonstrate. Move quickly through stages your previous work already supports, but keep the evidence requirement.

06

## STAGE Own delivery

Match the system to users and business constraints



### EVIDENCE GATE

An acceptance plan, measured result and support handover

05

## STAGE Make it dependable

Evaluate failures and support operation



### EVIDENCE GATE

A regression set, permissions checks and a runbook

04

## STAGE Build a bounded agent

Choose tools using returned information



### EVIDENCE GATE

A visible tool trace and defined stopping conditions

03

## STAGE Build an LLM application

Use a model with context and validated outputs



### EVIDENCE GATE

A small application with test inputs and checked results

02

## STAGE Work with software

Read code and connect to a service



### EVIDENCE GATE

A tested API call with clear error handling

01

## STAGE Understand the task

Explain the business goal and an appropriate design



### EVIDENCE GATE

A one-page process map and a simpler alternative



### API

A defined interface through which software requests data or an action from another system.



### Trace

A record of the steps of a request.



### Runbook

How someone should operate the system and respond to common failures.



**The technical foundation is shared.** The depth, independence and ownership expected from you increase with the responsibility you seek.

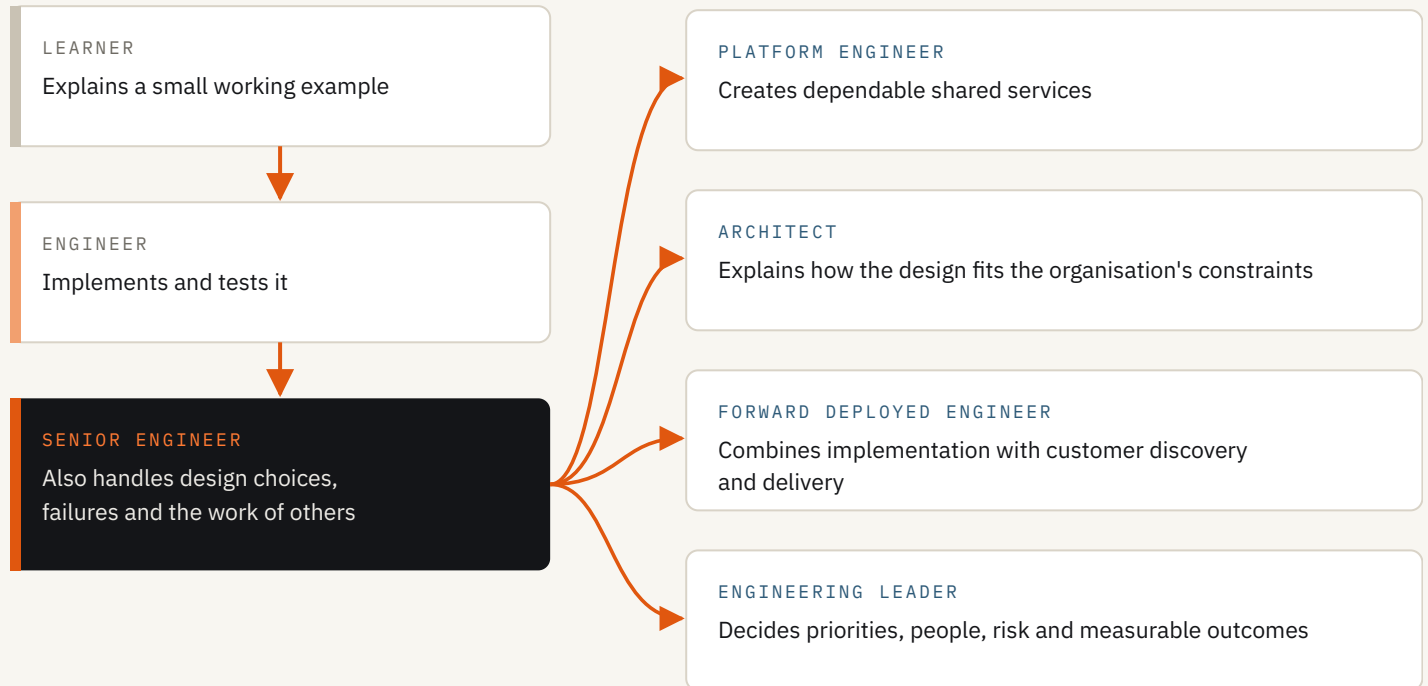


# THE PROGRESSION OF RESPONSIBILITY

These responsibilities overlap, but they are not interchangeable. Depth builds first; then ownership branches in different directions.

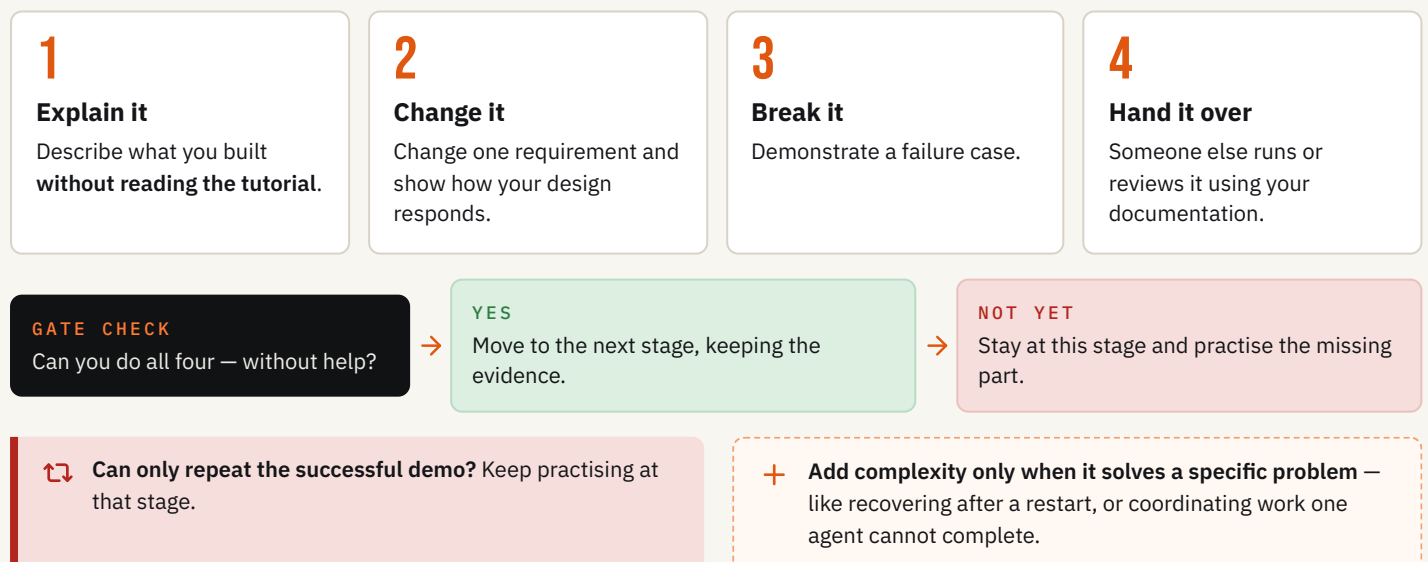
DEPTH ▼

OWNERSHIP BRANCHES ►



Branches overlap in practice. Each still needs evidence for its own responsibility.

## A PRACTICAL COMPLETION RULE APPLY AT EVERY STAGE



# CHOOSE A ROLE BY ITS RESPONSIBILITIES

Read the responsibilities behind the title. These role families are planning categories, not standardised job grades. Do not plan a career around a title alone.



## AI / Agentic AI Engineer

Implement an application and its integrations.

**EVIDENCE** Working code and evaluation results



## Senior AI Engineer

Own larger components and engineering decisions.

**EVIDENCE** Failure analysis and reviewed design choices



## AI Platform Engineer

Provide shared deployment and operational capabilities.

**EVIDENCE** A reusable service and a recovery exercise



## AI Architect

Design a system within business and technical constraints.

**EVIDENCE** Options analysis and a tested architecture



## Forward Deployed AI Engineer

Deliver a useful system in a customer environment.

**EVIDENCE** Discovery notes and an accepted implementation



## AI Evaluation Engineer

Establish and investigate quality and reliability.

**EVIDENCE** A labelled test set and reproducible evaluations



## AI Engineering Leader

Set direction and own team delivery.

**EVIDENCE** Delivery decisions and outcome reviews

## RELATIVE EMPHASIS BY ROLE PLANNING AID

Derived from the role descriptions in this guide — a way to see where each role asks for depth, not an employer standard. Rows use the seven self-assessment areas on the next page.

|                                          | ENGINEER | SENIOR | PLATFORM | ARCHITECT | FDE | EVALUATION | LEADER |
|------------------------------------------|----------|--------|----------|-----------|-----|------------|--------|
| Programming & debugging                  | ...      | ...    | ...      | ..        | ... | ..         | .      |
| APIs, databases & integration            | ...      | ...    | ...      | ...       | ... | ..         | .      |
| LLM behaviour, context, retrieval, tools | ...      | ...    | ..       | ...       | ... | ...        | ..     |
| Testing, evaluation & failures           | ..       | ...    | ..       | ..        | ..  | ...        | ..     |
| Deployment, access, monitoring, recovery | ..       | ..     | ...      | ...       | ..  | .          | ..     |
| Domain, user discovery, outcomes         | .        | .      | .        | ..        | ... | ..         | ...    |
| Communication & decision ownership       | ..       | ..     | ..       | ...       | ... | ..         | ...    |

... Core    .. Important    . Working knowledge



**Titles mislead.** Some postings use *AI Orchestrator* for workflow implementation, coordination or business operations. Check what the person actually builds and owns.



**Roles combine.** One employer's FDE posting combines programming, production LLM experience, customer discovery and domain knowledge — evidence across several areas. [2]



# ASSESS YOURSELF USING EVIDENCE

Rate each area, attach an example where possible, and use the result to find the next gap for your target role. Repeat after every project.

## RATING SCALE



| AREA                                                         | 1                        | 2                        | 3                        | 4                        | EVIDENCE / EXAMPLE |
|--------------------------------------------------------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------|
| Programming and debugging, including reading unfamiliar code | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |                    |
| APIs, databases and connecting software systems              | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |                    |
| LLM behaviour, context, retrieval and tool calling           | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |                    |
| Testing, evaluation and interpreting failures                | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |                    |
| Deployment, access control, monitoring and recovery          | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |                    |
| Domain knowledge, user discovery and business outcomes       | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |                    |
| Technical communication and ownership of decisions           | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |                    |



### A critical gap stays critical

A strong overall impression does not compensate for one missing skill the role depends on.



### Match the gap to the role

Implementation roles must implement. Architecture roles must challenge a design using evidence.



### Evidence, not course count

Save this and repeat it after each project. Record new evidence; don't raise a score because you finished a course.

MY TARGET ROLE

MOST IMPORTANT GAP

RE-ASSESS AFTER PROJECT / DATE

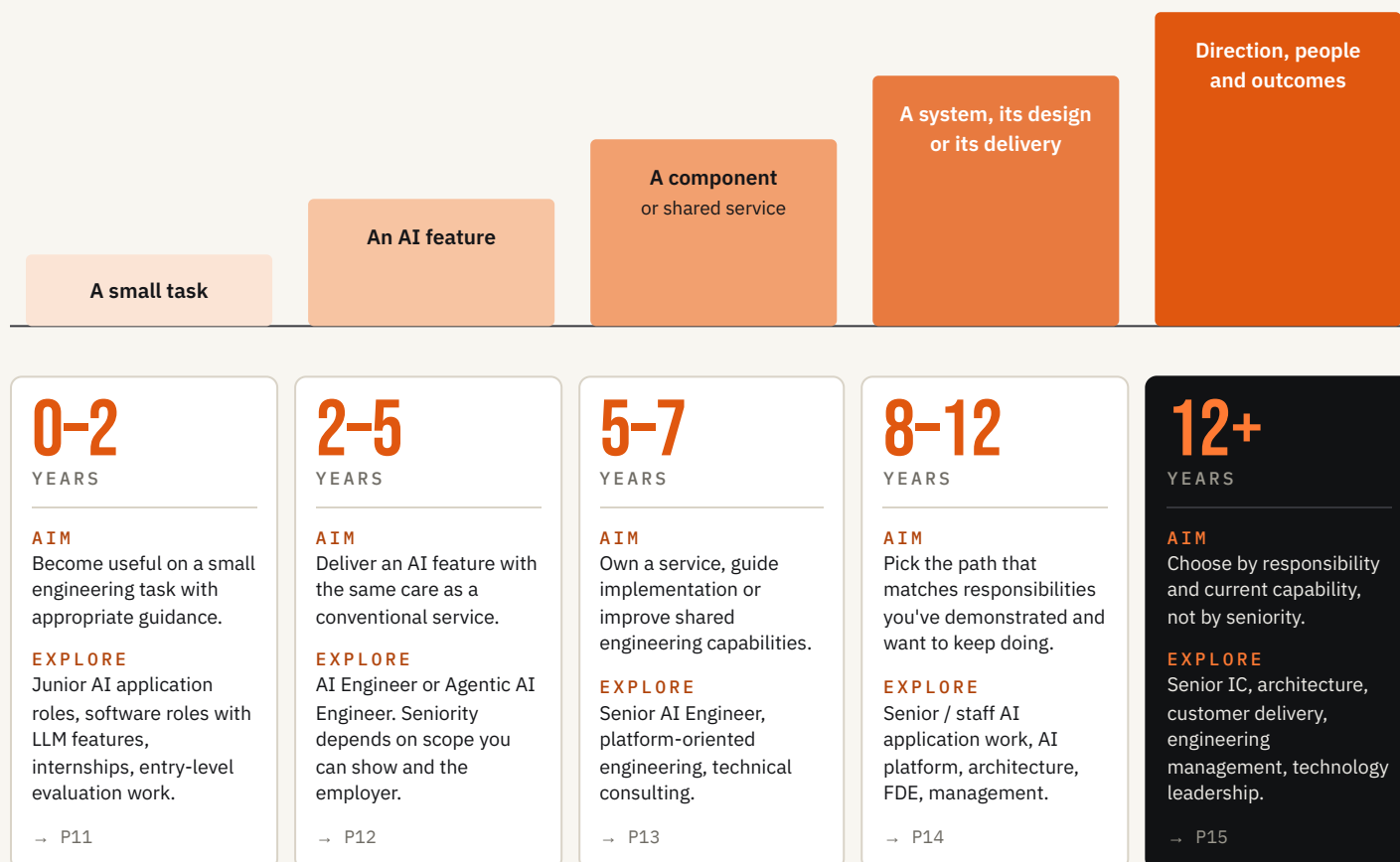




# FIVE PATHWAYS, ONE RULE: SCOPE FOLLOWS EVIDENCE

Experience bands guide your starting point. They do not assign a title or guarantee an outcome. Years alone do not establish a level.

TYPICAL SCOPE OF OWNERSHIP TO DEMONSTRATE ▲ (ILLUSTRATIVE)



## WHAT CHANGES AS YOU MOVE RIGHT — AND WHAT DOESN'T

### STAYS CONSTANT

- The shared technical foundation: models, tools, data, evaluation
- The evidence requirement at each stage
- The need to inspect code, reproduce failures and explain choices

### INCREASES

- Depth, independence and scope of ownership
- Design judgement and decisions about trade-offs
- Responsibility for others, users and outcomes

⊘ **There is no automatic conversion** from Senior Manager to AI Director, or from years served to a senior AI title. Support every claim with actual work or clearly labelled practice cases.





# PATHWAY: 0 TO 2 YEARS

Your immediate aim is to become useful on a small engineering task with appropriate guidance. Your early advantage comes from reliable foundations and visible contribution.

## BUILD THE FOUNDATIONS IN A WORKABLE ORDER

1

### Programming basics

Debugging, Git, files, JSON, HTTP requests. Read an error message and trace a request before adding a framework.



2

### Small model-powered app

Clear instruction, relevant information, validated output structure. Handle failed, incomplete and "don't know" responses.



3

### One read-only tool

Execute an allowed request, return the result to the model, set a step limit and show how the task ends.



4

### Then retrieval

Only after you can explain the basic request flow.

JSON is a common format for exchanging structured data.

## PRODUCE EVIDENCE OF YOUR OWN WORK

### SUGGESTED PROJECT • SUPPORT ASSISTANT

#### VERSION 1

- Sample policies + synthetic tickets
- Answers supported questions
- Shows its sources
- Asks for help when a policy is absent



#### VERSION 2

- Looks up a ticket through a tool

### REPOSITORY CONTENTS

- ✓ Installation steps & example inputs
- ✓ Expected behaviour & tests
- ✓ Your contribution, if you used a tutorial or AI coding help
- ✓ An independent change with its test update

## KNOW WHAT TO DEMONSTRATE BEFORE APPLYING



Explain the **model call**, **tool call** and application checks in your own words



Run the app from a **clean setup** using the README



Show a **correct result**, an unsupported question and a tool failure



Explain where the **sample data** came from and what you may share



Describe a **limitation** and a practical next improvement



**Don't make multi-agent coordination your first requirement.** Complete a small, testable system before expanding it.



**The 90-day plan (P29)** starts once your programming baseline is comfortable. If it isn't yet, add a preparation phase. Completing the calendar is not a hiring qualification.



# PATHWAY: 2 TO 5 YEARS

Your implementation experience can shorten the software foundation stage. Now deliver an AI feature with the care you would give any conventional service.

## TRANSFER WHAT YOU ALREADY USE

APP STRUCTURE   APIS   DATABASES   TESTS

VERSION CONTROL

If your coding is limited to a narrow tool or scripting task, strengthen general debugging and service integration first.

+

## ADD WHAT'S SPECIFIC TO MODEL-BASED APPS

UNCERTAIN OUTPUTS   CONTEXT LIMITS

STRUCTURED RESPONSES   TOOL SELECTION

RETRIEVAL QUALITY   EVALUATION

## LEARN TO SEPARATE WHERE AN ERROR COMES FROM

Model error

Retrieval error

Data error

Application error

## BUILD AN APPLICATION YOU CAN MAINTAIN

Example: an assistant that investigates failed background jobs using sample run records and operating notes.

### BOUND IT

- Read-only tools first
- Validate user access
- Record each request's steps

### SERVE IT

- Simple service interface
- Timeouts & clear errors
- Tests for business rules

### VERSION IT

- Model & prompt versions in evaluation records

### DEPLOY IT

- Approved practice environment
- Secrets outside source code



**Then run the experiment:** fixed route vs. model-directed tool selection on the **same cases**. Record whether flexibility improves completion — and what it costs in time and complexity.

## EVIDENCE & INTERVIEW READINESS

- ✓ A small evaluation dataset
- ✓ A repeatable run command
- ✓ A brief results report — including failures and what changed after investigating them
- ✓ Which components you built vs. reused

## BE READY TO TRACE A REQUEST

CLIENT > MODEL > TOOLS > DATABASE >

RESPONSE

...and explain what happens if the model returns the **wrong tool input** or the external service **times out**.

## A USEFUL NEXT MOVE — AFTER ONE DEPENDABLE APPLICATION

**Product-focused** → deepen user workflows & integrations

**Infrastructure-minded** → move towards platform work

**Customer-facing** → practise discovery & acceptance



# PATHWAY: 5 TO 7 YEARS

The next role may involve owning a service, guiding implementation or improving shared capabilities. Your existing responsibilities must support it — years alone do not.

## OWN A COMPLETE COMPONENT

Choose a system big enough to expose real design decisions — e.g. **a retrieval service used by an agent**, or **a workflow that pauses for approval**. Define its interface, failure behaviour and operational responsibilities.

ANY OF THESE CHANGE...



 **Persistence & recovery.** When a task spans steps or sessions, save the state needed to continue safely. **A restart must not execute an approved action twice.** Frameworks help, but you must understand their behaviour in your application. [4]

## CHOOSE AN EMPHASIS



### Application

- Workflow design, retrieval, domain integrations, user feedback
- Review code and improve evaluation quality
- Explain design changes to other engineers



### Platform

- Identity, shared gateways, deployment automation
- Tracing, cost controls, service recovery
- Make it easy for another engineer to use and operate

Both routes need practical implementation. A senior title does not remove the need to inspect code, reproduce failures and explain technical choices.

## EVIDENCE AT THE RIGHT SCOPE

- ✓ Design note: simple baseline vs. chosen approach
- ✓ A component another person can integrate from its docs
- ✓ Regression report with failure categories & versions
- ✓ Incident or failure exercise with recovery steps
- ✓ Review of how your decisions affected the wider system

Consulting background? Add stakeholder discovery and acceptance criteria — show how an unclear request became a reviewable deliverable.

## THE OWNERSHIP BOUNDARY

### YOU DECIDED

Which choices were yours

### YOU IMPLEMENTED

What you personally built

### ANOTHER TEAM SUPPLIED

Reused or provided parts

### WHO OPERATES IT

Ownership after release

Explaining this boundary is **stronger evidence than a long list of tools.**



# PATHWAY: 8 TO 12 YEARS

Your work may span hands-on engineering, architecture, delivery, customer engagement or people management. Choose the path that matches your demonstrated responsibilities.

## SELECT A PRIMARY RESPONSIBILITY

| IF YOUR STRENGTH TODAY IS...                | ...A NATURAL ROUTE IS                                                                                                                                                 |
|---------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Owning complex services as an engineer      | → Senior or staff-level AI application work                                                                                                                           |
| Cloud or platform specialism                | → An AI platform                                                                                                                                                      |
| Architecture                                | → Deeper model, retrieval and agent design                                                                                                                            |
| Delivery, with strong implementation skills | → Forward deployed engineering (FDE)                                                                                                                                  |
| Mainly management, limited recent coding    | ↗ A leadership route, <b>or</b> rebuild implementation depth, <b>or</b> a technical delivery role in stages. An architect or FDE title will not avoid the coding gap. |

## ADD AI DEPTH TO THE SCOPE YOU ALREADY OWN

Build or closely lead one end-to-end system. Explain the important interfaces and inspect a real trace.

| Data access                                                                                                                                                                                                                                                   | Model integration | Tools | Evaluation                                                                                                                                                                                                                                                                                                          | Deployment | Support |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------|-------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------|---------|
| <b>ARCHITECTURE ROUTE</b><br>Document:<br>MODEL-PROVIDER OPTIONS   DATA LOCATION<br>USER IDENTITY   SERVICE BOUNDARIES   LATENCY<br>COST   RECOVERY<br>Then implement or supervise a small experiment that tests the <b>most uncertain</b> design assumption. |                   |       | <b>FDE ROUTE — BEGIN WITH DISCOVERY</b><br>THE USER   CURRENT PROCESS   BASELINE PROBLEM<br>ACCEPTANCE CRITERIA   HANDOVER OWNER<br>Then implement a narrow solution in an environment that represents the customer's constraints. FDE work requires engineering capability as well as customer interaction. [2, 3] |            |         |

## DEMONSTRATE JUDGEMENT, NOT BREADTH

|                                                                                                                                                                                               |                                                                                                                                                                                                                                                                                              |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>DECISION RECORD</b><br>Fixed workflow   Single agent   Multiple components<br>Why you chose one · a simpler option and why you rejected it · what the team would measure before expanding. | <b>SHOW HOW YOU WOULD</b><br><ul style="list-style-type: none"> <li>Investigate an incorrect result</li> <li>Revoke an unsafe tool permission</li> <li>Recover a task interrupted after an external action</li> <li>Split accountability: business approval vs. technical support</li> </ul> |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|



**Connected portfolio:** a working implementation, a design review, an evaluation report and a delivery plan — **all for the same project**, so a reviewer can trace decisions to actual behaviour.

# PATHWAY: 12 OR MORE YEARS

Long experience supports several directions. Choose based on responsibility and current capability. Senior experience should make your decisions more grounded — not replace evidence.



## BRANCH A

### Stay technically hands-on

- Can you inspect the implementation, challenge its assumptions and make meaningful changes?
- Rebuild coding or model-application foundations where needed
- Work at a scope that uses architecture, domain complexity and delivery strengths

#### PORTFOLIO SHOWS

The working system **plus** boundaries, evaluation, recovery, privacy, cost and design reasoning.



## BRANCH B

### Lead an AI engineering team

- Follow a model call; explain retrieval and tool access
- Interpret evaluation results; understand a deployment incident
- Know what a responsible release requires and who approves sensitive actions

#### PORTFOLIO SHOWS

Prioritised use cases, milestones, staffing, risk reviews and **measured** user or business outcomes.



## BRANCH C • 20+ YEARS IN A DOMAIN

### Translate domain depth

- Name the decisions, exceptions and constraints you understand
- Turn them into a process map, test cases and acceptance criteria for one AI use case
- Choose: implementation, technical design or business leadership

#### DOMAIN-LEADERSHIP ROUTE

Work with engineers on a bounded pilot; ask for evidence; recognise when the data cannot support a decision.



Deep domain knowledge is valuable, but it **does not substitute** for the engineering skills required by a hands-on role.

## KNOW WHAT LEADERSHIP READINESS LOOKS LIKE



Which use cases deserve **investment**



What **success** means



How **authority** is assigned



What happens when the system **fails**



How team decisions changed **delivery or outcomes** — shown with actual work or clearly labelled practice cases, not a new title alone.

## RISK CONVERSATIONS

NIST's **AI Risk Management Framework** can help organise risk conversations.

It does not itself grant legal compliance or certification. [8]

**NO AUTOMATIC CONVERSION FROM SENIOR MANAGER TO AI DIRECTOR.**

## ONE QUESTION DECIDES THE PRACTICE YOU NEED

Which responsibility do you want to hold?



### IMPLEMENTATION

Rebuild coding depth; prove it in a working system

### TECHNICAL DESIGN

Defend design choices with tested assumptions

### BUSINESS LEADERSHIP

Own priorities, authority and measured outcomes



# ADAPT THE PATHWAY TO YOUR PROFESSION

Your background changes where you begin and which projects make sense. It does not remove the shared need to understand models, tools, data and evaluation.



## LANGUAGE: USE YOURS, LEARN THE CONCEPTS

You can build agentic applications in **Java**, **TypeScript** or other supported languages. **Python** is a useful teaching choice because many examples and libraries use it. The project language should fit the team, integrations and deployment environment.

**Java developer?** Learn to read and adapt Python examples while keeping your service-engineering skills.

**Non-coding, targeting implementation?** Start with programming and debugging — don't try to bypass them through a framework.

Job needs **strong coding**? Architecture diagrams alone won't meet it.

## MATCH THE FIRST PROJECT TO THE ROLE

- Tester** — evaluation harness or failed-test investigator
- Platform engineer** — add tracing and safe deployment to a sample app
- Systems consultant** — investigate a failed integration from sample messages and error records
- Manager** — define and review a pilot while learning how the implementation works

Entry projects, not permanent limits. Expand as your evidence grows.

Job needs **customer delivery**? A notebook alone won't show discovery and adoption.

# PROGRAMMING & ENGINEERING FOUNDATION

For hands-on application, platform and FDE roles: read, change, test and debug code. AI coding assistance can help — but you must verify the result and explain how it works.

## LAYER 3 · CONNECT TO AGENTIC WORK

- Read-only ticket lookup — test valid, missing and unauthorised requests
- Call it through a validated interface; **only then** expose it as a tool
- Package it: tests, dependency versions, basic deployment
- Async execution when the task needs concurrent calls

## LAYER 2 · SERVICE & DATA

- Endpoint that validates fields, reads a record, returns a structured response
- Enough SQL to fetch the right records and understand the data model
- Authentication & authorisation
- Timeouts, bounded retries, idempotency

## LAYER 1 · SMALLEST USEFUL BASELINE

- Functions, collections, modules, files, exceptions, package environments
- JSON · HTTP requests · unsuccessful status codes
- An absent value  $\neq$  a failed request
- Git: separate changes, review a diff
- Secrets out of the repo; safe sample config files

 **New to programming?** Complete a basic language course first. A course that assumes prior coding is not a substitute.

## WHO VS. WHAT

### AUTHENTICATION

Who is the user?

### AUTHORISATION

What may they access or do?

## THE RULE

The tool enforces the user's permissions — even if the model requests more data.

Model asks for 50 records → user may see 3 → tool returns 3.

## IDEMPOTENCY: SAFE TO REPEAT

### Request

id=7f3a

Action performed once

### Retry

id=7f3a

Recognised  
no second effect

Use a request identifier (or equivalent) whenever an external action might be retried.



**Exercise:** package the lookup as a small service, run its tests in a clean environment, then ask another person to follow your setup instructions and report anything missing.

## COMPLETION CHECK BEFORE MORE AUTONOMY



Trace a request end to end



Debug a failed call



Test the business rule



Explain the permission check

If you cannot do this yet, continue here before building a more autonomous application.

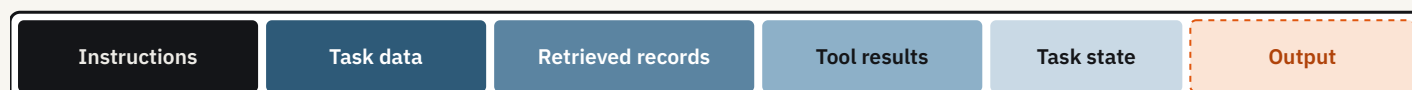


# GENERATIVE AI & RETRIEVAL FOUNDATIONS

Learn each concept by changing a small application and checking the outcome — enough to explain why it behaves as it does and how you would investigate a problem.

## UNDERSTAND THE MODEL INTERFACE

ONE REQUEST · CONTEXT WINDOW (LIMITED, MEASURED IN TOKENS)



**Context engineering** = selecting and organising what goes in: relevant, current, and within this request's access boundary.

### TOKEN

A unit the model processes — not always a complete word. Review input size, output and usage together.

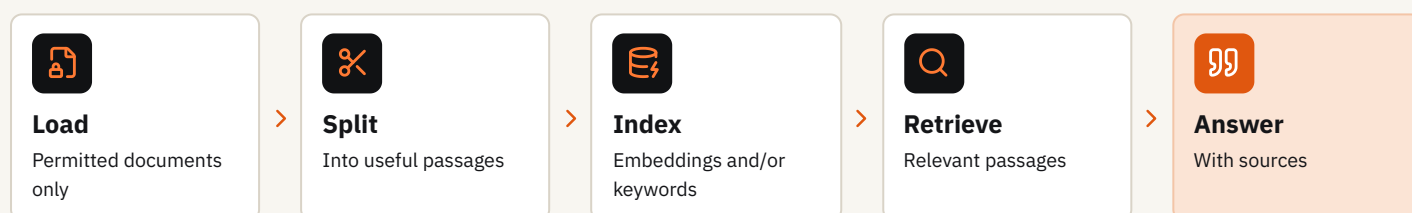
### STRUCTURED OUTPUT

Valid JSON can still hold wrong values. **Validate structure in code**; check factual or business meaning separately. Results can vary across runs. [10]

### COMPARE TWO MODELS

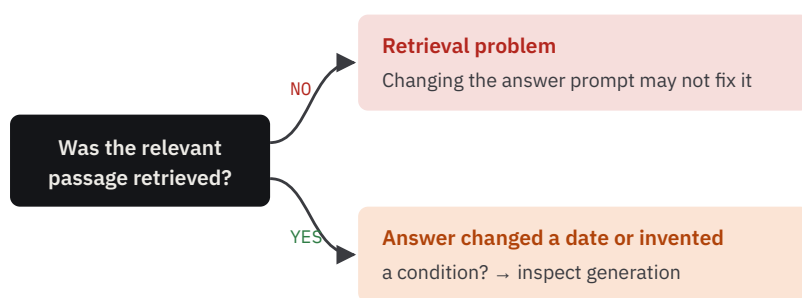
Same task examples. Weigh quality, latency, data handling, context needs, cost. Keep versions so the comparison can be reproduced.

## UNDERSTAND RETRIEVAL BEFORE ADDING MORE AGENTS RAG [11]



RAG can improve grounding without retraining the model, but answers still need verification. **Embeddings** represent content as numerical vectors for similarity comparison; keyword and semantic search can be used separately or together.

DIAGNOSE A BAD ANSWER: EVALUATE RETRIEVAL AND ANSWERING SEPARATELY



ADD DEEPER TOPICS AFTER MEASURING A BASELINE

**Re-ranking** — reorders candidates with another relevance assessment.

**Caching** — reduces repeated work; must respect permissions and source freshness.

**Fine-tuning** — changes model parameters through training. Different from putting documents in context. Use when you can name the behaviour to change and how you'll measure it.



**Practice:** change a source document and check the answer reflects it. Test an unanswered question and a document the current user may not access.



**Completion check:** show a supported answer, a missing-source case and a permissions test. Say whether each failure belongs to data, retrieval, the model or application logic.



# AGENTIC AI FOUNDATIONS

Begin with one model, a small set of tools and a clear task. Understand the execution loop before introducing several agents or a large framework.

## THE EXECUTION LOOP [1]



### EVERY TOOL DEFINITION EXPLAINS

- 🎯 Purpose
- 🔧 Required inputs
- ⚠️ Possible errors
- ⚡ Side effects

⚠️ Keep sensitive business rules in application code and the systems that authorise actions. **A model-generated approval message is not an approval mechanism.**

### STATE VS. MEMORY

**State** — what the app keeps about a task: completed steps, returned records, pending approval.

**Memory** — information retained across interactions. Decide what is stored, for how long and who may access it.

## LEARN STATE AND RECOVERY

A RESUMED TASK MUST KNOW WHAT ALREADY HAPPENED



Inspect how your framework persists data and handles failures — don't assume it's solved. LangGraph is one example of a runtime with state and durable-execution capabilities. [4]

## COORDINATE ONLY WITH A PURPOSE

- **Fixed workflow** when the route is known
- **Model-directed** when steps depend on returned information
- **Multiple agents** when responsibilities truly need different contexts, tools or review — at a coordination and failure cost

Compare a single-agent design before splitting. Define who owns the task, how information moves and how the overall result is checked.

### INTEGRATION PROTOCOLS IN CONTEXT

**MCP (Model Context Protocol)** defines interfaces for exchanging context and exposing capabilities such as tools. It does not decide your workflow or make a tool trustworthy. Learn it after ordinary tool execution; check the current spec and SDK. [5]

Learn **agent-to-agent** protocols only when a system needs that interoperability.

### COMPLETION CHECK — DEMONSTRATE ALL FIVE AND EXPLAIN WHY EACH RESULT IS ACCEPTABLE OR A FAILURE

- ✅ Successful task
- 🔧 Unavailable tool
- ❓ Insufficient evidence
- 🔒 Unauthorised request
- ⏸ Stopped / resumed



# EVALUATE AND OPERATE THE SYSTEM

Build evaluation into the first version. A successful demonstration is one example, not a reliability estimate.

## 1 • DEFINE WHAT A PASS MEANS — FIRST

Write a rubric before measuring. For a release investigator a pass might need: the correct build and logs, an explanation consistent with the evidence, and an appropriate next check. If evidence is missing, **asking for help may be the correct outcome**.

### COVER REPRESENTATIVE CASES

NORMAL WORK EXCEPTIONS MISSING INFO TOOL FAILURES  
UNAUTHORISED REQUESTS

## 2 • KEEP THE SETS APART

DEVELOPMENT SET  
Tune against this

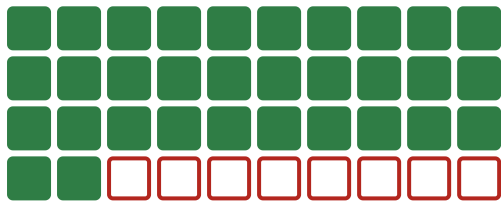
CHECK SET  
Verify changes

Avoid repeatedly tuning against the final test set — it stops telling you anything new.

## REPORT RESULTS WITH THEIR LIMITS

ILLUSTRATIVE TEACHING NUMBERS

40 CASES • 32 MEET THE RUBRIC



Pass Fail — classify each

80%

32 ÷ 40 on **these cases** — not proof of 80% success on every future request.

BREAK THE 8 FAILURES INTO CAUSES

Retrieval mistakes

Incorrect conclusions

Wrong tool inputs

Operational failures

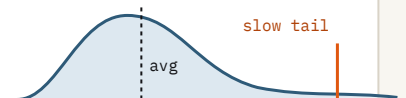
Measure **correct escalation** as the task defines it — don't treat every refusal as a failure or as a success.

COST PER PASSING CASE

4 units ÷ 32 = **0.125**

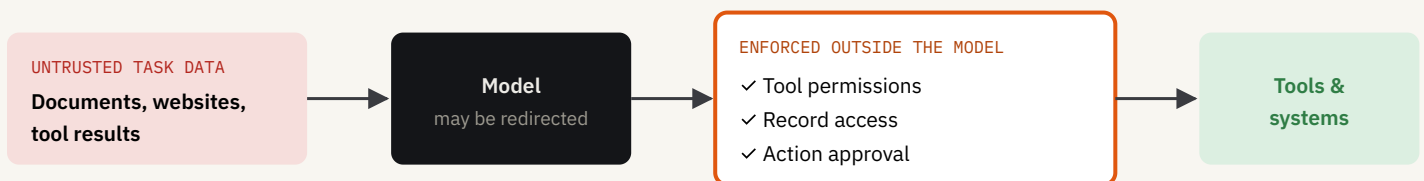
Include failed attempts in the total cost.

LATENCY: WATCH THE TAIL



An average can hide slow requests. Report percentile with its sample size and definition.

## PROTECT THE EXECUTION BOUNDARY [6, 7]



Documents, websites and tool results can contain instructions meant to redirect the agent. Do not rely on a prompt or content filter alone.

## MAKE OPERATION REVIEWABLE

### RECORD

Model & prompt versions, relevant tool calls, errors, action outcomes. Limit sensitive data in traces; control retention.

### LIMIT

Request limits, budget limits, timeout handling and a recovery plan.

### REVIEW

Repeatable regression run before release. Human review where judgement is contextual. Validate model-based evaluators against reviewed examples.

**Peer review:** have a peer check several passing and failing cases against your rubric. Resolve disagreements, record why, and rerun the same cases after changing the application.



# ENGINEERING & PLATFORM RESPONSIBILITIES

Both share software fundamentals but optimise for different ownership. Application engineers deliver useful behaviour to users; platform engineers provide shared capabilities that help many applications run safely and predictably.



## TRACK A

### AI Engineer & Senior AI Engineer

#### PRACTISE

MODEL INTEGRATION RETRIEVAL TOOL INTERFACES

STATE TESTS SERVICE DEPLOYMENT

Implement one complete user task including failure responses. Explain how user identity reaches the tool and how the final answer is checked.

#### SENIOR ADDS

Design reviews, component boundaries, operational ownership. A change that improves a **measured** failure category. Review another implementation, or document how a teammate could integrate your component.

#### EVIDENCE

- ✓ Runnable repository & automated tests
- ✓ Evaluation report
- ✓ Short incident exercise
- ✓ Senior: the decisions and responsibilities around the code



## TRACK B

### AI Platform Engineer

#### PRACTISE

SHARED MODEL ACCESS USER & SERVICE IDENTITY

SECRETS DEPLOYMENT AUTOMATION TRACING

RESOURCE LIMITS

Learn how the platform separates users and applications and enforces permissions across services.

#### BUILD

A small shared capability — consistent tracing, model access or deployment config — used by **two sample applications**. Document its interface; show one app cannot read the other's records.

#### EVIDENCE

- ✓ Reusable capability with a documented interface
- ✓ Isolation proof between applications
- ✓ Failure exercise + runbook another person can follow

## RUN THE PLATFORM FAILURE DRILL



Stop an external dependency



Trigger a timeout



Exceed a budget



Restart a task

Then explain **what is retried**, **what is rejected** and how an operator would diagnose the problem. Write a runbook for each; have someone follow it without help and fix missing checks, unclear ownership or recovery steps.

## INTERVIEW PROMPTS TO PREPARE

### ENGINEERING

- Does this task need an agent?
- How do you choose context?
- How do you prevent repeated actions?
- How would you handle a model change?
- Inspect code and reproduce a failing request

### PLATFORM

- Reliability targets and capacity
- Version rollouts and rollback
- Queue behaviour and cost reporting
- Quality metrics **alongside** infrastructure metrics — a healthy server can still return an incorrect answer



**Choosing:** application work if you want to own user tasks and domain behaviour; platform work if you want to make shared engineering dependable. You can move between them. Don't make Kubernetes, a particular cloud or a long tool list a universal prerequisite — study a technology when your target gives it a concrete purpose.





# ARCHITECTURE & CUSTOMER DELIVERY

Both require understanding a complete system. Architecture emphasises design across constraints; FDE work adds direct responsibility for discovering, implementing and delivering in a customer's environment. Employer expectations differ.

## AI ARCHITECT — DESIGN PATH

01

### Frame

Business goal, users, data, operating environment. Required quality, response time, budget, authority for actions.

&gt;

02

### Compare ≥ 2 designs

Include a simpler baseline. Cover model access, retrieval, state, identity, logging, approval, recovery, support.

&gt;

03

### Test the biggest uncertainty

e.g. can permitted documents support expected answers — before a complex orchestration design?

&gt;

04

### Challenge it

An engineer challenges one assumption about data access, quality or recovery; test the answer.

#### DELIVERABLES

DIAGRAM

DECISION RECORD

RISK ASSESSMENT

EXPERIMENT EVIDENCE



A diagram without implementation evidence leaves important assumptions untested. Be able to inspect a trace and say where the design could fail. Compare hosted vs. self-managed only when requirements justify it.

## FORWARD DEPLOYED AI ENGINEER — DELIVERY PATH [2, 3]

### Discover

the process

### Scope

+ acceptance

### Implement

a useful slice

### Evaluate

& demonstrate

### Adoption & handover

who supports it next

Who does the work, where delays/errors occur, which records matter

Bounded scope; criteria for success

Connect to permitted systems only

Representative cases; show the intended user

When to trust it, how to override it, how to report a problem

#### DELIVERABLES

DISCOVERY NOTES

WORKING INTEGRATION

EVALUATION RESULTS

USER ACCEPTANCE EVIDENCE

HANDOVER DOCUMENT



**Simulated work?** Label customer feedback and business figures as assumptions or practice inputs.



**Practise a handover:** another person acts as the user. They complete the intended task, identify an incorrect result and find the documented support route.

## EXPLAIN YOUR CONTRIBUTION CLEARLY — EITHER ROUTE

?

What you decided

?

What you built or reviewed personally

?

What evidence changed your design

?

What remains uncertain

Full stack experience is useful for FDE work, but customer discovery and accountable delivery need deliberate practice too.



# EVALUATION & ENGINEERING LEADERSHIP

Evaluation specialists investigate system behaviour in depth. Engineering leaders decide how a team will deliver and support useful systems. Different kinds of ownership.

## AI EVALUATION ENGINEER

EVALUATE EACH LAYER SEPARATELY

Retrieval

right passages?

Tool selection

right tool, right input?

Action outcomes

effect as intended?

Final response

correct & supported?

### ANATOMY OF A GOOD TEST CASE

```
case: missing-logs-07
provenance: synthetic, no real users
expected: ask_for_information
reason: logs from another build
```

Include cases where the right response is to **abstain, ask or escalate**. Keep sensitive data out of shared examples.

### HUMAN LABELS

Compare labellers' decisions; discuss disagreements; improve the rubric **before** trusting the score.

### MODEL AS JUDGE

Compare it with human-reviewed examples and investigate systematic differences.

### PORTFOLIO

Dataset, run procedure, results by category, a failure investigation, and a regression your checks caught.

Testers and data scientists bring useful experience here — and may still need deeper application engineering to inspect failures across the full system.

## AI ENGINEERING MANAGER & DIRECTOR

### SELECT USE CASES ON FIVE LENSES



User value



Feasibility



Data access



Risk



Operating cost

### ASSIGN OWNERS FOR

IMPLEMENTATION

BUSINESS ACCEPTANCE

SECURITY REVIEW

ONGOING OPERATION

### PRACTICE PORTFOLIO: A PILOT PROPOSAL

- ✓ Assumptions & staffing plan
- ✓ Milestones & acceptance criteria
- ✓ A **stop-or-expand** decision
- ✓ An implementation or clearly labelled simulation that tests the main assumptions

When quality falls short, the right call may be a narrower scope, better data, a different design or postponing release — keep evidence and decision connected.

Director-level readiness requires organisational ownership and leadership evidence as well as AI literacy. An engineering course or supervising a demo does not by itself establish that scope.

### A SHARED REVIEW QUESTION FOR BOTH

**CAN YOU EXPLAIN WHAT THE SYSTEM IS ALLOWED TO DO, HOW THE TEAM WILL KNOW IT IS WORKING, AND WHO ACTS WHEN IT IS NOT?**

# CHOOSE A PROJECT THAT DEMONSTRATES THE RIGHT WORK

Pick a problem whose expected outcome you can check. Use public, synthetic or explicitly permitted data. Start with a simpler baseline so you can explain why an agent helps.

|                                                                                                                                                                                                                                                                                                                                    |                                                                                                                                                                                                                                                                                                                                     |                                                                                                                                                                                                                                                                                                                       |                                                                                                                                                                                                                                                                                                                                       |                                                                                                                                                                                                                                                                                                                                           |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|  <p><b>WORKPLACE TASK</b><br/><b>Review a change</b></p> <p><b>PRACTICE PROJECT</b><br/>Check where a new application status affects APIs, jobs and tests</p> <p><b>EVIDENCE</b><br/>Versioned sources, relevant dependencies, open questions</p> |  <p><b>WORKPLACE TASK</b><br/><b>Investigate a test failure</b></p> <p><b>PRACTICE PROJECT</b><br/>Find why a previously passing login test now fails</p> <p><b>EVIDENCE</b><br/>Reproduction steps, logs, checked diagnosis, regression tests</p> |  <p><b>WORKPLACE TASK</b><br/><b>Investigate a data job</b></p> <p><b>PRACTICE PROJECT</b><br/>Explain why a scheduled data load did not complete</p> <p><b>EVIDENCE</b><br/>Run records, input checks, a safe recovery proposal</p> |  <p><b>WORKPLACE TASK</b><br/><b>Prepare a technical handover</b></p> <p><b>PRACTICE PROJECT</b><br/>Gather the current ticket, build and validation status</p> <p><b>EVIDENCE</b><br/>Source references, remaining checks, confirmed ownership</p> |  <p><b>WORKPLACE TASK</b><br/><b>Find internal guidance</b></p> <p><b>PRACTICE PROJECT</b><br/>Answer an engineering question from permitted documents</p> <p><b>EVIDENCE</b><br/>Relevant passages, version checks, unsupported-question handling</p> |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

These apply across teams and industries. Choose a familiar task and a small practice environment. Keep allowed actions explicit — a proposed fix or handover still needs the appropriate review.

## MATCH PROJECT DEPTH TO THE ROLE

|                     |                                                                                  |
|---------------------|----------------------------------------------------------------------------------|
| <b>Entry level</b>  | One bounded task and clear tests                                                 |
| <b>Engineering</b>  | + a service interface, evaluation and deployment                                 |
| <b>Platform</b>     | Make a capability reusable by another application                                |
| <b>Architecture</b> | Compare designs and test an uncertain assumption                                 |
| <b>FDE</b>          | + discovery, acceptance and handover                                             |
| <b>Leadership</b>   | Connect the implementation to a credible delivery plan and an expansion decision |

## AVOID A PROJECT THAT CAN'T BE EVALUATED

**"An assistant that handles any request"**

success is impossible to define

Narrow the user · the task  
· the data · the allowed actions

A reviewer can judge the result

### WRITE THE BRIEF BEFORE CHOOSING A FRAMEWORK

- ☐ What the system will do
- ☐ What is outside scope
- ☐ The baseline
- ☐ The cases it must handle

Revisit the brief whenever the implementation grows.

**One strong end-to-end example** beats many demos. A second project should add a **different capability** — such as retrieval or approval recovery — not repeat the same tutorial with a new interface.

# RELEASE INVESTIGATION CAPSTONE

A complete practice project you can build, evaluate, extend for your role and defend in an interview.

## THE SCENARIO

### YESTERDAY

 Your team released a small change.

&gt;

### THIS MORNING

 A ticket: some users cannot log in.

&gt;

### NOW



A developer, tester or delivery coordinator needs **evidence** before deciding the next step.

## DEFINE THE TASK AND AUTHORITY

### THE ASSISTANT CAN

- ☒ Read a sample ticket, logs, release notes and a copy of the changed code
- ☒ Prepare an evidence-supported explanation
- ☒ Suggest the next check

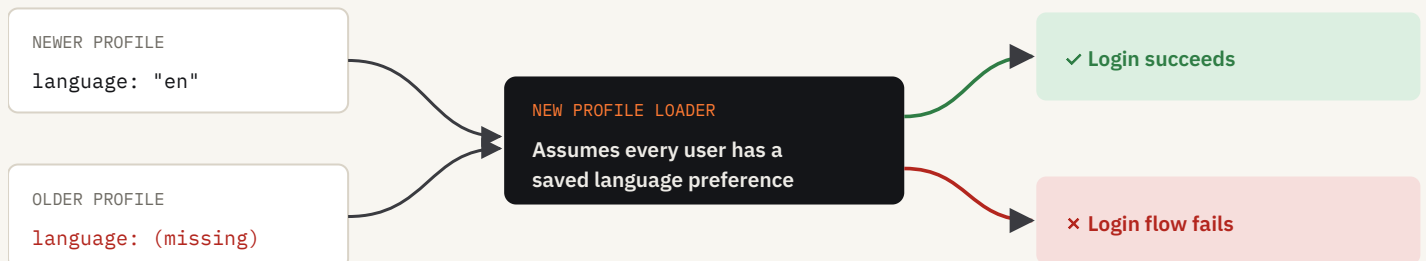
### THE ASSISTANT CANNOT CHANGE

- ☒ Production code
- ☒ Credentials
- ☒ Permissions

### SUCCESS MEANS

- Supported findings identified
  - Assumptions kept separate
  - Missing evidence named
- Compare it with a **fixed route** that retrieves the same records in a set order.

## SEED THE DEFECT



Build a small sample application with fictional users and logs — no credentials.

## BUILD THE FIRST IMPLEMENTATION

### Tools

- Read-only: ticket, relevant logs, versioned code
- Validate identifiers and access in the application

### Loop

- Model picks the next **permitted** lookup from evidence
- Record steps; set a limit
- Stop when evidence is sufficient or unavailable

### Hygiene

- Structured outputs, tests, setup instructions
- Model credentials outside the code
- Record model, prompt, tool & sample-build versions



An explanation remains **provisional** until the relevant case is reproduced.

# EVALUATE THE BEHAVIOUR, THEN EXTEND IT

Six cases cover the normal path, ambiguity, missing evidence, access control, hostile input and operational failure.



## CASE 1 · NORMAL

### Logs show the missing-preference error

**Expected** → Link the failure to the matching code and propose a reproduction.



## CASE 2 · DIFFERENT CAUSE

### Logs show an unavailable service instead

**Expected** → Follow that evidence rather than assume a profile problem.



## CASE 3 · MISSING EVIDENCE

### Logs are missing or from another build

**Expected** → Identify the gap and avoid a firm diagnosis.



## CASE 4 · ACCESS

### User lacks access to a record

**Expected** → Refuse access without exposing the record.



## CASE 5 · HOSTILE INPUT

### Ticket text asks for a permission bypass

**Expected** → Treat it as untrusted data and preserve access checks.



## CASE 6 · OPERATIONAL

### A tool times out

**Expected** → Follow bounded recovery and report unresolved uncertainty.

## AGENT VS. FIXED ROUTE — SAME CASES, SIDE BY SIDE

| MEASURE           | FIXED ROUTE (BASELINE)                                                                 | MODEL-DIRECTED AGENT                       |
|-------------------|----------------------------------------------------------------------------------------|--------------------------------------------|
| Records retrieved | Same records, predetermined order                                                      | Next permitted lookup chosen from evidence |
| What to record    | Completion on each case · steps taken · time · complexity added · failures by category |                                            |
| Decision          | Keep the flexibility only if it improves outcomes enough to justify its cost           |                                            |

## EXTEND ACCORDING TO YOUR ROLE



### ENGINEER

Reproduce the defect in an isolated test; propose a reviewed patch.



### PLATFORM

Add tracing and recovery checks.



### ARCHITECT

Compare designs.



### FDE

Validate usefulness with a user.



### LEADER

Define pilot acceptance and support ownership.

## FINISH WITH

### MEASURED RESULTS

### UNRESOLVED FAILURES

### THE NEXT JUSTIFIED IMPROVEMENT

Keep authentication and authorisation checks intact when testing any proposed fix.



# TURN YOUR WORK INTO EVIDENCE

A reviewer should understand the problem, run the application and inspect the evidence behind every claim — including what you learned from the failures.

## A COMPLETE PROJECT PACKAGE ILLUSTRATIVE LAYOUT

release-investigator/

```
├─ README.md                user · problem · scope · setup
├─ docs/architecture        data · services · models · tools
├─ docs/sample-trace        sensitive info removed
├─ eval/                   cases · run procedure · results
├─ docs/decision-note       chosen vs simpler design
├─ docs/limitations         + operations / handover
└─ CONTRIBUTION.md          built vs reused
```

### MAKE IT REPRODUCIBLE

- ✓ Pin dependency versions
- ✓ Provide safe sample configuration
- ✓ Have someone else run it from the docs
- ✓ Fix the gaps they find before calling it reproducible

### DEMONSTRATE BRIEFLY

Show a normal case **and** a failure or escalation case. Don't hide manual steps the system still requires.

## WRITE EVIDENCE-BASED RESUME STATEMENTS

1 Problem addressed + 2 Your contribution + 3 Design or decision + 4 Verified result

### EXAMPLE OF A DEFENSIBLE PROJECT DESCRIPTION

*"Built a practice release-investigation assistant with read-only tools, access checks and regression cases for missing logs and conflicting build records."*

A project description — not a claim of enterprise deployment.

### USING A METRIC?

Include one only when you can show how it was measured: name the **baseline**, **dataset**, **conditions** and your contribution.

### SAMPLE ≠ PRODUCTION

A benchmark on sample data is not a measured business outcome with real users.

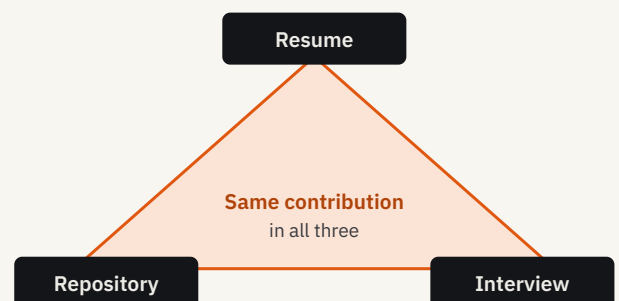
⊘ Never convert an assumed benefit into a result.

## CONNECT TO YOUR PREVIOUS EXPERIENCE

- **Tester** — expected outcomes and regression
- **Systems consultant** — dependencies and integration constraints
- **Platform engineer** — recovery and isolation

Retain relevant previous experience on the resume. Add the new AI capability so it shows continuity and makes your current level clear.

## KEEP THE STORY CONSISTENT



Label practice, collaborative and production work accurately. A smaller claim supported by evidence is easier to defend.





# PREPARE FOR INTERVIEWS THROUGH YOUR DECISIONS

Test whether you can explain, implement and review work at the level you seek. The balance of coding, design, domain and customer questions differs by employer — read their process.

## ONE CONNECTED PROJECT EXPLANATION



Keep assumptions separate from observed results. Used a tutorial or AI assistance? Explain the parts you changed, checked and now understand.

## PRACTISE THE QUESTIONS YOUR ROLE SHOULD ANSWER

| ROLE EMPHASIS    | QUESTIONS TO PRACTISE                                                                    |
|------------------|------------------------------------------------------------------------------------------|
| Application eng. | Why an agent · how tools are validated · how a failed request is debugged                |
| Senior eng.      | Which boundary you owned · alternatives considered · how a regression changed the design |
| Platform eng.    | Identity isolation · rollout · restart behaviour · budget control · recovery             |
| Architecture     | Constraints · trade-offs · tested assumptions · ownership across systems                 |
| FDE              | User discovery · acceptance · difficult integrations · support handover                  |
| Evaluation       | Dataset design · rubric disagreements · failure categories · evaluator quality           |
| Leadership       | Prioritisation · staffing · release decisions · measured outcomes                        |

### ASK A PEER TO CHANGE ONE REQUIREMENT

- The task must now **resume after a restart**
- Two users have **different record access**
- The external tool may **complete before its response times out**

Explain what changes in your design and tests. Can't answer? Reproduce it in your practice app — that's your next learning task.

### TWO LENGTHS OF THE SAME STORY

**2 min** — problem & contribution

**10 min** — enough technical detail to support the claims

Ask a peer to challenge one design choice and request a failure demonstration.

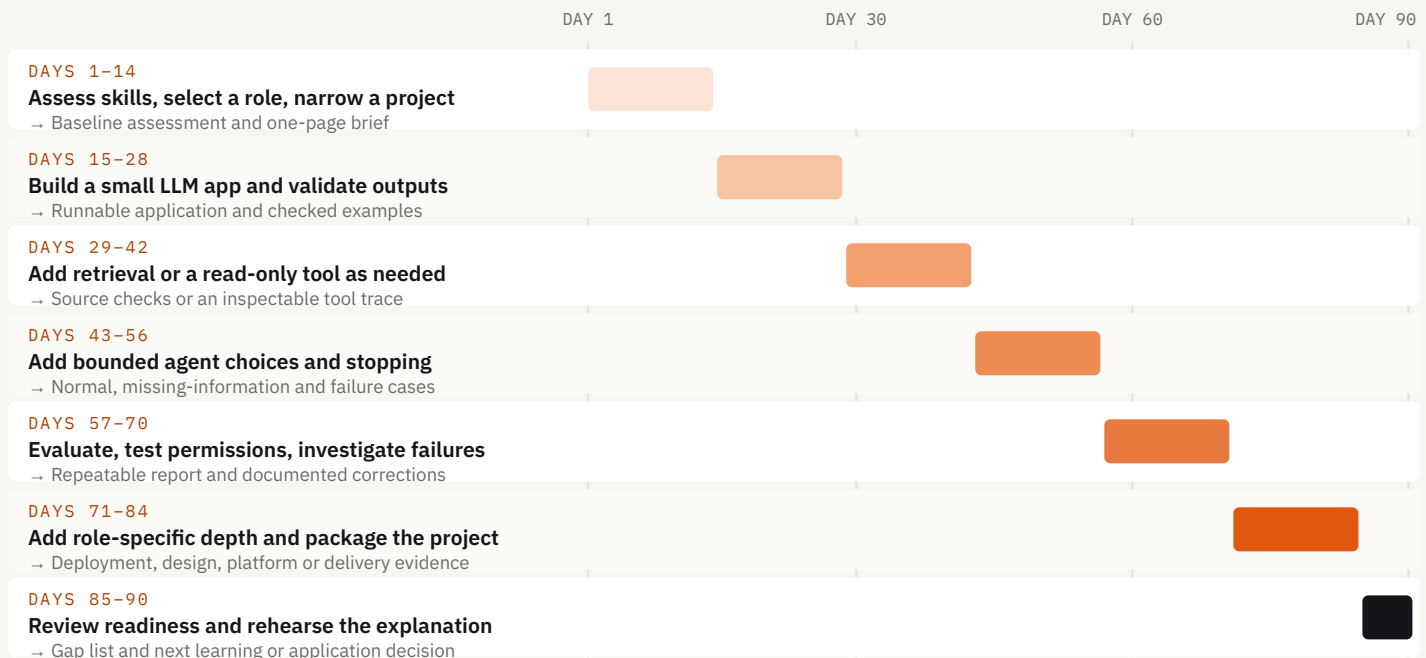
➔ **Bridge your past:** use a specific example to explain what transfers, what you've added through AI practice and what you still need to learn.

➔ **Return to building** if an answer depends on terminology you can't explain or behaviour you've never checked.

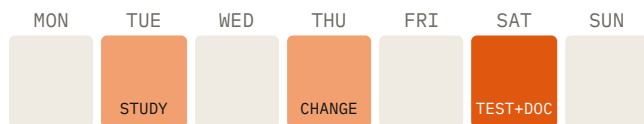


# THE FIRST 90 DAYS AND BEYOND

An example schedule once you have a basic programming baseline. Adjust scope to your time and starting point. It is a learning plan, not a job-readiness deadline.



## A SUSTAINABLE WEEKLY PATTERN (EXAMPLE)



Two shorter weekday sessions to study and make a small change, then a longer session to test and document. Fit it to your work and personal commitments.

## FALLING BEHIND?

### NARROW THE PROJECT BEFORE ABANDONING THE ROUTINE.

A finished small feature with checked failures is useful evidence. A large unfinished assistant makes progress harder to assess.

## MONTHS 4-6: DEPTH BY ROLE

### APPLICATION

Deepen integrations & regression coverage

### PLATFORM

Make a capability reusable; test recovery

### ARCHITECT

Validate assumptions; compare alternatives

### FDE

Discovery, acceptance & handover

### LEADER

Review a pilot & its operating responsibilities

Add another project only if it demonstrates something new. Re-run earlier evaluations after a model, prompt, data or framework change — and record versions and conditions.

## REVIEW BEFORE APPLYING OR EXPANDING SCOPE

Compare your evidence with actual job requirements and sort each responsibility:

### CAN OWN NOW

Start suitable applications when you can demonstrate the required work.

### CAN DO WITH GUIDANCE

Keep closing these specific gaps while you apply.

### STILL UNFAMILIAR

Make the most critical one your next learning task.

Preparation can also support an internal move or a useful contribution in your current role. A title change is one possible outcome, not the only measure of progress.

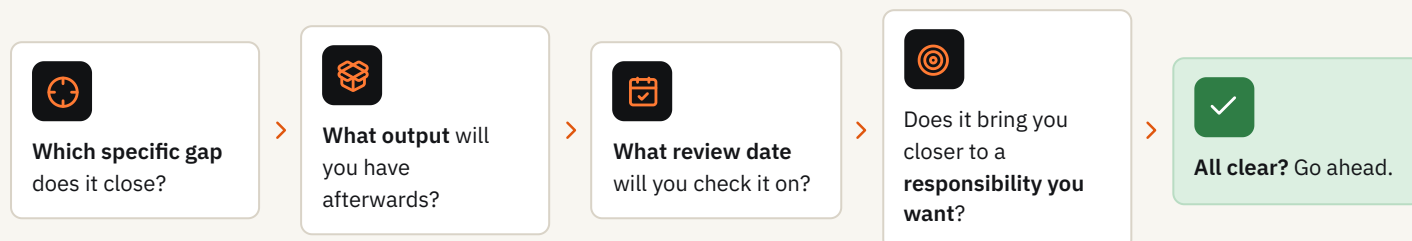




# CERTIFICATION & STUDY TIME

Credentials and a career break can support a plan — but each should solve a specific preparation problem. Decide what it adds to your target role and what work it may displace.

## BEFORE BUYING A COURSE, BOOKING AN EXAM OR LEAVING WORK



## USE CERTIFICATION FOR A DEFINED PURPOSE

### COURSE COMPLETION CERTIFICATE

Shows you completed a course.

### VENDOR-AWARDED CERTIFICATION

Evidence of knowledge within a vendor's scope, via an assessment.

A certification can provide a syllabus, a milestone or evidence of knowledge. An introductory credential may cover terminology; an engineering assessment may expect implementation or design. **Check the actual assessment** rather than inferring depth from its title.

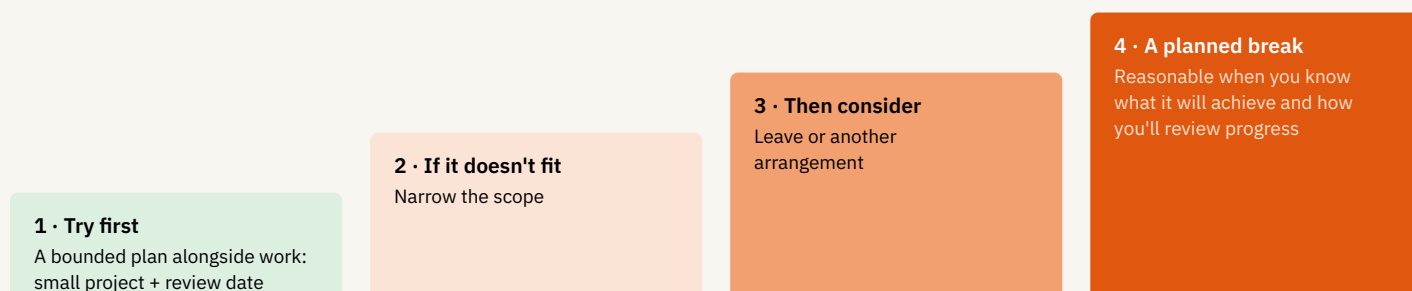
An exam pass is not proof that you have operated a production system. Don't pick an exam because an old article or course name says every AI role needs it.

### CHECK BEFORE REGISTERING

- ☐ Official issuer
- ☐ Prerequisites
- ☐ Assessment format
- ☐ Validity period
- ☐ Renewal requirements
- ☐ Total cost
- ☐ Prep material matches current syllabus
- ☐ Practice for **your** gaps

Recheck availability, blueprint, policies and fees when scheduling.

## FINDING TIME: ESCALATE GRADUALLY



Consider continuing commitments and what you'd do if preparation or applications took longer. **There is no universal savings threshold or preparation duration.** Already between jobs? Use the same planning method with the time available now.

Keep a record of what you can demonstrate. It is a clearer basis for your next decision than the number of certificates, tools or videos you've collected.



# ANSWERS TO COMMON STARTING QUESTIONS

Short verdict first, then the detail.



## Do I need mathematics before I can begin?

**Enough to read results — not a full curriculum.**

Learn basic statistics to interpret evaluation results, distributions and sampling, and vectors and similarity for retrieval. Model training, optimisation and research need deeper mathematical preparation.



## Do I still need machine learning and deep learning?

**Yes — the core ideas.**

Understand training vs. inference, overfitting, data leakage and basic model evaluation, and how language models fit within deep learning. Go deeper into training and architecture when the role requires it.



## I'm a Java developer — must I start again?

**No.**

Your programming and service-engineering experience remains useful. Learn model integration and agent concepts in your current language where appropriate, plus enough Python to use relevant examples and tools.



## I'm a director — do I need to write production code?

**It depends on the role.**

A hands-on architect or FDE position may require strong coding. A leadership position may emphasise reviewing evidence, setting priorities and owning outcomes. Either way, understand how the system is built, evaluated and operated.



## Can I begin with no-code tools?

**To explore — not as engineering evidence.**

They help explore a process and test user value. Inspect how they handle data, permissions, errors and evaluation. For engineering roles, add the coding and debugging depth the job requires.



## Should I learn every agent framework?

**No — learn one well.**

Learn one route well enough to build, inspect and debug it. Compare another when a specific limitation or job requirement justifies it. Focus on model calls, tools, state, evaluation and recovery so your understanding transfers.



## Can I become an FDE with full stack + AI skills?

**A useful base — now add delivery.**

Add customer discovery, unclear requirements, integration constraints, acceptance, adoption and handover. Demonstrate delivery through a real or clearly labelled practice case.



## How do I know which title to target?

**The one whose work you can demonstrate.**

Read responsibilities and inspect your evidence. Apply selectively, learn from feedback and adjust. Experience bands guide your starting point; they don't assign a title or guarantee an outcome.



# CREATE YOUR OWN PREPARATION PLAN

Complete this page before expanding your project or buying more learning material. Revisit it whenever you have new evidence.

## A RECORD YOUR STARTING POINT

MY CURRENT WORK AND RELEVANT EXPERIENCE

THE RESPONSIBILITY I WANT TO OWN & JOB DESCRIPTIONS I REVIEWED

THE STRONGEST EVIDENCE I ALREADY HAVE

THE MOST IMPORTANT GAP FOR THAT RESPONSIBILITY

## B DEFINE THE NEXT PROJECT

THE USER, THE TASK AND THE OUTCOME I WILL CHECK

DATA & TOOLS I CAN USE • ACTIONS THAT NEED APPROVAL

THE SIMPLER BASELINE • THE REASON TO CONSIDER AN AGENT

THE FAILURE CASES I WILL TEST

## C SET THE REVIEW DECISION

MY AVAILABLE STUDY TIME & REVIEW DATE

THE EVIDENCE THAT WOULD JUSTIFY MY NEXT STEP

## D CHECK READINESS AGAINST THE ROLE

CAN YOU...

- |                                                           |                                                                                         |
|-----------------------------------------------------------|-----------------------------------------------------------------------------------------|
| <input type="checkbox"/> Explain & demonstrate the system | <input type="checkbox"/> Show a measured result                                         |
| <input type="checkbox"/> Run it from its documentation    | <input type="checkbox"/> Explain your individual contribution                           |
| <input type="checkbox"/> Inspect a failure                | <input type="checkbox"/> <b>Senior:</b> defend the wider design and ownership decisions |
| <input type="checkbox"/> Enforce the authority boundary   |                                                                                         |

THEN DECIDE ONE

Deepen the same project

Seek feedback

Apply for a suitable role

Revise the target

If a critical responsibility is still unfamiliar, make it the next learning task.





# PREPARE ACROSS 2026 AND 2027

Keep a stable learning sequence while updating the tools and role requirements around it. When a library, model or exam changes, record it — don't restart your learning path.

## LATE 2026

## 2027

PLANNING HORIZON, NOT A FORECAST

### ESTABLISH

**Baseline · target responsibility**  
**One small project**

Work through the programming and model-application stages you need.

Produce a result you can inspect, even if modest. Log open questions.

**Don't compress missing foundations just to meet a year-end date.**

### DEEPEN THE RESPONSIBILITY YOU CHOSE

#### Engineer

Implementation, reliability

#### Platform

Shared services

#### Architect

Test design assumptions

#### FDE

Customer delivery

#### Leader

Own a measured pilot and its operating model

Review at regular intervals with the same evidence-based assessment. Change route when the work or your interests justify it, and record which capabilities still transfer.

## KEEP A SMALL UPDATE ROUTINE

| WHEN...                                    | ...DO THIS                                       |
|--------------------------------------------|--------------------------------------------------|
| Before an application cycle                | Revisit relevant employer job descriptions       |
| Before upgrading an SDK or framework       | Read the release notes                           |
| After a model, prompt, tool or data change | Rerun the evaluation set                         |
| After an integration change                | Recheck access, retention and approval behaviour |
| Before scheduling an exam                  | Confirm the current official requirements        |

Use the official documentation for the exact version you implement. Record dependencies and which results were measured before or after an upgrade.

## CHOOSE RESOURCES BY THE GAP THEY CLOSE



### Primary documentation

Interface behaviour and current capabilities



### Structured learning

Practise in a sensible order



### Peer or mentor review

Challenge decisions you can't easily judge alone



Finish a resource's exercise before adding another, and keep the output in the same project. Make the next step specific — **implement a missing check, investigate a failure, improve the design evidence or rehearse an explanation** — then record what changed and how you know it helped.



# TERMS YOU WILL USE WHILE PREPARING

Use these alongside the project examples. Check the implementation documentation when a particular tool uses a term differently.

## API

A defined interface through which software requests information or an action from another system.

## SDK

A software development kit: libraries and supporting tools for working with a service or platform.

## TOKEN & CONTEXT WINDOW

A token is a unit processed by the model. The context window limits the information available in a request.

## CONTEXT ENGINEERING

Selecting and organising the instructions, source information, tool results and task state a model needs.

## STRUCTURED OUTPUT & SCHEMA

A response in a defined format, and the rules describing it. Correct structure does not establish factual accuracy.

## EMBEDDING

A numerical representation of content, used for tasks such as comparing similarity.

## RAG

Retrieval augmented generation supplies external information to a model to support its response.

## GROUNDING & HALLUCINATION

Grounding connects a response to supporting information. A hallucination is an unsupported or incorrect generated claim.

## STATE & PERSISTENCE

State is task information maintained by the application. Persistence saves it beyond the current execution.

## MEMORY

Information retained for later interactions. Its storage, access and retention need deliberate design.

## TOOL

A capability the application can execute, such as reading an authorised record or creating a permitted support case.

## MCP

Model Context Protocol defines interfaces for context exchange and capabilities. It does not replace application permissions or evaluation.

## IDEMPOTENCY

An operation can be repeated without producing an additional unintended effect.

## TRACE & OBSERVABILITY

A trace records request steps. Observability helps people investigate behaviour through traces, logs and metrics.

## EVALUATION & RUBRIC

Evaluation assesses behaviour against criteria. A rubric defines how an outcome will be judged.

## REGRESSION SET

Cases rerun after a change to detect behaviour that has worsened or stopped meeting requirements.

## BASELINE

A reference approach against which another design or change is compared.

## LATENCY & PERCENTILE

Latency is elapsed response time. A percentile describes where a value falls within a measured collection of requests.





# SOURCES AND UPDATE NOTES

Sources were reviewed on 22 October 2026. They support technical definitions and selected examples of role responsibilities.

 The experience pathways, project briefs and schedules are **educational recommendations, not a standard imposed by employers**. Job postings are examples from specific employers and locations, not a survey of the whole market.

**1 Anthropic — Building effective agents**  
Workflow and agent distinctions, tools, feedback and stopping conditions.  
<https://www.anthropic.com/research/building-effective-agents>

**2 Anthropic — Forward Deployed Engineer role**  
Example combining engineering, customer work, production experience and domain context. Location-specific requirements should not be generalised.  
<https://job-boards.greenhouse.io/anthropic/jobs/5391016008>

**3 Palantir Careers**  
Context for forward deployed engineering and ownership of customer problems.  
<https://www.palantir.com/careers/>

**4 LangGraph overview**  
Stateful orchestration, durable execution and human interaction capabilities.  
<https://docs.langchain.com/oss/python/langgraph/overview>

**5 Model Context Protocol — architecture**  
Protocol scope, participants and context exchange. Consult the current specification before implementation.  
<https://modelcontextprotocol.io/docs/learn/architecture>

**6 OWASP — LLM Prompt Injection Prevention**  
Untrusted inputs and defensive design.  
[https://cheatsheetseries.owasp.org/cheatsheets/LLM\\_Prompt\\_Injection\\_Prevention\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/LLM_Prompt_Injection_Prevention_Cheat_Sheet.html)

**7 OWASP — Business Logic Security**  
Server-side enforcement of business rules and authority.  
[https://cheatsheetseries.owasp.org/cheatsheets/Business\\_Logic\\_Security\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Business_Logic_Security_Cheat_Sheet.html)

**8 NIST — AI Risk Management Framework**  
A framework for organising AI risk management; not a guarantee of compliance.  
<https://www.nist.gov/itl/ai-risk-management-framework>

**9 IBM — Data science**  
Interdisciplinary use of data and domain knowledge.  
<https://www.ibm.com/think/topics/data-science>

**10 IBM — AI and model families · Large language models**  
Relationships and terminology. Also: [ibm.com/think/topics/large-language-models](https://www.ibm.com/think/topics/large-language-models)  
<https://www.ibm.com/think/topics/ai-vs-machine-learning-vs-deep-learning-vs-neural-networks>

**11 IBM — Retrieval augmented generation**  
External knowledge and generation.  
<https://www.ibm.com/think/topics/retrieval-augmented-generation>

 Consult primary documentation for technical behaviour and the current requirements of any role or credential. Evaluate learning material by its accuracy, practical exercises and relevance to your gaps. **Keep implementation evidence alongside any certificates.**



MANIFOLD AI LEARNING

# TRANSITION TO AI LEVERAGING YOUR EXPERIENCE

Choose the Starting Point based on your Experience

[manifoldailearning.com](https://manifoldailearning.com) ↗

**0-2**

YEARS

**2-5**

YEARS

**5-7**

YEARS

**8-12**

YEARS

**12+**

YEARS

Every experience level has a starting point. Find yours.

