Knowledge Before Answers

Train-the-Trainer Guide

How to help large cross-functional teams understand and practice knowledge-graph-informed AI design

1. Knowledge before answers
2. Caveats before confidence
3. Human control before automation

Facilitator-first / hands-on QA

WHY IT EXISTS

The problem is not AI polish. It is unsupported confidence.

> **Most teams can talk about AI features. Fewer teams can explain what the AI must understand before it answers.**

The gap

Prompt-writing alone does not create trustworthy AI. Teams need a way to translate human reality, rules, source authority, caveats, and decisions into AI behavior.

The risk

A polished answer can hide missing knowledge, weak source authority, unclear intent, or unsupported confidence.

The goal

Give cross-functional teams a repeatable way to build, test, and refine KG-informed assistants in a short working session.

AI behavior to design

- Ask
- Caveat
- Refuse
- Explain
- Escalate
- Simplify

Behavior > graph theory

LEARNING OUTCOMES

Participants leave with a practical mental model.

They should see the relationship between knowledge, instructions, QA, and trust.

01

Explain

what a KG is in product and UX terms.

02

Map

AI behavior to layered knowledge, not one big prompt.

03

Spot

where answers need caveats, source authority, escalation, or refusal.

04

Run QA

that tests behavior against the graph, not just polish.

05

Improve

instructions or knowledge based on observed bot behavior.

Trainer move: keep returning to “What must the system understand before it answers?”

MENTAL MODEL

Three artifacts. One learning loop.

Knowledge graph

Durable meaning layer: entities, relationships, constraints, caveats, source authority, escalation paths, and human-control moments.

Instructions

Tool-specific adapter: role, behavior, tone, routing, refusal, and response rules.

QA prompts

The proof: pressure-test whether the assistant uses the intended knowledge layer.

Build → Test → Refine

If the bot fails, decide whether the gap is in the graph or the instructions.

PRIMARY TEACHING DIAGRAM

The six-layer KG stack

Each layer answers a different question. The connector decides which layer should govern the response.

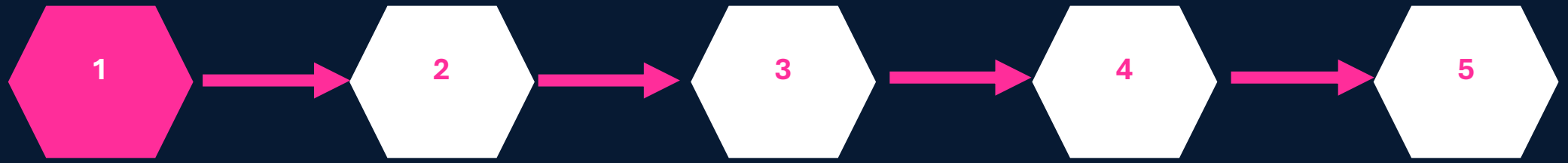
1	**Customer reality**	Who is this for?
2	**AI requirements**	What must be known?
3	**Journey & state**	Where are they now?
4	**Decision support**	What choice is this?
5	**Voice & trust**	How should this sound?
6	**Connector**	Which layer governs?

Layered knowledge becomes one coherent experience.

Ethics is not a final check. It is embedded in every layer.

WORKING SESSION SETUP

Give the room a sequence before anyone starts building.



Recommended room roles

Facilitator keeps the room in the layer • Builder creates the bot • Scribe captures QA outputs • Domain reviewers watch for source, caveat, trust, and human-control gaps

Do not start with the connector bot. Build layer bots first so defects are easier to diagnose.

2-HOUR AGENDA

Keep the session moving.

0:00–0:10	**Set purpose**	Why knowledge before answers matters.
0:10–0:25	**Teach the stack**	Walk the six layers and the working loop.
0:25–0:40	**Demo one bot**	Show instructions + knowledge + QA output.
0:40–1:15	**Small-group build**	Teams build or review assigned layer bot.
1:15–1:40	**QA run**	Run prompts, capture Pass / Refine / Revise.
1:40–1:55	**Refinement discussion**	What was missing: graph, instructions, or source?
1:55–2:00	**Close**	Name one change teams will apply to current AI work.

If the session is shorter, skip the connector bot and focus on one layer.

QA proves behavior, not polish.

Pass

The bot uses the intended layer correctly. Capture the answer as a positive example.

Pass with refinement

Mostly right, but the graph or instructions need a sharper rule, caveat, or example.

Revise

The bot answers from the wrong layer, overclaims, ignores the graph, or creates trust risk.

QA prompts should create pressure.

Use prompts that trigger ambiguity, missing source authority, budget tension, user distress, accessibility needs, confidence gaps, or conflicting goals.

A bot that sounds polished but ignores the graph has not passed.

Diagnose the layer before rewriting everything.

Domain drift	Answers from generic context.	Add a stronger domain lock.
Over-answering	Gives a plan when it should clarify.	Tell it to stay in the layer.
Source overconfidence	States policy, pricing, or availability as fact.	**Add source authority and freshness rules.**
Too much machinery	Exposes graph terminology to users.	Add guest-facing language examples.
Caveat and abandon	Says “check the source” with no next step.	Add safe interim guidance.

Often the problem is the instructions or missing knowledge, not the model itself.

CROSS-FUNCTIONAL ENGAGEMENT

Give every function a way into the conversation.

One shared prompt

What must the AI understand before it answers?

Use the question to pull expertise into the graph instead of debating a single answer.

Product	What outcome are we optimizing?
UX / Research	What human reality must be modeled?
Content / Brand	What must be explained plainly?
Data / Engineering	What source and relationships are required?
Legal / Compliance	What cannot be inferred or claimed?
Operations / Service	Where will humans recover and improve?

Do not ask everyone to agree on the answer. Ask what the AI must understand before it answers.

TRAINER TALK TRACK

Repeatable phrases keep the session anchored.

> **The graph is the durable meaning layer; instructions are the adapter.**

> **QA proves behavior, not polish.**

> If the assistant overclaims, the graph needs a boundary.

> If the assistant asks the wrong question, the instructions need a role correction.

> If the assistant gives one answer too quickly, ask what layer it skipped.

CLOSE

What good looks like

Shared language

The team can explain the difference between knowledge, instructions, and QA.

Working artifact

Participants leave with at least one testable KG-informed assistant or layer bot.

Better questions

The team asks what the AI must understand, caveat, refuse, or escalate before it answers.

Knowledge before answers. Caveats before confidence. Human control before automation.

Ask: Where do we currently have answers before knowledge?