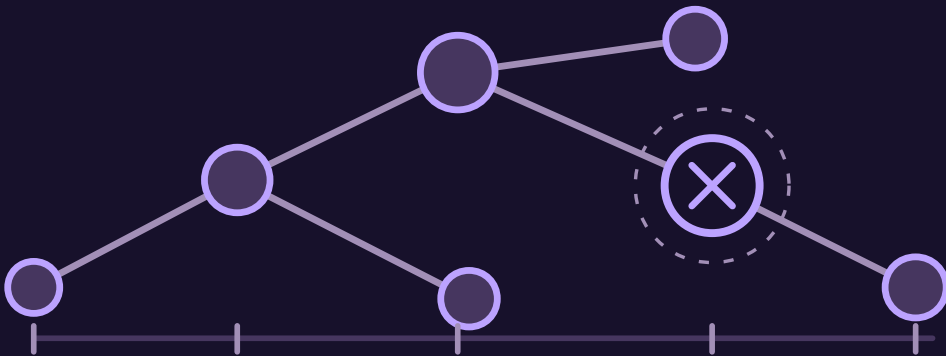




PRACTICAL AI

# Trace and Triage AI Application Failures

Connect request paths, operational signals, and answer-quality evidence



**Sanjaay Gowda T S**

FREE SAMPLE / INTRODUCTION + CHAPTER 1 / EDITION 1.0.0

[sanjaaygowdats.ai](https://sanjaaygowdats.ai)

# About this free sample

## Trace and Triage AI Application Failures

Sanjaay Gowda T S

First edition 1.0.0, 2026-10-13

Copyright 2026 Sanjaay Gowda T S. This is a labelled free sample, not the complete ebook. You may share this sample unchanged. It contains the cover, sample permissions, contents, introduction, Chapter 1 and a closing product link. The remaining chapters, worked solutions and companion files are not included. The full ebook text, diagrams and artwork have separate single-reader terms; this sample permission does not authorize sharing the full purchased PDF or EPUB.

The companion source code, tests and fictional data are separately MIT licensed. You may adapt or share that project with its included licence notice. MIT does not apply to the ebook text or book artwork. The books and companion projects have no technical activation requirement or buyer-specific watermark.

This first edition was prepared with AI assistance. The case studies are fictional teaching examples. Any measured outputs come from the supplied examples, not production customer data. The teaching scope, assumptions and limitations are explained in the chapters. Independent human technical, editorial and assistive-technology reader review is pending. Automated tests and accessibility checks do not establish certification or suitability for every use.

Run examples in a separate working folder using the supplied synthetic data. Review inputs and calculations before adapting the techniques to real operations. Model software, service access, subscriptions, ongoing updates and consulting are not included.

Author website: [sanjaaygowdats.ai](https://sanjaaygowdats.ai)

Ebook terms: [sanjaaygowdats.ai/ebooks/license](https://sanjaaygowdats.ai/ebooks/license)

# Inside this free sample

|                                          |   |
|------------------------------------------|---|
| Start here.....                          | 4 |
| 1. Start with the failure question ..... | 5 |
| Continue with the full guide .....       | 6 |

This free sample contains the introduction and Chapter 1 only. The closing link opens the complete guide. The companion project and worked solutions are supplied with the complete ebook.

# Start here

An AI application can return a successful HTTP response while giving an answer that misses the user's question. It can also produce a good answer and fail to deliver it. If every outcome is compressed into one success flag, the operational record cannot explain either incident. Useful observability starts by distinguishing what completed, what was judged acceptable, and what the user actually received.

This book follows a fictional support assistant with retrieval, generation, validation, and delivery stages. The companion contains synthetic span records and a deterministic triage program. It validates trace structure, keeps quality judgments separate from transport status, and routes traces to transport, quality, latency, or healthy queues using a documented priority rule. There is no telemetry backend, model invocation, or real customer data.

You need Python 3.10 or later, basic JSON familiarity, and a willingness to ask what evidence a field really provides. Run `python main.py --input example-input.json` from project, then `python -m unittest discover -s tests -v`. The expected output and eleven tests make the examples reproducible. They establish the behavior of this teaching program, not the reliability of an AI service.

OpenTelemetry distinguishes signals such as traces, metrics, and logs. [R1] We use that vocabulary to design an application evidence model, while keeping the executable fixture deliberately simpler than an actual OpenTelemetry integration. The fixture's JSON is not OTLP and its `ok` field is not the SDK's status enum.

The outcome is a practical incident method: define the symptom, inspect a correlated request path, separate observations from hypotheses, reproduce the smallest failure, and turn the verified cause into a regression case.

# 1. Start with the failure question

## Replace "add logging" with an investigation

Imagine an operator receives this report: "The assistant said the fictional parcel had arrived, but the tracking page did not." Before instrumenting anything, write the questions that would resolve the discrepancy. Which tracking revision was retrieved? Did retrieval return zero records? Did generation receive the retrieved facts? Did validation detect the contradiction? What exact response reached the user?

Those questions define the evidence requirements. Logging every prompt may still omit the document revision or the delivery outcome. Conversely, a short structured record with a case reference, retrieval revision, quality decision, and response reference can support an efficient investigation. The purpose of a field is to answer a question, not to make the event look detailed.

Start with three recurring investigations: an incorrect answer, a slow answer, and an operation that failed to reach the user. For each, identify the minimum evidence needed to distinguish plausible causes. This creates an instrumentation plan that follows user outcomes rather than implementation fashion.

## Separate observations and explanations

An observation might be "retrieval returned zero records" or "the generation attempt took 700 milliseconds." A hypothesis might be "the index revision omitted a document" or "the provider was overloaded." The first can be recorded directly. The second requires additional evidence. A telemetry field named `root_cause` should not silently contain an untested guess.

| Incident question               | Direct observation              | Hypothesis needing a test      |
|---------------------------------|---------------------------------|--------------------------------|
| Why was the answer unsupported? | Retrieved record count was zero | Index update missed the parcel |
| Why did the request run slowly? | Generation span lasted 700 ms   | Remote service was overloaded  |
| Why was no result visible?      | Delivery acknowledgement absent | Browser disconnected           |

In the companion, the routing label is explicitly a heuristic. A transport error takes priority over quality failure, and quality failure takes priority over a slow span. This helps choose an initial queue. It does not prove which component caused the overall incident.

## Define the unit being observed

A user action can create multiple requests, retries, and background jobs. Decide which identifier denotes one logical user operation and which identifiers denote individual attempts. If a retry receives a new logical-operation identity, operators may mistake one troubled action for several independent failures. If all attempts share one span identity, attempt timing and outcomes become ambiguous.

The fixture gives each trace one root operation and each span a unique identity within that trace. Real applications may need links between traces for asynchronous work. Start by documenting those relationships before choosing the collection backend.

### Exercise 1

for a fictional answer that cites an expired policy, write four investigation questions and one directly observable field for each. Avoid putting the presumed cause into the field name.

# Continue with the full guide

This free sample ends after the introduction and Chapter 1. The complete guide contains nine teaching chapters, worked exercise solutions, primary references and a separate Python companion project. The full PDF, reflowable EPUB and project files are supplied together.

Use the complete guide to work through the remaining exercises and inspect the supplied examples. The companion code and fictional data are separately MIT licensed; the complete ebook has the single-reader terms described on its product page.

The examples are teaching fixtures, not production customer results. Independent human technical, editorial and assistive-technology reader review remains pending. Software access, subscriptions, consulting and guaranteed results are not included.

[View the complete guide and purchase options.](#)

You may share this labelled free sample unchanged. Keep the full purchased PDF and EPUB private. Thank you for reading.

## Primary references

### 1. [R1] OpenTelemetry: Signals

<https://opentelemetry.io/docs/concepts/signals/>

Trace, metric and log signal distinctions