



PRACTICAL AI

Build a Prompt Regression Test Suite

Keep improvements, regressions and old failures visible.



Sanjaay Gowda T S

FREE SAMPLE / INTRODUCTION + CHAPTER 1 / EDITION 1.0.0

sanjaaygowdats.ai

About this free sample

Build a Prompt Regression Test Suite

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

Ebook terms: sanjaaygowdats.ai/ebooks/license

Inside this free sample

Start here.....	4
1. Define a regression contract	5
Continue with the full guide	7

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

A prompt change often has a persuasive demonstration. The new version fixes one awkward answer, sounds more natural and appears ready to replace the old version. The difficult question is what happened to the cases you did not show. A regression suite preserves those cases and gives them a voice in the decision. It should also reveal old failures, incomplete experiments and the limitations of its own checks.

This book builds an offline comparison harness for a fictional equipment library. Every baseline and candidate response in the companion is a synthetic, hand-written fixture. The prompt texts are also teaching examples. They are fingerprinted but never executed. No language model, API, search index or customer workflow is tested. You will learn the mechanics of trustworthy comparison before deciding how to collect real responses.

Use Python 3.10 or later and basic dictionaries, lists, JSON and tests. From the extracted project directory, run:

SHELL

```
python --version
python main.py --input example-input.json
python -m unittest discover -s tests -v
```

On Windows, `py -3` can replace `python`. On macOS or Linux, use `python3` if necessary. No packages, accounts or keys are required. `Expected-output.json` records the actual deterministic fixture comparison. If the command cannot find the input, confirm that the terminal is in the directory containing `main.py` and `example-input.json`.

The deliverable is a suite with independent expectations, preserved outputs, interpretable reasons and release gates that do not confuse "nothing got worse" with "everything works." The tests exercise the comparator. They do not demonstrate that any real prompt produces the fixtures. That distinction remains visible in every report and in the handoff you create at the end.

1. Define a regression contract

A regression needs a prior requirement

A response sounding different is not necessarily a regression. It becomes one when a previously satisfied requirement fails under a new candidate. Start with the requirement: renewal answers must direct users to contact staff, exception answers must retain the damage qualification, unknown facts must produce abstention, and citations must use supplied identifiers. These fictional rules are deliberately concrete enough to inspect.

The comparator evaluates the same cases against baseline and candidate output maps. A baseline pass becoming a candidate failure is a regression. A failure becoming a pass is an improvement. Two passes are an unchanged pass; two failures are an unchanged failure. A missing output makes the comparison incomplete rather than being silently assigned to one of those four states.

Baseline	Candidate	Classification
Pass	Pass	Unchanged pass
Pass	Fail	Regression
Fail	Pass	Improvement
Fail	Fail	Unchanged failure
Missing either	Any	Incomplete comparison

These definitions are local evaluation conventions. The important discipline is to declare them before collecting evidence. Otherwise a team can relabel an inconvenient failure as a harmless style change after seeing which candidate it favors.

Separate behavior from exact wording

The renewal requirement uses a required phrase, contact staff. That is an intentionally narrow mechanical proxy. A semantically equivalent answer saying "Speak with the library team" would fail. A longer answer containing contact staff plus an unsupported promise might pass. A phrase check tests text occurrence, not full behavioral correctness.

Decide which requirements deserve exact mechanical tests and which need reviewed judgments. Exact citation membership is suitable for code. Whether a qualification is complete may need source comparison. Tone can need a defined rubric and independent assessors. Putting every dimension behind one pass flag is tempting but leaves the meaning of success unclear.

Define scope before optimizing

The fixture suite covers four scenarios, not the equipment library's entire support task. It includes a renewal instruction, an exception, an unknown fact and a citation-version mismatch. It does not cover multilingual questions, conflicting policies or tool actions. Name these exclusions before using the suite to approve a wider product change.

A prompt regression suite is useful even when incomplete because it protects specific known requirements. Its value decreases when its scope is exaggerated. The honest claim is that the candidate meets these retained checks under these inputs, plus whatever separate review has actually been performed.

Exercise: specify one protected behavior

Choose the exception case and write the protected behavior independently of its current answer. Give one acceptable paraphrase and one answer that sounds plausible but drops the qualification. Explain which part your current mechanical check can distinguish and which part needs source review.

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.