NORDIC TESTING DAYS 2026

Vibe-Coding for Testers

Building Custom Testing Tools Without Pinging Your Engineering Team

Pavel Fleisher · Quality Engineering Lead · MeetingPackage

2h hands-on workshop

Hi, I’m Pavel 👋

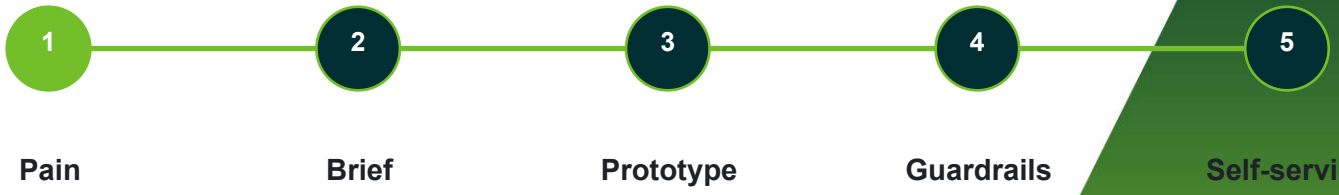
Quality Engineering Lead @ MeetingPackage

1. **Product + consultancy QE** — Nearly a decade across quality engineering, automation and software development.
2. **Automation-first mindset** — I care about workflows, evidence, tooling and boring things that save hours.
3. **Today’s bias** — Small tools should solve real QA pain, not create shiny technical debt.

fleisher.ee

From repeated QA pain to a small safe tool.

We will build the thinking path, not just admire the AI magic trick.



This is not an AI guru ritual.

No AI replaces engineers speech

No production hacks

No “trust me bro” generated code

We keep the fun. We also keep the seatbelts.

Too small for the roadmap. Too frequent to ignore.

“Can you quickly check this state?”

“Can you export HAR and explain what failed?”

“Can you summarize this regression run?”

“Can you prepare test data for this weird case?”

When to tool it?

Once

handle manually

Twice

watch the pattern

Three times

design a small tool

Do not automate because AI makes it easy.
Automate because the pain became a pattern.

Do not vibe-code the mess.

First structure the workflow:

Who has the pain?

What input do we already have?

What output is useful?

What should the tool never do?

Good small tool vs risky small monster

Good candidate

- repetitive
- low-risk
- easy to validate
- no secrets
- clear owner

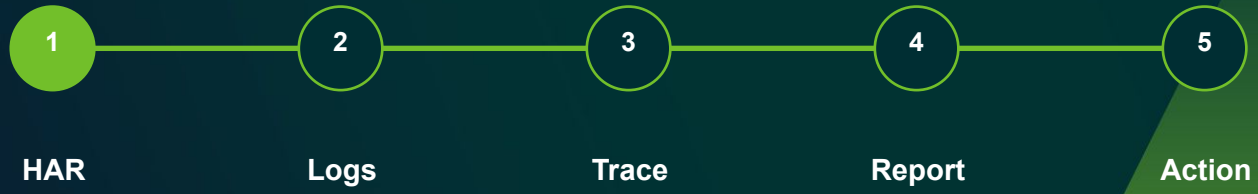
Bad candidate

- touches production
- uses sensitive data
- irreversible actions
- business-critical
- needs ops support

CORE CONCEPT

Signal-to-Action tools

Small QA utilities that turn raw testing signals into structured, reviewable output.



Small QA tools that never reach the roadmap

Messy bug notes	**Bug Report Builder**
Raw HAR files	**HAR Signal Analyzer**
Noisy console logs	**Log Summarizer**
Unclear release risk	**Regression Scope Builder**
API changes	**API Diff Viewer**
Flaky tests	**Failure Triage Helper**

SHARED EXAMPLE

QA Evidence Assistant

A local tool that turns tester notes, HAR files and console logs into a structured bug report.

Manual testers: better evidence

Automation engineers: follow-up test ideas

Developers: fewer “can you add more info?” loops

BEFORE

The bug report nobody loves

Checkout failed after clicking Pay.
Chrome, staging.
Please check.

Dev response in 3... 2... 1...

“Can you send logs, HAR, exact request, account state and steps?”

Network signals are already there

failed requests: 4xx / 5xx

slow or duplicated requests

auth failures: 401 / 403

validation failures: 400 / 422

conflict states: 409

first failed request after user action

AFTER

A report that speaks engineering

Bug: checkout fails after booking conflict

Signals:
- POST /api/bookings returned 409
- response says: slot unavailable
- no visible UI error after failed response

Likely risk:
backend returns conflict correctly, frontend does not handle the state

Follow-up:
add UI test for 409 BOOKING_CONFLICT

Raw HAR can be radioactive.

- cookies
- authorization headers
- tokens
- emails
- session data
- request bodies

Parse and redact locally before AI touches anything.

LIVE DEMO TASK

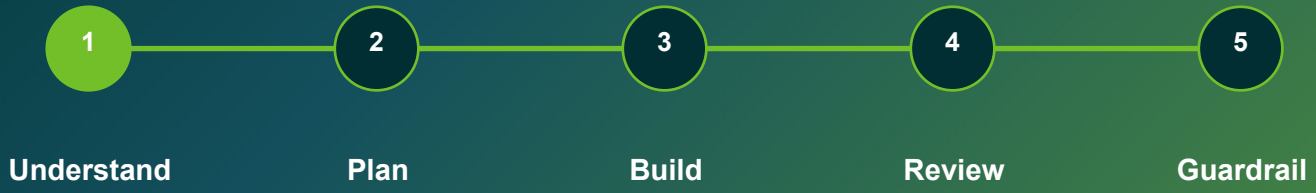
Add booking conflict detection

Detect:
- HTTP 409
- URLs containing booking / reservation / checkout / availability
- response text containing conflict / unavailable / already booked / slot taken

Output:
- detected signals
- bug report snippet
- follow-up test idea

Prereq: Cursor or Antigravity installed. Starter repo provided.

Cursor / Antigravity flow



Do not start by asking the tool to “make it better”.
That is how small monsters are born.

PROMPT 1

Understand first

Inspect this project and explain:
1. what this app currently does
2. where HAR parsing happens
3. which files are safe to modify
4. what risks you see

Do not edit files yet.

PROMPT 2

Make one bounded change

Add detection for HTTP 409 booking conflict signals in HAR files.

Constraints:
- local processing only
- no external APIs
- redact sensitive values
- do not modify unrelated tools
- keep output understandable for manual testers

PROMPT 3

Review like a mini-PR

Review this tool as:
1. manual tester
2. automation engineer
3. frontend developer
4. security reviewer
5. future maintainer

List concrete issues and improvements.
Do not rewrite the code yet.

YOUR TURN

Pick one pain. Shape one tool.

Beginner: bug report or evidence builder

Intermediate: HAR, API diff or regression scope

Advanced: mock response or test failure triage

Goal: a working mini-tool, a tool brief, or a reviewed prompt you can take home.

Before code, answer these

QA pain point:
Who has this problem:
Raw signal we have:
Useful output:
What it must never do:
Safe data:
How we validate it:
Owner:
When engineering gets involved:

AI makes the prototype faster.
QA makes it useful.
Guardrails make it safe.

The future of QA tooling is not every tester becoming a full-stack engineer.
It is testers getting better at shaping small tools around real workflow pain.