What to Measure

As you onboard and adopt Functionize Studio

Take the baseline in week one. It is half an hour, it is the easiest thing to skip, and it is the hardest to recover. Most teams start measuring after things have already improved — and then cannot prove anything. None of the five numbers below come from Studio; they come from your existing process, and they are the ones everyone outside QE cares about.

WEEK-ONE BASELINE — FILL THIS IN BEFORE YOU BUILD A SINGLE TEST

Number	What it is	Yours today	Where to get it
Escaped defects	Issues that reached production last quarter	.....	Your incident or ticket system
Manual regression hours	Person-hours hand-testing per release	.....	Ask the team — they know
Release cadence	How often you ship; how long a release is held for testing	.....	Your release calendar
Maintenance share	Share of automation hours spent *repairing*, not adding	.....	Your existing suite's history
Critical journeys covered	Revenue-critical flows, and which are automated today	.....	Your own list
Date captured	.....		
Captured by	.....		

THE SEVEN KPIS WORTH OWNING

KPI	Target	Where the number comes from
1. Escaped defects	Down, quarter on quarter	Your ticket system, tagged "could a test have caught this?"
2. Critical journey coverage	100% of your critical list (usually 10–20 flows)	Your list, checked against your Studio projects
3. Time to feedback	Minutes, not hours	Duration on an orchestration's run history
4. Maintenance share	Under a quarter of automation effort	The History panel — every change, with author and date
5. Flake rate	Near zero; treat recurring flake as a defect	The Warning count in Reports is a good proxy
6. Pass rate	Read with the test count beside it, never alone	Reports → Pass Rate Breakdown by Project
7. Mean time to diagnose	Minutes	Time it yourself for two weeks

Two ways to measure badly. Counting tests — "we have 400 tests" says nothing about whether you would catch a broken checkout; report what is *proven*, not what was *written*. **Chasing a coverage percentage** — it drives people to write cheap tests over valuable ones. Ask instead: **if the whole suite passes, what do we now know is true?**

A REPORTING RHYTHM THAT WORKS

Weekly — for the team

Reports on a 7-day range. Read the *trend*, not the number.

- Sort the project table by pass rate; look at the bottom
- Are warnings growing?
- Pick one thing to fix

Monthly — for leadership

A 30-day range, exported. Report three things only:

- Escaped defects
- Critical journey coverage
- Manual regression hours displaced

Pass rate is an internal diagnostic. Leave it out.

Quarterly — for the budget

Compare against the week-one baseline above.

This is the conversation that renews the investment — and the reason the baseline was worth half an hour.

TARGETS FOR A FIRST QUARTER

- ☐ **100% of critical journeys covered** — usually ten to twenty tests
- ☐ **Smoke suite under ten minutes**, running at least daily
- ☐ **Flake near zero** — every recurring warning triaged, not tolerated
- ☐ **Maintenance below a quarter** of automation effort
- ☐ **Every escaped defect gets a test**, without exception

Deliberately modest. Over-promising in month one is how programs get canceled in month six.

JUSTIFYING THE SPEND, MONTH OVER MONTH

1. **Hours displaced.** Manual regression hours before, versus now. The easiest number for a finance conversation, and usually the largest.
2. **Escaped defects prevented.** Failures your suite caught before release that would previously have reached production. Tag them as you go — reconstructing them later is guesswork.
3. **Credit consumption, with its shape explained.** Building a test costs roughly what fifty runs cost. Month one is mostly building and looks expensive; by month three you are mostly running, and the run-rate settles. Present usage without that curve and a normal build-heavy first month reads as a cost overrun.

> **The efficiency metric worth tracking: credits per test maintained, per month.** It should **fall** as your suite matures, because repairs cost a fraction of rebuilds, manual edits are free, and reusable components stop you regenerating the same login twenty times. A **rising** figure means people are regenerating tests instead of fixing them — a training issue, not a pricing one.

WHERE THE NUMBERS LIVE IN STUDIO

Screen	What it gives you
Reports	Execution Breakdown (passed / failed / warning) · Test Execution Trends per day · Pass Rate Breakdown by Project · date range · Export → pass rate, flake rate
Orchestration history	Duration and result per suite run → time to feedback
Test History panel	Every run and every change, with author and timestamp → maintenance activity, run-duration drift
The test list	Status, last updated, last run, browsers, filter by tag → tests that have not run in months

Printed on the Reports page itself: *"Data updates on a schedule and may not reflect the latest changes."* Reports is for weekly and monthly shape, not for checking whether the fix you made two minutes ago worked. For that, open the run.