Regression Test Strategy

A template to adapt for your organization

Regression testing answers one question: does what worked yesterday still work today? It is the largest test effort in most organizations, the most repetitive, and the best candidate for automation — which is exactly why it needs a strategy rather than an accumulated pile of tests.

> ***How to use this template.*** *Everything in italics is guidance for you and should be deleted once you have replaced it. Square brackets mark a value to fill in. Work top to bottom the first time; after that, treat sections 3 and 4 as living documents and revisit them every quarter. Keep it short. A strategy nobody reads governs nothing.*

1. Scope and triggers

Trigger	What runs	How long it may take	Blocks release?
Every deploy to [env]	Smoke tier	Under 10 minutes	Yes
Daily	Critical tier	[ ]	No — but read same day
Nightly / per release	Full regression	[ ]	[ ]
After a production incident	[the new test for it]	[ ]	[ ]

2. Tiering

> *Three tiers is enough. A single undifferentiated pile cannot be trusted, because a failure in it carries no information about how urgent it is.*

Tier	What it answers	Size	Read within
Smoke	Is this deployment fundamentally alive?	[5–10 tests]	The hour
Critical	Do the revenue-critical journeys and permission boundaries work?	[10–20 tests]	The same day
Regression	Does everything else still work, including one test per escaped defect?	[ ]	Next working day

Promotion rule: a test may only move up a tier once it has passed reliably where it is. Nothing enters the smoke tier because it feels important — it enters because it has passed three times in a row.

> A gate that fails at random gets bypassed within a month, and then nothing is gated. Protect the smoke tier above all.

3. What goes in the regression suite

1. Every critical journey from the master strategy.

2. One test per escaped defect — the highest-yield list you will ever have, and it is already written down.
3. Permission boundaries, including the negative cases.
4. The long, repeated manual checks your team quietly resents.

Flow	Business risk if it breaks (H/M/L)	Run frequency	Deterministic?	Data controlled?	Automate?
[journey]					
[escaped defect REF-000]					

4. Flake policy

Flake is a defect, not weather. It is a defect in the test, the data or the environment that happens to be intermittent. The cost people quote is the small part; the real cost is teaching a team that red sometimes means nothing.

Cause	Typical share	What to do
Timing — a step ran before the page was ready	~45%	Never a fixed wait. Describe the state, or raise the wait at project level
Uncontrolled data		Make the test create what it needs, or pin values you own
Order dependency	~12%	Split it out, or have the test set up its own state
Environment		Fix once at project level, not in every test

When a test fails intermittently there are exactly three options, and "rerun it and move on" is not one:

1. **Fix the cause.** Almost always the right answer.
2. **Demote it** out of any tier that gates something — **with a date**.
3. **Delete it.** A test nobody trusts and nobody will fix costs runs, attention and credibility.

> Quarantine with no date is how flake becomes permanent.

5. Who reads red

> *This single decision predicts whether the suite is still being read in six months better than anything else in this document.*

Suite	Named reader	By when	First action	Done means
Smoke	[a person, not "the team"]	Within the hour	[ ]	App fixed, test fixed, or demoted with a date
Critical	[ ]	Same day	[ ]	
Regression	[ ]	Next working day	[ ]	

> Never "it passed on the rerun".

6. Maintenance

In selector-based suites, maintenance commonly exceeds half of all automation effort — which is why those suites stop growing. Track it.

> Share of automation effort spent repairing rather than adding (today)
>
> Target
>
> Reviewed

Quarterly pruning, one hour:

- Archive tests for retired features.
- Find tests that never fail — break one on purpose and see whether it notices.
- Find tests that always fail. Fix, demote or delete.
- Check the smoke tier is still under its time limit.
- Fold duplicated flows into reusable components.

> **With Functionize Studio:** tests describe intent rather than page structure, so a redesign that preserves behavior does not produce a wave of red — and repairs are targeted rather than rebuilds. The maintenance share is the number the platform targets directly, so it is the one to watch.

7. Reporting

Audience	Cadence	What they get
The team	Weekly	Trend, not the number. Warnings growing? One thing to fix.
Leadership	Monthly	Escaped defects, critical journey coverage, manual hours displaced
Budget holders	Quarterly	The same three, against the week-one baseline