AUTOMATED TESTING PROPOSAL

AI-POWERED REGRESSION TESTING THAT KEEPS UP WITH YOUR RELEASES

We move your Web, API and Batch-job regression from manual runs to automated tests, built with Playwright, BDD and AI.

≥50% less regression effort

Web · API · Batch testing

AI-powered test design

OVERVIEW

Faster releases need faster regression testing

We automate your regression so it keeps pace with development, with the same quality checks in place.

PROBLEM

Manual testing slows down release cycles

Feature delivery becomes faster with AI while regression testing stays manual. Release cycles get longer.

OUR APPROACH

Automated regression testing

AI drafts the tests, engineers review them, and CI runs them on every change.

- **AI test design**
- **Playwright + BDD**
- **CI/CD execution**

IMPACT

Faster and safer releases

50% **LESS EFFORT**
Test design & scripting

50% **FASTER**
Shorter regression runs

✓ **QUALITY**
Same or better quality

Targets confirmed against your baseline in week 1.

WHY NOW

Why now: The bottleneck has moved to testing

Every release has more features to re-test, but regression is still done by hand. The bigger the product gets, the less manual regression can keep up.

Early-stage

Small scope. Manual regression fits inside the release window.



One release cycle

- **Build:** feature development
- **Test:** manual regression

MANUAL TESTING KEEPS UP

At-scale

Scope grows every sprint, and regression grows with it.



One release cycle: regression takes most of it

- Build: faster, AI-powered delivery
- **Test:** growing regression of dependencies

MANUAL TESTING BECOMES THE BOTTLENECK

COST OF INACTION

What happens if regression stays manual?

Manual regression forces a trade-off between speed, quality and cost.

Cut test scope

Run only "critical" cases to hit the release date.

Coverage drops → escaped defects rise

Test later

Push full regression to the end of the cycle.

Bugs found late cost more to fix

Add more testers

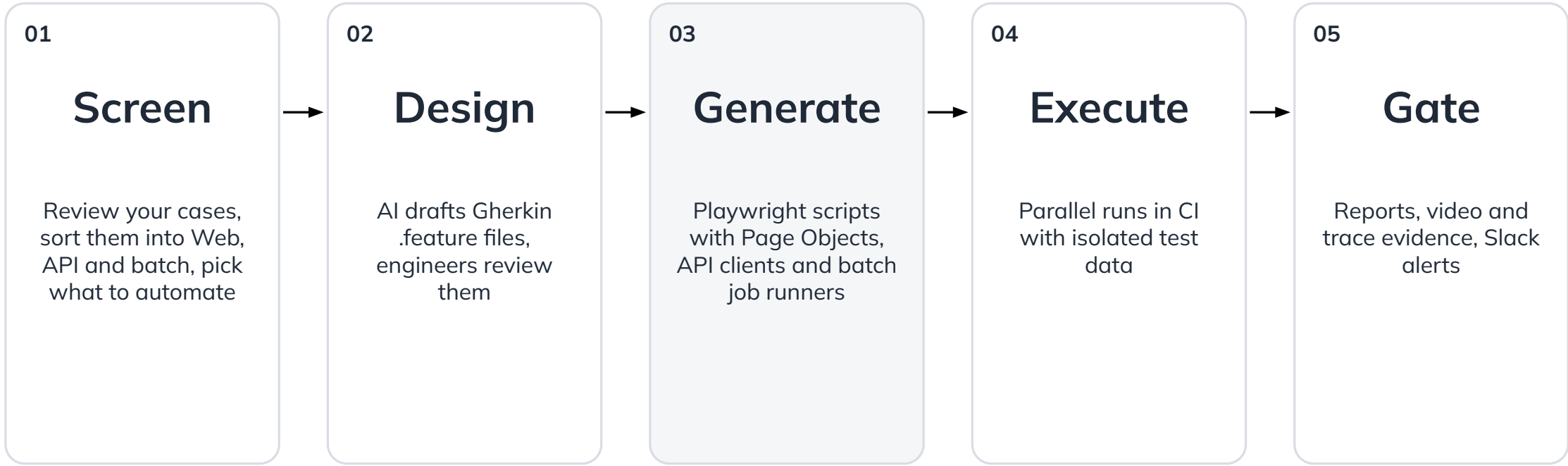
Scale QC headcount with every release.

Cost grows with every release, and tired testers miss things

HOW AUTOMATED TESTING WORKS

The solution: AI-Powered automation pipeline

Each case becomes a versioned test that runs on every change.



When the UI changes, AI suggests locator fixes for an engineer to approve

HUMAN X AI DELIVERY

AI automates execution. Engineers review results.

AI does the heavy lifting end-to-end. Engineers act as quality gatekeepers before merging.

WHO DOES WHAT

	WHAT AI AUTOMATES (HEAVY LIFTING)	WHAT OUR ENGINEERS REVIEW & APPROVE
Design	Auto-drafts complete scenarios from cases	Verify business logic accuracy
Scripting	Generates Playwright scripts & API code	Review PR & approve code structure
Maintenance	Detects broken locators & generates fixes	One-click approve or reject fixes
Reporting	Auto-summarises & categorises test failures	Final sign-off on defect vs environment issue

COVERAGE

One framework for Web, API and batch jobs

The same pipeline tests UI flows, services and back-end jobs. Web, API and batch tests share one repo, one CI pipeline and one report.

Web

UI / END-TO-END

Critical user journeys, cross-browser (Chrome, Safari), Page Object Model.

API

SERVICES

REST endpoints, schema & response validation, shared auth context.

Batch

BACK-END JOBS

Scheduled, ETL & import/export jobs: data in → job run → data out verified.

HOW WE TEST BATCH JOBS

1 · Arrange

- Seed isolated test data via a helper function
- Tag test data per run for isolation
- Back-date data for time-based rules

2 · Trigger & wait

- Invoke jobs directly: schedule, file or queue
- Use real inputs, marked as test
- Detect completion via indirect signals

3 · Verify

- **Read results via the helper, no direct DB**
- **Check external services via sandbox APIs**
- **Re-running a job gives the same result**

TEST-AS-CODE

Why Test-as-Code is the enabler

AI works well when tests are structured, versioned and executable, not scattered in spreadsheets. Your tests live in Git and get reviewed like any other code.

Structured

Steps, data and expected results in a format AI can read.

Composable

Page Objects, fixtures and flows reused across Web, API and Batch.

Versioned

Git history, PR review and traceability to test case IDs.

Executable

Runs in GitHub Actions per PR, tag and environment.

EXAMPLE TEST ASSET

checkout.feature

```
@core @TC-1024

Scenario: Member checks out with saved card

  Given a logged-in member with items in cart

  When they pay with the saved card

  Then the order status is "Confirmed"

  And the Orders API returns the new order
```

IMPACT

Target impacts on release cycle and business

We record your current numbers in week 1 and compare them at the end.

−50%

REGRESSION EFFORT

Less effort to design, script and maintain regression tests

+50%

REGRESSION SPEED

Faster regression runs and earlier release feedback

✓

QUALITY

Repeatable evidence and quality gates on every release

HOW WE PROVE IT

Authoring & maintenance effort

Man-hours per cycle vs. your last 2–3 manual cycles

Execution time

Regression run time in CI (parallel) vs. today's manual run

Quality signal

Coverage, escaped defects and flaky rate vs. recent releases

CASE STUDY

Case Study: Automated regression in 6 weeks

A web platform in production for years, releasing new features every sprint. Overcoming the manual testing bottleneck to maintain fast, high-quality release velocity.

CONTEXT

E-Commerce web platform, in production for over **8** years. New features every sprint. **445** manual test cases.

PROBLEM

Frequent releases make manual regression an escalating bottleneck. QA cycle stretches longer every sprint, leading to missed test cases and delayed releases when demand exceeds capacity.

SOLUTION

We **automated 60% of the cases** with Playwright, BDD and AI support. Web and API suites now run in CI/CD, with a report after every run.

RESULT: BEFORE / AFTER

	BEFORE	AFTER (AUTOMATED)
Regression run	Manual, 2 days per release	**Automated in CI, 2–3 hours**
QC effort	2 man-days per release	**1 man-day, for the 40% that stays manual**
Coverage	Cut when the release is tight	**Full suite every run (60% automated, 40% manual)**
Evidence	Manual notes	**Attachments and trace report per run**

GETTING STARTED

What the first 4–6 weeks look like

An example of how we begin: one high-value scope, measured against today, then a plan for what's next.

01

Baseline & Scope

~1 WEEK

Pick a critical flow and record today's effort, cycle time and quality.

02

Test-as-Code Foundation

1–2 WEEKS

Set up the framework, Page Objects, isolated test data and CI runs.

03

AI + Automation

1–2 WEEKS

Generate and refine coverage, then automate the regression runs.

04

Measure & Scale

~1 WEEK

Compare results with the baseline and recommend next steps.

WHAT YOU HAVE AT THE END

- Reusable Test-as-Code assets
- Measured KPI report
- Automated execution in CI/CD
- Know-how to scale

WHAT WE NEED

What we need from your team

We confirm this list with you in the kickoff meeting.

TEST CASES

Test case list/documentation in any format (spreadsheet, TestRail, Jira/Xray) along with relevant feature documentation. Scope is based strictly on existing cases without writing new ones.

TEST ENVIRONMENT

A stable staging environment that matches production. We don't need production access.

PEOPLE

A QA lead for about **~5** hours a week, plus developer contacts for feature, API, batch, and data questions.

ALSO NEEDED

	DETAILS	NEEDED BY
Access	Repo access, test accounts per role, sandbox keys	**Before week 1**
Test data rules	What we can create, what must stay masked	**Week 1**
CI/CD	GitHub Actions by default	**Week 6**
Sign-off	One decision maker for scope and results	**Week 1 and week 6**

INVESTMENT

Package and pricing

A fixed price for each module, calculated from the team effort below.

FULL IMPLEMENTATION

End-to-End Test Automation

6 weeks / module 2.5 FTE (3.75 MM)

Web, API & batch regression automation

GitHub Actions pipeline & dashboard setup

Docs, AI playbook & knowledge transfer

Quality baseline vs. result report

Fixed price per module, agreed after screening

HOW THE PRICE IS CALCULATED

ROLE	ALLOCATION	M.M.
Automation Lead	0.5 FTE × 6 wks	0.75 MM
Engineer – Web	1.0 FTE × 6 wks	1.50 MM
Engineer – API & Batch	1.0 FTE × 6 wks	1.50 MM

Total: 2.5 FTE ≈ 3.75 MM
(Rate will be disclosed in a separate quotation)

TERMS

- Rate, currency and payment terms: provide separately
- AI tool licences included in the package
- Final price confirmed after we screen your test cases

AFTER HANDOVER

Three ways to keep the tests running after handover

Tests need upkeep as your product changes. You choose who does it.

Your team maintains

Your QA team updates the tests, using our playbook and training.

Included in the package

We support you

Script fixes, new tests for new features, monthly health check.

Monthly fee provided separately

We run it for you

We run, monitor and maintain the suite, and triage every failure.

Monthly fee provided separately

SECURITY & COMPLIANCE

How we protect your code and data

AI only sees what you allow it to see. Nothing touches production.

- **NDA first:** Signed before we receive any test cases or code
- **100% owned by you:** All test code, docs and reports. We use open-source tools, so no lock-in.
- Test data is synthetic or masked, and tagged per run so it can be cleaned up afterwards.
- We work in staging and sandbox environments only

- **Certifications**

TÜVRheinland ZERTIFIZIERT
Management System
ISO/IEC 27001:2022
www.tuv.com
ID 01153 2135631

AICPA SOC
SOC for Service Organizations | Service Organizations

ISTQB®
International Software Testing Qualifications Board

AUTOMATED TESTING PROPOSAL

Book a free consultation with us!

Email: info@supremetech.vn
Website: www.supremetech.vn