CASE STUDY

gallabox

HOW GALLABOX SCALED FROM ZERO TO 1,800+ AUTOMATED TESTS TO KEEP PACE WITH 3 RELEASES A WEEK

Customer Overview

Gallabox is a fast-moving SaaS platform built around WhatsApp Business, helping businesses use WhatsApp for customer conversations, engagement, and business workflows.

With approximately three releases every week, Gallabox needed a testing approach that could keep pace with continuous product changes.

Before testRigor, Gallabox relied entirely on manual testing, with its QA team working through 400–500 documented test scenarios and no dedicated automation resources.

Today, Gallabox has grown to 2,000+ documented test scenarios, with 1,800+ scenarios automated, while regression testing that would take more than three days manually can be executed in approximately 5–6 hours.

1,800+	5–6 Hours	2,000+	3 Releases
Automated test scenarios	Regression execution	Documented test scenarios	Every week

The Problem

Before adopting testRigor, Gallabox's regression testing was entirely manual. With approximately 400–500 documented test scenarios, two manual testers, and three releases every week, the existing process could not keep pace with the company's release velocity.

This approach created several challenges.

Regression Testing Could Not Keep Pace

Manually executing hundreds of test scenarios for three weekly releases placed significant pressure on the small QA team.

Lack of Automation Resources

Gallabox had no dedicated automation engineer, making it hard to build and maintain an automation suite with traditional tools.

High Technical Barrier

The team explored Selenium and Cypress, but these coding-intensive approaches were not practical for its existing manual QA team.

Pressure on Release Sign-Off

Gallabox needed faster and more reliable regression results to confidently approve frequent releases without compromising product quality.

As the Gallabox QA team described it, testRigor helped transform regression testing from a time-consuming manual process into a faster, repeatable approach that could keep pace with three releases per week.

What Gallabox Wanted

Faster Regression Testing

Reduce the days of manual effort required to validate a release.

Reliable Automation

Build a stable regression suite that could support three releases every week.

No-Code Automation

Enable the existing QA team to automate scenarios without requiring extensive programming expertise.

Faster Debugging

Make it easier to understand why tests failed and determine whether the issue was an application defect or a test problem.

Faster Automation Creation

Quickly convert existing manual regression scenarios into automated tests.

Maintain Release Velocity

Prevent regression testing from becoming a bottleneck as the product and test suite continued to grow.

Key Objectives

- Move from manual regression testing to automation
- Support three releases every week
- Reduce regression execution time
- Build automation without depending heavily on coding expertise
- Convert existing manual scenarios into automated tests
- Make failed tests easier to investigate and debug
- Increase regression coverage as the application grows
- Reduce repetitive manual QA effort

> **"We needed a stable and dependable tool that could maintain our regression suite. That's the major problem testRigor solves for us."**
>
> **Vignesh Jayamuthu**
> **Senior QA, Gallabox**

Solution Highlights

- Grew from **zero to 1,800+ automated scenarios**
- Expanded documented test scenarios from approximately **400–500 to 2,000+**
- Reduced regression execution from **more than 3 days manually to approximately 5–6 hours**
- Supports approximately 3 releases every week
- Enabled manual testers to participate in automation
- Made it easier to convert manual regression scenarios into automated tests
- Uses **Claude with testRigor MCP** to create automation scripts within minutes
- Provides screenshots, recordings, and logs for easier failure analysis
- Reduced repetitive manual regression effort
- Integrated automated execution notifications with **Slack**
- Built a scalable regression process for a rapidly changing SaaS platform

Before testRigor

- 400–500 documented test scenarios
- Zero automated scenarios
- Regression testing took more than 3 days manually
- Manual regression struggled to keep pace with 3 releases per week

During and After testRigor

- 2,000+ documented test scenarios
- 1,800+ automated scenarios
- Automated regression runs in approximately 5–6 hours
- Automation helps QA reliably support 3 releases per week

The Result

Gallabox can now maintain its release velocity without regression testing becoming a bottleneck, reducing execution time from more than three days manually to approximately **5–6 hours** for over 800 workflows. This enables the team to confidently support approximately three releases every week while focusing manual effort on higher-value testing.

The company has also expanded from **400–500 documented scenarios to more than 2,000,** including approximately **1,800 automated scenarios.** As a result, regression-related defects reaching production have dropped drastically and are now close to zero, with most remaining issues tied to new functionality.

Summary

Objective	Result
Build Test Automation	0 → 1,800+ automated scenarios
Expand Test Coverage	~400–500 → 2,000+ documented scenarios
Accelerate Regression	3+ days manually → ~5–6 hours automated
Support Release Velocity	~3 releases per week
Accelerate Test Creation	Claude + testRigor MCP converts scenarios into automation within minutes
Reduce Manual Effort	Majority of regression moved to automation
Simplify Debugging	Screenshots, recordings, and logs help pinpoint failures
Reduce Regression Escapes	Regression-related production bugs reported as close to zero

testRigor can also help you

"If you're looking for a reliable and stable automation regression testing tool, I would recommend testRigor."

Vignesh Jayamuthu
SENIOR QA, GALLABOX

By making test automation accessible in plain English, testRigor helps QA teams create, maintain, and troubleshoot reliable tests without requiring deep programming expertise. It reduces automation maintenance and repetitive manual effort, enabling teams to expand coverage, accelerate releases, and deliver with greater confidence. testRigor's capabilities make automation simple across web, mobile, desktop, APIs, databases, AI features, and mainframe applications.