SauceLabs

WHITE PAPER

The Complete Guide to Low-Code Test Automation

LEARN MORE AT SAUCELABS.COM

UPDATED MARCH 2022

Introduction

It's hard to overstate how challenging software development is today.

The market for web and mobile applications is more competitive than ever. Customers expect their applications to deliver a flawless, beautiful, and intuitive experience every time. Hiring and retaining high-quality software developers and engineers feels next to impossible given the talent shortages emerging organizations are facing.

How can companies balance these competing challenges and expectations? How can they scale their development operations effectively to deliver at speed without compromising the customer experience?

This is where low-code test automation can help.

Table of Contents

4 What is Low-Code Test Automation?

4 Why Low-Code Test Automation?

5 How AI-Powered Low-Code Can Address Test Automation Challenges

6 Key Low-Code Testing Strategies

6 Understand the Costs of Not Adopting Low-Code Test Automation

6 Measure the Impact of Low-Code Test Automation

6 Include End-To-End Test Automation

7 Ensure Development and Testing Happen in Parallel

7 How to Choose the Right Low-Code Test Automation Platform

7 For Your Organization

7 Skillset and Type of User

8 Use Cases

8 Collaboration and Reusability

8 Key Features to Look for in Low-Code Test Automation Platforms

9 What is the ROI of Low-Code Test Automation?

9 Cost Savings

9 Resource Savings

9 Reduced Test Debt

9 Case Study: Aryaka Networks Accelerates End-to-End Platform Coverage Using Sauce Low-Code

9 The Challenge

10 The Solution

10 The Results

11 Conclusion

SauceLabs

What is Low-Code Test Automation?

Low-code testing tools simplify application testing by removing most, if not all, manual coding from the process. This allows so-called “citizen testers”—typically product managers, product owners, business analysts, etc. without specialized technical skills—to participate in the testing process and ultimately get more involved in the software development lifecycle (SDLC).

Why Low-Code Test Automation?

According to Gartner’s IT Automation Predictions for 2021, improvements in automation capabilities will refocus 30% of IT operation efforts from support to continuous improvement. The citizen-code/no-code/low-code movement promises a new era of rapid and continuous application development whether teams are leveraging enterprise-packaged SaaS tools (like Oracle, Salesforce, and WorkDay) or building their own custom web and mobile applications.

By essentially democratizing software development and lowering the barriers to testing through an intuitive and globally understood framework, low-code test automation offers a more agile alternative that can help companies scale their development organizations to keep up with today’s most pressing demands and expectations. In short, low-code test automation can bring enormous value to your business.

Low-code also helps to address one of the most pervasive problems faced by QA teams in today’s world of rapid, continuous development: test debt. Test debt is incurred when there is a failure in testing that is not fixed before the code is released to production. For example, if your test automation suite fails to identify the correct UI element in your application, then you have created some extra work for yourself: you will need to course correct that automation script as soon as possible and fix it before it causes trouble in production.

Test debt can be more insidious if it’s not constantly measured and kept in check. However, just knowing about it is not enough. QA teams need to be able to reduce test debt in a coordinated manner. Low-code testing enables this coordination by enabling collaboration between cross-functional teams without the barrier of needing a highly technical skillset. Low-code testing represents a unique opportunity to drive out a major structural cost center—test debt–and replace it with an “always-on” infrastructure that is not only more efficient but also more powerful in terms of its scalability, coverage, awareness of change, and responsiveness to downstream factors. High-performing product teams receive the compound benefits of more frequent deployments, shorter lead times, and lower change fail rates.

How AI-Powered Low-Code Can Address Test Automation Challenges

CHALLENGE	HOW AI-POWERED LOW-CODE TEST AUTOMATION HELPS
Poor agility and speed	**Rapid application development (RAD):** Low-code test automation supports heightened agility when developing and deploying new software. Using the power of AI, low-code helps transform testing from a bottleneck to an integrated accelerator since tests can be executed in-sprint.
Dynamic applications are difficult to automate	**Self-Healing Scripts:** The changing nature of dynamic objects in modern low-code application platforms (e.g., Oracle, Salesforce) can lead to unstable scripts. These scripts are better maintained by AI-powered test automation tools that can self-heal dynamic objects and frames.
Integrations and upgrades	**Seamless API testing:** Intelligent test management across multiple applications increases the breadth, scope, and velocity of end-to-end tests across various platforms. It also accelerates deployment and reduces defects, especially when delivering APIs.
Influx of Shadow IT	**Embracing new tools to reduce shadow IT:** IT executives can embrace the value new tools and integrations provide while ensuring there is a process in place to test and maintain these tools. Low-code test automation acts as the glue that holds this together since end-to-end regression test suites can be created across all of your company's critical tools and non-technical employees are not hampered by the lack of technical knowledge.
Maintenance burden	**Lower maintenance burden:** Increasing the velocity of code deployments leads to more testing and script maintenance. Compared to traditional test automation frameworks, low-code test automation removes the need to manually code automation scripts.
Tech talent shortage	**Empower citizen testers:** With a low-code test automation platform, anyone in the organization can test your software. This helps companies address staff deficits and ensure in-sprint test automation can still be achieved.
High IT costs	**Lower IT costs:** Companies can lower their IT development costs by adopting a low-code test automation tool to deliver new innovations to their customers with confidence.

Key Low-Code Testing Strategies

Understand the costs of not adopting low-code test automation

There are costs to not adopting low-code test automation and allowing test debt to accumulate in your testing lifecycle. These costs manifest on the P&L balance sheet rather than in bloated cash/interest payments. Here are three ways test debt creates costs for your business:

- **Headcount** - More human resources are needed to maintain test debt, but your overstretched technical team already spends too much time on tedious, low-fidelity work.
- **Revenue** - System outages can delay critical business processes (e.g., the ability to convert a prospect to a sale), losing revenue in the process and leading to a less efficient marketing spend.
- **Lower staff productivity** - Addressing technical debt distracts staff from completing higher-value work.

The lack of formal oversight over technical debt is one key difference from financial debt, where there are usually credit committees, asset and liability management teams, and treasury staff that monitor debt levels like a hawk. In QA departments, few of these controls exist. This leads to surprise costs on the P&L balance sheet.

Measure the Impact of Low-Code Test Automation

Here are some metrics to look at:

- **Bugs** - How many bugs make their way into production and disrupt the customer experience? QA teams should keep track of both fixed and unfixed bugs. Keeping track of unfixed bugs allow teams to amplify their focus on key aspects of future test automation. Taking note of the fixed bugs helps teams measure how effective their test debt management is.
- **Code Quality** - Test automation cannot guarantee code quality, but it can evaluate the quality of software at a variety of levels, from individual components or modules throughout the whole system. Code quality is a measure of how well the code can respond to changes in requirements, typically related to maintainability and performance. Poorly written or poorly maintained code can lead to inefficient use of resources, difficult bugs to fix, security vulnerabilities and increased operational costs. QA teams should keep track of unit test coverage and design patterns used in test automation. By ensuring that individual units of code have been tested by at least one unit test, this increases test coverage and reduces errors in later stages of production. In addition, taking a qualitative measure of the design pattern and applying a universal approach to all test automation makes it easier for team members to build on the existing library of test assets. Less guesswork means less test debt.
- **Churn** – Churn happens in test automation when the test scripts get refactored, updated or replaced. Measuring churn helps QA teams recognize the constant level at which test automation assets need to be re-done. Having an objective view of this helps organizations plan and solve for these challenges more quickly, allowing them to be more proactive in how they manage test debt.

Include End-To-End Test Automation

End-to-end tests are important because they can exercise your system from start to finish, unlike unit or integration tests, which only cover one small area of functionality. While end-to-end tests have historically been time-consuming and difficult to set up, low-code testing tools make it easier for testers on any team (including those with just one tester) to drive end-to-end test automation.

Here are the key benefits of including end-to-end testing in your low-code test automation strategy:

- Ensures the health of your entire application
- Applies behavior-driven development to end-to-end tests to ensure that functionality is optimized for customer experiences
- Tests the logic of your business flows

End-to-end testing needs to be planned from the inception of your project. Once your team understands that end-to-end testing is a key strategy, you can integrate test automation as a method to reduce any repetitive actions.

Ensure Development and Testing Happen in Parallel

AI-powered low-code testing can drive out a tremendous number of technical challenges when innovating and delivering new products. Citizen testers can access hundreds of prebuilt test cases across the various modules and add them to any test scenario with a single click. These test cases can be written in plain English when using natural language processing (NLP) to manage critical business processes.

Low-code testing also further enables accelerated testing by ensuring that powerful API tools work in parallel to precisely detect and diagnose bugs in-sprint. This orchestration significantly accelerates the time to resolution and helps resolve issues before deployment. This gives IT owners the confidence to accelerate releases and integrate APIs across the product landscape with reduced risk.

AI-driven low-code testing removes human error from repeatable processes and what were once highly technical tasks. These efficiencies also allow for a massive increase in regression testing with intelligent scope management, allowing teams to innovate with API-first strategies without increasing risk.

How to Choose the Right Low-Code Test Automation Platform

For Your Organization

Low-code test automation success requires more than just the platform's technology and functionality. You also need to objectively evaluate whether the platform supports the projects you plan to tackle, suits the skills of your people and organizational processes, and offers access to a robust testing tool chain.

Here are some important criteria to consider as you explore low-code test automation solutions.

Skillset and Type of User

The ideal low-code test automation platform enables all team members to adapt the software according to their role. This can be done through a collaborative interface that provides standard no-code options for non-technical users and allows more technical power users to create reusable components for use by anyone across the organization. These power users can also scale up the complexity of the test assets to achieve more regression coverage in your environment.

Here are some questions to consider when evaluating the organizational fit of a low-code test automation platform:

- How accessible is the platform to various user profiles in your organization?
- Is there a certain level of experience and coding skills needed to achieve business value?

- Could more tech-savvy users build a more complex set of test cases to help achieve better regression?
- To what extent are coding skills necessary and to what degree can your team create all the test assets needed?

Use Cases

As you know, not every type of application and use case suits the same set of functionalities. You should evaluate the flexibility of a low-code platform by asking which applications/use cases best match your organization's requirements and whether the platform can go beyond those requirements. Next, focus on complexity-related use cases: ask if the vendor has examples of complex use cases that they have.

Some test automation tools can execute against a certain application really well. For example, they might be extremely good at pre-packaged enterprise applications like Salesforce or Oracle. They might be very good at custom applications or cover a broad range of use cases from legacy migration, operational efficiency or various industry-specific products. Having a better grasp of the niche a test automation tool focuses on is as simple as evaluating the case studies of that vendor.

Collaboration and Reusability

Collaboration and reusability are major components to consider when evaluating low-code testing tools. The ability to reuse test assets/logic helps create consistency amongst your test suites, which can lead to increased usability, accuracy of information, and regression while decreasing in-sprint testing cycles.

Here are some questions to consider when evaluating whether a low-code test automation platform support reusability:

- Are there pre-built test modules available, such as common Oracle or Salesforce scenarios?
- Can these modules be used more than once so the effort to create new scripts is decreased?

Key Features to Look for in Low-Code Test Automation Platforms

Key Features	Traditional Test Automation	Low-Code Test Automation
Ability to add custom code	Yes	Yes
Codeless user interface	No	Yes
Point and record interface	No	Yes
Reusability	Yes	Yes
Scalability	Yes	Yes
Cross-platform accessibility	Yes	Yes
English-to-code test cases	No	Yes
Self-maintaining scripts	No	Yes
End-to-end testing	Yes	Yes
Designed for	Developers, automation engineers	Citizen testers, business users, QA specialists, developers, automation engineers

What is the ROI of Low-Code Test Automation?

Low-code test automation tools cut down on the time, cost, risk and general inefficiencies of standard practices today. With these tools, a disjointed collection of engineers and citizen testers can transform into more collaborative and efficient teams.

Cost Savings

When companies transition to intelligent low-code test automation, they typically see cost savings of 25% to 75%. Not all core elements of your business can be completely automated, but identifying a few areas—generating automation scripts, managing and healing test scripts, cross-browser and cross-platform automated testing—can allow your team to focus on other aspects of the business. This helps reduce labor costs and improve employee productivity.

Resource Savings

Low-code automation can take English-written test cases created by anyone and generate automation scripts within a few clicks. This means anyone can be a tester. The burden of writing automation code is removed from the process, meaning a new cohort of citizen testers are provided the autonomy to play a major role in delivering software quality at speed in a more efficient and effective way.

Reduced Test Debt

Intelligent testing provides a more efficient and effective way to manage test debt since the burden of tasks like maintaining scripts is removed from the already overwhelmed engineering team. AI-powered test automation also provides the possibility to drive out major cost centers across the software testing lifecycle: the cost of outsourcing, the cost of recruiting, the cost of staffing up/onboarding, the cost of manual testing, the cost of managing change, and so on. This contributes to an intelligent infrastructure that rarely experiences downtime caused by a lack of human coordination.

Case Study: Aryaka Networks Accelerates End-to-End Platform Coverage Using Sauce Low-Code

The Challenge

The internal business technology team led by Venkat Ranga, Head of Business Information Systems, was tasked with developing a large-scale implementation journey that involved many critical business processes and applications like Salesforce. They needed a way to get more out of their bi-weekly release goals—and with an average of eight new end-to-end user scenarios introduced every sprint, their current QA process was unable to keep up.

“As a leader, first and foremost, you have to look at how to create value for the organization using the technology,” Venkat shares. “In the traditional world, the technology team is waiting for requirements to come from the business so they can build the solution. But now, this technology team is also working very closely with the business and enabling the solution. If I end up giving 3-4 days development time within the two week sprint for the QA, I am not really achieving a lot. My goal is to see how I can really reduce that time so that I can push through more changes within the system.”

Venkat's team had to evaluate and overhaul many critical business processes, including quote to cash, procure to pay, record to report, hire to retire, and others by mobilizing a multifaceted solution to optimize how they are delivered. This major implementation journey relied on the integration of many connected systems, such as Salesforce CPQ, Zuora, and NetSuite, and they had to ensure that introducing new features to the technology infrastructure would not compromise the existing code base or functionality.

The Solution

Venkat and his team found the answer to their challenge through Sauce Low-Code AI-driven codeless studio. They initially started to explore Sauce Low-Code for its rapid Salesforce end-to-end testing capabilities, but quickly realized that the tool could be used with all their custom and enterprise applications.

As a low-code platform, Sauce helped Aryaka deliver transformation projects faster by accelerating the testing of full regression cycles and keeping the regression suite up-to-date. Many presets come out-of-the-box for Salesforce-specific test assets, which offer unmatched workflow flexibility and shorten the time it takes to complete in-sprint QA.

In addition, Venkat understood that to keep his team at the forefront of their career trajectory, the role of business analyst and quality assurance manager had to be combined. "Now that most of the [technical Selenium script writing and maintenance] work is automated, I'm starting to move them into BA [business analyst] roles because they understand the system, they understand the business processes, they have been testing this for the last 18 months, and they can talk about exactly what is going on in the business," Venkat added.

The Results

Venkat and his team have been using Sauce Low-Code for six months and the AI-driven codeless studio continues to play a role in the overall business transformation. Aryaka has seen significant reduced cost of operations and they have been able to deliver more user stories in their sprint cycles.

> "I cater at most three days for testing in a two-week sprint and my goal was to cut down to one day. We were able to accomplish that with Sauce Low-Code."
>
> **—Venkat Ranga (Head of Business Information Technology Systems)**

Low-code business technology teams are starting to be very prevalent in the workforce. Under Venkat's leadership, Aryaka Networks has embraced this change which has contributed to the overall reduction of their testing efforts. They have increased business value by adding more user stories to their sprints and expanded the QA's responsibility to take up a business analyst role with the power of Sauce Low-Code's low-code and no-code test automation tool.

Conclusion

Today's software development teams are more resource-crunched than ever, but the pressures to deliver quality at speed are only increasing. AI-driven low-code test automation offers a compelling solution to this problem by lowering the barriers to testing across your organization.

By making everyone a developer and every developer a tester, low-code testing effectively democratizes the software development process and breaks down organizational silos, empowering cross-functional teams to collaborate more effectively and achieve innovation breakthroughs that wouldn't have been possible before. By mitigating risk, accelerating innovation, and increasing efficiency across the entire software development lifecycle, low-code test automation can bring enormous value to your business through reduced costs, higher-quality products, and happier customers.

About Sauce Labs

Sauce Labs is the leading provider of continuous test and error reporting solutions that gives companies confidence to develop, deliver and update high quality software at speed. The Sauce Labs Continuous Testing Cloud identifies quality signals in development and production, accelerating the ability to release and update web and mobile applications that look, function and perform exactly as they should on every browser, operating system and device, every single time. Sauce Labs is a privately held company funded by TPG, Salesforce Ventures, IVP, Adams Street Partners, and Riverwood Capital.

For more information, please visit
→ saucelabs.com

saucelabs.com/sign-up

FREE TRIAL

SAUCE LABS INC. - HQ 450 Sansome Street, 9th Floor, San Francisco, Ca Usa 94111

SauceLabs