An introduction to Salesforce testing.

Contents.

Introduction.

Why Salesforce testing is important.

Since its launch on February 7th, 2000, Salesforce was little more than a contact database to store customer details, track sales, and send the odd email or two.

Fast forward to now, and Salesforce is much more than a simple customer relationship management (CRM) system. With circa 20% of the global CRM market and over 150,000 customers, Salesforce's dominance doesn't look like it's letting up anytime soon.

As the numbers grow, so do the possibilities of how enterprises use Salesforce to run their organizations. By adding processes, workflows that require complex business rules, and application integrations, Salesforce is pivotal for organizations to enable growth and deliver an exceptional customer experience.

To do this, you need a robust testing Salesforce strategy. But despite efforts to standardize deployments, with thousands of possible integrations, and business logic to support, no two companies will implement Salesforce the same way.

Add to this the nudge to upgrade from the original Salesforce Classic to the snazzy modern UI of Salesforce Lightning, you're adding another layer of complexity when testing.

This guide will highlight the main testing challenges to overcome and best practices, so your Salesforce platform supports your organization to meet business objectives after every update, upgrade and implementation.

5 challenges to overcome when testing Salesforce Lightning.

Overview.

Such is the critical role Salesforce plays for thousands of organizations, testing it has become crucial. Salesforce must deliver a single source of truth, so testing functionality changes after every release, app integrations, and business-critical workflows are vital to keeping your business moving.

These five challenges below must be considered before you begin testing your Salesforce platform.

1.

Shadow DOM.

As more organizations transition over to Salesforce Lightning for improved functionality, be prepared for changes to how web components render the UI and its composition.

The challenge here is Salesforce's use of the Shadow Document Object Model (Shadow DOM), which wraps up and protects the internal DOM, which is how the content of a webpage is structured.

By protecting the internal DOM or shadow tree, Salesforce's web components are prevented from being accessed and manipulated by programming languages such as HTML, CSS, and Javascript. This can prevent Object or DOM test automation tools from working because the scripts they use are typically written in Javascript.

2.

Salesforce updates.

Salesforce's main releases occur three times a year - Winter, Spring, and Summer. There are also plenty of other upgrades and maintenance updates to contend with. For the three major releases, your teams have time to prepare before testing begins, but some updates are pushed directly to your production environment.

And with every new release, update, and upgrade, new functionality and UI improvements are introduced, resulting in changes to object IDs. These changes dramatically increase test maintenance as scripts created by automation tools that solely rely on static IDs break.

The same thing happens when organizations upgrade or migrate from Salesforce Classic to Lightning. The core functionality of your Salesforce platform remains the same, but the changes in the code alter object IDs.

Not only do tests fail, but a lot of work and effort is required by your teams just to uncover which part of the DOM has changed.

3.

Validating complex UI.

Compared to Salesforce Classic, Lightning offers improved UI for greater functionality, which is great for your users, but it does make testing more challenging.

New features, such as iFrames, drop-downs, and pop-up windows, while offering a better experience, can take time to load on the screen. While loading, Salesforce Lightning adds an overlay that masks all underlying object identifiers, causing the same problem associated with Shadow DOM.

This overlay may cause issues for object-based tools because the built-in wait conditions in the code may cause a test to fail before functionality returns. In reality, it is the slowness of Salesforce loading which is the problem. The object-based tool can't see the onscreen loading panel in the code, meaning you need a solution capable of validating the UI, as well as verifying the code.

4.

Custom workflows.

Salesforce offers more than just simple customer relationship management; it can create custom user workflows to support business-critical processes. Internal processes and procedures can be automated, helping organizations save time, increase productivity and improve efficiency.

These custom workflows and business rules will typically follow a step-by-step process and include intricate rules, such as mandatory fields, business rules, and "if/then" statements, to ensure they function correctly.

But because Salesforce's usability is so fluid, you can't rely on only testing at the code level; testing must validate the UI as well. If business rules are incorrectly implemented, mandatory fields may not be entered, which critical internal processes require.

For instance, what would happen if a business rule was wrong, and a user could not complete a multimillion-dollar deal?

With the sheer number of potential actions, paths and, routes a user can take, manually testing every permutation takes too long. Manual testing can also introduce human error when identifying any bottlenecks or gaps preventing workflows from supporting business logic.

5.

App integration.

To further support business-critical workflows, Salesforce allows the integration of a range of applications. Salesforce testing is no longer just about testing Salesforce –it's about testing the whole ecosystem of applications that are used alongside it.

Apps within your CI/CD pipeline, such as Github, Jira, and Slack, can be integrated to enable DevOps. Hubspot and LinkedIn can be linked to support marketing and sales, along with a whole host of other apps.

However, with the introduction of different codebases and dynamic Object IDs, the higher the likelihood of failed tests either when testing manually or using the wrong tools.

Best practices
for Salesforce Lightning testing.

Align IT and business for custom workflows.

QA Engineers understand how to write scripts. Business Analysts understand which business process needs to be integrated with your Salesforce platform. By bringing both together early, custom business-critical workflows can be identified and created to maximize efficiency and productivity throughout your organization.

Test a wide variety of user experiences.

Create numerous test cases for various roles across your organization, as your team will be using Salesforce in very distinct ways across multiple technologies.

User experience (UX) testing is also beneficial when mandatory fields with specific rules are used for custom workflows. These rules can be verified in the code, but to ensure they are implemented correctly and support business logic, they must be validated at the UI level.

And because Salesforce's UX is highly flexible, exploratory testing is recommended to cover the multiple paths a user can take along a custom workflow.

Don't only test Salesforce.

Salesforce has grown to become an essential piece of software for hundreds of thousands of organizations by supporting the integration of multiple applications and plugins to enable business growth.

End-to-end testing is a vital step in any framework to ensure all integrations, such as Google Cloud, ServiceNow, and Slack, support business-critical workflows within your Salesforce platform.

Automate to keep pace.

Numerous upgrades that break tests. Complex UI. Custom workflows. Multiple integrations. The Salesforce ecosystem is constantly expanding, and manual testing just can't keep pace with new apps, user flows, and the constant demands when trying to scale your organization.

If you are trying to test all these updates and new configurations manually, the efficiency of your Salesforce is impacted significantly. Productivity is reduced, as is your ability to use the platform to support business objectives, leading to a poor customer experience.

Test automation is the only way to ensure updates and system improvements happen fast enough for your teams to outperform your competitors.

Testing of internally-developed apps.

Internally-developed apps often support business-critical workflows. While powerful, Salesforce DX requires additional tools for app development beyond continuous integration tools like Jenkins. Therefore, third-party DevOps tools are often needed for effective release management. This demands that workflows be accounted for in your testing plan.

5 key areas for your testing checklist.

1.

User interface.

UI testing is imperative to validate how the users see and interacts with Salesforce. Custom workflows also introduce mandatory fields that enable business rules, which must also be validated at the UI level. If your users enter the wrong data or accidentally skip a field, sales deals could be lost, planning skewed, or an order might never be fulfilled.

And with a lot of Salesforce instances behind hosted in the cloud, UI testing should extend to browsers – organizations will have a preference about which browser they use. UI testing in this manner will ensure all users have the same experience across different browsers and devices, such as a smartphone or laptop.

Offline capability testing must be included too, as some users, particularly salespeople, will need access to customer or prospect information while in the field.

2.

Performance.

Closely linked with UI testing, testing your Salesforce platform's performance is necessary to check it can withstand the stress of real-life usage.

With numerous users interacting with it across various technologies and integrated applications, testing ensures Salesforce can perform under peak conditions. Maintaining performance with no lag or downtime is essential for users to remain productive even when hundreds or thousands access the system at once.

Don't forget to test:

→ Peak number of users
→ Peak conditions (max users, end of month etc.)
→ Operating capacity
→ Application and custom work low integrations
→ Response time
→ Loading speed

3.

End-to-end.

As Salesforce systems become more complex, end-to-end testing is required to maintain operational excellence to support business objectives. Custom workflows, application integrations, and front-end and back-end functionality are vital areas that must be tested to identify any bottlenecks that may impact productivity.

But as more Salesforce instances move to the cloud, integrations extend to apps and processes that support customer interactions, such as:

→ Website

→ Customer service

→ Mobile applications

→ Social Media

→ Email

→ Web chats

→ Point of Sale

→ ERP

One thing to remember before you begin end-to-end testing – ensure any test automation solution you use verifies if the codebase is correct AND tests the UI to make sure it functions as expected.

4.

Functionality.

Functionality testing should cover all necessary functions, accessibility considerations, and custom workflows to ensure seamless operation for all users from all departments.

Sales teams need all customer information — be it leads, online forms, or prospect follow-ups — linked together to have a complete view of all interactions. Marketing requires campaign stats, email effectiveness, and social media exchanges all in one place. Customer services need to be able to prioritize tickets raised and inform relevant teams of any customer-related issues or queries, and so on.

Combine testing technologies so the codebase is verified, and UI is validated to ensure your Salesforce platform functions as expected.

5.

Exploratory.

Salesforce's ability to allow organizations to create business rules to drive custom workflows increases the need to test every permutation of how users interact with the underlying code and at the UI level.

Just because a custom workflow has five clear steps, users will not necessarily move down that linear path in sequence. If one step can be accessed from a different route but fails to conform to the business rule, a user may need to go back and refresh the page to complete an action, which takes time and will impact productivity.

By testing every possible user journey, custom workflow, and business rule, you can stop this from happening.

And combining exploratory testing with AI, coverage is significantly increased through auto-generating test cases to cover all possible user journeys. Even the best manual tester would struggle to develop all necessary user journeys to match this intelligent way of increasing test coverage.

Conclusion.

Keysight's Salesforce testing strategy.

[Recognized by Forrester](#) and Gartner as [leaders in software test automation](#), Keysight's Eggplant AI-assisted testing solution is uniquely positioned to meet the challenges of Salesforce testing head-on.

Recognized by Forrester and Gartner as leaders in software test automation, Keysight's Eggplant AI-assisted testing solution is uniquely positioned to meet the challenges of Salesforce testing head-on.

By having the ability to interact with a range of technologies, custom workflows, application integrations, Eggplants testing capabilities far exceed other testing tools that solely rely on verifying the codebase. Here's why:

Model-based approach.

Auto-generate user journeys that span different devices, browsers, and operating systems, as well as business rules that support custom workflows. One model is all you need to test both versions of Salesforce as well. Salesforce as well.

No code/low code approach.

Easily bridge the gap between IT and the Business by using a no-code/low code approach, allowing anyone to carry out software testing, regardless of technical ability. People across your organization can take advantage of test flows being auto generated by simply clicking through a test model to design a user journey.

Test any technology at any level.

Intelligent testing that interacts with any text or image on a screen to validate the UI. When validating the UI, the same test snippets can also be used across all versions of Salesforce. Eggplant can also use objects for scenarios where data needs to be extracted from a table, plus it can make back-end API calls to verify updated data is correct.

AI-assisted automation.

Intelligent testing is available via AI-assisted automation to generate essential and business-critical user flows throughout Salesforce. The AI engine also monitors exactly how Salesforce is being used to auto-generate future tests based on real user journeys.

Test from the user perspective.

Automate processes just as a human would to ensure Salesforce is functioning as expected by testing not only what you think your users will do but everything they can do. Don't rely only on the code being verified to pass a test; validate the UI with Keysight's intelligent computer vision and be sure no false positives are returned.

So, to dramatically reduce test maintenance, future-proof your Salesforce platform, and maintain business continuity, contact Keysight today to prevent any tests from breaking with every release and UI change. Now and in the future.

Eggplant is now part of Keysight Technologies.

Keysight Technologies Inc. (NYSE: KEYS) is the world's leading electronic measurement company, transforming today's measurement experience through innovations in wireless, modular, and software solutions. With its Hewlett-Packard and Agilent legacy, Keysight delivers solutions in wireless communications, aerospace and defense and semiconductor markets with world-class platforms, software and consistent measurement science. The company's nearly 12,600 employees serve customers in more than 100 countries.

eggplant®

PART OF
KEYSIGHT
TECHNOLOGIES