A Strategic Guide to Salesforce Testing

eBook

KEYSIGHT

eggplant
Test Automation Software

Contents

CHAPTER 1

Why Testing Salesforce is Important

CHAPTER 1

Why Testing Salesforce is Important

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

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

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

You need a robust Salesforce testing 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 requirement to upgrade from the original Salesforce Classic to the modern user interface (UI) of Salesforce Lightning — you are adding another layer of test complexity.

This eBook highlights best practices to overcome Salesforce testing challenges to ensure the platform supports your organization so users can meet business objectives after every update, upgrade, and implementation.

CHAPTER 2

Five Challenges and the Test Strategies to Use

CHAPTER 2

Five Challenges and the Test Strategies to Use

Salesforce plays a critical CRM for thousands of organizations; testing the platform has become crucial. Salesforce must deliver a single source of truth, so testing functionality changes after every release, application integrations, and business-critical workflows are vital to keeping your business advancing.

The following are five key challenges and strategies to consider when testing your Salesforce Lightning platform.

Challenge 1

Shadow Document Object Model

As more organizations transition to Salesforce Lightning for improved functionality, they must prepare for how web components render the UI and its composition. The challenge is Salesforce's use of the Shadow Document Object Model (DOM), which wraps up and protects the internal DOM, which structures the content of a web page's content.

Programming languages such as HTML, cascading style sheets (CSS), and JavaScript cannot access Salesforce's protected internal DOM or shadow tree web components. These tools also rely on underlying object identifications (IDs) in the code to precisely identify the focus of testing.

Further problems arise because these IDs are dynamic, meaning they change after every new release. Consequently, pre-existing scripts from an object or DOM tool will break as they search for IDs that no longer exist.

Strategy

The best practice to solve this challenge is to use a testing solution that combines object-based testing with image-based testing. This combination of tools validates the user experience while verifying the object IDs within the DOM. Users can access the metadata when the test engine interfaces directly with the Salesforce API. An object, such as an account, contact, or case, and connected inputs, such as a checkbox, picklist, or text areas are detectible to the test software.

Challenge 2

Salesforce updates

Salesforce's main releases occur three times a year, along with additional upgrades and maintenance updates. Your teams have time to prepare for the three main releases before testing begins. However, some updates release automatically to your production environment.

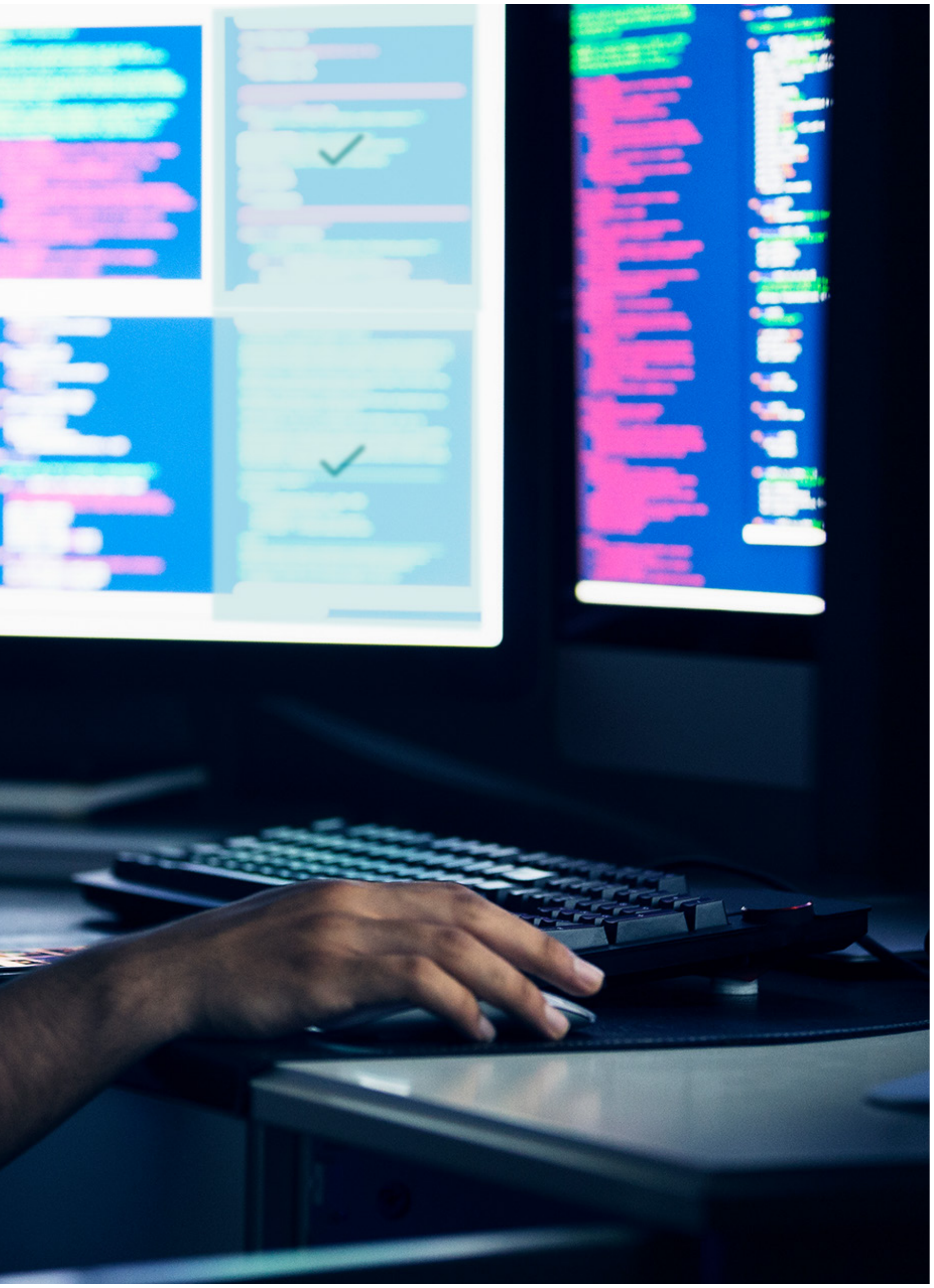
Every new release, update, and upgrade introduces new functionality and UI improvements, resulting in changes to object IDs. These changes dramatically increase test maintenance. Because some automation tools create scripts that solely rely on static IDs, these scripts will break.

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

Teams will often rush testing, resulting in releasing a version of Salesforce that does not allow the business to function as expected. Not only do tests fail, but a lot of work and effort is necessary for your teams to uncover which part of the DOM has changed.

Strategy

Overcoming these challenges depends on test automation tools that do not rely solely on object ID verification. It is beneficial if the software can reuse the same test automation assets across Salesforce's Classic and Lightning versions, including third-party applications and systems.



Challenge 3

Validating a complex UI

Compared to Salesforce Classic, Lightning introduces new navigation, layouts, list views, filters, dashboards, integrations, and workflows. While these improvements are helpful for users, each feature increases the number of user journeys to test.

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, the slowness of Salesforce loading is the problem. The object-based tool cannot see the onscreen loading panel in the code, meaning you need a solution capable of validating the UI, as well as verifying the code.

Strategy

Instead, it would be best to have a testing solution that does not rely solely on objects and can handle various new features and UI functionality changes from the user's perspective.

Another way to overcome this challenge is to use a model-based approach powered by artificial intelligence (AI). This strategy enables you to use the same script to test different versions of Salesforce across various browsers, devices, and operating systems.

Challenge 4

Custom workflows

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

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

Because Salesforce's usability is so fluid, you cannot rely on only testing at the code level; testing must validate the UI as well. If users incorrectly implement business rules, they may miss some mandatory fields, which critical internal processes require. For example, 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 bottlenecks or gaps preventing workflows from supporting business logic.

Strategy

Modern testing solutions generate an accurate map of Salesforce data flows. This process creates a library of pre-written automation scripts that save users days of time and effort compared to a manual testing framework.

To customize tests, users can manipulate the data within a model to replicate workflows that are specific to their business needs.

Challenge 5

Application integration

To further support business-critical workflows, Salesforce enables 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.

Applications within your continuous integration / continuous delivery (CI / CD) pipeline, such as GitHub, Jira, and Slack, can integrate and enable development and operations (DevOps). HubSpot and LinkedIn can link to Salesforce to support marketing and sales, along with a whole host of other applications.

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.

Strategy

Your testing plan needs to handle third-party application integrations effectively with the various customization options available to your teams. User journeys may span multiple applications, so make sure to test these thoroughly.

A comprehensive test automation tool is critical for modern businesses when testing Salesforce. The most effective solution is software that interfaces directly with the Salesforce API to auto-generate all test automation assets by collecting Salesforce metadata.

CHAPTER 3

Salesforce Testing Best Practices

CHAPTER 3

Salesforce Testing Best Practices

Salesforce is critical to organizations around the world. Departments including sales, operations, marketing, and finance depend on it functioning as expected. A vast number of third-party applications can integrate with it as well.

As a result, numerous user journeys, input fields, and custom workflows unique only to your business are the foundation of your Salesforce instance. This complexity means that you must test every action, integration, and customization across various user interfaces to support business-as-usual. Here are some best practices for testing Salesforce.

IT and business alignment for custom workflows

Quality assurance engineers understand how to write scripts. Business analysts understand which business process needs to integrate with your Salesforce platform. By bringing both together early, teams can identify and create custom business-critical workflows to maximize efficiency and productivity throughout your organization.

Test a variety of user experiences

Create numerous test cases for various roles across your organization, as your team uses Salesforce across multiple technologies to meet their specific business requirements.

User experience (UX) testing is also beneficial when using mandatory fields with specific rules for custom workflows. You can verify these rules in the code, but to ensure they run correctly and support business logic, you must validate these fields at the UI level.

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

Test other applications

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

Numerous upgrades break tests. Complex UI, custom workflows, and multiple integrations require multiple tests. The Salesforce ecosystem is constantly expanding, and manual testing cannot keep pace with new applications, user flows, and the constant demands when scaling your organization.

Trying to test all these updates and new configurations manually will significantly impact the efficiency of your Salesforce instance. There is a reduction in productivity, 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.

Test internally-developed applications

Internally-developed applications often support business-critical workflows. While powerful, Salesforce's developer experience (DX) requires additional tools for application development beyond continuous integration tools like Jenkins. Developers often need third-party DevOps tools for effective release management. It is critical that your testing plan accounts for workflows.

CHAPTER 4

Salesforce Testing Checklist

CHAPTER 4

Salesforce Testing Checklist

☑ User interface

UI testing is imperative to validate how the users see and interact with Salesforce. Custom workflows also introduce mandatory fields that enable business rules, which you must also validate at the UI level. If users enter the wrong data or accidentally skip a field, sales teams could lose deals, inaccurate planning, or fulfillment centers might never receive an order.

Since many hosted Salesforce instances are in the cloud, UI testing should extend to browsers — organizations may have a preference about which browser they use. UI testing in this manner ensures all users have the same experience across different browsers and devices, such as a smartphone or laptop.

It is crucial to include offline capability testing because some users, particularly salespeople, will need access to customer or prospect information while in the field.

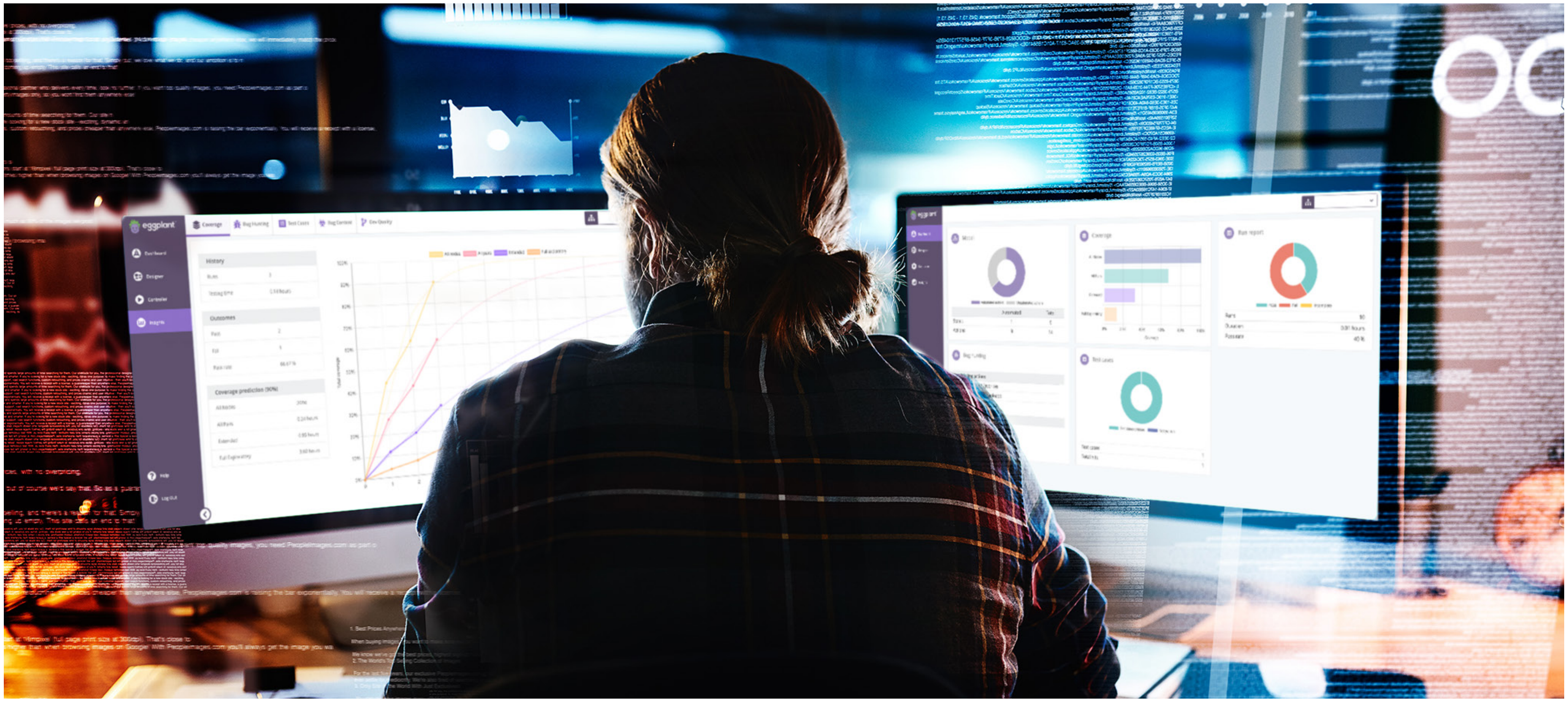
☑ Performance

Testing your Salesforce platform's performance is necessary to check that 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 many conditions. Maintaining performance with no lag or downtime is essential to remain productive even when hundreds or thousands of users access the system simultaneously.

Essentials to test:

- number of users during specific conditions (daily / monthly)
- capacity during operation
- workflow integrations and custom applications
- response time
- speed while loading



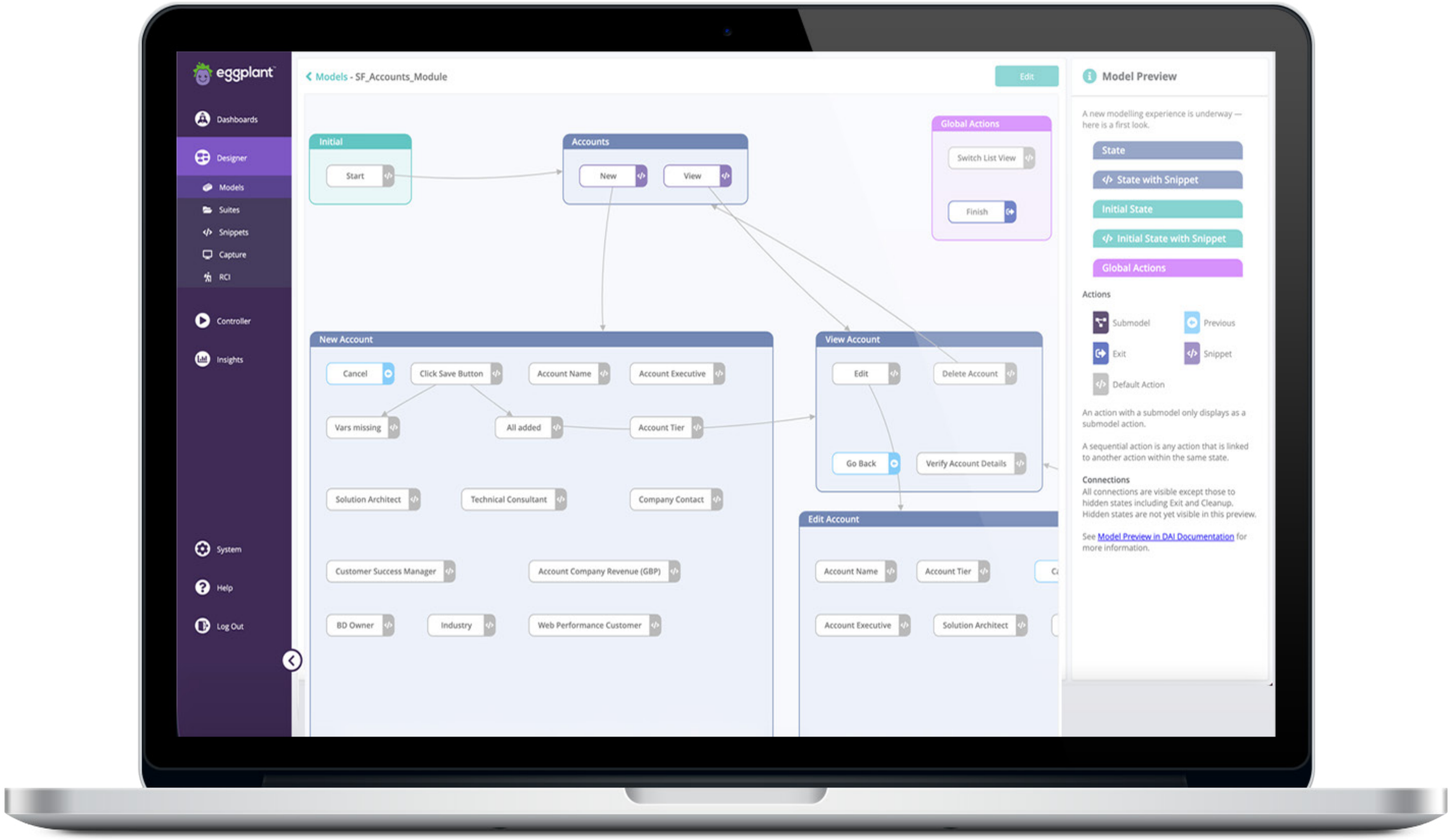
☑ End-to-end

As Salesforce systems become increasingly complex, end-to-end testing ensures that businesses maintain operational excellence to support their objectives. Custom workflows, application integrations, and front-end and back-end functionality are vital areas to test so you can identify any bottlenecks that may impact productivity.

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

- website
- customer service
- mobile applications
- social media
- email
- live chat and chatbots
- point of sale
- enterprise resource planning (ERP)

Before you begin end-to-end testing, it is essential to confirm that the test automation solution you use verifies that the codebase is correct. This process includes testing the UI to ensure it functions as expected, as shown in Figure 1.



A new modelling experience is underway — here is a first look.

An action with a submodel only displays as a submodel action.

A sequential action is any action that is linked to another action within the same state.

All connections are visible except those to hidden states including Exit and Cleanup. Hidden states are not yet visible in this preview.

See Model Preview in DAI Documentation for more information.

Figure 1. Example of a Salesforce test model built using Eggplant's digital automation intelligence (DAI)

☑ Functionality

It is critical to combine testing technologies to verify the codebase and validate the UI to ensure your Salesforce platform performs correctly.

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

Sales teams need critical customer information quickly — be it leads, online forms, or prospect follow-ups — linked together to have a complete view of all interactions. Marketing requires campaign statistics, email effectiveness, and social media exchanges all in one place. Customer service requires ticket prioritization to inform relevant teams of any customer-related issues or queries.

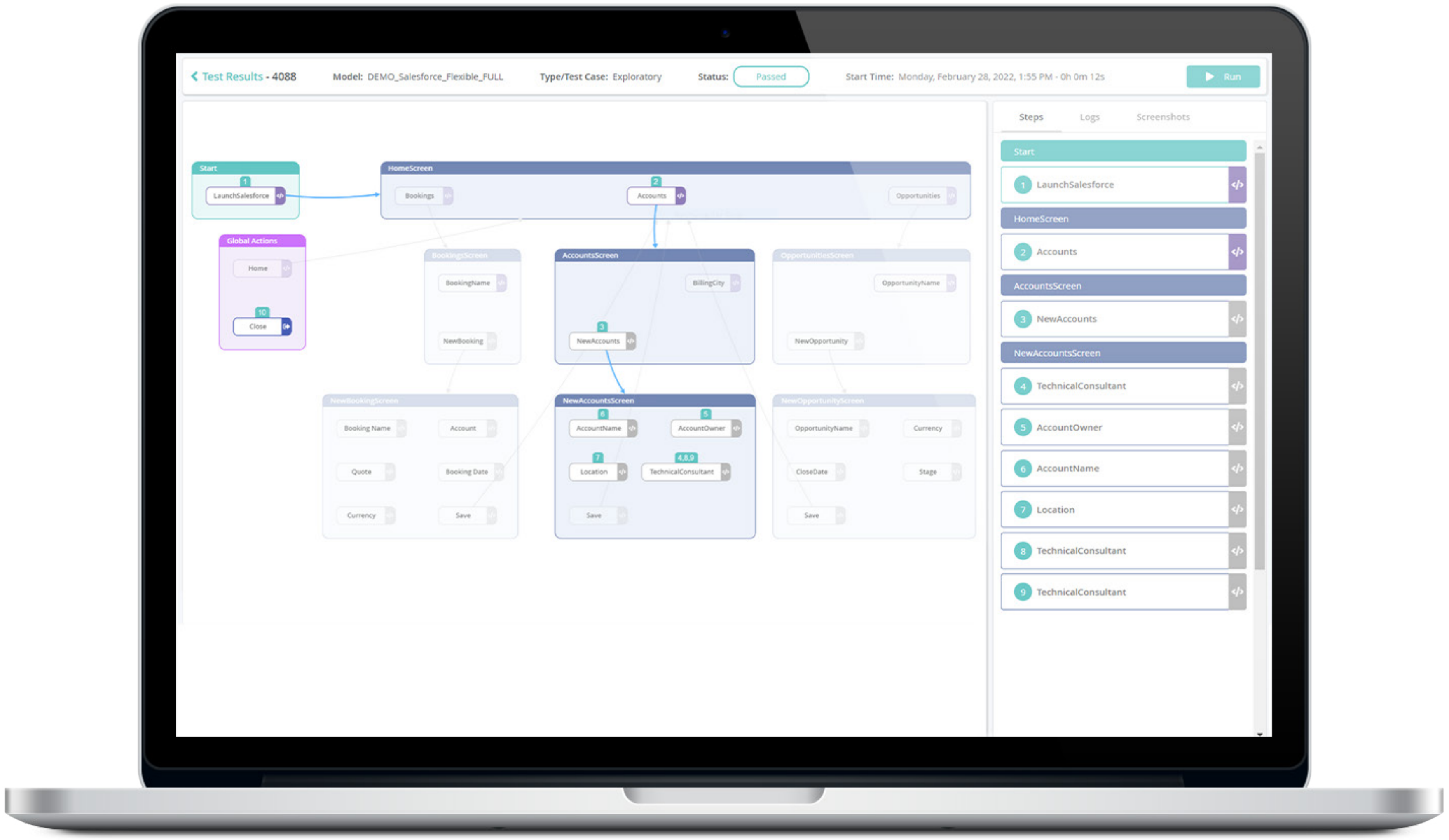
☑ Exploratory

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

Users will not necessarily move down that linear path in sequence because a custom workflow has five clear steps. If the end-user can access one step from a different route, but the step fails to conform to the business rule, a user may need to go back and refresh the page to complete an action — however, that extra step takes time and impacts productivity.

You can prevent this from happening by testing every possible user journey, custom workflow, and business rule.

Coverage significantly increases by combining exploratory testing with AI auto-generating test cases, as shown in Figure 2. This process covers 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.


Test Results - 4088 Model: DEMO_Salesforce_Flexible_FULL Type/Test Case: Exploratory Status: Passed Start Time: Monday, February 28, 2022, 1:55 PM - 0h 0m 12s

Start
1 LaunchSalesforce
HomeScreen
2 Accounts
AccountsScreen
3 NewAccounts
NewAccountsScreen
4 TechnicalConsultant
5 AccountOwner
6 AccountName
7 Location
8 TechnicalConsultant
9 TechnicalConsultant

Figure 2. Test coverage heatmap of full exploratory testing of a Salesforce model using Eggplant DA

CHAPTER 5

Eggplant Testing Strategy

Eggplant Testing Strategy

Eggplant Salesforce solution is an AI-assisted test automation solution that uniquely meets the complex challenges of Salesforce testing.

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. The following is how Eggplant software can help you with your Salesforce test needs.

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.

No code / low code approach

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

Test technology at any level

Use intelligent testing that interacts with any text or image on a screen to validate the UI. When validating the UI, testers can use the same test snippets across all versions of Salesforce. Eggplant software can also use objects for scenarios when extracting data from a table, plus it can make back-end API calls to verify that the updated data is accurate.

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 exactly monitors how Salesforce auto-generates future tests based on real user journeys.

Test from the user’s perspective

Automating processes just as a human would ensure that Salesforce functions as expected, as shown in Figure 3. Easily conduct and perform usability tests based on the current business requirements. Do not rely only on verifying the code to pass a test; validate the UI with Eggplant’s intelligent computer vision and be sure to return no false positives.

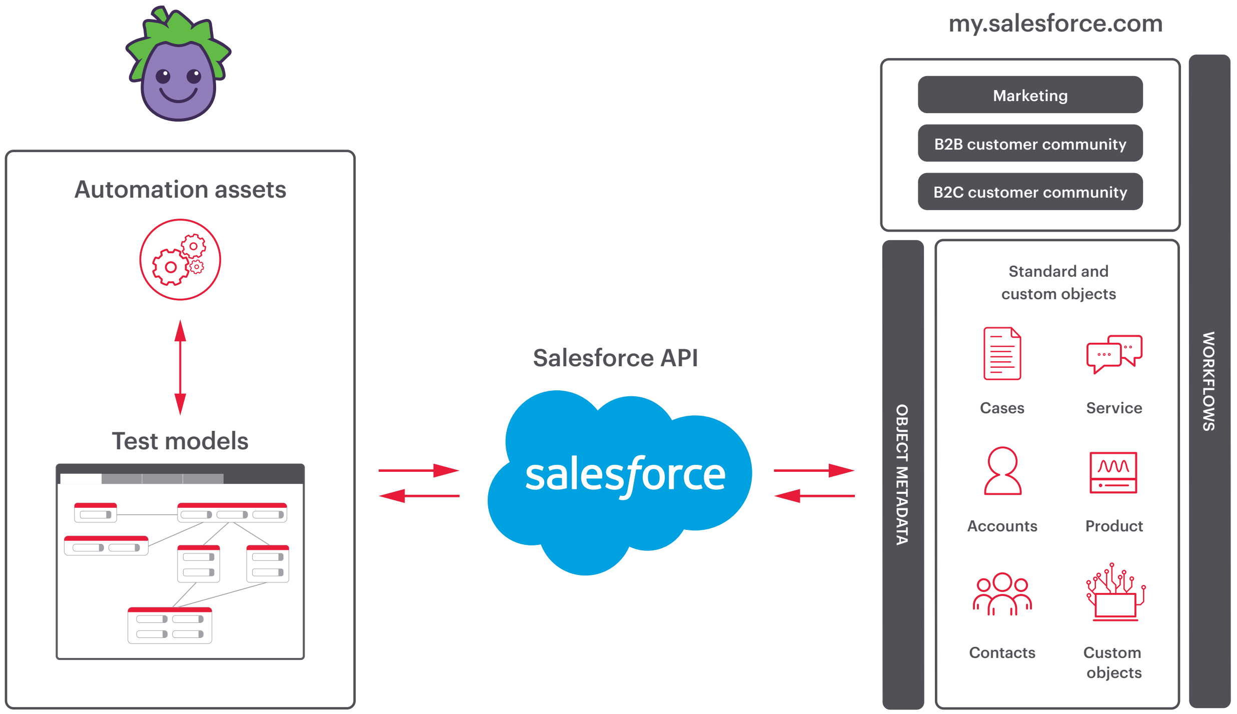


Figure 3. Architectural flow for Eggplant's Salesforce solution

SOLUTION BRIEF

Eggplant Salesforce Solution

Accelerate Deployments By Auto-Generating Test Assets

eggplant
Test Automation Software

Salesforce sits at the center of daily customer relationship management (CRM) operations for many global enterprises, but no two CRM platforms are the same. User workflows, business-critical rules, and third-party systems are a few of the customizations that make each Salesforce environment unique.

Customizations are vital to ensure Salesforce's user experience improves process efficiency and productivity. However, changes at the user interface (UI) level alter the codebase, making testing a challenge without the correct tools.

Testing is made even more complex by Salesforce's use of the Shadow Document Object Model (Shadow DOM). Salesforce encapsulates the DOM to protect its web components from being randomly manipulated by programming languages, such as HTML, CSS, and JavaScript.

Shielding the DOM makes it difficult for object or DOM testing tools to access the code because often scripts are written in JavaScript. These tools also rely on underlying object IDs in the code to precisely identify the focus of testing. Further problems arise because these IDs are dynamic and change after every major new release for Salesforce Lightning due to functionality enhancements and UI improvements. Consequently, pre-existing scripts built by object or DOM tools will break as they search for IDs that no longer exist.

Key Features

By directly integrating with Salesforce's API, the Eggplant Salesforce Solution accelerates testing and deployment by using:

- Auto-generated test assets
- UI validation combined with code verification
- Intelligent user journey mapping
- One model to test both Salesforce Classic and Lightning versions
- Business-critical workflow testing
- No code/low code

KEYSIGHT TECHNOLOGIES

Find us at www.keysight.com

Page 1

Download the solution brief

Learn More

Reduce test maintenance, future-proof your Salesforce platform, and maintain business continuity with Eggplant's Salesforce solution. To learn more, visit eggplantsoftware.com/salesforce-solution

KEYSIGHT