Are “robots” breaking your quality?

A WHITE PAPER

BROUGHT TO YOU BY

provar

THE LEADER IN SALESFORCE QUALITY

salesforce PARTNER

NEXST IT

TESTING EXCELLENCE COMPANY

Why are we writing this article?

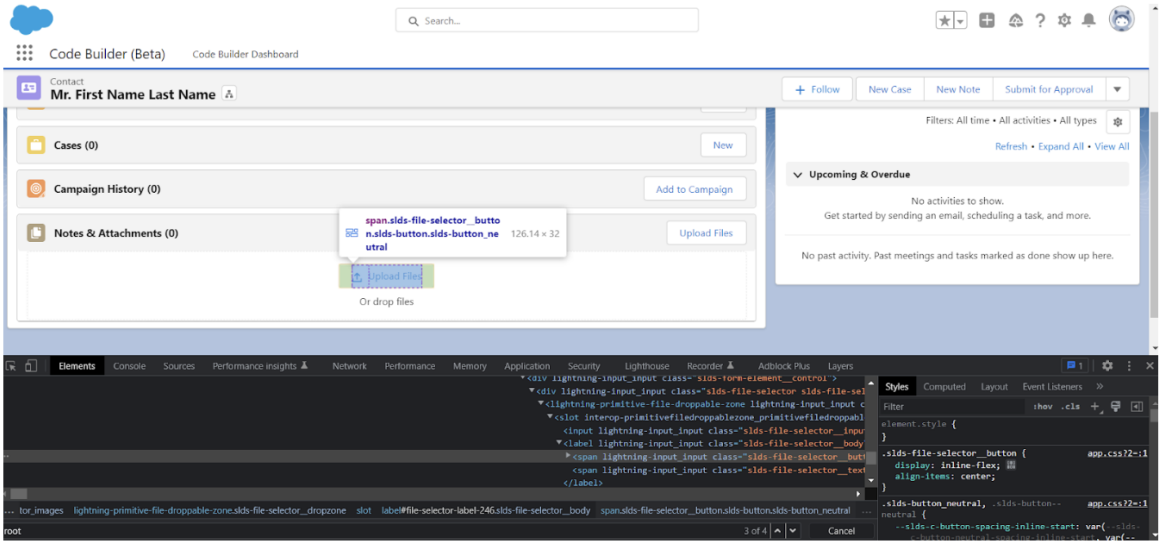
The purpose of this article is to highlight the specific pros of using Provar - such as “Metadata based automation” - versus the cons of using “robotic” frameworks for Salesforce test automation.

Narrative

Salesforce test automation is a critical aspect of maintaining the quality and functionality of Salesforce-based applications. With the help of test automation, businesses can ensure that their Salesforce platforms are working as intended and can catch any potential issues before they impact end users. A lot of times, agile teams depend on “robotic” frameworks to simulate testing for their implementations. While this setup works in the beginning, it can quickly become a bottleneck - not just for the QA team, but for the the whole delivery cycle because test automation for Salesforce is inherently complex. Some of the reasons could be:

- Constant updates and changes: Salesforce is constantly evolving and updating, with new features and functionality being added on a regular basis. This means that your test automation framework needs to be able to adapt to these changes and continue to provide accurate and reliable results.

- Salesforce has a multi layer front end system. It is a powerful and feature-rich platform, with many different components and functions at the UI layer. This system results in the complicated front end layers and poses a challenge for UI automation tools. Here’s a sample screen from Salesforce:



A lot of times, teams ask us if “robotic” solutions based on the Robot framework are good solutions for automating Salesforce test cases, and unlike most engineering questions, this one’s answer is pretty straight forward.

Robot Framework is a generic Python based free and Open Source framework for simulating user behavior on software applications. While it can be used to automate simple forms, things can get out of hand even
for simple Salesforce applications, such as creation of an Account :

```
LaunchApp               Sales

ClickText               Accounts
ClickUntil              Account Information         New

TypeText                Account Name                Salesforce                  anchor=Parent Account
TypeText                Phone                       +100000000000000            anchor=Fax
TypeText                Fax                         +100000000000000
TypeText                Website                     https://www.salesforce.com
Picklist                Type                        Source
Picklist                Industry                    Health

TypeText                Employees                   100
TypeText                Annual Revenue              10 million
ClickText               Save

ClickText               Details
VerifyText              Salesforce
VerifyText              17,000
```

As you can see there are two problems right away:

- The automated test exponentially gets complicated with the length of the user scenario. Imagine covering the whole scenario of creating an account, contact, opportunity, adding products, setting up the price and editing quote lines; it's easy to see that the number of lines of “ClickText”, “TypeText '' might cross triple digits.

- The test case relies on front end elements such as “anchors” which might be flaky by design and might break with the test runs.

But go deeper and we’ll find two hidden problems:

- The tester must understand the underlying technologies and dependent libraries (For example: libraries to interact with Excel, mobile testing etc) to build out resilient test cases. This could be a huge lift in terms of training/hiring for the team.

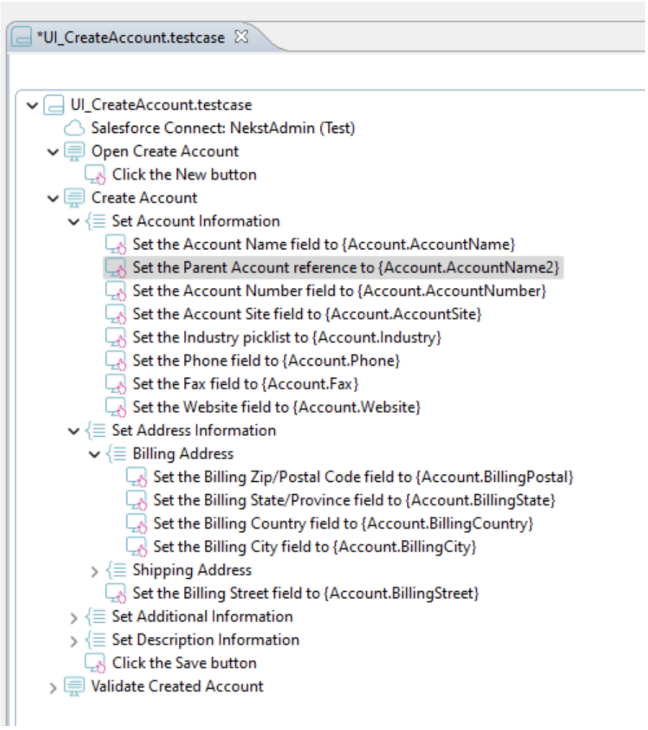
- Python based Test automation frameworks can get really complicated for non-technical testers. Here’s a sample for matching phone number :

```
${phone_num}=           GetFieldValue           Phone
Should Match Regexp     ${phone_num}            ^[+]\\d{1}$
```

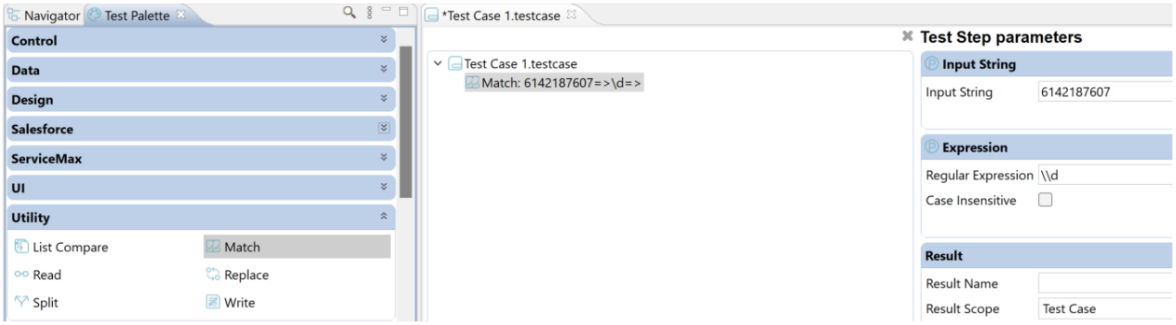
As we can see, Robot based tools are well suited for tests around web applications, but they are not a good fit for automation testing of applications within the Salesforce ecosystem (For example: Salesforce API calls, Sales Cloud, Service cloud etc.).

On the other hand, **Provar Automation** overcomes all of the four problems above (and many more) with its Metadata based low-code/no-code test automation framework.

From the below screenshot, Provar tests are evidently easier to create, organize and execute:



And here is an example, how string matching is managed by Provar, and empowers the user to match simple strings or provide regular expressions. :



Most of the time there is a change on a screen or on a web element, Provar dynamically generates the required locators at runtime by mapping test elements using metadata so that the same test can run across different user experiences

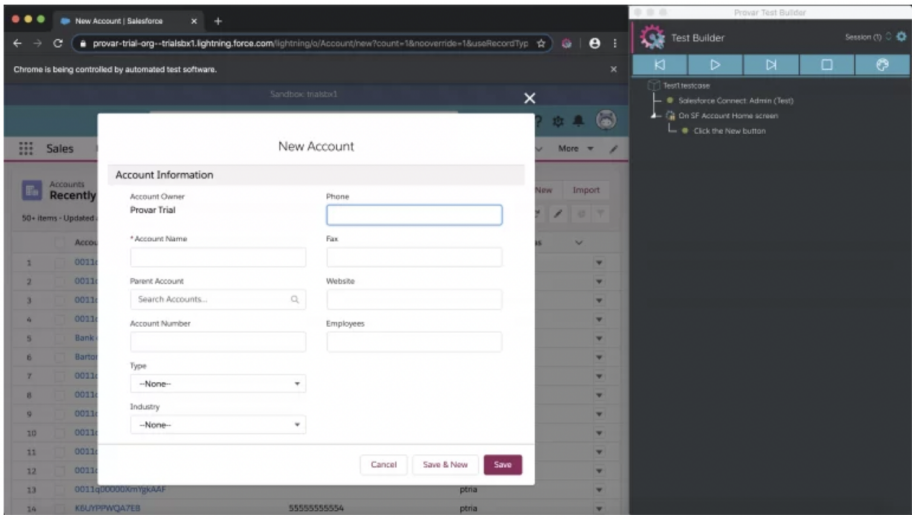
(Classic and Lightning), environments and even different Salesforce releases.

Additionally, Provar follows standard design patterns such as “Page Objects”, so that the cognitive load on teams around training is minimized. Provar tracks Salesforce UI changes closely throughout the year, in addition to Browser updates, so that the maintenance over-head for automation is minimized.

When implementing Salesforce test automation, it's important for businesses to have a clear understanding of their testing goals and objectives. This will help them to choose the right tools and strategies for their specific needs and to ensure that their test automation efforts are successful. **Provar Manager** helps agile teams organize, analyze and optimize their end to end testing journey, from planning to execution, in a single native Salesforce application. With the complete suite of tools spanning across Provar Automation, Provar Manager and built in CLI capabilities, Provar streamlines the quality management process for the whole team with powerful inbuilt integrations.

In addition to the above, Provar simplifies the test automation process exponentially with the below benefits as compared to open source solutions such as:

- **Setup of multiple environments with different users and seamlessly switch between them.**
- Cross browser compatibility with support for latest versions.
- Ease of data extraction or data creation in Salesforce.
- **Provar support system and community.**
- Ability to step back/continue/restart as part of the test authoring process.
- With Provar’s Test builder, the web element mapping is seamless and robust as below.



Overall, Salesforce test automation is an essential part of maintaining the quality and reliability of Salesforce-based applications. By using automated testing tools, businesses can ensure that their applications are working as intended and can catch potential issues before they impact end users.

Learn more by reaching out to Provar experts via this **link** or contact **product@provartesting.com**