functionize

White Paper

Intelligent Functional Testing

The field of software functional testing is undergoing a major transformation. What used to be an onerous manual process took a big step forward with the advent of Selenium and other testing tools. But those tools remain heavily developer-centric and are primarily tethered to rigid script-based approaches that do not scale. And now, with the rise of cloud, big data and machine learning, companies need to re-think their testing strategies and technologies yet again.

The core concept behind this new approach is that, in this era of agile development and rapid customer response, testing can no longer be an adjunct to the development process: it needs to be an integral and strategic component of the workflow. Functionize is doing just that with the introduction of our ground-breaking scalable autonomous test platform. The driving force behind this platform is Adaptive Event Analysis™, a patented and proprietary hybrid of algorithms which span supervised, unsupervised, reinforcement learning and computer vision, enabling lightning-fast test creation and execution, self-healing maintenance, and actionable analytics.

Functionize goes well beyond conventional test recording tools to perform exhaustive and intuitive testing of highly complex applications across multiple environments including both desktop and mobile platforms. The integral usage of computer vision, machine learning, and big data processes bring previously unrealized levels of support to your test suites. Many tests will automatically accommodate trivial changes, so that you can maintain focus on fixing critical issues.

functionize

Table of Contents

What is Functional Testing?

The purpose of functional testing is to map documented specifications to the design and construction of software, and to confirm that the version of an application being tested operates precisely according to specification. In a typical functional test, as part of the specified workflow, the elements on the web page are interacted with and tested to ensure that they model the expected user experience correctly. Furthermore, the discovery and prompt reporting of issues to enable required changes is imperative for accurate functional testing. In this paper we will illustrate how Functionize seamlessly empowers and enables comprehensive functional testing using its unique patented technologies.

As part of a web application being considered for functional testing, there can be any number of elements and workflows that must be examined and evaluated to ensure an acceptable test suite. The web app as a whole and each operational sub-unit/ module must be subjected rigorously to the following test procedures.

- Identify all functions and workflows that should together form the test suite.
- Design a range of input data to match the specifications of each functionality.
- List the expected output for each input accordingly.
- Execute a comprehensive set of tests on all input and output combinations.
- Compare the observed results with expected output.
- Report the variances and initiate corrections to match test specifications.

An ideal test suite should exhaustively cover the critical operations of the web application, and should ensure an acceptable level of coverage across all aspects. This includes the behaviour and interactivity of the user interface elements including menus, form elements as well as dynamic data populated from backend databases. Special attention should be paid to asynchronous file loading and effect of response delays on the intended user experience. The color scheme standard of the site must be verified for consistency, which includes link colors and CSS consistency with existing pages and components. Single Page Applications (SPA) should also pay attention to dynamic data updates using AJAX calls along with the peculiar behavioural characteristics of frameworks like AngularJs, Vuejs, or ReactJS. Additional considerations of a good test suite include performance metrics around the page loads, AJAX calls and element interactions - this can include tabs, scrolls, buttons, checkboxes, radio elements, etc. All these must be examined under varying permutations to discover potential issues and unexpected outcomes. Indeed, the scope of modern functional testing is beyond the capability of human testers, and that is precisely why automation tools are mandated.

Moreover, the evolution of efficient development workflows in Agile development with its stated goal of continuous integration, testing, and deployment makes automation imperative for QA. An important motivation of this white paper is to further demonstrate that common automation tools which require scripting, such

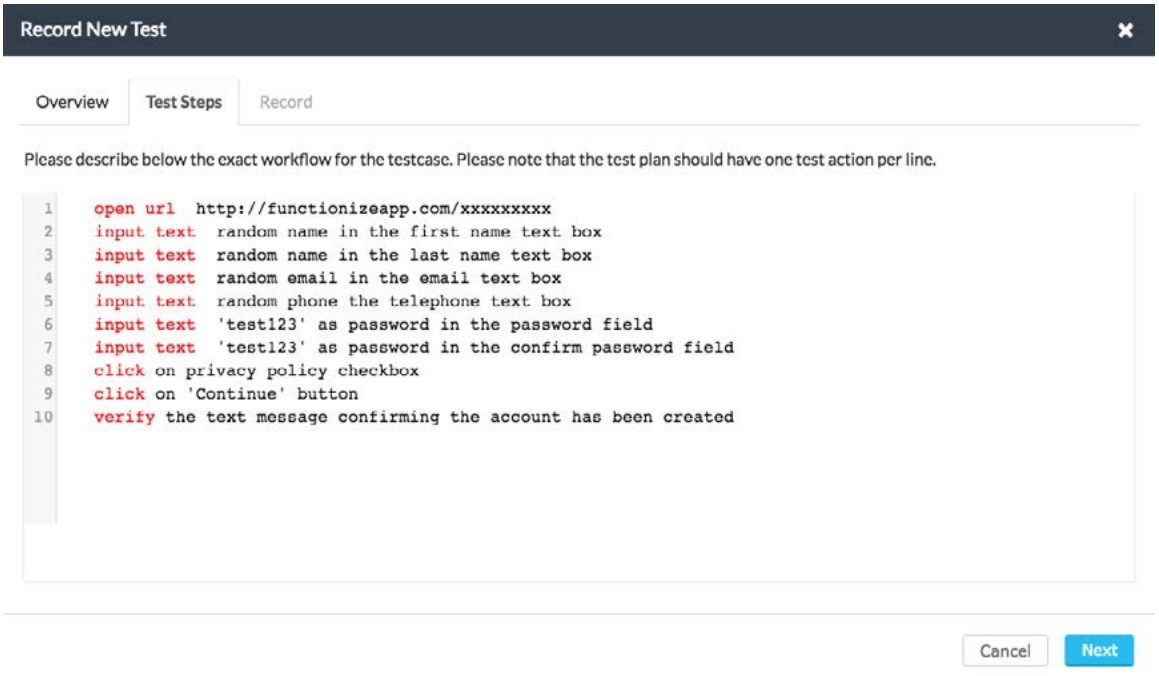
as Selenium and JMeter, are nearing obsolescence due to the limits of their scope of application and maintainability. These tools require an inordinate amount of effort and resources to be applied towards scripting of new tests and maintenance of the existing ones - which is not practical in this age of ever increasing design and functional complexity of web apps.

Functionize Case Study - Functional Web App Testing

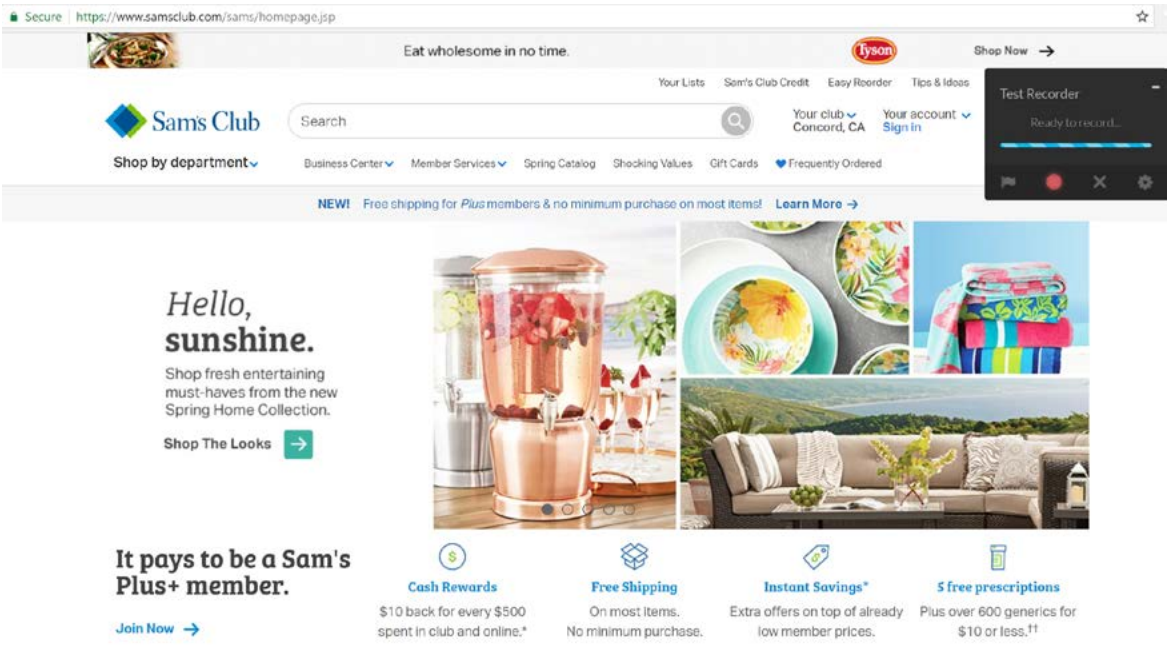
The below case study illustrates how Functionize can be used to seamlessly manage all aspects of functional testing. We show how to effortlessly create a sample test workflow consisting of data collection, element/data verifications and assertions. We then move onto exploring the Functionize tools that allow for quick discovery of potential failures and easy maintenance of the test suite.

There are two ways to create a test case within Functionize - using either a plain English test plan, or alternatively, using the Functionize Smart Recorder.

Creating a test plan with plain English involves writing out the sequence of steps you want as part of the test details, and submitting it to our natural language processor (NLP) and Artificial Intelligence engine for analysis and data modelling. The NLP engine uses automated test creation techniques and human created sample tests to automatically identify form inputs, dynamic data assertions and page interactions - and returns a perfectly working test case following the workflow specified.



As part of this case study, we will be using the Functionize application to test and confirm Sam's Club functionality for adding a new TV set to the shopping cart, and ensuring that the sample workflow functions correctly. With the Smart Recorder, a user creates test cases as if they are natively browsing on a website using a proprietary Chrome extension. Point Functionize at a target url and the Smart Recorder automatically captures all actions performed on the webpage during the course of the browsing session and converts them into a reusable test case capable of being executed across browsers and platforms, including mobile.



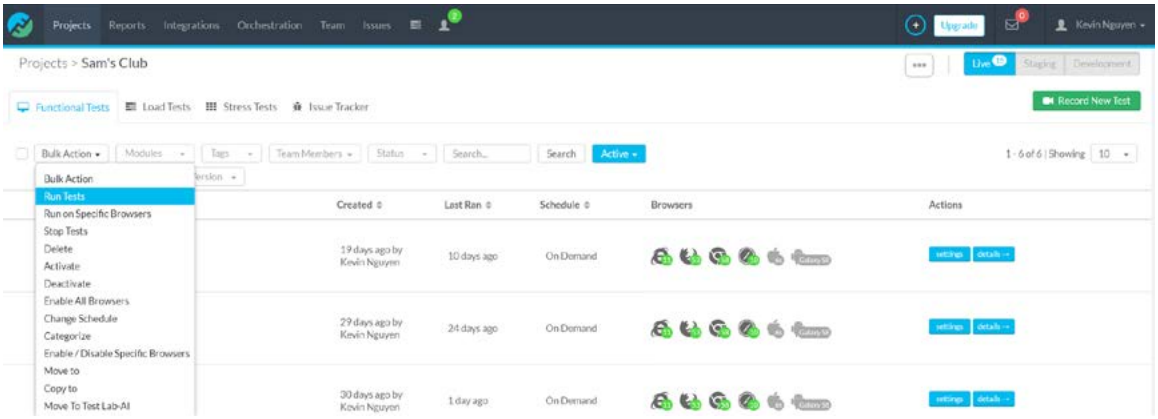
Once created, the test case is available in the Functionize application, presenting the user with options to execute, maintain and manage it as part of a functional test suite. The platform itself is designed to present test cases and their execution reports in a graphical and intuitive format, aiding in fast debugging and maintenance. The test case review interface allows one to compare the execution status of each action across multiple browsers in a single glance. Review example test case here.

Example of Test Creation Workflow

The following case study example depicts how quickly a test is created within Functionize. The end user begins by opening the Smart Recorder and dropping assertions on the main navigation items on a target website. In this example, the end user drops assertions on the main navigation items on Sam's Club. When the main navigation items change, Functionize will automatically notify the user of the change and include visual screenshots of before and after the change took place. In this example, the user continues browsing by searching for TV's in the general search bar. Functionize is able to account for dynamic search results by using the inbuilt advanced verifications. The next step involves adding the TV to the cart, and lastly, by verifying the selected TV is in the final check out screen.

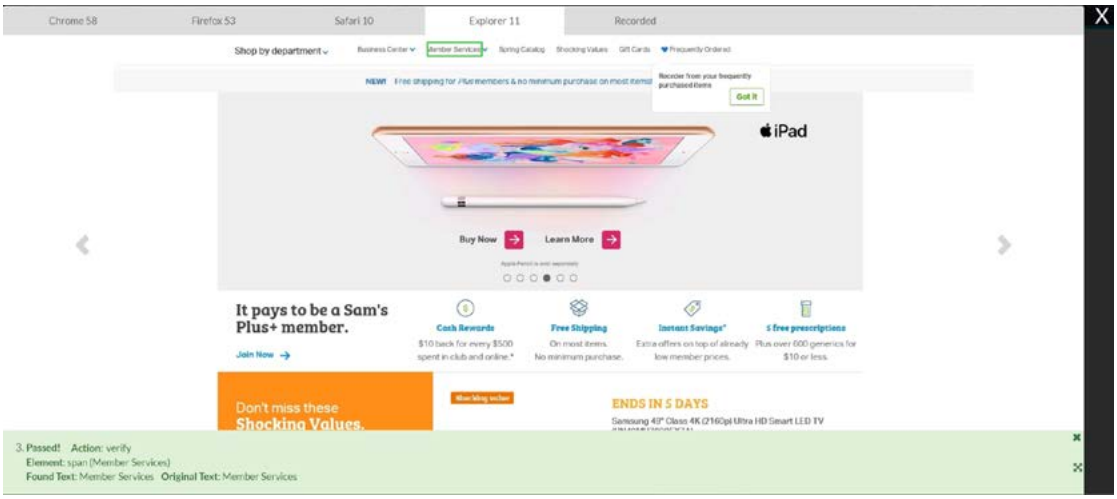
Test Execution

Once a test suite is created, the end user can navigate to the main user interface and execute an individual or bulk test suite with the click of a button. This includes selecting any number of browsers and mobile devices as well. These actions can also be remotely executed using the Functionize public API and command line tools, as part of an agile development and release process.



Reviewing a Test Case

Functionize provides multiple tools to enable an exhaustive review of test case execution and any potential issues that might be uncovered. These include screenshot comparisons across multiple browsers at the same time, videos of test case executions, and computer vision comparisons of visual elements. Functionize also automatically tracks page and resource load data across all pages in the workflow, enabling the user to uncover any potential performance issues.

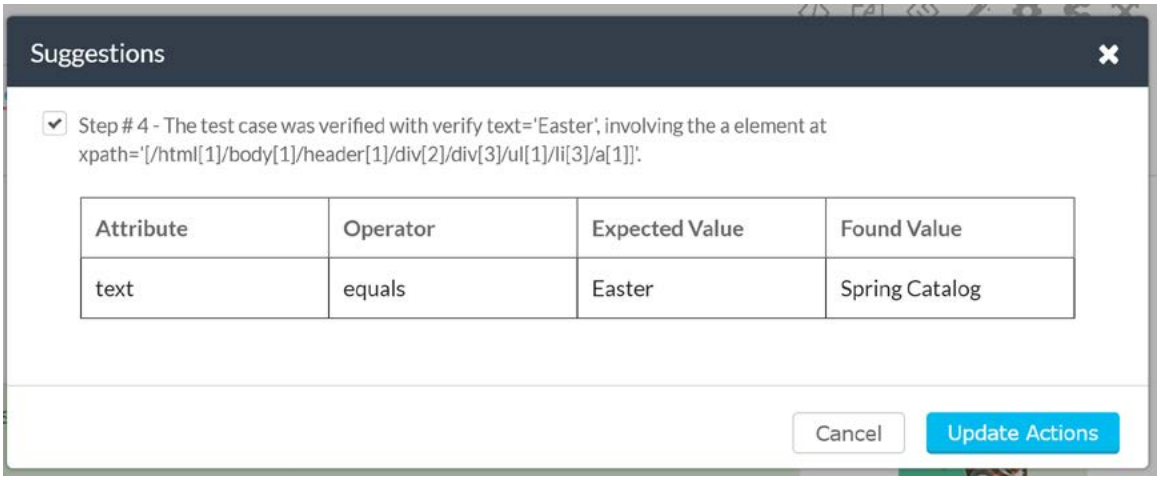


Maintaining a Test Case

Functionize provides several options to easily maintain and update test cases in a visual environment with features like intelligent suggestions, reusable page objects, and a live edit environment. These allow testers to quickly target the bottlenecks in successful executions of the designed workflow and correct only the sections that are necessary.

functionize

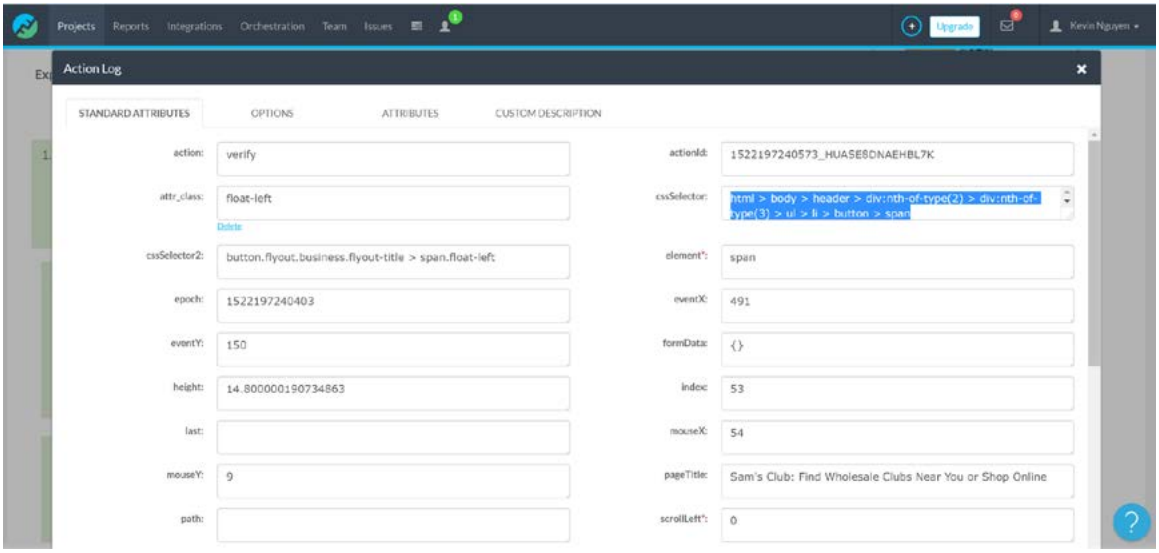
Learn more at:
www.functionize.com



Screenshot 1. One Click Updates

Functionize automatically tracks hundreds of selectors per test step during test creation. This enables Functionize to detect element relocations and structural changes, vastly reducing manual effort to maintain tests. In addition to the standard Xpath and CSS selectors, Functionize also captures proprietary selectors for the targeted element, as well as the elements in the vicinity, size and location of the element, as well as *previously known* sizes and locations of the target - this analytical data gathering capability is at the heart of the *machine learning core of Functionize*: the patented ARIMA based technology in Functionize which we call **Adaptive Event Analysis.**

As a practical example of this patented technology, if a structural change alters the original element such that the original xpath is invalidated, Functionize can still locate the element using previously known attributes and smart selectors described above. This intelligent feature significantly reduces the number of new test cases which must be created in each DevOps cycle, as well as eliminating the need for scripting test cases. Overall maintenance of test cases is considerably reduced using the smart tools built into Functionize.



Screenshot 2. Smart Selectors

functionize

Learn more at:
www.functionize.com

Additional Features and Benefits of Functionize

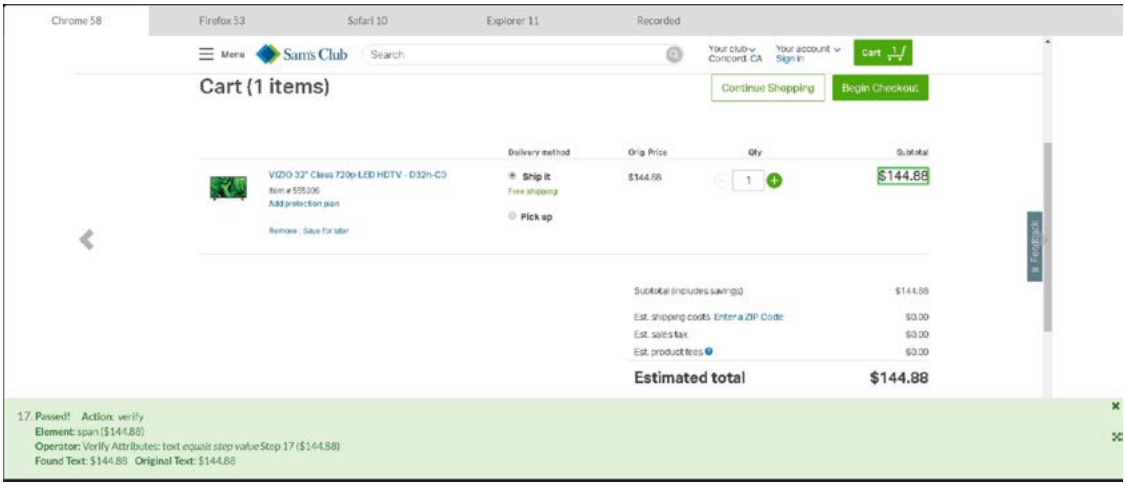
Additional features include reusable page objects (sequence of steps that can be shared between test cases), conditional branching, A/B testing, sophisticated looping actions and manually scripted actions. Functionize also supports a Robot compatible framework if scripting is preferred by the user.

By incorporating more intelligence into test creation, we substantially reduce the risk of errors and unexpected app behavior because the probability of the Functionize app missing an error is much less than that of a human tester. Functionize never forgets a test case, and every execution of a test improves its stability.

Functionize supports unbelievable flexibility in test case development. Opening an existing test case we can scan through the screenshots captured by the Functionize Test Cloud since every step of the test is automatically captured by computer vision. Additionally, every element on a page is automatically recognized visually while the user browsers the site to create a test.

The Real Magic Inside Functionize

In the test case scenario on Sam's site, a search for TVs will result in varying numbers, models and descriptions of TV inventory items each day. Dynamic validation of these elements is also made possible by Adaptive Event Analysis. Screenshot 4 shows verification of the correct price of the TV is in the shopping cart. Testers glide through screenshot verifications; when a test case needs updating, Functionize will autonomously self-heal or suggest a fix with a 1-click update for the broken test step.



Screenshot 3. Dynamic Verification

functionize

Learn more at:
www.functionize.com

The Outcome of Functional Testing

Functionize combines all phases of testing into a smooth, seamless testing experience. Our autonomous testing platform, an industry first, enables a single human tester to model and observe the experience of hundreds and even thousands of tests from one convenient vista: the Functionize test panel. In fact, the three primary forms of testing: functional, performance, and load testing, are so smoothly integrated in the Functionize experience that companies can model their entire user experience in one application.

Optimizing the user experience is a critical component to any company's success. The Functionize autonomous test platform captures and reports all elements of this experience, enabling companies to deliver a superior experience, reducing customer churn and elevating customer satisfaction—both of which translate directly to the bottom line.

References

https://www.semanticscholar.org/topic/Functional-testing/168200

https://link.springer.com/chapter/10.1007/978-3-540-79124-9_7

156 2nd Street | San Francisco, CA 94105
+1-800-826-5051 | www.functionize.com