ULTIMATE HANDBOOK TO START AUTOMATION TESTING

OSP

TABLE OF CONTENTS

1 INTRODUCTION

Thousands of software applications are created and released every year. Each of these apps goes through several tests to ensure it works as expected and meets the required quality standards before being launched in the market. An important technique used during the software testing process is automation testing.

Automation testing is a technique in software testing that improves the execution speed of verification or any other repetitive high-volume tasks and workflows. Automation testing interacts with applications in a human-like manner and uses assertions in programming to validate test steps.

Manual regression testing is a lengthy and tedious process as the manual tester evaluates each step of the test case procedure. Hence, businesses are now replacing their manual test cases with automated test cases as it executes the test steps automatically, saving precious time.

2 BENEFITS OF AUTOMATED TESTING

Automation testing is becoming increasingly important as technological advances are leading to the development of software programs at a rapid rate. Fast and reliable tests are critical in software development, especially for practices like continuous integration and delivery, and manual verification does not fulfill this requirement.

No doubt automation helps a company save time and money, but its benefits extend way beyond that, some of which are:

- Validates newer versions of software applications
- Allows testers to focus on exploratory-type testing
- Improves benchmarking accuracy
- Flexible programming options
- Reduces false failures due to human errors
- Faster release of software applications
- Provides timely feedback to developers on failing checked-in software
- Scope for reusability that allows you to use it for new builds of your application

3 CREATING A TEST AUTOMATION FRAMEWORK

An automation framework is a combination of test tools, guidelines, and practices created to improve efficiency and effectiveness of automation testing by minimizing test script maintenance.

Some points to bear in mind to build a robust test automation framework are:

- Set your manager's and team's expectations
- Reiterate that automation is a collaborative effort by the complete team
- Break your automation framework into abstraction layers
- Add automation in your definition of done (DOD)
- Use proper synchronization methods
- Use a version control system
- Use a naming convention for standardization and easy understanding of source code
- Determine a strategy for training and retraining your framework users
- Set your tests apart from your framework

- Incorporate parallel testing to shorten execution time
- Use reusable and low-cost maintainable test automation modes
- Look for existing libraries/tools before inventing your own
- Do code reviews on all automated tests
- Ensure reports and logs are maintained to help the debugging process
- Create unique IDs for all elements for easy identification in testing
- Practice code refactoring regularly to identify non-functional attributes
- Include easy to maintain design patterns such as page objects
- Do not duplicate code
- Create an effective test data management strategy
- Create reusable methods and utilities
- Allow for mocking and stubbing in testing your programs

4 OPEN SOURCE TOOLS FOR AUTOMATION

Now that we know how to build a test automation framework and choose the right test stack, the next step is to identify the best testing tool for your process.

Here, we take a look at some of the readily available open-source free automation tools and how they differ from one another.

Selenium

This is one of the widely used testing tools for browser automation. However, selenium scripts don't work for non-browser applications.

Appium

This open-source test automation tool works best for native, hybrid, and mobile web apps.

Watir

This open-source Ruby library follows human-like interactions with a browser by clicking links, filling out forms, and checking the text.

Sikuli

This application follows image-based automation testing.

WinAppDriver

This service supports UI test automation of Windows applications.

White Framework

This .NET based framework automates rich client applications based on Win32, WinForms, WPF, Silverlight, and SWT (Java) platforms, without the use of any proprietary scripting languages.

AutoIt

This freeware BASIC-like scripting language helps to automate the Windows GUI and general scripting.

Serenity

This automation framework provides a lot of built-in functionality that integrates with Selenium and BDD tools like jBehave and Cucumber JVM.

Gauge

A cross-platform test automation solution that supports multiple languages, including Ruby, Java, C#, Python, and Javascript.

Sahi

This automation and testing tool also has a proprietary version that includes additional features such as test distribution and report customization.

Robot Framework

This keyword-driven framework for acceptance testing and acceptance test-driven development that allows better readability and easy creation of tests.

RedwoodHQ

This open-source tool creates a website interface that allows multiple testers to work together and perform their tests from one web-accessible location.

Galen

This framework used for the design (UX)/layout and functional testing tests the look and feel of responsive websites that help improve user experiences.

JavaScript Test Automation

Jest, Mocha, Jasmine are some of the open-source JavaScript testing frameworks. These work best for web application testing, unit testing, as well as E2E(End-to-End) testing for applications developed using different environments.

Visual Validation Tools

Gemini, Needle, Phantom CSS are some of the open-source visual validation tools that help to detect visual bugs in the UI. It also checks that each UI element appears in the right color, shape, position, and size.

5 GUIDELINES FOR AUTOMATING TESTS

There are several benefits of automation testing, however, you need to consider the time, effort, and resource utilization, and then identify which tests should be automated and which tests require manual testing.

The following table lists some points that can help you make the right decision to get the best value out of test automation.

AUTOMATION TESTING IS IMPORTANT FOR	AUTOMATION TESTING IS NOT NECESSARY FOR
Business-critical paths - the elements or user flows that are essential for the business's success.	Intermittent tests with unpredictable and unreliable results as they do not generate pass and fail conditions.
Long-running tests generally scheduled to run during breaks or overnight	Tests that record user experiences, especially on the ease of use of the application.
Tests with the same workflow, which use different input data for each test run.	Urgent tests that require quick feedback, e.g., development of a new feature
Tests run against multiple configurations such as different OS & Browsers.	Exploratory testing - random testing based on domain expertise.
Long tests that need a lot of data as inputs such as long forms.	Tests run only once, except tests that work with a lot of data.
Tests used to measure performance, such as stress and load tests.	Tests that need visual validation, as manual testing is required to test the page images captured through automated testing.
Tests that cannot be completely automated, except if it will save a lot of time.	Tests that cannot be completely automated, except if it will save a lot of time.
Tests that capture snapshots to demonstrate the application's behavior or to ensure that different web pages maintain cross-browser compatibility.	

6 CHOOSING YOUR AUTOMATION TEST STACK

While there are many automation test stacks available, here are our recommendations for different tests - unit tests, integration tests, and end-to-end tests.

Best test stacks for real-time testing locally

01

FOR UNIT TESTING

Wallaby.js is what we recommend as this intelligent test runner monitors every keypress and returns a pass or fail condition in real-time. It provides continuous, parallel testing and immediate feedback, even in an unsaved file. Wallaby also displays various types of results directly, including code coverage, error messages, and console messages.

02

FOR INTEGRATION TESTING

We think Chimp.js is suitable for integration testing as it authenticates your business domain logic with every file save. In the domain testing mode, this tool checks the file system for a change and reruns the service tests. Real-time feedback allows testers to make changes to the domain logic faster.

03

FOR END-TO-END TESTING

Chimp.js also works well for end-to-end testing as it runs a test on every file save to verify the end-to-end feature you're presently working on. This automation test stack helps you concentrate on the core task and saves a lot of time by rerunning tests only on specific tags instead of all the end-to-end tests. Other great features include immediate end-to-end feedback and usability on any web application regardless of the backend.

04

FOR ORCHESTRATION

We recommend Gulp.js as this streaming build system seamlessly combines all tasks to work together and in a precise order. It works with non-node.js technologies as well and allows you to run complex tasks locally and also refreshes the page automatically after editing.

7 TEST AUTOMATION CHECKLIST

Automation testing helps to shorten the testing time, improve test coverage, and release products faster.

Creating a test automation solution is easy, but you should always look at building an automation project that is portable, extensible, and easily maintainable. To ensure the success of your automation project, you should define clear objectives beforehand and state all considerations of long term maintenance in the scripting strategy.

This checklist shows how proper groundwork, approach, and implementation can help you build effective and efficient test automation solutions.

1 Initial Meeting

All the people involved in the project (automation experts, development engineers, and other stakeholders) come together to consider the objectives, requirements, and strategies for test automation.

2 Gather Requirements

The automation team collects critical requirements and analyses and executes the best test automation solution. Some important points to be considered in this step are:

- Technology employed for app development
- Product & domain expertise (including functional specifications and product roadmap)
- Product structural design (including any specific back end hooks like APIs, Web Services or DB connectivity hooks)
- Automation tool preferences
- Challenges encountered in earlier automation attempts

3 Record Use Cases

The automation team creates a list of test cases and sets priorities to deal with the most critical transactions first and make timely decisions about the framework plan and tool selection.

4 Test approaches & procedures

The automation team makes a note of all the details, such as the structure of tests, test language, the test interface, and the input/output test data.

5 Review automation tools

In this step, the developers carefully review all the paid and open-source automation tools to choose the most suitable application.

6 Sign-off

The automation team examines the use cases and provides an effort estimation for sign-off so that all team members are aware of scope and plan.

7 Build Elementary Framework

The automation team creates a basic framework necessary for automating use cases.

8 Write test scripts

The use cases are automated in line with the earlier discussed objectives to determine the right elements. The automation team also creates the data, writes the scripts, and validates the cases.

9 Execute script batches

The script is executed in batches while creating the test scripts to confirm the app interacts as expected. It also helps to identify synchronization issues and avoid sudden test script failures.

10 Demo & Reporting

The automation team presents the final framework, automated tests, and reports and trains other stakeholders for future set-ups and implementations.

Regular communication within the internal and external teams and a structured process helps create useful and dynamic test automation solutions.

8 CONCLUSION

Quality at Speed is the buzzword in software development as organizations are competing to deliver software applications faster without compromising quality. Automation testing, if utilized to its full potential, can help organizations in achieving these goals.

OSP has built customized seamless and well-integrated automation test solutions using Artificial Intelligence that has helped clients release bug-free products. These systems include a host of advanced features beneficial for test optimization, test generation, execution, and reporting.

We are a leading software development company aiming to empower, and inspire the world with next-gen solutions. We help in simplifing every step of the development process, from system architecture design to quality delivery. Our intelligent processes enable quick deployment of enterprise-grade solutions against the toughest, and most complex challenges.

We are re-imagining how technology can empower Healthcare, AI, Analytics, and Financial organizations to build solutions for every day use in business applications.

With 10+ years of experience, and 200+ customers worldwide, we're leveraging technology to build the future today.

---

www.osplabs.com

Discuss Your Project ▸

solutions@osplabs.com

Texas | California | Maryland | Mumbai