Automation Testing Framework for a Leading Retail Software Solution Provider

The Customer

The client offers retail inventory solutions through intuitive methodologies integrating retail sciences and predictive analytics. Their products are designed to streamline supply chain complexities and help retailers in understanding their customers better.

The Challenges

The client offers intuitive and intelligent software solutions to retailers that help them in solving their inventory complexities and enables better understanding of their customers. The nature of their industry demands a wide window of environments to test their products.

The client's existing automation framework wasn't able to customize, support and run their test codes in different test environments without having to rewrite the hard-coded lines in their program every single time. This process was not just inefficient but was also dangerous to do on an everyday basis.

Business Challenges

The existing automation framework was difficult to be implemented as a solution for such a complex product and the lack of proper documentation of the use cases added up to the difficulty in establishing an organized business communication to perform testing effectively.

Since the nature of the client's product was very diverse, the business logic that is to be applied in the test cases was complex and they varied according to the environment. The testing process in place previously caused process overheads and threats as they demanded constant changes in the source code.

Technical Challenges

→ The existing/new SHELL scripts had to be integrated with selenium test scripts to support running in multiple environments.

→ In order to accommodate the growing demands of their product, the existing framework needed re-factoring.

The Solution

Aspire modified their automation testing framework with improved reliability and quicker turnaround time by creating automation scripts for 400 new test cases that can be run against multiple test environments. The processes adopted at the client's end was revamped by inducing orderliness in the way testing is conducted, defects are tracked and documented.

→ Apart from developing 400 test cases to suit the products' needs, Aspire also developed and delivered the execution status report for weekly and monthly reporting that improved the overall process efficiency.

→ With the extensive testing modules, Aspire was able to identify critical bugs that occurred during development. These defects helped the clients in refining their original product.

Highlights

Aspire thoroughly documented their framework-customization procedures for a proper understanding of the process in the long run.

→ Jenkins jobs were setup for running shell scripts and other automation tests.

→ DB verification steps were added in the automation procedure to validate the business logic.

→ Data tables were used efficiently to support testing in multiple environments.

Technology Snapshot

→ BDD with cucumber

→ Tools: Selenium with Cucumber

→ Platform : Linux, Windows

→ Languages: Java & Perl Scripting

The Result

With Aspire's implementation of 400 test scripts, the customer can now run all of them against multiple environments for every patch release without having to touch the source code. Their efforts in re-factoring existing framework and re-organizing defect detection, tracking and documentation has induced tremendous efficiency in the client's overall testing operations.

→ By automating 400 test cases that can run in two parallel environments, the total execution time has been reduced by nearly 60%.

→ Reduced the process time by setting up Jenkins jobs, updating test data & source code and adding use cases in feature files.

Situation Comparison

→ The customization of the existing framework, removal of hardcoded values and introduction of YML test data has enabled running test scripts against multiple environments without creating duplicate tests for each environment, a practice that was adopted before this solution was implemented.

→ Specific scenario tags were created as opposed to individually correcting the hard-coded values in the test data every single time.

Future Impact

This model can be scaled for addition of new environments by just creating YML files and adding data to it. The existing test case will take up this YML data to run in the new setup.