Efficiency in Action with Automation: A Mobile Trade-In App Case Study

About

Trade-in and Upgrade solutions for the mobile device ecosystem of mobile operators, retailers, OEMs, insurers, and online brands. These solutions support lifecycle management for pre-owned mobile and connected devices including new revenue, market insights, environmental sustainability, customer churn reduction, reduce the cost insurance and warranty programs, and promotions that help consumers offset the cost of new handsets. By extending the life of pre-owned devices, it provides consumers in developed and emerging markets with access to affordable, high-quality wireless technology; builds economic opportunity and enables information access to new users; and protects our planet from further e-waste.

Background

The client's previous experience with another organization left them unsatisfied. The automation performed by the previous organization was deemed incorrect, leading to the emergence of Priority 1 (P1) bugs in the production environment. Additionally, the test coverage was inadequate.
In response to the client's dissatisfaction, the decision was made to abandon the old automation framework. Then we introduced our proprietary framework **RaptorVista**, which aimed to provide a more reliable and comprehensive solution.

This framework was designed to support the execution of scripts in various environments with a single script, streamlining the testing process.

Our framework offered a significant improvement over the previous automation solution. Notably, the framework incorporated its own written wrappers on top of existing libraries available in the market, such as

- File util,
- Email util,
- Csv util,
- Properties util and more.

Impressed with our framework, the journey towards automation began in April 2019 with just a few programs. Over time, the automation initiative has expanded and matured, currently handling automation for more than 15 different vendors, reflecting substantial growth and progress.

Laying the Foundation

The initial phase involved understanding the product thoroughly and evolving our framework. To achieve this, a multifaceted approach was adopted, including:

- Q&A sessions and training provided by the previous organization,
- live demonstrations,
- one-on-one consultations with business users,
- examination of business requirements,
- user stories,
- existing test cases,
- QA processes, and
- access to the existing environment, databases, and third-party systems.

Furthermore, deployment guidelines, system configuration, installation instructions, troubleshooting, changelogs, and bug tracker data were scrutinized. This approach was implemented successfully with initial clients, and it led to the automation of nearly all Priority 1 (P1) test cases.

Expanding the Framework

During the expansion of the framework, not only were existing libraries utilized, but custom wrappers were created to meet specific business needs. These included utilities for:

- Generating IMEIs util
- State code util

Apart from above mentioned utils some generic utils were also added to **RaptortVista**, those were added in addition to some pre-existing utils that includes:

- PDF reader
- Image verification

Gradually, the coverage of P1 test cases expanded to include more vendors, further proving the effectiveness of our automation scripts.

API Automation

Earlier the client's system was monolithic but later architectural changes were made towards microservices. So, to automate these microservices a separate project was introduced.

This helps the team to catch the issues on first stage I.e., at API level.

Enhancing Efficiency

- To reduce manual effort even further, **TestRail APIs** were integrated into the framework. This allowed the creation of test runs and the automated marking of test cases as pass or fail along with logs to get more insight in case of failure.
- Additionally, support for backing up test suites in TestRail was introduced, aiding the manual testing team.

- Earlier we did executions on VMs but it was time consuming, costly and less efficient. So, later we migrated to **AWS CodeBuild** and **Test container support (Docker Containerization)** to support parallel execution.
- To add on to the above, we also integrated **DB through AWS RDS** implementation.

Daily Run

To ensure there is **no P1 bug** in product, on daily basis P1 testcases are executed on the latest build. As trust in the automation initiative grew, the client requested daily execution of smoke tests for multiple clients. Initially, these tests were also conducted on virtual machines (VMs). However, due to its drawbacks the process was migrated to **AWS CodeBuild**. Reports from these executions are directly shared with stakeholders.

Library and Tool Updates

Regularly, libraries and tools integrated into the framework were updated to benefit from the latest features and security patches. This helps maintain compatibility and performance.

Load and Performance Testing

In response to the success of previous tasks, the client expressed a desire to perform load and performance testing. JMeter scripts were created to fulfill this requirement.

Benefits

Today, our automation scripts take on an average to execute **11000 testcases** that saves over **1000 hours** of manual testing, greatly improving efficiency and accuracy in the testing process. This case study illustrates how a well-planned and executed automation strategy can lead to significant improvements in quality assurance, testing, and reporting, ultimately benefiting both the development team and the end users.

Our goal was to achieve comprehensive test coverage, ensuring that all critical aspects of the application were thoroughly tested.

Since the implementation of the **RaptorVista** framework, the client has not encountered any critical P1 issues, showcasing the robustness and reliability of the automated testing process