deviQA

10+ YEARS OF EXPERIENCE | **1M+** HOURS WORKED | **100+** ENGINEERS | **300+** HAPPY CLIENTS

Case Study: CipherHealth

Mobile Automation Testing, Dedicated QA Team, Web Testing, Performance Testing

Project Overview

What CipherHealth had in the testing processes when they came to us and what they got after they started working with us.

Before improvement

- Customers faced a huge amount of issues
- Low percentage of app coverage by auto tests
- Long running tests
- Uninformative test results

After improvement

- Decreased the number of client issues and speed up the process of delivering new, well-tested features for customers
- We build an automated test system that includes integration, performance, acceptance, and end-to-end testing solutions for web and mobile parts of application across all environments
- Speed up auto tests by running them in multiple threads and simultaneously for different areas
- Configured a mailing system that sends test results to every relevant person and Slack channel with other useful information
- Covered main functionality for iOS and Android applications
- Added performance tests for different services of the system
- Each QA was assigned to the separate product team to improve workflow and communication

1,400 TESTS DESIGNED | **20** PARALLEL THREADS | **50** JENKINS JOBS FOR TESTING | **25,000** HOURS OF WORK | **3** HEADLESS BROWSERS | **60%** MAJOR BUGS DETECTED

QA Team:
4 Senior Test Engineers

Project length:
5 years

TECHNOLOGIES & TOOLS

- Selenium WebDriver
- Cucumber
- Capybara
- Jenkins
- Appium
- Cipher API
- Faraday
- Google API
- Ruby
- Jmeter

Project Summary

Our client created patient engagement solutions to improve the communication process. CipherHealth is a platform to enhance patient outcomes through care team coordination. It provides such types of services like Digital rounding, Post-Discharge, Follow-Up for patients, Appointment reminder systems, Health Outreach, etc.

Our goal was to improve the quality of the product, decrease the number of missing issues, improve test suite structure, increase test coverage, design and develop the automated testing framework as well as continuously add new automated tests.

Background Significance

Before we joined our client already had some checks which they usually performed by their in-house developers as well as project managers.

These checks were basic ones. As a result of test case insufficiency, the customers faced with a huge amount of issues.

Code changes usually developed and moved to the client without integration testing.

Once we joined, we decided to add two automated QA engineers who will coordinate the testing process, create an approach for test automation, build test suite architecture, etc.

The first one took a position of QA lead and was responsible for connection with management and business team from client-side as well as for creating a testing strategy and testing approach. Another test automation engineer started working on creating the architecture of the automation test suite.

Both test engineers worked closely with the client team, and business leads on each area of the project. After a few months, another 2 test automation engineers were added to the project from our side to speed up the process of test coverage, work more closely with each dedicated team and expand test coverage for mobile apps (IOS and Android platforms).

The main tasks for the QA team included the development of Web automation tests, reporting bugs, implementing mobile and performance testing, creating a simple reporting system, and integrating into CI/CD.

Aims and Goals

There were several goals to bring automated testing in place:

- Functional and integration testing for each part of the application;
- To cover as much as possible cases to minimize manual work;
- To support at least 2-3 latest operating system both for IOS and Android devices;
- To have the ability to run tests using time trigger and code changes as a trigger;
- To have tests splitted by areas;
- Tests for every area should not take more than 1h to run;
- To perform cross-browser, cross-device testing;
- To run tests in multiple threads;
- To avoid random test failures;
- To have ability test HL7 messaging.

Methodology and Work Undertaken

To implement client requirements, we decided to use the following technology stack:

- Ruby;
- Capybara;
- Cucumber;
- Appium;
- Faraday;
- Jenkins;
- Jmeter;
- Google API.

Appium is the right candidate for both iOS and Android automated test development as it has about the same API for both platforms.

Cucumber allows us to write test scenarios in a human-readable format. So, even non-technical people can check reports and see where we have a bug or why the test is failed. Additionally, Cucumber's report has screenshots for each test failed and readable stack trace that allows understanding the nature of the issue.

Our client already had some Jenkins jobs configured for building changes. So, from our side, we implemented the integration of automated tests into the existing build process.

To have an ability to run tests independently from building the application, the performance, mobile, and Web test automation builds were moved to separate Jenkins jobs.

Automated tests for the mobile device were integrated into Jenkins Pipelines to build the application from the most actual branch that contains the latest changes.

Continuous Integration system was configured to exclude human work as much as possible. Each build for mobile application had automation tests running as the latest step of every building process.

To avoid server slowness issues and, as a result, to affect other automated testing jobs, performance tests were running by time trigger.

We split the smoke test suite of automated tests for Web (which takes 10 mins to run) as well as the regression test suite (5 hours to run) into separate jobs.

Smoke tests build was triggered by pushing the changes from the developer (using the special keywords) while the regression test suite was configured to run by time trigger.

For the reporting system, we used cucumber reports. We configured a mailing system that sends test results to every relevant person (e.g., to the developer who made the code changes, head of development, and QA team). Also, test results were integrated with Slack and send job status and other useful information to the channel.

Challenges and Obstacles Overcome

Using API calls for creating preconditions steps.

Some of the scenarios which the client had in his test suite before we joined the project contained a significant amount of steps that were defined as preconditions before tests run.

As a result, it makes our tests slow. To improve the speed, we made refactoring and replaced UI methods in prerequisites to use API. Unfortunately, we didn't have any documentation related to API. To make it happen, we used a proxy for recording all API calls required and converting them to tests. This improvement made our tests faster and more stable.

Testing audio messages and comparison text from audio with text on UI

One of the key features of CipherHealth's app is the ability to create audio records for people who have a problem with their eyes.

It looks like a list of questions to compile and convert to audio files. So, as a part of the quality assurance process, we had to check that the questions were correctly converted into audio files.

We designed and developed a QA tool that allows downloading audio files, parsing text from audio, and compare it with actual text on UI. To implement it, we made in-depth research of third-party libraries available on the market. During researching, we noticed that there were two possible tools (Google Speech-to-Text / IBM Watson Speech-to-Text) that match our expectations and provide high-level quality to convert audio speech into text.

Decreasing time of build

The requirement was to reduce the time for automated smoke tests to run to less than 15 minutes. Every time during the automated tests run, thousands of objects (tables, reports, etc.) are created. As a result, after one workday, the application contains a massive amount of useless data, which causes slowness of loading on some pages and as decreasing the total time of job execution.

We have implemented a "cleaning" job approach on Jenkins by using Rails console and API calls. Besides that, we added parallel running in threads to clean data faster as much as possible. So, after each test run a clean job procedure launches and cleans data created during the test run. The clean job procedure triggers only if the run was successful.

From the beginning of autotest development, the QA lead defined strategy to make test scenarios as much atomic as possible. At first glance, it required more work and doing the same steps multiple times. But as a result, when we introduced running tests in parallel with several threads inside, we didn't spend time refactoring scenarios as all of them were independent. Thus we have decreased job execution from almost 1.5 hours to 20-30 mins.

QA Tool for Performance Testing

The client required us to do some performance testing as they expected a fast-growing customer base.

One of the difficulties was that the application contains different modules, and the loading of one part of the application required some pre-setup work in another.

We have created a QA framework that performs "load" and "performance" testing of different parts of the application. It has flexible configs to control load pressure, the volume of data on the system as well as which part of the app should be tested. Each test contains all pre-requirements and pre-conditions needed. Using this framework, we have performed several rounds of investigations with system engineers and, as a result, improved the performance of the app and isolated specific "bottlenecks".

Performance Testing solutions designed by DeviQA were integrated into Jenkins as well. Jenkin's job has a flexible configuration that allows users to select all required settings (e.g., load only a part of the application, configure the number of users as well as the time of running under that load, etc.)

HL7 messaging

One of the core features of the CipherHealth application is importing patients via HL7 messages. Each HL7 message has a complex structure, and it's hard to manually create and send messages. QA team has developed its own tool to generate and send messages with a flexible configuration of each field and message types. Also, we added the ability to send messages in threads for different care providers. These updates allowed us to measure how many messages the server is able to process. All testing results were provided to the DevOps and development team that allowed them to find some places where performance could be improved on the server-side.

HIPAA, HITECH and Security

Since we are talking about Health Care application, we have to follow the HIPAA Privacy and Security rules. During the testing process, we should ensure that the new feature update or bug fix will not affect PHI or ePHI. Besides that, we should test all limits and permission restrictions that users use to access patients' information.

Access to real PHI is strictly regulated inside the company. To exclude security breaches due to hacker attacks, only employees with an appropriate access level have access to any servers related functionality. Any interactions with servers are made through third-party service and tightly controlled and logged for future reference.

Due to multiple security measures in some cases, HIPAA requirements create difficulties and road blockers in QA work.

When the QA team doesn't have access or partial access to real PHI, we can miss some used data combinations that may cause issues. Some issues have too many pre-requisites and some historical information that we simply can't reproduce using a test environment.

Results

DeviQA did a massive amount of work and completely satisfied the requirements of CipherHealth. During 5 years, we designed and developed more than 1400 automated scenarios that run in 60 Jenkins jobs. Triggering build for each code changes gave us the ability to don't use manual QA from the client-side at all. Multithreading, as well as using API in prerequisites, allowed us to speed up the build time significantly. DeviQA achieved coverage of almost 80% of the apps, both iOS and Android.

The solutions designed decreased the number of client reports by more than 70% which made customer support unhappy as they didn't have enough work anymore.

We are happy that we designed and implemented stable, efficient, fast, and maintainable solutions that allowed us to achieve our goals. We build an automated test system that includes integration, performance, acceptance, and end-to-end testing solutions.

We decreased the number of clients issues and sped up the process of delivering new, well-tested features for customers.

Client partners from the business side could get results of success releases without any problems and see robust growing applications that get pleasure for the customer to use it.

Services Provided

DeviQA designed the mobile and web automation test suites architecture, developed scripts for the scenarios created, integrated the tests into Continuous Integration process, which already existed on the project, and was responsible for Quality of CipherHealth products.

Mobile Automation Testing →

4 dedicated senior QA engineers joined the project to improve the quality of the products. They were quickly integrated into the existing model of work. Besides the fact that a solid development process already existed on the project, they proposed improvements which were accepted by management.

Dedicated QA Team →

The team was not only focused on mobile test automation, they were also responsible for Web apps testing. QA engineers designed and created test documentation which covered more than 90% of tricky cases and was involved in manual web testing process too.

Web Testing →

We used JMeter as a main performance testing tool to understand how quickly the application responded, what number of users it could handle and how it operated under load. Monitors were setup on server side to get information about the CPU, RAM, and other metrics.

Performance Testing →

Let us know your details
so we can get back to you for discussion!

Telephone: +1 805 491 9331, +44 1922 214429
Skype: deviqa_co
Email: info@deviqa.com

https://www.deviqa.com/

1. **Free proof of concept** We will conduct a free proof of concept and prove that you can trust our quality assurance services
2. **12 hours to start** Within 12 hours, our team is ready to start your project
3. **Quick team resize** The size of the QA team on your project could be resized in no time
4. **Daily progress reports** We send an email report with detailed statistics and progress on daily basis