CASE STUDY

Coveros Automates Testing to Ensure Quality at REProphet

Software Design and Development
Test Automation and Test Execution
Continuous Integration and Deployment

REProphet is a small startup that provides financial management software to real estate agents. Their vision was a traditional web application that would provide an interface tailored to the specific problems that real estate agents face when balancing their finances. Their goals included providing a simple intuitive interface, integration with bank information, ROI justifications, and estimated end-of-year tax assessments. Coveros was brought on at a time when development under their current contractor had stalled and they faced an approaching deadline to get a minimum viable product (MVP) to market.

"Coveros understood the implications of releasing high quality, secure software to beta and regular users. They ensured the software produced met the high standards we were looking for to gain the right momentum for a full launch." - **Matt Erdmann, Co-Founder REProphet**

CHALLENGES

REProphet had contracted with an independent developer to build a production web application in PHP. As they approached a deadline to produce an MVP they found the application to be lacking in functionality and exhibiting significant defects. Additionally, security was considered as an afterthought.

- Existing software was undocumented, lacked required functionality, and exhibited significant defects
- MVP, User stories, goals, etc, were undefined, only a high level vision was provided
- Existing software had no functioning CI/CD process or tests of any kind
- Personally Identifiable Information (PII) was stored in plain text, and the database and connections were not secured
- Scalability was not feasible with respect to their chosen application architecture

SOLUTION

REProphet engaged Coveros to take over development, add missing features required to build the MVP, and support the production launch of the application. As no tests currently existed, Coveros decided instead of spending time up-front writing tests for all of the code to instead include tests for all new features. For each story, the definition of done included identifying, implementing, and passing unit and functional tests.

DEVELOPER TESTING

With each completed story, PHPUnit, and QUnit tests were written, for front-end and back-end functionality. Clover was run to determine code coverage, and these results were all pushed into SonarQube. Using this methodology, over the course of 6 months, Coveros was able to bring code coverage from 0% to over 65%.

SonarQube was used to analyze all code, and ensure the quality of the code itself was high. Because a lot of legacy code was being dealt with, no initial quality thresholds were set up. A static code analysis quality gate was added to ensure no critical, blocker, or major bugs were introduced into the code, all tests passed, and that code coverage didn't drop below the current level. In this way, we ensured the code was continually getting better, without having to go back and refactor/fix legacy code.

FUNCTIONAL TESTING

Functional tests were stored in Git in the same repository as the code was. This made it simple to write a feature, and test at the same time, and keep everything in sync. While strict TDD wasn't followed, the process enabled ease of the process, and facilitated testing early and often beyond the unit level.

After developer testing passed, functional Selenified tests were then executed, in several stages. Tests were tagged as smoke, acceptance, and regression and run in that order, to provide quick feedback to developers, letting them know early if the application looked wrong, before taking a deep dive to verify all of the functionality of the application. Similar to the unit tests, this suite grew as functionality was reworked and expanded, without an exceptional amount of time needing to be dedicated to testing before new development could proceed.

SECURITY TESTING

Security testing was a high priority for REProphet as the application has user PII associated with it, and was directly tied to users' financial institutions. As a result, multiple security checks were put in place. An unauthenticated application scan ran in parallel with the system smoke test using OWASP's Zed Attack Proxy (ZAP). When the acceptance tests were executed, a more indepth ZAP scan ran, with an authenticated user. Issues such as improper SSL setup, insecure headers, PII in plain text, and more were easily and automatically detected. Unlike the functional tests, If any issues were identified (not just new), the build failed, and resolving these issues were prioritized. Additionally, Selenified's proxy capabilities were utilized, so that all functional tests ran through ZAP, providing further security analysis of the system.

Manual system scans were also run using w3af, as a manual process before release, to ensure the system configuration was secure; checking firewalls settings, applications installed, etc. Finally, performance tests were run using JMeter to ensure the application could handle the expected load. The Selenified tests were run through JMeter as a proxy, and recorded, and then re-ran through JMeter increase the number of users. This was run as a manual process outside of the CI/CD process.

CASE STUDY

Coveros Automates Testing to Ensure Quality at REProphet

Software Design and Development
Test Automation and Test Execution
Continuous Integration and Deployment

TECHNOLOGY SOLUTIONS & RESULTS

Software Development

- PHP/Laravel framework
- JQuery
- NGINX
- MySQL

Test Automation

- PHPUnit
- Clover
- Selenified
- OWASP ZAP
- JMeter
- w3af

Continuous Integration

- AWS (console and cli)
- Gradle
- SecureCI (GIT, Jenkins, SonarQube, Nexus)

BUSINESS VALUE

Coveros was able to help REProphet successfully get to launch on their planned date and start onboarding their initial customer base. Due to the functional and non-functional tests added, confidence in the developed software and desired features was obtained. Multiple bugs security and feature issues were identified and resolved early in the development process, before the software ever went live. These included mis-calculated ROI values, date and filter problems with data viewing, insecure system configuration problems, and improper data handling. These tests not only provided confidence to REProphet that the software was functioning as desired, but also produced traceable reporting. As a result, issues were resolved at a low cost, without any impact to user data or performance.