coveros

Coveros Automates Testing to Ensure Quality in a Startup's New Application

CASE STUDY

CHALLENGES

- Stalled development
- Undocumented, defective software
- Undefined goals
- No CI/CD process
- Unsecure database
- No application scalability

SOLUTIONS

- Automated testing
- Static code analysis quality gate
- Specified MVP and definition of done
- Integrated CI/CD process
- System scan and security checks
- Scalable architecture

CASE STUDY

Coveros Automates Testing to Ensure Quality in a Startup's New Application

REProphet is a small startup that provides financial management software to real estate agents. Their vision was a traditional web application with an interface tailored to the specific problems 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.

REProphet had contracted with an independent developer to build a production web application in PHP. As they approached a deadline to produce a minimum viable product (MVP), they found the application to be lacking in functionality and exhibiting significant defects.

CHALLENGES

- Development had stalled
- Existing software was undocumented, lacked required functionality, and exhibited significant defects
- The MVP, user stories, and goals were undefined; only a high-level vision was provided
- Existing software had no functioning CI/CD process or tests of any kind
- Security was an afterthought: Personally identifiable information (PII) was stored in plain text, and the database and connections were not secured
- Scalability was not feasible with respect to the 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 existed, instead of spending time up front writing tests for all the code, Coveros decided to include tests for all new features. For each story, the definition of done included identifying and implementing unit and functional tests.

DEVELOPER TESTING

With each completed story, the team wrote PHPUnit and QUnit tests for front-end and back-end functionality. The team ran Clover to determine code coverage, and these results were all pushed into SonarQube. Using this methodology, over the course of six months, code coverage was brought from 0 percent to over 65 percent.

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

"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, Cofounder, REProphet

FUNCTIONAL TESTING

Functional tests were stored in Git, in the same repository as the code. This made it simple to write a feature and test at the same time, keeping everything in sync. While the team didn't follow strict test-driven development, the process was close enough that it facilitated testing early, often beyond the unit level.

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

SECURITY TESTING

Security testing was a high priority for REProphet, as the application has associated user PII and is 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 executing the acceptance tests, Coveros ran a more in-depth ZAP scan 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 ones), the build failed, and the team prioritized resolving these issues. Selenified's proxy capabilities were also utilized, so all functional tests ran through ZAP, providing further security analysis of the system.

Coveros ran system scans using w3af as a manual process before release, checking firewall settings, applications installed, etc. to ensure the system configuration was secure. Finally, performance tests using JMeter confirmed the application could handle the expected load. The Selenified tests were run through JMeter as a proxy, recorded, and then rerun through JMeter to increase the number of users. This was run as a manual process outside of the continuous integration and delivery (CI/CD) process.

CI/CD PROCESS

Coveros set up SecureCI, an open source CI/CD platform, to manage the application build. This provided the basic tooling and development components needed for building the app: Git, Jenkins, SonarQube, Nexus, and AWScli. The above functional and nonfunctional testing was integrated into a DevOps pipeline to coordinate building and testing the application.

Jenkins orchestrated testing the code, first running the developer tests described above and then deploying

the application. Once the application was deployed, Coveros ran the functional and security tests outlined above, using SonarQube to keep track of metrics and enforce quality gates. If tests passed, the application was promoted to higher environments so that longer running tests could be executed and, eventually, deployed to production.

TECHNOLOGY SOLUTIONS

Software Development

- PHP/Laravel framework
- jQuery
- NGINX
- MySQL

Test Automation

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

Continuous Integration

- AWS (console and command line interface)
- Gradle
- SecureCI (Git, Jenkins, SonarQube, Nexus)

BUSINESS VALUE

Coveros was able to help REProphet successfully launch their application on their planned date and start onboarding their initial customer base. Due to the functional and nonfunctional tests added, REProphet gained confidence in the developed software and desired features.

Multiple bugs and security and feature issues were identified and resolved early in the development process, before the software ever went live. These included miscalculated 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.

CONNECT WITH COVEROS

coveros.com

info@coveros.com

929.341.0139

twitter.com/coveros

linkedin.com/company/coveros