Introduction

Efficient testing methods are becoming increasingly important to ensure teams produce high quality applications. Implementing testing methods effectively help drive the following results:

Client Partnership

Low Defect Rate

Speed of Delivery

Methods

This one-pager identifies critical testing strategies that should be embedded in every Appian delivery team and recommends others that are dependent upon the client and solution being developed. At the start of each project, determine which of the project-specific methods will be utilized by your team.

Design Documents

By writing design documents, developers can identify test cases before development starts. This is a core tenet of Test Driven Development. Designs should be reviewed as a team at the start of the sprint so others can identify additional risks and test scenarios.

Expression Test Cases

Expression Rule Test Cases speed up bug discovery by ensuring changes are backwards compatible by unit testing rules as they are developed. They can be configured to run automatically with the Automated Rule Testing shared component.

Manual Unit Testing

Units test are executed on the smallest piece of code that can logically be tested. For the Appian objects that can't be verified in an automated fashion, manual tests should be carried out.

Peer Testing

A code review by a peer helps ensure best practices and common design standards are followed. In addition, functional peer testing should be carried out to ensure all requirements are satisfied.

Acceptance Testing

Acceptance testing should be done at the conclusion of every sprint to continuously collect feedback from SME's, PO's, and end-users. Formalized User Acceptance Testing (UAT) is held at defined points within the project to collect feedback prior to the solution's release.

Exploratory Testing

Exploratory tests (ET) use general patterns of interacting with the system, called heuristics, to identify defects. Work with your team to incorporate ET into your established testing efforts (SIT, peer, regression testing, etc.).

Regression Testing

After major changes/deployments and established points in the project, (hardening sprints, production rollout, etc.) regression tests should be executed to ensure recent changes did not cause unanticipated impacts to existing functionality.

System Integration Testing

System Integration Testing (SIT) ensures the application works cohesively. During SIT, the following are tested: database connection, integrations with outside systems, and work flows within the application. Work with your project team to determine the best cadence for SIT.

Deployment Verification Testing

All release candidates should be deployed and verified prior to a prod deployment. Smoke tests are executed to ensure all environment properties, Appian objects, integrations, and data have been promoted successfully.

Performance/Load Testing

Performance testing is a way to measure page load times and resource utilization across the platform. The results of performance testing will inform business decisions, scaling considerations, and application design.

Automated End to End Testing

FitNesse for Appian with the FitFam shared component and Cucumber for Appian allow developers to automate functional and regression testing of the user interfaces and workflow. The preferred tool will vary based on tester, client infrastructure, and familiarity.

Accessibility Testing

All teams should be designing with accessibility in mind, but some clients enforce strict accessibility standards. Screen readers like JAWS and NVDA can be used verify if the application is 508 compliant.

○ *Required* ○ *Project Specific*

Conclusion

Ensuring quality requires an investment throughout the project: design, development, testing, acceptance, and deployment. The strategies outlined are proven to help teams deliver successful projects by allowing for feedback to be identified earlier, when it's less costly to address.