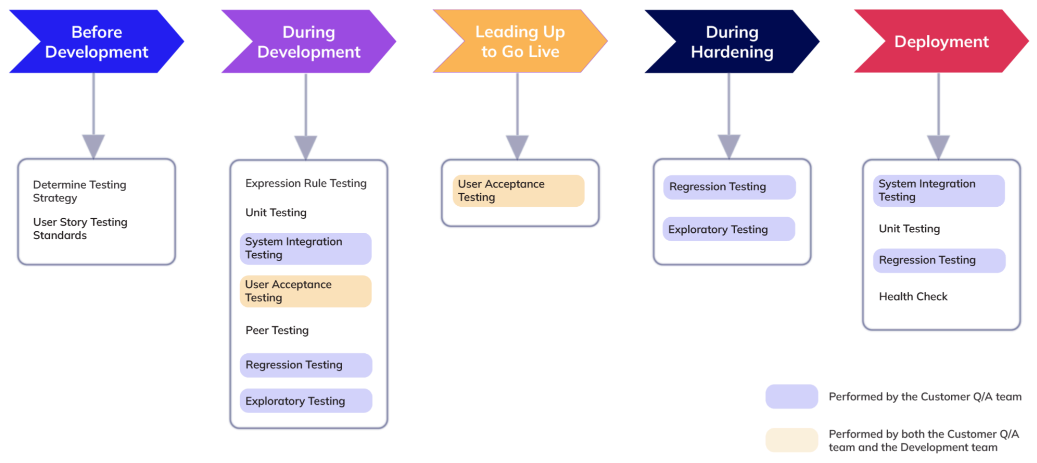
Testing One Pager

Introduction

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.



Systems Integration Testing ● ●

Systems integration testing is used to test that all components work together seamlessly and cohesively. This testing includes workflows, integrations with outside systems, and database connections.

Unit Testing ●

Unit testing is used by the developer to test individual units of code. Objects such as expression rules and decision rules have an expected result, so code is written to test all possible scenarios. More information can be found in Functional Testing - Unit Testing and Creating Expression Rule Test Cases.

Expression Rule Testing ●

In expression rules, we use test cases to determine if the expected results are being achieved based on certain input. They can be configured to run automatically with the Automated Rule Testing shared component. See Creating Expression Rule Test Cases to learn more.

User Acceptance Testing ● ● ●

User acceptance testing makes sure that the application meets the needs and requirements of the users. Typically performed at the final phase of development, it should also be done throughout the duration of the project.

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.

Health Check ●

Health Check is a process that provides insights into an application. It can identify services and nodes that pose a high risk to the performance of the application, ensure that best practices are being followed, and provide graphs detailing historical trends in the environment.

Exploratory Testing ● ●

Exploratory testing is a method of testing that’s used to explore areas of risk in your application. It can identify edge cases that could lead to failure of your application and bugs that might otherwise go undetected.

Regression Testing ● ●

Regression testing checks to see that any changes or updates in the software doesn’t negatively impact any existing functionality. This will ensure backwards compatibility when the changes go live.