




eGuide

# **Best Practices for Conducting Generative AI-Based Test Automation**



The best practices for developing and running an efficient and effective test automation program have been discussed by many in the industry. It's important to determine your strategy, and then implement your test automation in a way that supports your DevOps process. It should also support release timing, provide actionable information for bug resolution and go/no-go release decisions, and meet your risk tolerance specifications.

However, the best practices for automation may look different when you are bringing to bear the power of a generative AI platform for creating and executing your automated tests. The purpose of this guide is to highlight the key areas where Best Practices need to be defined differently and provide you with suggested Best Practices that support the transformative aspects of using a generative AI-based testing platform and framework.

## Why You Need to Follow Unique Best Practices With a Generative AI-Based System

In the past, a Test Automation program was designed around parameters like test reusability presumptions. Given the effort required to create a test script and then maintain it, a test was only automated when it would be used enough times to justify this time investment. But generative AI changes the calculation—with AI writing tests instantly and autonomously, scripts are now disposable, so your program no longer has to be limited by the reusability constraint.

Let's get more specific about what generative AI can do for you **today** with respect to test automation, and in particular, the capabilities that Appvance's AIQ platform provides.

### AI can write test scripts

The AIQ platform can generate 2 types of test scripts automatically with its AI models:

- **Regression Test**

These tests are based on anonymized log data capturing the behaviors of actual users. AI then uses this data to reconstruct the use cases and write corresponding test cases so that your Regression Test Suite is always up to date, covering the flows of your actual users.

- **Exploratory Tests**

AI can create test scripts covering every possible user flow in an application. Think of these AI-generated tests as the exploratory testing your team would conduct but on steroids. You can now cover even the newest code submitted by your dev team and ensure that it is good for release on day one. These exploratory tests provide complete Application Coverage™, which is only possible to achieve with the use of these AI-generated tests.

This test creation capability is state of the art, but more types of test creation by AI are sure to be coming soon.



## AI can create (synthetic) test data

With the enormous number of test scripts created by AI, and the ubiquity of continuous testing needs, it's hard to keep up with creating test data to satisfy your needs. AI can step in to alleviate this burden with some quick instruction.

## Best Practices For a Generative AI-Based Test Automation Program

In light of the capabilities of generative AI, we've (re)considered the best practices for creating and executing your test automation program. Here are the areas we feel you need to look at differently, and we also provide our take on the new Best Practices requirements in each area.

### Test Design / Strategy

As always, your program needs to begin with the design and strategy phase, where you match your testing strategy and plan to your business goals. The capabilities of AI impact this phase significantly. So here are our recommendations for your revised Best Practice approach.

### Recommended Best Practices

#### 1. Rethink Test Scripts:

With AI in the picture, the need to convert every test case into a test script diminishes. Instead, you can focus on identifying critical test scenarios and generating test scripts for them. Consider test cases that require complex decision-making or involve interactions with adjacent systems, as those most warrant explicit and thorough testing.

#### 2. Error Reporting:

AI is capable of detecting a larger number of errors compared to traditional testing approaches. To manage the influx of reported errors, establish rules for immediate reporting and prioritization of critical issues. Classify issues based on severity and impact, addressing high-priority concerns first.

### 3. Evolve Test Case Development:

While AI generates a comprehensive set of tests, it does not eliminate the need for human input entirely. Savvy QA managers play a crucial role in guiding AI-driven testing. For instance, it is often valuable to have AIQ focus on creating test cases for unique scenarios, edge cases, and critical functionalities. This helps ensure that its training is comprehensive and effective.

### 4. Enhance AI Training:

Speaking of training, to effectively train AIQ, shift the focus from user flows to documenting business rules. Clearly define the expected behavior, constraints, and conditions of the application-under-test (AUT). By providing explicit instructions regarding business rules, you enable AIQ to understand the desired outcomes and identify potential deviations.

### 5. Regression Testing Frequency:

With AI-powered testing, it becomes feasible to perform full regression tests after every build. However, the decision to do so should consider factors such as the size and complexity of the AUT, time constraints, and available resources. It may be more practical to prioritize regression testing for critical areas of the application.

### 6. Reevaluate Test Coverage:

The old-school metrics of Test Coverage and Code Coverage have been supplanted by Application Coverage™, which is the new standard of testing completeness. This is because Application Coverage mimics user experience and can now be comprehensively achieved via generative AI. You can read more about why comprehensive Application Coverage is not just achievable with a generative AI-based system like AIQ, but should now be expected, in this recent blog post.


## Design for Test

Developers and quality assurance teams must collaborate to succeed in today's fast-paced software development industry. By working together from the beginning, Dev and QA can incorporate best practices that make test automation easy and efficient, thereby accelerating release cycles and improving quality. Together, they foster collaboration and enable comprehensive test automation.

## Recommended Best Practices

### 1. Collaborate with Dev to create a test-specific environment

It is essential to establish a dedicated test environment to facilitate test automation. By working closely with Dev, QA can identify the specific requirements for testing and find workarounds that meet those needs without impacting the production environment. This ensures that test cases run smoothly and are not affected by external factors. For instance, the test environment can include workarounds for multi-factor authorization (MFA), as discussed below.



## 2. Assign element IDs and consider testability

To enhance the reliability of test automation, Dev should assign unique element IDs to every element in the application. Traditional accessors, such as XPath or CSS selectors, can be dynamic and unreliable. Element IDs provide a stable identifier for automation scripts, making them less prone to failures caused by UI changes. Moreover, during the design phase, Dev should consider what elements need to be tested, ensuring consistency in domain and data types across the application.

## 3. Organize and label common elements

To maximize efficiency, it's essential to identify common elements and organize them systematically. Dev should design the application in a way that allows for easy identification and retrieval of common elements during test automation. By ordering, labeling, and organizing common elements consistently, QA can create reusable test procedures that can be shared across the team. This approach streamlines the development of test cases and minimizes duplication of effort.

## 4. Leave clues for the test team

Dev can support the test team by leaving breadcrumbs or clues in the application code. By writing to the log before and after critical steps, the test team can easily verify that the expected events occurred. This practice assists in debugging and troubleshooting, making it easier for QA to identify potential issues and ensure the application functions as intended. Collaboration between Dev and QA in this aspect promotes transparency and accelerates the test automation process.

## 5. Involve the test team in design conversations

One of the key aspects of "Design for Software Test" is involving the test team in design conversations right from the beginning. By including QA professionals in discussions and decision-making processes, their unique perspective and expertise can contribute valuable insights. The test team can provide feedback on the design, suggest additional requirements, and identify potential challenges for test automation. This collaboration ensures that the application design incorporates testability, making it easier to create comprehensive and effective test cases.

The collaboration between Dev and QA teams is crucial for successful test automation. By following the five best practices listed above, organizations can create a harmonious working environment where both teams work together to ensure fast release cycles and high quality. By incorporating testability into the application design, assigning element IDs, organizing common elements, leaving clues for the test team, and involving QA in design conversations, organizations can streamline the test automation process, which is essential to the delivery of high-quality software. Embracing these practices encourages collaboration between Dev and QA, ultimately resulting in faster time-to-market, improved product quality, and enhanced customer satisfaction.

## Test Data

The overall best practice for the provision of test data is to design and generate it, so-called Synthetic Data Generation. This involves generating synthetic data that encompasses various data combinations and scenarios, an approach that ensures comprehensive test coverage now that AI has enabled such a wide array of tests.

AIQ has robust Synthetic Data Generation capabilities, including a vast library of fictional names, streets, cities, email addresses, colors, sizes, part numbers, etc. These can be generated in any combination to create representative test data. Further, AIQ can add regular expressions (also known as Regex) to the test data to conform to a particular pattern, e.g., product codes or customer codes, or to create dates in the future (for a delivery date) or dates in the past (for a birth date).

It is important to have test data that tests all the corner cases (the domain of each data element) and the valid and invalid combinations (positive and negative testing). Plus, it must be a stable dataset. AIQ's test data generation provides that stable dataset.

## Multifactor Authentication (MFA)

Multi-Factor Authentication (MFA) is an essential security measure to protect applications from unauthorized access. However, MFA poses challenges for test automation teams who need to strike a balance between comprehensive automation and MFA-enhanced security. But there are a series of test automation best practices to use when MFA is in the mix. These support effective automation with uncompromised security.

### Recommended Best Practices

#### 1. Understand the purpose of MFA

MFA is designed to defeat brute-force attacks and unauthorized access attempts. It is crucial to understand the rationale behind MFA when developing appropriate test automation best practices. Recognize that while automation is important, the primary objective of MFA is to safeguard the production application and its users' data.

#### 2. Devise an MFA workaround for testing

Work with the development team to devise a workaround that caters specifically to the test environment when an application-under-test (AUT) uses MFA. Here are some techniques:

- Use a token that always works

Create a test-specific token that bypasses MFA and can be used exclusively for automation purposes.

- Disable MFA altogether in the test environment

Temporarily disable MFA during testing to streamline the automation process. However, be cautious to enable it again for production environments.

- Provide an API call for setting credentials

Develop an API endpoint that allows automation scripts to set the required MFA credentials programmatically.

- Store the token in the database

Have the development team store the MFA token in the test database, allowing automation scripts to retrieve it during tests.

- Utilize web SMS services

Integrate a web SMS service to retrieve the MFA token automatically during test automation.

### 3. Ensure that MFA is back in place when the application goes to production.

Conduct manual tests to verify that the MFA workaround implemented for test automation purposes doesn't exist in the production build. This ensures the integrity of the MFA process and avoids security vulnerabilities.

### 4. Maintain Separate Environments

Maintain a clear separation between the test and production environments. Test environments should have distinct configurations that facilitate efficient test automation. Ensure that the MFA workaround implemented for testing purposes DOES NOT carry over to the production environment, where MFA should function as intended.

Now that you're armed with these revised Best Practices, you can achieve maximum efficiency and greater performance in your test automation program. If you haven't adopted a generative AI-based system like AIQ yet, contact us to get a demo and see its power in action.