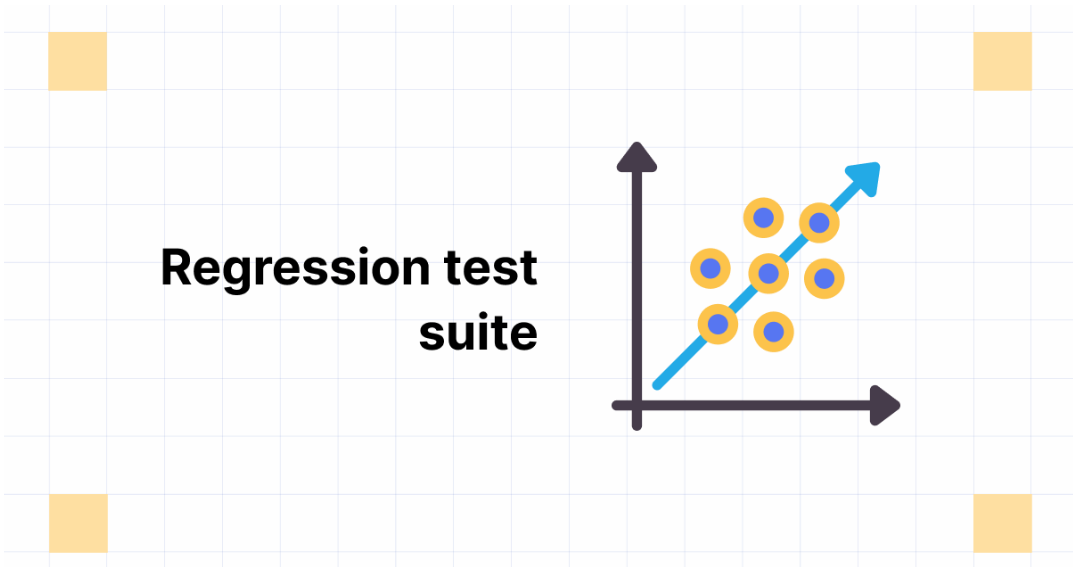
# *How to Build an Effective Regression Test Suite*

Aaron Thomas; September 30, 2024, https://testsigma.com/blog/regression-test-suite/



Regression Testing is a **type of testing in the software development cycle** **<u>that runs after every change</u>** **to ensure that the change introduces no unintended breaks**. Regression testing addresses a common issue that developers face — the emergence of old bugs with the introduction of new changes.

Regression testing has undergone remarkable transformations over the past decade in the ever-evolving landscape of software testing. The days of enduring weeks, or even months, of regression testing cycles on a single candidate code build are now behind us. Today, the spotlight is on creating regression test suite that epitomize agility, effectiveness, and efficiency.

Irrespective of the software development methodology employed, the pressure to deliver top-notch code swiftly to satisfy customer demands and stay competitive remains constant. In the realm of Agile, regression testing could span just 1-2 days, though some enterprises might extend this to 3-5 days. However, what takes precedence is not the duration of the testing phase but the efficacy of the regression testing suite itself.

## What are the benefits of a regression test suite?

A regression test suite offers several benefits, including:

- **<u>Detecting Regressions</u>**: It identifies unintended defects or issues introduced by new changes, ensuring the application's stability.
- **<u>Ensuring Continuity</u>**: By retesting existing functionalities, it safeguards against the unintended impact of new features or modifications.

- **Enhancing Confidence**: Consistent regression testing boosts confidence in software quality and aids in maintaining a positive user experience.
- **Speeding Up Fixes**: Rapid identification of regressions enables prompt fixes, reducing the potential for prolonged disruptions.
- **Cost Savings**: Addressing issues early prevents costly post-release fixes and enhances overall software development efficiency.

## How to Create a Regression Testing Suite ?

Creating an efficient regression test suite involves considering various approaches that balance effectiveness and speed within a short execution window. There are a couple of methods to explore, each catering to different project dynamics and goals. Let’s explore these options and five essential actions to ensure a successful regression testing suite.

### Option 1: Group-based Regression Testing

One approach involves categorizing tests into logical groups that cover the full spectrum of your application’s functionality. You could organize tests by features, functions, or integrations. For each release, you’d execute a different group in rotation. This approach ensures that your application is thoroughly tested over multiple releases, offering comprehensive coverage. Alternatively, focus solely on the tests relevant to the most recent changes, ensuring efficient use of testing resources.

### Option 2: Continuous Testing

Continuous testing involves seamlessly integrating testing into your development cycle. Your quality assurance team continually tests new features and addresses bugs. Manual regression testing can be performed alongside automated tests that run conveniently—perhaps during overnight hours or weekends. This approach eliminates the need for separate testing cycles and maintains a continuous feedback loop.

## 5 Essential Steps for an Effective Regression Test Suite:

### 1. Prioritize Tests

Identify critical and high-priority tests that assess core functions and essential workflows. These tests play a vital role in identifying defects that need immediate attention. Consider using codeless automation or code-based automated tests for these crucial functions to catch issues early and prevent customer impact.

### 2. Establish Smoke Tests

Create a “smoke test suite” comprising critical and high-priority tests. Execute these tests daily, bi-weekly, or with each build to quickly identify potential problems. Automate these tests to save time and facilitate swift failure analysis. Smoke tests ensure the foundational stability of your application before further testing.

### 3. Incorporate Manual Testing

Include a suite of tests that cover basic functionality across the application. These workflows need to be simplified for automated tools to handle accurately. Manual testing allows for exploring edge cases and end-to-end scenarios that automated tests might overlook.

### 4. Validate Integrations

Execute a regression suite that thoroughly tests API connections, back-end processes, and data feeds. These integrations might not be visible to users but are crucial for seamless functionality. While creating these tests might be labor-intensive, their value in preventing failures must be balanced.

### 5. Address Performance

Don’t neglect performance testing, as it significantly impacts user experience. An application that slows down with each release can erode customer trust. Develop a performance regression suite that uses testing tools or manual tests to assess speed and identify bottlenecks. Prioritize customer satisfaction by ensuring consistent application performance.

In conclusion, a well-crafted regression test suite can enhance the quality and reliability of your software. Whether you opt for grouped testing or a continuous approach, prioritizing critical tests, incorporating smoke tests, manual testing, validating integrations, and assessing performance are all essential steps toward building a robust and effective regression testing strategy.

## Maintain the Regression Test Suite

Maintaining a regression test suite is crucial to ensure its continued effectiveness and relevance. Here are some key practices to follow:

**Regular Review**: Periodically review the regression test suite to identify tests that have become obsolete or redundant due to changes in the application.

**Remove Outdated Tests**: Eliminate no longer applicable tests, reducing the suite’s size and focusing resources on tests that matter.

**Update for Changes**: Whenever the application undergoes significant updates or changes, update corresponding regression tests to reflect these modifications.

**Add New Tests**: Continuously add tests for new features, functionalities, and integrations to maintain comprehensive coverage.

**Prioritize Tests**: Reassess the priority of tests based on evolving project requirements and user needs.

**Automated Monitoring**: Implement automated monitoring tools to detect changes that might impact the application’s behavior, triggering updates to tests accordingly.

## Best Practices for Regression Test Suites

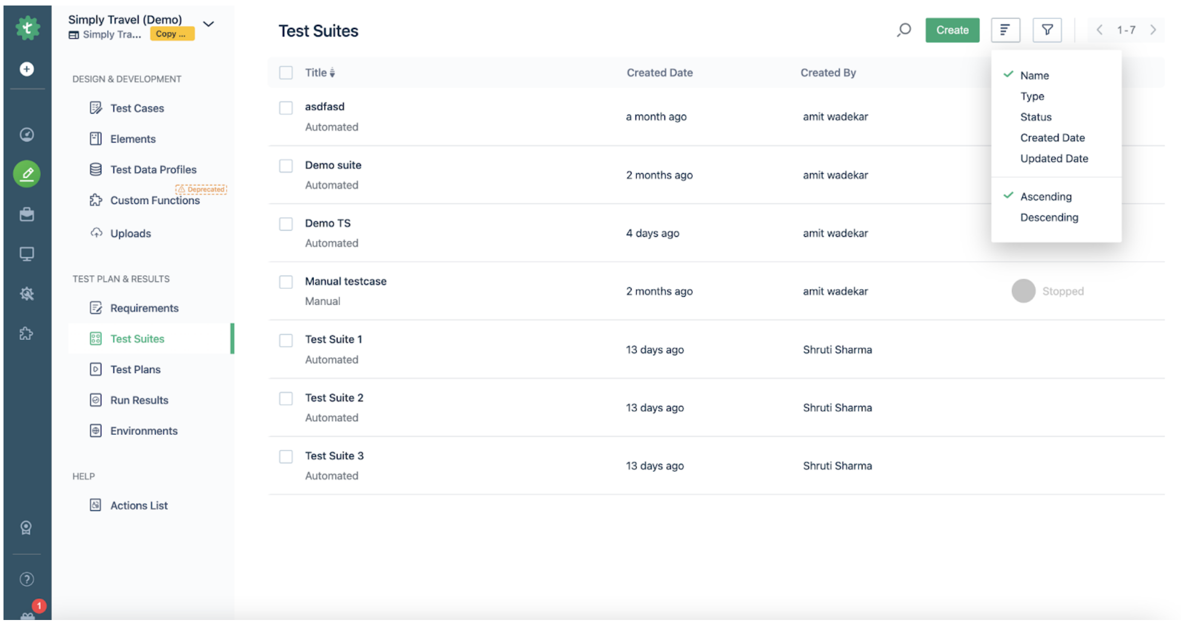
Building an effective regression test suite requires adherence to certain best practices. Here’s a quick rundown:

- **Test Isolation**: Ensure tests are independent and don’t rely on each other’s outcomes, minimizing false positives and making debugging easier.
- **Version Control**: Keep test scripts and related resources under version control to track changes and collaborate effectively.

- **Automation Balance**: Strike a balance between manual and automated tests, recognizing that not all scenarios are suited for automation.
- **Regular Execution**: Execute regression tests frequently to catch issues early and maintain confidence in application functionality.
- **Test Data Management**: Manage and update test data effectively to simulate real-world scenarios accurately.
- **Documentation**: Document test cases, procedures, and expected outcomes to facilitate consistent knowledge sharing.
- **Maintenance Strategy**: Establish a clear plan for test maintenance, including responsibilities and timelines.
- **Collaboration**: Foster collaboration between developers, testers, and stakeholders to ensure everyone's insights contribute to a comprehensive test suite.
- **Continuous Improvement**: Review and refine the test suite to adapt to changing requirements and lessons learned.
- **Parallel Execution**: When possible, parallelize test execution to reduce overall testing time.
- **Reporting**: Generate clear and actionable reports after each regression test cycle to communicate findings and facilitate decision-making.
- **Feedback Loop**: Establish a feedback loop between testing and development teams to address issues promptly.
- **Performance Testing**: Regularly assess the performance of your regression test suite itself to optimize its efficiency.

By following these best practices, you can maintain a regression test suite that evolves alongside your application, effectively catching regressions and ensuring high software quality.

## Automating your Regression Test Suite (Explain with Testsigma tool as an example)



Test Suites

| Title | Created Date | Created By | |
|---|---|---|---|
| asdfasd<br>Automated | a month ago | amit wadekar | |
| Demo suite<br>Automated | 2 months ago | amit wadekar | |
| Demo TS<br>Automated | 4 days ago | amit wadekar | |
| Manual testcase<br>Manual | 2 months ago | amit wadekar | Stopped |
| Test Suite 1<br>Automated | 13 days ago | Shruti Sharma | |
| Test Suite 2<br>Automated | 13 days ago | Shruti Sharma | |
| Test Suite 3<br>Automated | 13 days ago | Shruti Sharma | |

Automating regression tests using tools like Testsigma can streamline your testing process. Here's a step-by-step guide on how to create and execute test cases using the Testsigma recorder, along with examples:

### Test Case Creation with Testsigma Recorder

1. Log in to Testsigma: Access your Testsigma account and navigate to the project where you want to automate regression tests.

2. Navigate to Recorder: Click on the "Recorder" option to launch the Testsigma recorder tool.

3. Record a New Test Case:

Click on "New Test Case" to start recording a new test.

Enter a name and description for the test case.

Select the browser or device you want to test on.

4. Record Actions:

Open the URL of the application you want to test.

Interact with the application by clicking buttons, entering data, and navigating through various pages.

5. Add Assertions:

Insert checkpoints in your test to verify expected outcomes. For example, verify the presence of a certain element or the correct text on a page.

6. Parameterization (Optional):

If you want to run the test with different inputs, you can use parameterization to vary data during test execution.

7. Save and Finish Recording:

Save the recorded test case. Once assessed the test steps, run the test case.

## Re-execution of All Test Cases in the Regression Test Suite

1. Navigate to Test Suites:

In your Testsigma project, create a new test suite or access an existing one.

2. Add Test Cases:

Add all your regression test cases to the test suite.

3. Execute the Suite:

Click on the “Run” button to execute all test cases within the test suite.

Observe the test execution progress and results.

## Execution of Selected Test Cases Based on Impacted Areas

1. Tag Test Cases:

Tag test cases with relevant keywords or labels based on impacted areas. For example, tags could be “Cart Checkout,” “Payment,” etc.

2. Create Test Suites:

Create separate test suites for each impacted area, adding test cases with matching tags.

3. Execute Selected Suites:

Run the specific test suites related to the impacted areas.

## Execution of Selected Test Cases Based on Priority

1. Prioritize Test Cases:

Assign priority levels to test cases based on their criticality.

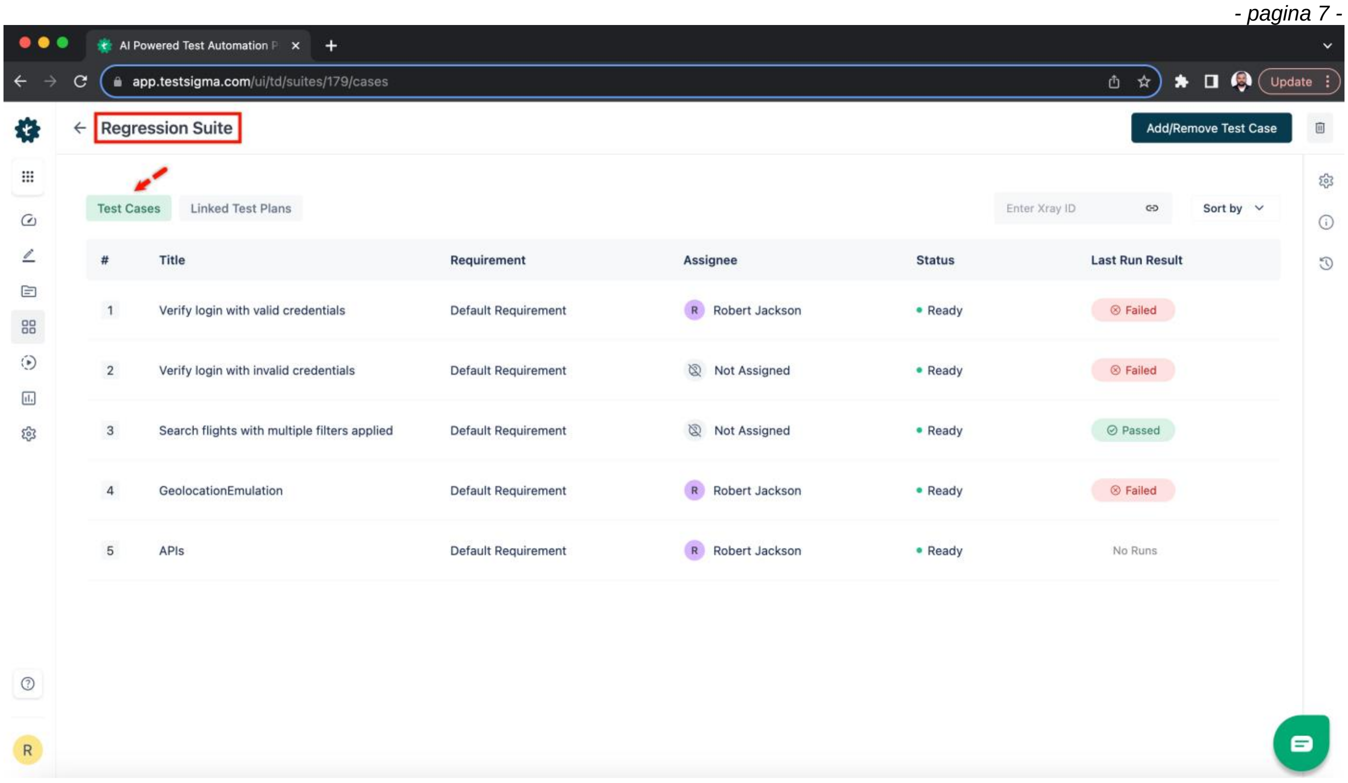
2. Create Test Suites by Priority:

Organize test cases into test suites according to their priority levels.

3. Execute High-Priority Suites:

Begin by executing the test suites containing high-priority test cases.

Automating regression tests with Testsigma involves creating test cases using the recorder, organizing them into test suites, and executing them based on various criteria such as re-execution, impacted areas, and priority. This approach enhances testing efficiency, reduces manual effort, and ensures thorough regression testing to maintain software quality.



Regression Suite

| # | Title | Requirement | Assignee | Status | Last Run Result |
|---|---|---|---|---|---|
| 1 | Verify login with valid credentials | Default Requirement | Robert Jackson | Ready | Failed |
| 2 | Verify login with invalid credentials | Default Requirement | Not Assigned | Ready | Failed |
| 3 | Search flights with multiple filters applied | Default Requirement | Not Assigned | Ready | Passed |
| 4 | GeolocationEmulation | Default Requirement | Robert Jackson | Ready | Failed |
| 5 | APIs | Default Requirement | Robert Jackson | Ready | No Runs |

## Tools for Automating Your Regression Test Suite

Automating regression tests can significantly enhance efficiency and accuracy. Several tools can help streamline this process:

1. Testsigma: A cloud-based test automation platform that emphasizes codeless automation. It allows users to create, execute, and maintain automated tests without writing code, making it suitable for technical and non-technical team members. Testsigma offers integrations with various tools and supports web, mobile, and API testing.

2. Selenium: A widely-used open-source tool for automating web browsers. Selenium supports multiple programming languages and offers browser interaction, form submissions, and data validation features.

3. Appium: Designed for mobile app testing, Appium is an open-source tool that automates mobile applications on iOS and Android platforms, enabling cross-platform testing.

4. Cypress: Known for its speed and reliability, Cypress is a JavaScript-based end-to-end testing framework. It offers real-time reloading, debugging, and an easy-to-use interface.

5. TestNG: A testing framework inspired by JUnit and NUnit, TestNG provides advanced functionalities such as parallel execution, test grouping, and test dependency management.

6. JUnit: A popular testing framework for Java applications. JUnit simplifies the creation and execution of unit tests, ensuring code reliability.

7. TestComplete: This commercial tool offers automated desktop, web, and mobile application testing. It supports multiple scripting languages and has a wide range of testing features.

8. Robot Framework: An open-source automation framework that uses a keyword-driven approach. It supports web, desktop, and mobile application testing and offers easy extensibility.

Using these automation tools, including Testsigma, you can accelerate your regression testing process, reduce human error, and ensure consistent and reliable software quality across various platforms.

## Summary

This comprehensive guide explored the nuances of creating and maintaining an effective regression test suite. We've discussed strategies like group-based testing and continuous testing, highlighted key actions to prioritize, and touched on automation tools that can streamline the process. By adhering to best practices, such as regular updates, test isolation, and collaboration, you can ensure your regression test suite remains a powerful asset for software quality assurance.

## Frequently Asked Questions

### How many test cases are in a regression suite?

The number of test cases in a regression suite varies depending on factors like application complexity, critical features, and available resources. It's crucial to balance comprehensive coverage and efficient testing, focusing on high-priority scenarios and maintaining a manageable suite size.