**Introduction to Non-Functional Testing**

In this module, you'll add nonfunctional tests to the pipeline. These tests check characteristics like performance and reliability.

In Run functional tests in Azure Pipelines , you and the Tailspin Toys web team added UI tests to the pipeline. UI tests are a form of functional testing. Functional tests verify the functionality, or behavior, of software. In other words, functional tests verify that each function of the software does what it should.

But what about the nonfunctional aspects of an application, such as performance and reliability? After your service is operational and deployed to an environment, how can you determine the application's performance under both realistic and heavy loads? Does your application expose any loopholes or weaknesses that might cause an information breach?

In this module, you'll explore the types of nonfunctional tests you can run. You'll run load tests in your pipeline by using Apache JMeter.

**Learning objectives**

In this module, you will:

* Define nonfunctional tests and learn how they compare to functional tests.
* Identify some popular kinds of nonfunctional tests you can run.
* Run automated load tests in the pipeline by using Apache JMeter.
* Convert test results to a format that Azure Pipelines can visualize for you.

**Prerequisites**

The modules in this learning path are part of a progression. To follow the progression from the beginning, be sure to first complete these learning paths:

* Evolve your DevOps practices
* Build applications with Azure DevOps

We also recommend that you start at the beginning of the Deploy applications with Azure DevOps  learning path.

If you want to go through just this module, you need to set up a development environment on your Windows, macOS, or Linux system. You need the following resources:

* An Azure DevOps organization .
* An Azure subscription .
* A GitHub  account.
* Visual Studio Code .
* .NET Core 3.1 SDK .
* Git .

You can get started with Azure and Azure DevOps for free. You don't need an Azure subscription to work with Azure DevOps, but here you'll use Azure DevOps to deploy to resources that exist in your Azure subscription.

The development environment lets you complete the exercises in this and future modules. You can also use it to apply your new skills to your own projects.

**Note**

Keep in mind that you can use Azure DevOps to build and deploy almost any kind of application written in any language. In this module, you'll be working with a .NET Core application written in C#.

You don't need to be an expert in .NET or C# to complete this module. You can apply the patterns you learn here to your own projects that use your favorite programming languages and frameworks.

**Meet the team**

You met the *Space Game* web team at Tailspin Toys in previous modules. As a refresher, here's who you'll work with in this module.



Andy is the development lead.



Amita is in QA.



Tim is in operations.



Mara just joined as a developer and reports to Andy.

Mara has prior experience with DevOps. She's helping the team adopt a more automated process that uses Azure DevOps.

# What is nonfunctional testing?

* 7 minutes

In Run functional tests in Azure Pipelines , you and the Tailspin team added Selenium UI tests to the pipeline. UI tests are a form of functional testing. In this part, you explore the kinds of nonfunctional tests you can run in a pipeline.

The team first defines nonfunctional tests. They talk about some types of these tests. Then they decide on a nonfunctional test to add to their pipeline.

## Daily standup meeting

The team is having their daily standup meeting. Mara and Amita are demoing the changes they made to the release pipeline. Amita shows the UI tests running on a laptop. The team watches how quickly each web browser comes up and runs through the tests that Amita would normally do manually.

**Amita:** I must admit, I initially resisted this form of automation. But now I see how it gives me the freedom to do so much more. I'm already beginning to write my own UI tests. And I still have plenty of time to do usability testing to verify that our websites are intuitive and meet the user's needs.

**Tim:** I'm happy this automation helps speed up things for you, Amita. This improvement should help us move changes from Test to Staging more quickly. Given that improvement, is there anything we should do in Staging to help increase the quality of our releases?

**Andy:** UI tests are a form of functional testing. I suspect, Tim, that you're more concerned about performance, or the nonfunctional parts of the application. What kinds of tests do you normally run?

**Tim:** Normally, after our sites are in production, I run performance, load, and stress tests. But I'd like to start running other kinds of tests as well, such as compliance tests and security tests. All of those tests are difficult to run manually. By using automation, we can run them both earlier and more frequently.

**Mara:** Tim, you've developed a DevOps mindset! I think we can help you. First let's discuss how nonfunctional tests compare to functional tests. Then let's talk about the kinds of nonfunctional tests we can run.

## How do nonfunctional tests compare to functional tests?

In Run functional tests in Azure Pipelines , we defined functional tests and nonfunctional tests.

In short, functional tests verify that each function of the software does what it should. In other words, functional tests verify an application's functionality.

Nonfunctional tests check nonfunctional aspects of an application, such as performance and reliability. You can also do nonfunctional tests on systems that aren't apps, such as infrastructure components. One example of a nonfunctional test is to determine how many people can simultaneously sign in to an application without causing a problem, such as slower response times.

Taking the Space Game website as an example, a functional test might verify that the leaderboard appears correctly and that it shows the correct records when the user selects a filter. A nonfunctional test might verify that leaderboard filtering finishes in less than one second, even when many users connect to the website at the same time.

Nonfunctional testing always tests something that's measurable. The goal is to improve the product. You might do that, for example, by improving how efficiently the application uses resources or by improving response times when many customers use it simultaneously. Here are some of the questions that nonfunctional tests can answer:

* How does the application perform under normal circumstances?
* How does the application perform when many users sign in concurrently?
* How secure is the application?

## What kinds of nonfunctional tests can I run?

There are many kinds of nonfunctional tests. Many of them fit in the broad categories of performance testing and security testing.

### Performance testing

The goal of performance testing is to improve the speed, scalability, and stability of an application. Testing for speed determines how quickly an application responds. Testing for scalability determines the maximum user load an application can handle. Testing for stability determines whether the application remains stable under different loads. Two common types of performance tests are load tests and stress tests.

#### Load testing

Load tests determine the performance of an application under realistic loads. For example, load tests can determine how well an application performs at the upper limit of its service-level agreement (SLA). Basically, load testing determines the behavior of the application when multiple users need it at the same time.

Users aren't necessarily people. A load test for printer software, for example, might send the application large amounts of data. A load test for a mail server might simulate thousands of concurrent users.

Load testing is also a good way to uncover problems that exist only when the application is operating at its limits. That's when issues such as buffer overflow and memory leaks can surface.

In this module, you use Apache JMeter to perform load tests. You use a set of simulated users that access the website simultaneously.

#### Stress testing

Stress tests determine the stability and robustness of an application under heavy loads. The loads go beyond what's specified for the application. The stress tests determine whether the application will crash under these loads. If the application fails, the stress test checks to ensure that it fails gracefully. A graceful failure might, for example, issue an appropriate, informative error message.

Scenarios in which applications must operate under abnormally heavy loads are common. For example, in case your video goes viral, you'll want to know how well the servers can handle the extra load. Another typical scenario is high traffic on shopping websites during holiday seasons.

### Security testing

Security testing ensures that applications are free from vulnerabilities, threats, and risks. Thorough security testing finds all the possible loopholes and weaknesses of the system that might cause an information breach or a loss of revenue.

There are many types of security testing. Two of them are penetration testing and compliance testing.

#### Penetration testing

Penetration testing, or pen testing, is a type of security testing that tests the insecure areas of the application. In particular, it tests for vulnerabilities that an attacker could exploit. An authorized, simulated cyber attack is usually a part of penetration testing.

#### Compliance testing

Compliance testing determines whether an application is compliant with some set of requirements, inside or outside the company. For example, healthcare organizations usually need to comply with HIPAA (Health Insurance Portability and Accountability Act of 1996), which provides data privacy and security provisions for safeguarding medical information.

An organization might also have its own security requirements. Software must be tested to make sure that it follows these requirements. For example, on Linux systems, the default user mask must be 027 or more restrictive. A security test needs to prove that this requirement is met.

## What does the team choose?

**Andy:** Tim, nonfunctional testing is your area. If you had to choose just one type of testing to start with, which would you pick?

**Tim:** Like Amita, I want our users to have a great experience. From my perspective, that happens when the page loads quickly. Load time is especially important when we launch a new game. On launch day, gamers come to our site to download the game. Load testing helps me simulate expected usage.

**Mara:** What tool do you use for load testing, and what do your load tests measure?

**Tim:** I use Apache JMeter. I mainly look at response time, or the time between the web browser's request to our servers and when it receives data back. Under a typical load, I like to see average request times of less than one second. No more than 10 percent of requests should take more than one second.

**Mara:** I'm interested in learning more. Can we pair up sometime and try to add load testing to the pipeline?

**Tim:** Absolutely. Perhaps I can walk you through my process. Then we can map that to tasks in Azure Pipelines.

**Mara:** That sounds great! As with our functional tests, we can decide exactly where these tests fit in the pipeline. We might decide to run the tests on a Microsoft-hosted agent, an agent we provide, or some other type of test infrastructure.

## The plan

**Mara:** OK. So here's what we're going to do:

* Tim is going to show me how he runs load tests by using Apache JMeter.
* We'll map the test steps to tasks in Azure Pipelines.
* We'll measure the load on the website and provide a report for all of you.

**Tim:** Great! See you later.