## UNIT 3

## What is Unit Testing?

Unit testing is testing the smallest testable unit of an application.

It is done during the coding phase by the developers.

To perform unit testing, a developer writes a piece of code (unit tests) to verify the code to be tested (unit) is correct.

## A Real-world Example

You have written a function to add two numbers:

int Add(int a, int b) { return a+b; }

The above function takes two numbers as input and returns their sum.

A unit test code would look something like this:

void TestAdd1() { Assert.IsEqual(Add(5, 10), 15) }

The above unit test “asserts” that 5 + 10 is equal to 15. If the Add function returns anything else Assert.IsEqual result in error and the [test case](https://tuskr.app/learn/test-case) will fail.

You will probably add a few more unit test cases like these:

void TestAdd2() { Assert.IsEqual(Add(500, 1000), 1500) }

void TestAdd3() { Assert.IsEqual(Add(0, 1000), 1000) }

void TestAdd4() { Assert.IsEqual(Add(-100, 100), 0) }

void TestAdd5() { Assert.IsEqual(Add(-100, -1100), -1200) }

After you write your test cases, you will run them to verify that everything is working correctly.

Later another developer, in addition to adding some of his code, accidentally modifies the ****Add**** function as:

int Add(int a, int b) { return a\*b; }

As you can see, instead of adding the two numbers, the code now multiplies them.

When the developer runs the unit tests that you have designed for this function, they will fail. The new developer can trace the failed test cases back to the function and fix the code.

## Automation

Automating unit testing is easy. For many teams, unit testing is integrated into their daily build process to promptly report any errors. Some of the most popular frameworks are [JUnit](https://junit.org/" \t "/home/computer/Documents\\x/_blank) for Java and [Jest](https://jestjs.io/) for Javascript-based framework like Node.js.

## Benefits of Unit Testing

Unit tests can be a great investment if done correctly.

**Unit testing helps in finding bugs early.**

Developers can write unit tests as soon as they finish writing code without having to wait for others. This makes it easier for developers to identify and fix bugs as they are usually quite familiar with the code they recently worked on.

**Unit testing makes the team in the long run.**

Software development is an interactive process. Design and implementation changes are commonplace. If you have unit tests in place, developers can quickly run them and get feedback on the quality of their work.

**Unit testing makes debugging easier.**

Unit testing simplifies the debugging process. When a test fails, only the newest changes made in the code need to be checked. If you don’t have unit test cases, small changes made over the range of several weeks or months need to be inspected.

**Unit testing can be automated.**

Unit testing can be easily integrated into the software build process, making it easy to report errors quickly.

**Unit testing decreases the total testing cost.**

Since unit testing can be automated, it can be performed daily. When bugs are nipped in the bud, integration and [system testing](https://tuskr.app/learn/system-testing) also becomes more effective.

## Disadvantages of Unit Testing

Unit tests can be a lousy investment if not done correctly.

**Unit testing increases the initial development time.**

It takes time to write good test cases. Since a developer does this, it adds to the development time. In the long run, good test cases can save time, but many think of unit testing as an expense rather than an investment and do not allocate the required amount of time.

**Unit testing can lead to an unnecessary proliferation of test cases leading to increased maintenance time.**

We have often seen a hundred test cases when just twenty would have sufficed. This inflation usually happens when someone other than the developer writes the test cases, and the incentive is based on the number of test cases written rather than the quality of the test cases. When the underlying API changes, reworking the test cases takes time but it never added to the code's quality in the first place.

**Unit testing can induce false confidence in the quality of the code.**

It is vital to ensure that unit tests examined all the code paths. To do this, developers have to install a code coverage software and verify the unit tests' efficacy. But this is rare. Unit testing is more often than not considered a to-do item rather than a necessity. As a result, you may often see a lot of redundant test cases and a lot of necessary boundary condition testing missing. Build

### Advantages of Unit Testing:

1. Unit Testing allows developers to learn what functionality is provided by a unit and how to use it to gain a basic understanding of the unit API.
2. Unit testing allows the programmer to refine code and make sure the module works properly.
3. Unit testing enables testing parts of the project without waiting for others to be completed.
4. Early Detection of Issues: Unit testing allows developers to detect and fix issues early in the development process, before they become larger and more difficult to fix.
5. Improved Code Quality: Unit testing helps to ensure that each unit of code works as intended and meets the requirements, improving the overall quality of the software.
6. Increased Confidence: Unit testing provides developers with confidence in their code, as they can validate that each unit of the software is functioning as expected.
7. Faster Development: Unit testing enables developers to work faster and more efficiently, as they can validate changes to the code without having to wait for the full system to be tested.
8. Better Documentation: Unit testing provides clear and concise documentation of the code and its behavior, making it easier for other developers to understand and maintain the software.
9. Facilitation of Refactoring: Unit testing enables developers to safely make changes to the code, as they can validate that their changes do not break existing functionality.
10. Reduced Time and Cost: Unit testing can reduce the time and cost required for later testing, as it helps to identify and fix issues early in the development process.

### Disadvantages of Unit Testing:

1. The process is time-consuming for writing the unit test cases.
2. Unit Testing will not cover all the errors in the module because there is a chance of having errors in the modules while doing integration testing.
3. Unit Testing is not efficient for checking the errors in the UI(User Interface) part of the module.
4. It requires more time for maintenance when the source code is changed frequently.
5. It cannot cover the non-functional testing parameters such as scalability, the performance of the system, etc.
6. Time and Effort: Unit testing requires a significant investment of time and effort to create and maintain the test cases, especially for complex systems.
7. Dependence on Developers: The success of unit testing depends on the developers, who must write clear, concise, and comprehensive test cases to validate the code.
8. Difficulty in Testing Complex Units: Unit testing can be challenging when dealing with complex units, as it can be difficult to isolate and test individual units in isolation from the rest of the system.
9. Difficulty in Testing Interactions: Unit testing may not be sufficient for testing interactions between units, as it only focuses on individual units.
10. Difficulty in Testing User Interfaces: Unit testing may not be suitable for testing user interfaces, as it typically focuses on the functionality of individual units.
11. Over-reliance on Automation: Over-reliance on automated unit tests can lead to a false sense of security, as automated tests may not uncover all possible issues or bugs.
12. Maintenance Overhead: Unit testing requires ongoing maintenance and updates, as the code and test cases must be kept up-to-date with changes to the software.

### **What is integration testing?**

Integration testing -- also known as integration and testing (I&T) -- is a type of [software testing](https://www.techtarget.com/whatis/definition/software-testing) in which the different units, modules or components of a software application are tested as a combined entity. However, these modules may be coded by different programmers.

The aim of integration testing is to test the interfaces between the modules and expose any defects that may arise when these components are integrated and need to interact with each other.

### **Common approaches to integration testing**

Four key strategies to execute integration testing are big-bang, top-down, bottom-up and sandwich/hybrid testing. Each approach has benefits and drawbacks.

****Big-bang testing:****The big-bang approach involves integrating all modules at once and testing them all as one unit.

Big-bang testing's advantages include the following:

* Its suitability for testing small systems.
* Its ease of identifying errors in such systems, saving time and speeding up application deployment.

However, big-bang testing has disadvantages, for example:

* Locating the source of defects can be difficult since different modules are integrated as one unit.
* Big-bang testing is time-consuming for a large system with numerous units.
* Testers could miss some interface links or [bugs](https://www.techtarget.com/searchsoftwarequality/definition/bug).
* Testers must wait until all modules are available, so they have less time to do the testing and developers have less time to fix any errors.
* Due to simultaneous testing, high-risk critical modules and peripheral modules dealing with user interfaces are not tested on priority (as they should be).

****Top-down testing:****The top-down approach is an [incremental approach](https://www.techtarget.com/searchcio/definition/incremental-innovation) that involves testing from the topmost or highest-level module and gradually proceeding to the lower modules. Each module is tested one by one, and then integrated to check the final software's functionality.

Advantages of top-down testing are as follows:

* It is easier to identify defects and isolate their sources.
* Testers check important units first, so they are more likely to find critical design flaws.
* It is possible to create an early [prototype](https://www.techtarget.com/searcherp/definition/prototype).

However, disadvantages to top-down testing are as follows:

* The examination of lower-level modules can take a lot of time, so testers may not test them adequately or completely.
* When too many testing stubs are involved, the testing process can become complicated.

****Bottom-up testing:****Bottom-up (also known as bottom-to-top) integration testing is the opposite of the top-down approach. It involves testing lower-level modules first, and then gradually progressing incrementally to higher-level modules. This approach is suitable when all units are available for testing.

Advantages of bottom-up testing are as follows:

* It is easier to find and localize faults.
* Less time is needed for [troubleshooting](https://www.techtarget.com/whatis/definition/troubleshooting) since testers don't have to wait for all modules to be available for testing.

Meanwhile, disadvantages of this type of testing include the following:

* Testing all modules can take a lot of time, so there may be delays in releasing the final product.
* Critical modules are tested only in the final stages, so testers may miss some defects and developers may not have enough time to fix found defects.
* Testing can be complicated if the software consists of multiple low-level units.
* It is not possible to create an early prototype.

### **What are the key steps in integration testing?**

The steps involved in integration testing are as follows:

1. Prepare the [test plan](https://www.techtarget.com/searchsoftwarequality/answer/Test-plan-and-test-strategy).
2. Design [test cases](https://www.techtarget.com/searchsoftwarequality/definition/test-case), [test scenarios](https://www.techtarget.com/whatis/definition/scenario), [use cases](https://www.techtarget.com/searchsoftwarequality/definition/use-case) and [scripts](https://www.techtarget.com/whatis/definition/script).
3. Run tests after unit/module integration.
4. Detect, report and fix errors.
5. Retest functionalities after fixing bugs.
6. Repeat the process until all bugs are found and fixed.

### **Integration testing example**

Consider a [video-streaming](https://www.techtarget.com/searchunifiedcommunications/definition/streaming-video) mobile application. Its core features include the following:

* Sign up/log in.
* View different monthly/yearly subscription plans.
* Choose personalized plans.
* Watch streaming video.

Once users download the applications, they see a sign-up form where they can enter their account information. After successful [authorization](https://www.techtarget.com/searchsoftwarequality/definition/authorization), they are redirected to a page listing different subscription plans. They can choose their own plan and then complete the payment.

Any errors in this logical flow could cause problems for the user and lead to losses for the app company. Integration testing can help find and fix such errors.

So, after each module is ready, testers conduct unit testing. And once all modules are available, testers test them together to check their interfaces and data flows. If no errors are detected, the end user should be able to successfully complete their transaction.