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

Benefits of Unit Testing

**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