**ADVANTAGE AND DISADVANTAGE OF TDD**

**Advantages of TDD :**

**1.You only write code that’s needed –**  
 Following the principles, you’ve got to prevent writing production code when all of your tests pass.

If your project needs another feature, you would like a test to drive the implementation of the feature. The code you write is the simplest code possible.

So, all the code ending up within the product is really needed to implement the features.

**2.More modular design –**  
 In TDD, you consider one microfeature at a time. And as you write the test first, the code automatically becomes easy to check. Code that’s easy to check features a clear interface.

This leads to a modular design for your application.

**Easier to maintain –**  
 Because the different parts of your application are decoupled from one another and have clear interfaces, the code becomes easier to take care of, you’ll exchange the implementation of a microfeature with a far better implementation without affecting another module.

you’ll even keep the tests and rewrite the entire application. When all the tests pass, you’re done.

**Easier to refactor –**  
 Every feature is thoroughly tested. you do not get to be afraid to form drastic changes because if all the tests still pass, everything is ok. Now, is extremely important because you, as a developer, improve your skills each and each day.

If you open the project after six months of performing on something else, most likely, you will have many ideas on the way to improve the code.

But your memory about all the various parts and the way they fit together isn’t fresh anymore.

So, making changes is often dangerous. With an entire test suite, you’ll easily improve the code without the fear of breaking your application.

**High test coverage –**  
 There’s a test for each feature. This leads to a high test coverage It develops gain confidence in your code.

**Tests document the code –**  
 The test code shows you ways your code is supposed to be used. As such, it documents your code.

The test code is a sample code that shows what the code does and the way the interface has got to be used.

**Less debugging –**  
 How often have you ever wasted each day to seek out a nasty bug? How often have you copied a mistake message from Xcode and looked for it on the web.

**Disadvantages of TDD :**

**No silver bullet –**  
 Tests help to seek out bugs, but they can not find bugs that you simply introduce within the test code and in implementation code.

If you haven’t understood the matter you would like to unravel, writing tests most likely doesn’t help.

**slow process –**  
 If you begin TDD, you’ll get the sensation that you simply need an extended duration of your time for straightforward implementations.

You would like to believe the interfaces, write the test code, and run the tests before you’ll finally start writing the code.

**All the members of a team got to do it –**  
 As TDD influences the planning of code, it’s recommended that either all the members of a team use TDD or nobody in the least. additionally, to the present, it’s sometimes difficult to justify TDD to the management because they often have the sensation that the implementation of latest features takes longer if developers write code that will not find themselves within the product half the time.

It helps if the entire team agrees on the importance of unit tests.

**Tests got to be maintained when requirements change –**  
 Probably, the strongest argument against TDD is that the tests need to be maintained because the code has got to. Whenever requirements change, you would like to vary the code and tests.

But you’re working with TDD. this suggests that you simply got to change the tests first then make the tests pass. So, actually, this disadvantage is that the same as before when writing code that apparently takes an extended time takes a long time.