So, you know about the benefits of writing tests, but we don't live in an ideal world, and every project has certain constraints. There are a few scenarios where I suggest you to think about automated testing a bit more realistically.

One scenario is where you're dealing with a legacy application that is written without unit testing in mind. Applications built this way are not inherently testable. If you want to make them testable, you have to spend a lot of time, and your costs will potentially be more than the benefits you will get.

So what should you do? Should you test or not. That's where you need to do your assessment. You may decide to refactor a small or a critical part of your application and make that part testable. This way, we can make sure the critical parts of the application continue to work as you implement new business requirements.

But if you want to aim for testing every part of the application.

Chances are your cost will outweigh the benefits. Another scenario, is where you're working at a start-up company and you have limited time to quickly turn a concept into working software in production. In that case, again, you may not have enough time to write tests for every piece of the application.

But at least you can write tests for the key parts plus, as I told you earlier in this section, if you have calculate functions you can unit test those. And this will in fact help you go faster, and spend less time on manually testing those functions. So you need to be realistic, and use your own judgement as to what parts of the application need to be tested.

And the last scenario is where you are the only developer on the team, who practices writing clean, and testable code. If the other developers on your team write messy code, like functions with a hundred lines of code or longer, you really can't implement unit testing at your organization, it just doesn't work. Its like swimming against a river. The first thing you need to do is to educate your team and help them write cleaner and more testable code.

In case you haven't watched my clean code course, I recommend you to take that course and get everyone on your team to watch that course too. That's your first step before writing testable code. If your code is not clean, writing tests for it is hard or impossible.