When it comes to automated testing, there are two schools of thought. On one side of the spectrum, we have this super smart developer who thinks, he's an amazing coder, he writes bug free code, and he doesn't need to write tests for his code.

In fact, he thinks writing automated tests is a waste of time

because they slow you down… He argues that they delay your delivery, and every time you change your application code, you have to deal with a bunch of broken tests, and all this is costly.

Oh and by the way, this is a true story! I didn't make this up! In

fact, I heard this from one of my previous co-workers! On the other side of the spectrum, we have this highly opinionated developer who thinks you're not a developer if you don't write tests… If you go for a job interview at their company, they're gonna give you a unit testing challenge to see if you're good at writing tests or not.

They have automated tests for everything and they don't have any manual testing at their company. So who's right? Neither of them! I want you to ignore both these extreme viewpoints and take the middle ground. In every application, it is absolutely essential to have a level of automated testing. Yes, it is true that writing automated tests take time!

These tests are not automatically generated! You need to write them by hand! And you need to keep them in good shape and maintain them just like your application code. However, if you don't spend this time writing these tests, then you're likely to spend more time fixing the bugs after deploying your application. In fact, multiple studies have shown that the later we catch bugs, the more costly it gets to fix them.

So, when you have a suite of automated tests, you can catch these bugs earlier in the software development lifecycle, specifically when you're writing the code. Uncle Bob, or Robert C Martin, has a metaphor and compares unit testing to double entry book keeping. Every transaction

is entered twice; once in the credit books, and once in the debit books.

The two entries participate in very different calculations but eventually result in a final result of zero. Of course, the effort is now double but accountants believe this practice is worth the effort because it reduces the chance for errors. Uncle Bob argues that writing tests is exactly like double entry book keeping! For every piece of logic we have in our application code, somewhere we have a test that verifies that logic is implemented properly. Some people complain that they don't have time to write tests! Here's a question for you… Imagine you're a pilot. Would you fly your passengers if you're behind the schedule? Of course not! If you were a surgeon, would you rush the surgery just to make sure you finish the operation according to the schedule? Of course not! So, if a pilot can delay a flight, you can delay your delivery schedule and focus on delivering quality software with less defects.

Your boss or client may think they're losing money if you don't release this new shiny feature in time, but when you rush the delivery,and don't write your tests, you end up writing bad code with many bugs and then you have to spend more time and money fixing those bugs. So, you need to explain this to your boss or client who is rushing you to delivery! So, the reality is that building software is costly. You either choose to pay that cost upfront by writing tests, or pay a far greater cost after releasing your software.