So, the third test case we want to write here, is where our new booking starts before an ExistingBooking, but finishes after that existing booking, in this case, HaveOverlap. So once again to save time, I'm going to select this code. duplicate and modify accordingly. So BookingStartsBefore AndFinishesAfter AnExistingBooking. Okay, so now we modify the logic, starts before Existing BookingsArrivalDate, but finishes After existingBookingsDepartureDate. And our Assertion is right, so we want to make sure this reference returns the existingBooking. So let's run the test, Okay, this test failed. So we expected to get a, which was our reference but we got an empty string. So this test is telling us that there is a bug in our production code, so, back to our BookingHelper class, this logic has a bug. This is why you write unit tests.

Look, with these three unit tests, you could catch a bug before deploying this application to production. Imagine if you want to test this manually, by filling out forms, setting different dates, click, click, click, it would take you like, 20 minutes to test all various possibilities. In contrast, we can run our unit tests and catch bugs in a split of a second.

So, how should we fix this? Well, here on Google, let's search for c#, overlap date range stackoverflow. Look at the first link here, So algorithm to detect overlapping period. You scroll down, okay, here's the simple logic, to test if two dates overlap, and chances are this logic has a bug, you don't know, you're going to write all our unit tests and then we will know if this logic test is correct or not, but this logic is actually simpler than what we have here, so, I'm going to copy this and paste it here as a comment. So, there are two scenarios. One is start of A is less than end of B,

So assuming that Booking object that we pass for this method is A, we can rewrite this as Booking.ArrivalDate less than b.DepartureDate. And, the second scenario is Start of B is less than end of A. So, that is b.ArrivalDate is less than booking.Departure date, and then we can get rid of these additional conditions. Look, our code is already shorter and it's already bug free. So, let's delete this comment here. Back to our test class, I'm going to put this cursor on the class name, and run all the tests. So you can see all the tests in this class are passing, so it looks like we have fixed the bug, but we don't know yet, so next we're going to

write more tests for this class.