A question that I get asked a lot is who should write these automated tests? Is it the job of a developer or a tester? Well, it depends.

Writing unit and integration tests is the job of a software developer.

So if you're building a new feature, modifying an existing feature, or fixing a bug, before committing your code to the repository, you're responsible to cover your code with a (?) of unit and integration tests. You cannot expect someone else to write the tests for the code that you have written. This is exactly like writing a bit of code that doesn't compile, and then passing it to your coworker asking them to fix the compilation issue. In one of the teams that I worked at before, I had a team member who was lazy at writing tests for his code. There were times when I had to extend his code, perhaps to make a change or to implement a new feature.

I realized his code had no tests whatsoever, so I had to write all the tests that he should have written in the first place, and then start changing his code. If you're that kind of developer, shame on you! This just demonstrates your laziness and lack of professionalism. No one cares how good of a coder or how intelligent you are if you cannot work with others in a professional manner. So, writing unit and integration tests is your job as a software developer.

End-to-End tests are a different story. Sometimes they're written by developers, other times they're written by test engineers. So a test engineer doesn't care how you have implemented a given feature. They don't care about your classes, and methods, and parameters, they test the application at a high level, like an end user, so, they run the application and test it manually, or they can write automated tests that simulate a real user.

Either way, they don't care about the internal implementation.