From sceptic to supercharged. How AI changed my day as a QA Engineer

BY MARTIN KLIMOVSKI · SEPTEMBER 21, 2026

Want to read the latest version? Visit the link below:
https://www.intigriti.com/blog/business-insights/how-ai-changed-my-day-as-a-qa-engineer

When the push came to start weaving AI into our everyday work, I had doubts at the start. But I have since come around, and here's why.

What I actually do (and why that matters)

I'm on the engineering team at Intigriti, but my role is Quality Assurance (QA), so I'm the one writing the tests, running the tests, and then collaborating with the developers when something doesn't work the way it should. That means my day is a mix of automation, bug reporting, code reviewing, and trying to verify and break things before users do.

There is a lot of important preparation needed to do what I do. This includes setup and documentation, and that's where the use of AI started to make sense for me.

The tools I work with

My main setup right now is Claude Code, alongside a few tools built specifically for QA and software testing. I use Playwright for our test automation, and there's a strong ecosystem of open-source tools that pair well with AI. The key is that the AI becomes the driving force, and the specialist tools on top of it give it direction. Without that structure, you ask AI to do something, and it'll give you ten different approaches, none of which are quite what you had in mind. Narrowing it down, teaching it the context of our platform and our repositories. That's what makes it actually useful.

Agents that do the grunt work

The piece of my workflow I'm most pleased with right now is the bug-logging agents I've built.

Here's the challenge I was solving: writing a good bug report takes time. You need to describe what happened, what was expected, provide context, attach screenshots, follow a format. I have done this hundreds of times. So, I built an agent that already knows how a bug report should look, already has the context of our open bugs in Azure DevOps, and just needs me to give it the raw information.

The way it works is deliberately two-stage.

- First, the agent gives me a draft. I can see how it's interpreted everything, how it's described the issue, and what it's included. Then, once I'm happy, I give it the green light, and it creates the bug report directly in Azure DevOps (ADO). I still review everything. But the time I spend on each report is a fraction of what it was.

Then there's the follow-up challenge: because the agent handles so much of the logging, it’s easy to lose track of what's already been reported.

- So, I built a second agent for exactly that. I tell it I'm not sure whether something's been logged; it goes through the recent bug reports in ADO, and it comes back with a clear answer and a summary.

Simple, but it saves real time.

Writing code, but not forgetting how to think

I also use AI for writing test plans and code, which I imagine is what most developers picture when they think about AI in engineering. You give it context, user stories, acceptance criteria, and with the right tooling, it can even log into the platform, do its own research, and produce code.

But here's the thing: I still review it. And I still want to write some of it myself. Not out of stubbornness, but because it’s important to think through problems myself. Basic stuff like how to structure a method, how to think through a function. That muscle atrophies if you let it. So, I try to stay deliberate. Sometimes I tell the AI what I want to do but ask it to hold back so I can write it first. I want it as a helper, not a replacement for my own judgment.

After I have completed a piece of work, I'll often run it past another agent I've set up for peer review. It has access to all the structural rules and conventions we follow in our Playwright repository, and it'll flag if I've drifted somewhere or if something could be written more cleanly. It's essentially a sanity check before I push the code for other developers to see.

Freeing up time for the good stuff

The broader shift I've noticed is that AI has taken a lot of the prep work off my plate and given me more time to do what QA is about, which is thinking creatively about how to test and break things.

When you've already got your documentation, your test cases, your test data, you can spend your energy on the real combinations. Trying unexpected paths. Looking for edge cases nobody thought of. That's the part of this job I really enjoy, and I'm doing more of it now than I was before.

I've also started testing the AI itself. Not in a formal way, but I'll deliberately hand it a tricky bug fix or a feature with a lot of subtle comparisons, just to see how it handles it. Often, it does well. And when it doesn't, that's useful information too.

A new playing field for all

I used to find a lot of the repetitive prep work draining. Now, because I've got an “assistant” handling that layer, my whole relationship with those tasks has changed. It's a new perspective. I don’t mind them anymore, because I'm not the one grinding through them.

That said, and I want to be clear about this, I review everything the AI produces. That's not optional for me. It's part of the workflow, not an afterthought.

The benefits that go both ways

Some of the features recently shipped, things like AI-assisted draft submissions and triage support, I think show the same pattern on a bigger scale.

It's useful for us on the engineering and QA side, and it's useful for the people using the platform. Everyone's finding their own use case for it, their own entry point into the workflow.

Was there fear that AI would completely change the game? For me, yes. But it's turned out to be more of an easing than an overturning. The pace of change is fast, new models keep coming, costs keep shifting,

and finding the right balance between using it to the fullest and not spending a fortune on it is something everyone's still figuring out.

What I am watching out for

AI is getting very good at writing code on its own. Give a model a large, well-documented codebase, and it can produce something close to production-ready. With every new generation of LLMs, the jump in capability isn't small. I saw it firsthand just the other day when a PR that was essentially good out of the box needed only minor tweaks. Imagine where that's at in a year or two.

The other side of the coin, and something I must voice, is that not everything needs to have AI slapped onto it. I've seen a screen protector for a phone marketed as AI-enhanced. That's not innovation; that's a sticker. If there's a genuine problem that AI makes meaningfully easier to solve, explore it properly, do the engineering, do the research, run the experiments. But don't just jump on the bandwagon because everyone else is.

The next frontier of security

One area I'm genuinely curious about and haven't had as much time to dig into as I'd like is security testing for AI itself. Last year I was at a conference with a colleague where we got a look at how you actually test LLMs: how you probe their limits, try to bypass their guardrails, and find the breaking points. It was fascinating.

If everything is increasingly AI-powered, then AI becomes the thing most likely to be attacked. For QAs and security testers, that’s a new surface area to get skilled in. I want to go back to what I saw at that workshop and see how much has shifted in the year since. It feels like the natural next frontier.

My summary

Am I still learning? Absolutely. Am I still tinkering with my agents and workflows? Every week. But the trajectory is clear. AI has made me more efficient and given me more space to do the parts that actually require a human brain. It's an enabler. A big one. And I think the engineers, testers, and designers who figure out how to use it well, without outsourcing their thinking entirely, are going to be in a genuinely good place.

The future isn't AI doing your job. It's you doing your job better, with AI in your corner.

Steps for security teams

Visit AI Security and Safety for more information.

Have a question about the content above? Contact the team.

AUTHOR

Martin Klimovski

Martin Klimovski is a Skopje-based Senior QA Engineer with 8+ years of experience in the software industry. He specializes in both manual and automation testing, with strong expertise in tools such as Playwright, Selenium, Postman, and k6. He focuses on building efficient QA processes, improving software quality, and integrating automation into modern development workflows. Alongside his professional work, he is also passionate about mentoring and teaching software testing, helping others grow in the QA field.