Ticket writing guidelines

> *“A well written ticket can eliminate confusion & saves development time”*

Why should we care about writing well written tickets?

(If you already know, then skip to the guidelines here)

Sometimes writing tickets can seem a bit of a chore when all we want to do is write a one-liner, giving a brief outline of the issue. This can actually end up being counter-productive and add more time to the whole process of getting your ticket into that ‘done’ column. A few of the benefits of a well written/spec’d out ticket include:

You have a clear understanding of the work involved

Quite often when starting work on a ticket, it’s tempting to jump straight in. Before long you realise that X won’t work because it’s going to need to integrate with Y, or another team members input will be needed (who just so happens to be off today).

When you spec out a ticket, it is like writing a mini plan for the work you are going to do and means less chance of unexpected complexity and delays.

It means your PR is more likely to be reviewed!

(And who *doesn't* want this?)

Nobody likes to be presented with a PR with 45 files changed and hardly any backstory to what is going on. When you have an associated ticket that clearly describes the problem or new feature, rather than a one liner description, it takes some of the ambiguity away, meaning that people are more likely to review it. Bonus points if you include a user story in your acceptance criteria!

It prevents confusion and saves time

There's nothing worse than having to double check details with stakeholders and team members, especially when you are halfway through the work. If you have spec'd out your ticket, then this is less likely to happen.

It helps other devs/team members

We can't predict the future and we never know when a ticket we have written will be reprioritised to the backlog, or if we will be ill.

A ticket we write may end up in the hands of another dev and they don't really want to spend their time trying to decrypt what exactly the problem is or even if it exists anymore!

Would another developer be able to pick your ticket up and have a good understanding of what it is all about?

It helps future you by acting as documentation

Picture this: You or somebody else finds a bug. You do a bit of investigation and then decide that it isn't a high enough priority for this sprint, so into the backlog it goes. Two sprints later and you go to pick it up and the description reads *'Xyz is not working anymore. Fix.'*

You remember investigating it, you remember finding the issue in some class somewhere…but you can't. quite. remember. what…

Bonus points: It makes your boss happy

A descriptive ticket gives squad leads/product owners more insight into what is happening when they can read the ticket like a story, and also helps when deciding what to bring in from the backlog in planning.

Enough! How do I write these tickets?

These are just guidelines & you shouldn't feel like you have put stuff in for the sake of it!

The main thing is that the ticket is written in language that is clear and concise to a wide audience (a whole range of people could be looking at your ticket - devs, designers, UX, QA, product owners etc.) and is descriptive enough for somebody else to pick up.

Title

This should be a high level overview of what your ticket is trying to achieve.

Why

If it’s a bug or other issue, include a description of the problem and steps to reproduce it.

If it’s a feature, why is it necessary, and what problem is it solving. You could include a user story here if that would help.

What

This is the place to really ‘spec’ out your ticket. Get down as many details as you can about how you are going to achieve what it says in the title!

If it’s a new feature, here you can list any external dependencies/other people that may potentially need to be involved in the ticket, and any problems that you may think will arise. Try to avoid just saying something like ‘build xyz feature’.

Acceptance

In short, the acceptance criteria for this PR being merged in!

This is quite a good place for a user story, especially if the ticket is front-end or ‘user facing’. Maybe even multiple user stories if there is more than one scenario.

There should be clear steps on how to run through the scenario(s). This is particularly important as the code needs to be smoke tested – it doesn’t matter how well the code is written, you just can’t predict how it will work in the hands of a user.

Don’t forget to include any details the reviewer might need, login details etc.

This is also a good opportunity for somebody who hasn’t been immersed in the problem to find new ways of breaking things!

An example of a scenario:

As a lender with an existing product

- I login
- I visit ‘some page’
- I click ‘some button’
- I should see ‘some information’