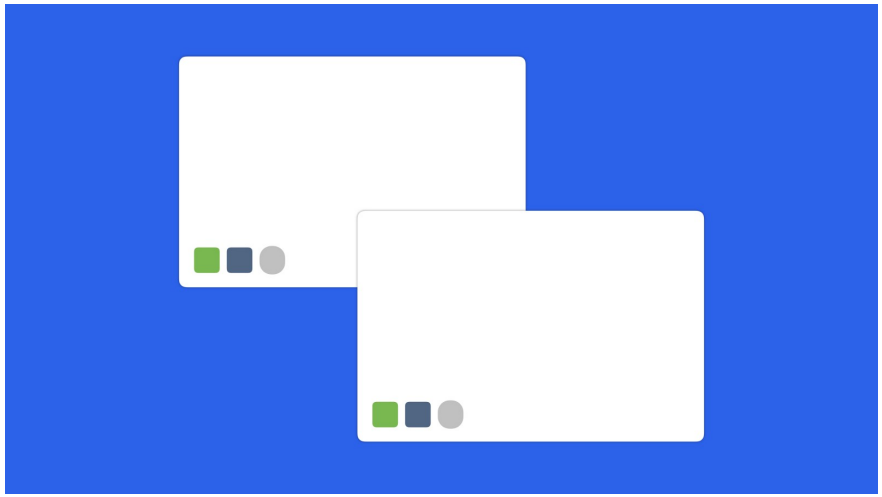
How to Write Good User Stories in Agile Software Development

Teagan Harbridge Follow
Mar 15, 2018 · 4 min read



For many software development teams striving towards agile, the idea of writing user stories can seem like another “thing” agile piles on top of their already busy workloads. We hear you, we’re busy too!

But if you’re reading this blog post, it means you definitely have some time to spare to write user stories. And good ones at that!

Let’s start off by looking at what a user story is, and isn’t.

> *A user story helps agile software development teams capture simplified, high-level descriptions of a user's requirements written from that end user's perspective. A user story is not a contextless feature, written is "dev" speak.*

. . .

How do we write user stories?

A user story often follows the following 'equation':

As a <type of user>, I want <some feature> so that <some reason>

Let's break this down one step further;

As a <type of user>—this is the WHO. Who are we building this for? Who is the user?
I want <some feature>—this is the WHAT. What are we building? What is the intention?
so that <some reason>—this is they WHY. Why are we building it? What is the value for the customer?

. . .

Let's look at a few simple examples;

As an internet banking customer
I want to see a rolling balance for my everyday accounts

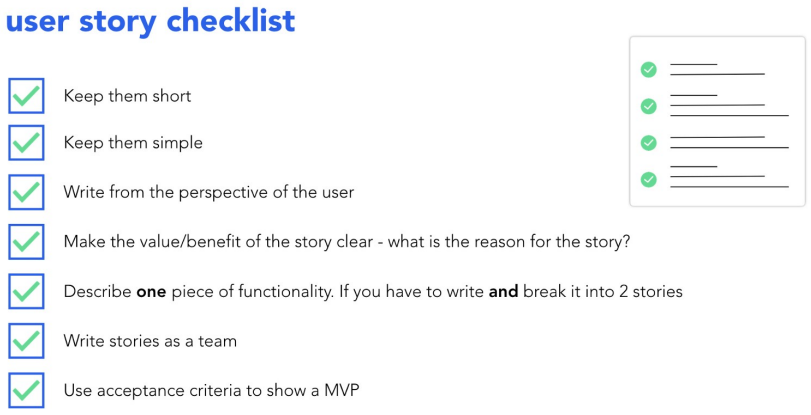
So that I can keep track of my spending after each transaction is applied

OR

As an administrator
I want to be able to create other administrators for certain projects
So that I can delegate tasks more efficiently

. . .

Following this equation, teams should make sure that their user stories are ticking all of the following checkboxes:



. . .

"Why can't we just write features or tasks instead?"

And it's a great question. Though most teams asking this question, usually don't understand the value of writing user stories, and the fact that they serve very different purposes to that of features.

user stories	tasks
a user story = the WHAT	the task = the HOW
user stories describe a piece of functionality from the point of view of the user	"what are the activities we need to perform in order to deliver outcomes (user stories)"
divided features into business processes	tasks are individual pieces of work

The fact is, it's easy to get buried in a contextless, feature developing cycle. The objective becomes more about clearing your way through a laundry list backlog, than it is about building solutions that add value to your customers. Your human customers. User stories bring that context and perspective into the development cycle.

. . .

Acceptance Criteria

User stories allow teams to have one hand on the needs, wants and values of their customers, and another, on the activities they need to accomplish to provide that value.

The link pairing these two things together, is acceptance criteria.

> *Acceptance Criteria or 'conditions of satisfaction', provide a detailed scope of a user's requirements. They help the team to understand the value of the story and set expectations as to when a team should consider something done.*

Acceptance Criteria Goals

- to clarify what the team should build before they start work
- to ensure everyone has a common understanding of the problem/need of the customer
- to help team members know when the story is complete
- to help verify the story via automated tests

Let's look at an example of a completed user story with acceptance criteria;

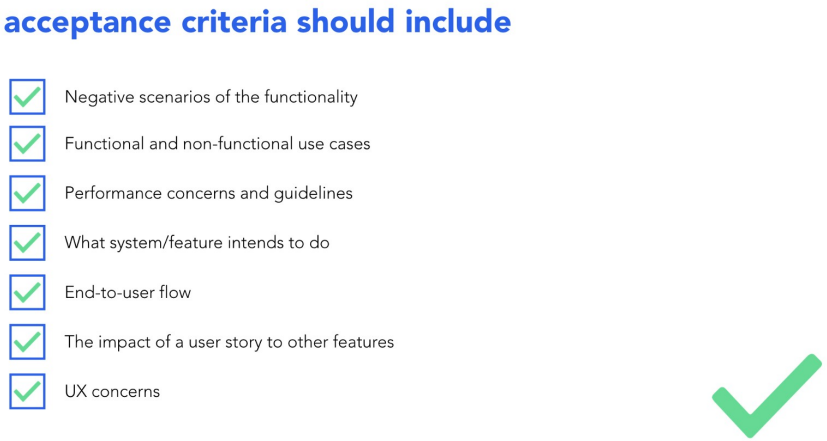
*| **As a** potential conference attendee, **I want to** be able to register for the conference online, **so that** registration is simple and paperless.*

Acceptance Criteria:

- Conference Attendance Form
- A user cannot submit a form without filling out all of the mandatory fields (First Name, Last Name, Company Name, Email Address, Position Title, Billing Information)
- Information from the form is stored in the registrations database
- Protection against spam is working

- Payment can be made via Paypal, Debit or Credit Cards
- An acknowledgement email is sent to the attendee after submitting the form

With this in mind, teams should make sure that their acceptance criteria is ticking all of the following boxes;



Acceptance criteria should **NOT** include the following:

- Code review was done
- Non-blocker or major issues
- Performance testing performed
- Acceptance and functional testing done

Why?

Your acceptance criteria should not include any of the above, because your team should already have a clear understanding of what your Definition of Done (DoD) includes, for instance:

- unit/integrated testing
- ready for acceptance test
- deployed on demo server
- releasable

. . .

Looking to get your team started writing user stories, but not sure where to start?