Published on *Rally Help* (https://help.rallydev.com)

Write a Great User Story

What is a user story [1]?

A user story represents a small piece of business value that a team can deliver in an iteration [2]. While traditional requirements (like use cases) try to be as detailed as possible, a user story is defined incrementally, in three stages:

- The brief description of the need
- The conversations that happen during backlog grooming [3] and iteration planning to nail down the details
- The tests that confirm the story's satisfactory completion

Well-formed stories will meet the criteria of Bill Wake's INVEST acronym:

Independent	We want to be able to develop in any sequence.
Negotiable	Avoid too much detail; keep them flexible so the team can adjust how much of the story to implement.
Valuable	Users or customers get some value from the story.
Estimatable	The team must be able to use them for planning.
Small	Large stories are harder to estimate and plan. By the time of iteration planning, the story should be able to be designed, coded, and tested within the iteration.
Testable	Document acceptance criteria [4], or the definition of done [5] for the story, which lead to test cases.

Why use user stories?

- Keep yourself expressing business value
- Avoid introducing detail too early that would prevent design options and inappropriately lock developers into one solution
- Avoid the appearance of false completeness and clarity
- Get to small enough chunks that invite negotiation and movement in the backlog [6]

- Leave the technical functions to the architect, developers, testers, etc.

How do I write user stories?

When you're getting started with stories, a template can help ensure that you don't inadvertently start writing technical tasks:

As a <user type>, I want to <function> so that <benefit> .

Examples:

- As a delivery team member, I want to know which tasks I own so that I can decide what to work on now.
- As a developer, I want to know which of my stories have failing test cases so that I can fix the code.
- As a product owner [7], I want to be able to drag-and-drop prioritize all the product backlog [8] items, so that I can easily adjust priorities based on changing needs.

Try to avoid the generic role "User" when writing user stories. User stories are about all of the "actors" who interact with the system or who realize some value or benefit from the system. Not all actors are end users. For example, an actor [9] could be another system or someone who wants certain functionality in order to buy your product but will never actually use the product. It may be useful to create aggregate actors (e.g. "consumer") and specialized actors (e.g. "browser" or "frequent shopper").

In Rally, this template should be entered at the top of the Description field [10]. This sets the tone for the details and acceptance [11] criteria, which will be entered below.

What size should a user story be?

A story should be small enough to be coded and tested within an iteration- ideally just a few days. When a story is too large, it is called an "epic". Backlog items tend to start as epics, when they are lower priority. For release [12] planning, epics should be broken down into smaller chunks, but not so small that you've moved into detailed design.

How detailed should a user story be?

Too broad

- "A team member can view iteration status."

Too detailed

- "A team member can view a table of stories with rank [13], name, size, package [14], owner, and status."
- "A team member can click a red button to expand the table to include detail, which lists all the tasks, with rank, name, estimate, owner, status."

Just right

- "A team member can view the iteration's stories and their status, with main fields."
- "A team member can view the current burndown [15] chart on the status page [16], and

can click it for a larger view."
- "A team member can view or hide the tasks under the stories."
- "A team member can edit a task [17] from the iteration status page."

When do I add detail?

Acceptance criteria provide the 'Definition of Done' for the story. As details about the story evolve, capture the critical ones as acceptance criteria. The product owner should list as many acceptance criteria as possible in order to clarify the intent of the story. Regardless of how detailed the acceptance criteria are, the team should have a conversation about them and adjust the acceptance criteria to capture the results of the discussion. Once an iteration has begun, testers can formalize acceptance criteria into acceptance tests.

In Rally, place the acceptance criteria directly beneath the value statement in the Description field. Delivery team members have a single location to see all of the story info with this method. Tip: by using bullet points [18], you can keep each criteria item brief and clear.

Who uses user stories?

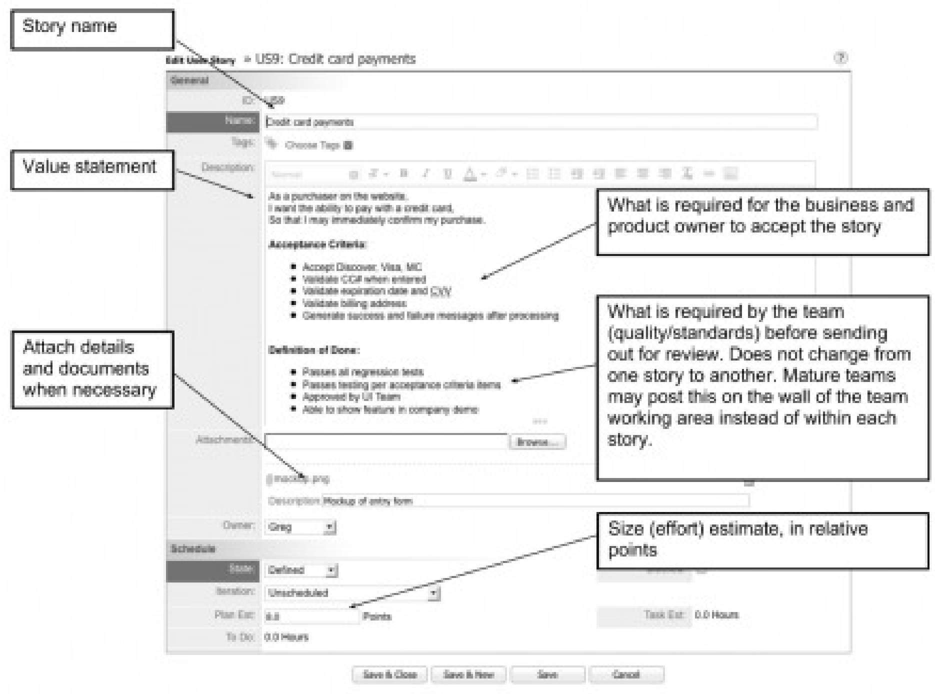
Creation — The customer [19], customer proxy, product owner and anyone else who identifies a need for the product can contribute user stories.
Ownership & Maintenance — The product owner owns the user stories and is responsible for writing, gathering, maintaining, and prioritizing.
Usage — Developers, testers, technical writers use user stories to be able to know what to implement and when they're done. Product owners track overall progress based on the status of the user stories. Management tends to track user stories rolled up to epics or features.

What are the top 3 mistakes that people make?

1. *Too formal / too much detail.* Product owners with good intentions often try to write extremely detailed user stories. If a team sees a story at iteration planning that looks like an IEEE requirements document, they'll often assume that all the details are there and they'll skip the detailed conversation.
2. *Technical tasks masquerading as stories.* A lot of the power of Agile comes from having a working increment [20] of software at the end of each iteration. If your stories are really just technical tasks, you often don't end up with working software at the end of each iteration, and you lose flexibility in prioritization.
3. *Skipping the conversation.* Stories are intentionally vague before iteration planning. If you skip the acceptance criteria conversation, you'll risk [21] moving in the wrong direction, missing edge cases or overlooking customer needs.

Example

Below is a sample user story in Rally. This story has been discussed by the team in several grooming and planning meetings as it moved up the backlog, and is close to being scheduled into an upcoming iteration. The purpose of this user story is to provide purchasers on an online site the ability to use a credit card [22] for payment.


Story name

Value statement

Attach details and documents when necessary

What is required for the business and product owner to accept the story

What is required by the team (quality/standards) before sending out for review. Does not change from one story to another. Mature teams may post this on the wall of the team working area instead of within each story.

Size (effort) estimate, in relative points

Edit User Story » US9: Credit card payments

General

ID: US9

Name: Credit card payments

Tags: Choose Tags

Description:

As a purchaser on the website,
I want the ability to pay with a credit card,
So that I may immediately confirm my purchase.

Acceptance Criteria:

- Accept Discover, Visa, MC
- Validate CC# when entered
- Validate expiration date and CVV
- Validate billing address
- Generate success and failure messages after processing

Definition of Done:

- Passes all regression tests
- Passes testing per acceptance criteria items
- Approved by UI Team
- Able to show feature in company demo

Attachments:

mockup.png

Description: Mockup of entry form

Owner: Greg

Schedule

State: Defined

Iteration: Unscheduled

Plan Est: 8.0 Points

Task Est: 0.0 Hours

To Do: 0.0 Hours

[23]

Source URL: https://help.rallydev.com/writing-great-user-story

Links:
[1] https://help.rallydev.com/glossary#user_story
[2] https://help.rallydev.com/glossary#iteration
[3] https://help.rallydev.com/glossary#backlog_grooming
[4] https://help.rallydev.com/glossary#acceptance_criteria
[5] https://help.rallydev.com/glossary#definition_of_done
[6] https://help.rallydev.com/glossary#backlog
[7] https://help.rallydev.com/glossary#product_owner
[8] https://help.rallydev.com/glossary#product_backlog
[9] https://help.rallydev.com/glossary#actor
[10] https://help.rallydev.com/glossary#field
[11] https://help.rallydev.com/glossary#acceptance
[12] https://help.rallydev.com/glossary#release
[13] https://help.rallydev.com/glossary#rank
[14] https://help.rallydev.com/glossary#package
[15] https://help.rallydev.com/glossary#burndown
[16] https://help.rallydev.com/glossary#page
[17] https://help.rallydev.com/glossary#task
[18] https://help.rallydev.com/glossary#points
[19] https://help.rallydev.com/glossary#customer
[20] https://help.rallydev.com/glossary#increment
[21] https://help.rallydev.com/glossary#risk
[22] https://help.rallydev.com/glossary#card
[23] https://help.rallydev.com/sites/default/files/multimedia/user_story_callouts.png