How to Write a Great User Story

Formats: Markdown (canonical) | Word (DOCX) | PDF **Updated:** 2026-03-22 **License:** CC BY 4.0 -- Kate Makrigiannis / k8mak.com

A 4-step quick start guide for writing user stories that are small, vertical, and focused on what real people need -- not what the feature list says.

When to use this

You have a feature or requirement and need to turn it into a story your team can build, test, and ship in 1-3 days. This guide walks you through the structure, the thinking, and the common traps. Works whether you're writing your first story or coaching someone who is.

Process

Step 1: Split features into user-focused story titles

Features are big. Stories are small. Start by breaking the feature into pieces a person would actually use.

The rule: A story title names what the user does, not what the system does. "Blood glucose entry" is a feature. "Log my morning blood glucose reading" is a story.

How to split:

- Think about one user doing one thing to get one outcome
- Each story should be a vertical slice -- touching the full stack from UI to data, delivering a complete piece of value
- If the title reads like a technical task ("Add API endpoint for glucose data"), rewrite it from the user's perspective

Watch for these traps:

- Stories split by technical layer (frontend story + backend story) are not vertical slices. They deliver nothing on their own.
- Stories that start with "As a developer..." are usually tasks, not stories. Reframe around the person who benefits.

Step 2: Define the outcome and purpose

Use the standard format to connect what the user wants with why they want it.

```
As a [specific user or persona],
I want to [concrete action],
So that [real outcome or benefit].
```

What makes this work:

- **"As a"** forces you to name who benefits. "As a user" is too vague. "As a newly diagnosed diabetic" tells the team exactly whose problem they're solving.
- **"I want to"** describes the action, not the feature. The user wants to "see my glucose trend over 7 days," not "access the glucose analytics module."
- **"So that"** names the real outcome. This is where engineers learn why the thing matters. "So that I can spot patterns and adjust my diet" gives the team context they'll use when making implementation tradeoffs.

Quick test: Read the story out loud. If it sounds like something a person would actually say, you're on track. If it sounds like a requirements document, rewrite it.

Step 3: Clarify acceptance criteria with Given/When/Then

Acceptance criteria define "done." They tell the team exactly what to build and what to test.

```
Given [a specific starting condition],
When [the user takes an action],
Then [the expected result happens].
```

Example -- HealthAlly blood glucose logging:

```
Given I am a logged-in HealthAlly user on the glucose tracking screen,
When I enter my blood glucose reading and tap "Save,"
Then my reading is stored with the current date and time,
And I see a confirmation that my reading was saved,
And the reading appears in my glucose history.
```

Rules for good acceptance criteria:

- Each scenario covers one path (happy path, error path, edge case -- separate scenarios for each)
- Use concrete values, not abstractions. "Enter 120 mg/dL" is better than "enter a valid reading."
- If you're writing more than 5-6 scenarios, the story is too big. Split it.

- Complex acceptance criteria is a signal, not a badge of honor. It means the story needs to be smaller.

Step 4: Wrap it up -- review with your team

A story is not done when you write it. It's done when the team understands it.

The review checklist:

- Walk through the story with a designer and an engineer before it goes into the backlog
- Ask the engineer: "Is this buildable in 1-3 days?" If not, split it.
- Ask the designer: "Does this cover the user's real workflow?" If not, revisit the acceptance criteria.
- Check for hidden assumptions. "The user is logged in" is an assumption. Name it.

Worked example

Feature: Blood glucose tracking in the HealthAlly diabetes management app

Story:

```
Title: Log my morning blood glucose reading

As a newly diagnosed diabetic using HealthAlly,
I want to log my morning fasting blood glucose reading,
So that I can track my levels over time and share them with my care team.
```

Acceptance criteria:

```
Scenario 1: Successful glucose entry
Given I am on the HealthAlly glucose tracking screen,
When I enter "95" in the glucose field and tap "Save,"
Then my reading is saved with today's date and time,
And I see "Reading saved" confirmation,
And "95 mg/dL" appears at the top of my glucose history.

Scenario 2: Missing glucose value
Given I am on the glucose tracking screen,
When I tap "Save" without entering a value,
Then I see "Please enter your glucose reading" below the input field,
And no reading is saved.

Scenario 3: Out-of-range value
Given I am on the glucose tracking screen,
When I enter "850" (above clinical range),
Then I see "This reading seems unusually high. Please double-check and re-enter."
And no reading is saved until confirmed.
```

Facilitator tips

If you're coaching a team on story writing:

- **Start with the title.** Most story problems are title problems. If the title describes a feature, the story will be too big.
- **Read stories out loud.** If the "As a / I want / So that" sounds robotic, the team hasn't found the real user need yet.
- **Use the AC complexity test.** Count the acceptance criteria scenarios. More than 5-6? The story is too large. Split it and try again.
- **Resist the urge to over-specify.** Stories are a conversation starter, not a contract. Leave room for the team to problem-solve during implementation.
- **Engineers who understand user nuance build better software.** The "So that" clause is not ceremony -- it's the information that helps developers make good tradeoff decisions.

How did it go?

After writing your stories, check:

- [ ] Each story title names what the user does, not what the system does
- [ ] "As a" names a specific person, not "a user"
- [ ] "So that" describes a real outcome, not a system behavior
- [ ] Acceptance criteria use Given/When/Then with concrete values
- [ ] Each story is completable in 1-3 days
- [ ] No stories are split by technical layer (all are vertical slices)
- [ ] A designer and an engineer have reviewed each story before it enters the backlog

Part of the k8 Agent Toolkit. Download other formats at k8mak.com/resources.