

Product Backlog Starter

Describe one backlog item, discuss its value and decide what happens next. This is an item worksheet to help you start, rather than a complete backlog management system.

Start with the problem

A product backlog is an ordered list of work that could improve a product. Start with the person who needs help and the problem they face. A proposed feature is one possible response, so check the need before committing to a solution.

Complete one item together

- Describe the need. Who is affected, what are they trying to do and where do they get stuck?
- Record the evidence. Use feedback, observation or data. Label assumptions that still need checking.
- State the outcome. Explain the user benefit and its connection to the Product Goal, the longer-term objective for the product in Scrum.
- Write acceptance criteria. These are observable conditions for this particular item to be accepted.
- Discuss order and uncertainty. Consider value, urgency, risk, dependencies and effort. Explain why this item belongs ahead of others.

Keep the detail proportionate

Use the blank page for one item, then copy it as needed. Add more detail as an item moves closer to delivery. A user story is an optional way to express a need: As a [person], I want [capability], so that [benefit].

In Scrum, the Product Owner is accountable for backlog management and ordering; Developers size the work. An Increment is a usable addition to the product. The Definition of Done sets its quality standard. Item-specific acceptance criteria do not replace it. User stories and this worksheet are optional practices, not Scrum requirements.

Next step: bring one completed item to a conversation with the people who would deliver it.

Scrum reference: scrumguides.org/scrum-guide.html

A completed backlog item

Fictional example: a learning website wants to help readers return to a useful guide. This is an illustration, not a claim about Coded Female readers or site functionality.

Item and user need

Item: Save a guide for later.

User: a reader learning alongside a full-time job.

Problem: the reader cannot easily find a guide again after leaving the website.

Evidence and assumptions

Illustrative evidence: 4 of 6 interview participants described losing useful links. Assumption to check: an on-site save option would be more useful than their existing bookmarks. Speak to more readers before choosing the solution.

Outcome and Product Goal

Outcome: readers can return to a saved guide without searching again.

Product Goal: make the website a useful place for readers to continue their learning. The save option is one possible contribution to that goal.

Acceptance criteria

- A signed-in reader can save a guide from its article page.
- The saved guide appears in that reader's saved list, with its title and a working link.
- Saving the same guide twice does not create a duplicate.
- The reader can remove the guide from the list.

Order and delivery questions

Proposed order: after reliable sign-in is available. Reason: the saved list depends on an account, and demand still needs checking.

Questions: do readers want an account? What happens when a guide is removed? Can the save control be used with a keyboard?

Next decision

The Product Owner and team will review reader feedback and delivery effort before deciding whether to build the save option. A prototype could help test the idea first.

Your backlog item

Complete one page per item. Keep unknowns visible and use the notes to support a conversation, rather than treating every field as a gate.

Item title and date

User and problem

Evidence and assumptions

Intended outcome and connection to the Product Goal

Acceptance criteria

Proposed order and reason

Dependencies and open questions

Next action and person responsible

Sprint Planning Checklist

Prepare for Sprint Planning with the whole Scrum Team: Product Owner, Scrum Master and Developers. In Scrum, a Sprint is a fixed period of one month or less.

Before planning

- The Product Owner brings the most important backlog items and explains how they support the Product Goal.
- Check availability, leave, support work and other commitments that affect capacity.
- Make dependencies, unanswered questions and known risks visible.
- Have the team's Definition of Done available so quality is part of the discussion.

During planning

- The whole Scrum Team agrees a Sprint Goal that explains why this Sprint is valuable and supports the Product Goal.
- Discuss what can be completed. In Scrum, Developers select the work through discussion with the Product Owner.
- Plan how to deliver it. Developers decide the approach and create the delivery plan.
- Invite concerns before agreeing the forecast. Ask: What are we assuming, and what might we need to change?

Before finishing

- Read back the goal, selected items and delivery plan. Together these form the Sprint Backlog.
- Check that everyone can find the plan and knows how to raise a blocker.
- Record follow-up actions and who will take them. Make space for quieter voices to contribute.

The Sprint Goal is the commitment. The selected work is a forecast that can be adapted as the team learns. Avoid turning capacity or estimates into individual performance targets.

Planning lasts at most eight hours for a one-month Sprint; it is usually shorter for shorter Sprints. Developers in Scrum means the people creating the usable product, not only programmers. The practical prompts here support planning; Scrum does not require this checklist.

Scrum reference: scrumguides.org/scrum-guide.html

A completed planning record

Fictional example: a small team is improving workshop registration on a learning website. The plan below illustrates a discussion, rather than a prescribed amount of work.

Product Goal and Sprint Goal

Product Goal: make workshop booking straightforward and reliable.

Sprint Goal: help visitors register and receive clear confirmation that their place is booked.

Capacity and commitments

Two-week sprint. One developer has two days of leave, and the team has ongoing support duties. The team chooses a smaller forecast based on this availability and its recent experience.

Selected work and approach

- Build the registration form with understandable validation messages.
- Show a confirmation page and send the booking email.
- Test the full registration journey, including keyboard use and failure cases.

The team will first check the email service, then deliver a small version of the complete journey. Testing and review are included in the plan.

Quality and dependencies

Use the team's existing Definition of Done. Confirm workshop details before publishing. The email service is a dependency; investigate its test environment at the start of the sprint.

Risk and follow-up

Risk: delayed email access could affect confirmation testing.

Action: the developer investigating email will check access on day one and raise a blocker immediately if needed.

Work left out

Calendar invitations and a redesigned events page are left for later. If the plan becomes unrealistic, the team will discuss scope with the Product Owner while protecting the Sprint Goal.

Your planning record

Complete this with the team. Keep the delivery plan in the place you normally work and use this page to capture the key decisions.

Sprint dates and Scrum Team participants

Sprint Goal and connection to the Product Goal

Availability and other commitments

Selected items and link to the delivery plan

Quality checks and Definition of Done

Dependencies and risks

Follow-up actions and people responsible

Work left out and reason

Product Metrics Planner

Choose a measure that helps you make a decision. Use this planner before building a feature or starting an improvement, then return to it when you review the result.

Begin with the outcome

A product metric is a measure of how a product is being used or performing. Start with the change you want for users, then choose a measure that would provide evidence of that change.

Define a useful measure

- Name the decision. What will you do differently when you have the result?
- Define the calculation. Specify the unit, such as people or attempts, who is included, exclusions and the time window. For a rate, define both parts of the fraction.
- Record a baseline. This is the starting value measured before the change. If it is unknown, plan how to establish it first.
- Choose a target and review date. Explain why the target is useful and achievable; treat it as a hypothesis to check.
- Add a guardrail. This is a second measure that helps reveal unwanted effects, such as more support requests or slower page loading.

Read the result with care

Look at observation counts as well as percentages. A minimum count alone does not establish statistical reliability. Check missing data, tracking changes and user groups. A before-and-after increase does not prove causation; other changes may explain it.

Combine the measure with reader or customer feedback. Use the findings to decide whether to continue, change the approach or investigate further. Activity counts such as visits can help, but they do not automatically show that users achieved their goal.

Next step: complete one plan for one outcome. Agree who will check the data and bring the result to the review.

A completed metrics plan

Fictional example: a learning website wants more people to finish workshop registration. All figures and thresholds are illustrative.

Outcome and decision

Outcome: visitors who start registration can successfully book a place.

Decision: keep the simpler form, revise it or investigate why visitors leave.

Primary measure

Completion rate = attempts completed within 24 hours of their own start ÷ eligible attempts started in the period × 100. Use a unique attempt ID, with one start and at most one completion per attempt. Exclude internal tests and identified bots consistently. This measures attempts, not unique people.

Baseline and target

Baseline: 120 completions from 200 eligible starts = 60%, measured over four weeks before release.

Target: at least 70% over four weeks after release. The target and an initial review threshold of 200 attempts are illustrative choices, not benchmarks or proof of statistical reliability.

Guardrail

Unique registration-help requests received within 24 hours of an eligible attempt's start ÷ eligible starts × 100. Link each request to an attempt ID; count each support ticket once, excluding duplicate tickets, internal tests and unrelated enquiries. Baseline: $6 \div 200 \times 100 = 3$ requests per 100 starts; aim for no increase.

Data and review

Source: form events and linked support tickets. Responsible person: product analyst. Use the same definitions in both periods and check tracking before release. Review four weeks after release, plus 24 hours so the final attempts have their full window. If linking requests is unreliable, revise the guardrail before using it.

How to use the finding

Consider keeping the form if the measures and feedback are encouraging. Investigate increased support requests. If evidence is limited, agree a further review. Check traffic sources and workshop demand; this comparison alone cannot show the form caused a change.

Conversion windows: amplitude.com/docs/analytics/charts/funnel-analysis/faq

Your metrics plan

Use one page per outcome. Define the measure before collecting data so the team can interpret the result consistently.

User outcome and decision to inform

Calculation including unit exclusions and time window

Baseline and measurement period

Target and reason including evidence needed

Guardrail definition and threshold

Data source and person responsible

Review date and data quality checks

Result limitations and next decision
