

PRODUCT MANAGEMENT FRAMEWORK

The MoSCoW Prioritization Framework

Must have · Should have · Could have · Won't have

A Practical Guide to Scope Prioritization & Release Planning

Prepared by	Harsh - Product Manager
Use case	Scope prioritization & release planning
Category	Product Strategy - Prioritization Frameworks

Table of Contents

1. What Is the MoSCoW Framework?	3
2. The Four Priority Categories	3
3. How the Categories Work Together	5
4. Worked Example	5
5. How to Apply MoSCoW - Step by Step	6
6. When to Use MoSCoW - and Its Limitations	7
7. Reusable Scoring Template	7

1. What Is the MoSCoW Framework?

MoSCoW is a scope-prioritization technique that sorts requirements, features, or backlog items into four buckets: Must have, Should have, Could have, and Won't have (this time) based on how critical each one is to a specific release or delivery deadline. It originated in the DSDM (Dynamic Systems Development Method) agile community and is widely used wherever scope has to be negotiated against a fixed date or fixed budget, such as regulatory launches, platform migrations, or contractual deliverables.

Unlike scoring models such as RICE, MoSCoW does not produce a numeric rank. It produces a categorical decision that is easy to communicate to non-technical stakeholders and easy to revisit in a scope-cut conversation: everything in a category is treated the same way when trade-offs have to be made.

Why PMs use it

- Gives stakeholders a simple, shared vocabulary for scope conversations no scoring math required.
- Makes it easy to protect a release date by cutting from the bottom of the list first.
- Forces an explicit, documented “not this time” instead of scope creeping in silently.
- Works well alongside RICE: RICE ranks the backlog, MoSCoW draws the release line.

2. The Four Priority Categories

Each item in scope is assigned to exactly one category. The categories are ordered by non-negotiability, not by size or effort: a Must have can be small, and a Won't have can be someone's favorite idea.

M - Must Have

Non-negotiable. The release fails without it.

- Required for legal, regulatory, or contractual compliance
- The product is unusable or unsafe without it
- Removing it invalidates the business case for shipping at all

S - Should Have

Important, but the release can survive without it.

- Adds significant value or removes real friction
- A workaround exists, even if it's painful
- Painful to cut, but not launch-blocking under time pressure

C - Could Have*Desirable, low-cost improvements if capacity allows.*

- Nice-to-have polish, convenience, or delight
- Low impact if left out entirely
- First candidates to add back in if the team finishes early

W - Won't Have (This Time)*Explicitly out of scope for this release not forever.*

- Acknowledged as valuable, but deliberately deferred
- Documented so it isn't silently re-litigated mid-sprint
- Revisited as a candidate for the next planning cycle

3. How the Categories Work Together

MoSCoW is most useful as a capacity-allocation guardrail, not just a labeling exercise. A common and effective rule of thumb, borrowed from the original DSDM guidance, is to cap Must haves at roughly 60% of total delivery effort and leave meaningful room for Should and Could items if Must haves alone consume the entire timeline, the team has no buffer left for the inevitable estimation slippage.

A practical allocation guideline:

- Must have - no more than ~60% of total capacity
- Should have - roughly 20% of total capacity
- Could have - roughly 20% of total capacity, first to be dropped if the timeline slips
- Won't have - zero capacity this cycle, carried into the next planning conversation

4. Worked Example

The table below applies MoSCoW to a claims-processing platform team preparing a compliance-driven Q1 release with a fixed regulatory deadline.

Item	Category	Rationale
State-mandated claim disclosure notice	Must	Required by regulation; missing it blocks the release entirely
Multi-factor authentication for adjusters	Must	Security requirement tied to the same compliance mandate
Bulk claim status export for CSMs	Should	High-value time saver; a manual CSV workaround exists

Item	Category	Rationale
In-app claim status notifications	Should	Reduces support tickets; email notifications already cover it
Personalized dashboard themes	Could	Nice polish; zero functional impact if deferred
Saved search filters	Could	Convenience feature; low cost to add if time allows
AI-generated claim summaries	Won't	Valuable, but not ready for this compliance-driven release

Reading the result: the two Must have are both compliance-driven and together define the non-negotiable floor of the release. The Should have are genuinely valuable but have a workaround, so they are the first place to look if the timeline compresses. The Could have add polish with minimal risk if cut, and the Won't have is not rejected it is explicitly parked for the next roadmap cycle rather than silently dropped.

5. How to Apply MoSCoW - Step by Step

1. Confirm the fixed constraint driving the exercise a deadline, budget, or contractual date since MoSCoW is built around a fixed constraint, not open-ended scope.
2. List every candidate requirement or backlog item in one place, gathered from stakeholders, compliance, and discovery research.
3. Draft an initial category for each item using the definitions in Section 2 do this independently before group discussion to avoid anchoring.
4. Bring stakeholders together to reconcile disagreements, especially anything nominated as Must have by only one function.
5. Check the Must have list against the ~60% capacity guideline; if it exceeds that, some "Musts" are really "Shoulds" in disguise.
6. Document the rationale for every Must and every Won't these are the two categories stakeholders will revisit under pressure.
7. Publish the categorized list as the scope baseline for the release, with Won't have items logged for the next planning cycle.
8. Revisit categorization at major milestones a Could have can become a Should have if new evidence emerges.

6. When to Use MoSCoW and Its Limitations

Best suited for

- Release planning against a fixed, non-negotiable deadline (regulatory, contractual, or executive-committed)
- Scope negotiations where stakeholders need a simple shared vocabulary, not a scoring model
- Situations where explicit, documented scope cuts are more valuable than precise ranking

Known limitations

- Provides no ranking within a category two Must haves can still compete for the same sprint capacity
- Vulnerable to “Must have inflation” if stakeholders treat the label as a negotiating tactic
- Doesn’t quantify reach, impact, or effort the way RICE does best used after a scoring pass, not instead of one
- Categories can go stale if not revisited as the release timeline evolves

PM Practice Tip

- Pair MoSCoW with a scoring model like RICE: use RICE to rank the backlog by expected value,
- then use MoSCoW to draw the release line once a deadline or capacity constraint is fixed.

7. Reusable Scoring Template

Copy the structure below into your own roadmap tracker. Keep the release constraint and the ~60/20/20 capacity guideline visible at the top of the sheet so every contributor categorizes consistently.

Item	Category	Rationale	Owner