# The
# Ultimate Guide

How to use the most popular prioritization frameworks

## to Prioritization
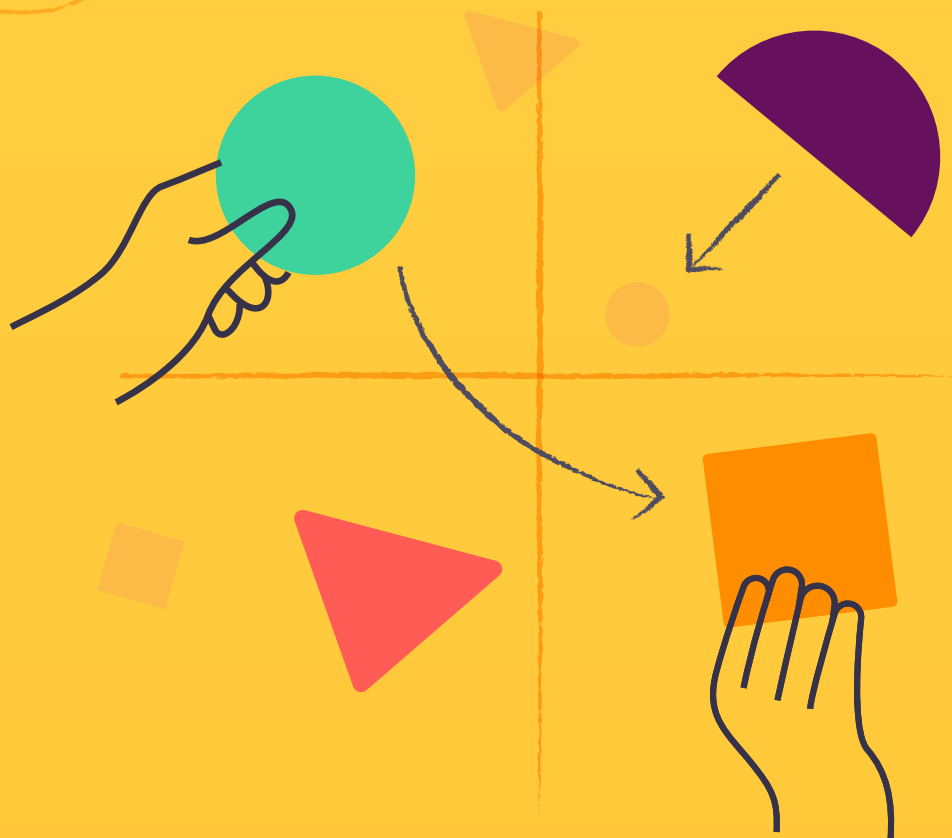
**roadmunk**

**The**
# Ultimate Guide
**to** Prioritization

Create simple and powerful roadmaps.
Roadmunk is a roadmapping tool
that makes it easy to share and
visualize your product strategy.

roadmunk.com

We've also created a community for
product people. Product to Product
is an event series, podcast, blog and
newsletter for creating + capturing
candid convos about product.

producttoproduct.com

For more like this, visit:
roadmunk.com/guides

# Table of Contents

# Foreword

"  **I'd like to build every single product feature idea that customers and stakeholders throw my way because they're all valuable.**

**I have so much time, money and employees available to work on these product ideas! Let's do them all!**

— said no PM ever.

Fellow PMs—I've been there. You have a limited amount of time, cash, and manpower you can spend on any given initiative. On top of that, you have a thousand equally-loud voices to listen to, all with different priorities and problems. It's the core of a PM's job—using data, research, information and insights to make evidence-based, value- and outcome-driven decisions that push the product, the business and the customer's goals towards success.

A good prioritization process gives you the confidence and the data you need to back up product decisions so you can get buy-in and alignment. Healthy product prioritization also lets you choose, classify and rank the many different inputs you get every day in a way that's easy to communicate to your stakeholders.

3
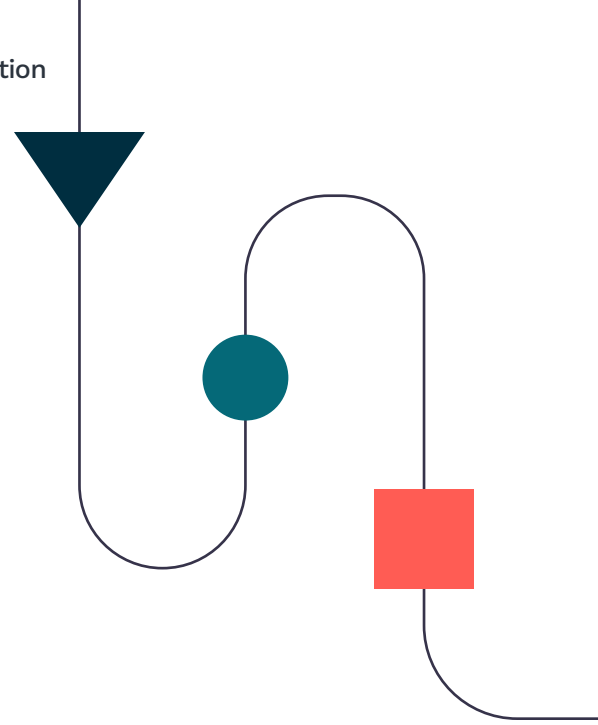
# Here at Roadmunk, we understand that good PMs are always looking for ways to improve their product prioritization process.

## That's why we built this guide full of actionable ways to approach idea management and product prioritization.

— You're very welcome

# Part 1

# Start with qualitative prioritization

Qualitative prioritization means prioritizing based on your product strategy. It's the qualitative, internal prioritization that happens before any type of feedback or feature request gets taken seriously by the product team.

Qualitative prioritization consists of taking the of proposed ideas, initiatives, updates and feature requests, then "scoring" them against how well they push the product vision, goals and metrics forward.

We'll go over the 3 main qualitative components you should be scoring each initiative and feature idea against:

**A** **Strategic value:** Strategic goals, themes and problems to be solved

**B** **Competitive landscape analysis:** Feature competitive analysis

**C** **Customer demand:** The volume of users that are facing the same problem

**A** # Strategic value

**Keep the higher purpose front & center using strategic prioritization**
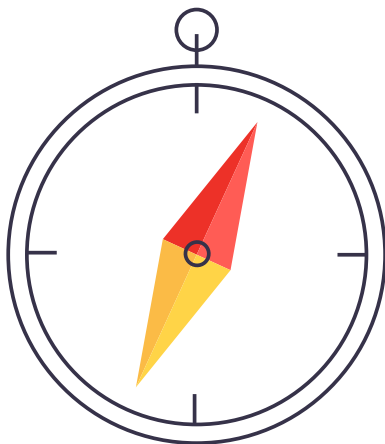
Regardless of which framework you use to prioritize your initiatives and features, there's one dimension they should all be measured against: how much they contribute to the overall product strategy and overall vision. And the best way to do that—as well as bring visibility to the features and initiatives that will keep the product on track—is by ranking features along a strategic scorecard.

Let's say your product is a financial management and budget planning application whose vision is to help everyone take charge of their finances in one place (like Mint). You'd then score your initiatives based on each goal that, once achieved, will push the product closer towards that vision.

So if the goals for the quarter revolve around **improving accessibility**, **improving UX**, and **decreasing churn**, you'd use a simple score to rank each feature idea based on how closely it pushes each of those goals forward. Like so:

| Idea | Accessibility | Improve UX | Decrease Churn | Score |
|---|---|---|---|---|
| **Budget warning: Prompt user before reching limit** <br> `Planned` Notifications | ■■□ | ■□□ | ■■□ | **44** |
| **Log-in to app with Face ID** <br> `Shipped` Authentication | ■■□ | ■■□ | ■■■ | **76** |
| **Custom layout for foldable phones** <br> `Planned` Interface | ■□□ | ■□□ | ■■■ | **55** |
| **Auto-suggest transaction categories** <br> `In Progress` Budgeting | ■■■ | ■■□ | ■□□ | **66** |

**Strategic value prioritization**

By bringing the focus of the prioritization back to those goals, it removes any risk of prioritizing features based on the loudest stakeholder in the room or competitive pressure. Keeping the strategy front and center during the prioritization phase gives product teams the confidence that they're building a valuable product for their specific audience's problems.

**B** # Competitive landscape analysis

**Keep a pulse check on your market position
using a feature competitive analysis**

When you look at how your
competitors are (or aren't) tackling
a feature, you can get a sense of
their weaknesses and strengths.
By seeing ranking features based
on which competitor offers it
and which doesn't—and at what
quality—product teams can find
underserved needs they can
then exploit and grow into.

**Competitive landscape analysis**

| Idea | Competitor 1 | Competitor 2 | Competitor 3 | Competitor 4 |
|------|:---:|:---:|:---:|:---:|
| **Two factor authentication**<br>Notifications | ◕ | ◔ | ● | ◔ |
| **Search improvements**<br>Authentication | ◔ | ◑ | ◔ | ◑ |
| **Responsive eCommerce site**<br>Interface | ◔ | ◑ | ◔ | ◑ |
| **Apple Pay integration**<br>Budgeting | ◔ | ◕ | ◑ | ◕ |
| **Help Bot**<br>Interface | ◑ | ◔ | ◕ | ◔ |

● Check out our in-depth guide to conducting a feature competitive analysis **here** ▸

# **C** Customer demand

**Prioritize by popularity (number of requests per feature)**

Customer demand is simply the number of times a feature has been requested by your users. These requests come in via support, CS, sales, product marketing and even social monitoring. While it's definitely not a reliable way to prioritize features (strategy and vision should always come before customer demand), it's a factor that can't be ignored.

The number of times a feature is requested—as well as the customer insights gathered from qualitative user research methods like interviews, experiments and tests—can have a tectonic effect on product strategy. So it's worth it to keep track of this information (i.e. the needs expressed by your users), especially in new products that are still looking for ways to stand out in a saturated market.

**Customer demand**

| Idea | Number of requests |
|------|--------------------|
| **Budget warning: Prompt user before reching limit**  Planned  Notifications | 31 |
| **Log-in to app with Face ID**  Shipped  Authentication | 14 |
| **Custom layout for foldable phones**  Planned  Interface | 11 |
| **Auto-suggest transaction categories**  In Progress  Budgeting | 6 |

# Tips for managing internal & external expectations

► **Be realistic and transparent with your customers and stakeholders**

► **Establish expectations.** It's important to establish how much of a role customer feedback will have in your product planning decisions. And it's equally important that your users know that. Not all products rely on customer feedback as a source of research, but the products that do should make sure their users know about it.

► **Be transparent.** Make sure your optimism is rooted in confident estimates made using real data. When you discuss the scope and the possible delivery timeline for any given feature, acknowledge your limits.

► **Get to the root of the problem and offer solutions.** Find the problem buried in the solution request and discuss alternative workarounds.

► **Close the loop.** Loop them back in if the feature they requested gets the green light.

▶ **Have product strategy meetings with customer-facing teams**

You're probably familiar with this scenario: A PM is the de facto feedback inbox. He or she is the repository that receives all the feedback coming from customer-facing teams. This PM holds regular meetings with all the teams who possess this insight and keeps track of all the different Google docs and Trello boards that are relevant to planning the roadmap.

It's an overwhelming volume of information that's only useful if everyone who submits it has a deep understanding of the product strategy, direction and vision. What does 'understanding' mean in the context of product prioritization? It means giving your customer-facing teams the strategic knowledge to discern between the noise and the signals in the customer feedback.

Give your customer feedback touchpoints and stakeholders answers to the questions:

- What do we define as value and impact?

- What are the ideal customer outcomes when using a potential feature?

- What are the ideal business outcomes for a potential feature (metrics for measuring success)?

- How are we defining the 'worth' in the problems worth solving?

This also applies to your stakeholders. When you effectively communicate the product strategy to everyone involved, you're giving them a filter for discerning which features would just bloat the product and not stick in the long-term, and the features that have the potential to push all the strategic goals in the right direction.

# Part 2

# Quantitative prioritization using prioritization frameworks

Prioritization isn't just about building a stack of features in a certain order. It's mostly about piecing together the many pieces of information coming from different sources of product insight and research into one coherent direction. Narrowing down a list of demands and feature requests, and committing to them on the roadmap, can be the most challenging part of a product manager's job.

Another prioritization challenge product managers face is knowing how many team members, stakeholders and customers they should involve in the prioritization process—as well as how to keep their motivations for pushing an idea aligned with the product strategy and vision.

When it's time to prioritize quantitatively, PMs ask themselves questions like:

**1** What information is noise and which is a signal that should be listened to more closely?

**2** Do our most requested features align with the product strategy and the long-term business goals we have for the company?

**3** Will these features solve the right problems we set out to fix with our product when we crafted our vision?

**" The most popular feature and the second most popular feature don't necessarily belong together in the same product. You've got to have a deliberate strategy where you go: We have a particular type of customer that we're trying to serve and we are trying to solve their biggest problems in a way that makes us money. That's a complex problem of finding the overlap in multiple different areas, not to mention things that a team can reasonably do technologically."**

- Bruce McCarthy, Product Manager and author of Roadmaps Relaunched
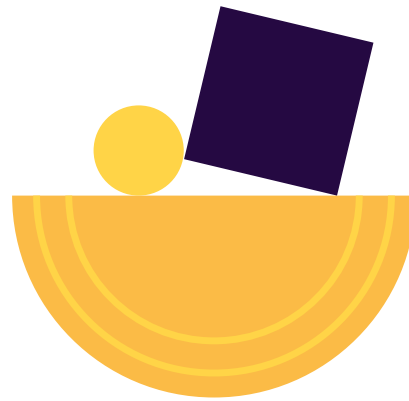
Ideally, good quantitative prioritization frameworks allow you to silence the voice of the loudest person in the room using scores, rankings, charts, and matrixes made up of values that took data and strategy and turned them into logical estimates everyone can align on.

We've rounded up a list of the most commonly used and popular product prioritization methods, the pros and cons of using each one and some best practices for how to get the best out of each one.

# Scoring methods

If you want to keep the process lean and easy-to-standardize, then weighted and unweighted scoring prioritization might be the best framework for your team. It all starts with choosing the criteria you'll evaluate each feature by. These values usually fall under one of two dimensions: **How important is it?** values vs **How difficult is it to build?** values.

Different teams will label it different terms, but generally speaking this is how they can be categorized when it's time to choose X vs. Y.

Scoring methods give product teams a quick and easy way to visualize a set of quantified priorities. This method of prioritization makes room for healthy discussions among stakeholders on what they believe value and effort means, which in turn helps product managers find the strategic alignment holes and fix them.

## How important is it?

**Value**
- Potential revenue

**Benefit**
- To current customers
- To potential customers

**Impact**
- On the business goals
- On the strategic goals

## How difficult is it to build?

**Cost**

**Effort**
- Development effort
- Operational effort
- Implementation effort

**Risk**

**Complexity**

# Value vs. Cost or Value vs. Risk:

## Which one's better?

Prioritization is a game of constraints, and the winner is never the company who gets to build the most features. Rather, a company that does prioritization well is one where the most impactful features are being built perfectly within a limited set of resources.

Scorecards are a great way to easily compare highly valuable features (desirability) vs. how realistic it is to build them (feasibility). A simple scorecard is the perfect lean method for aligning everyone on what internal, strategic criteria makes a good feature. It's also a great exercise in setting realistic expectations for what can be built with the resources the company has. For this guide, we'll explain the most popular prioritization scorecards.

Usually a combination of:

**Value:**

Value is defined as any benefit obtained from building the feature/initiative. It can refer to business value, like revenue, or value to the customer.

Quantified against one of three values:

**Cost:**

All the costs associated with building a feature. This includes development, operational, implementation and maintenance costs. Cost can be anything from time to build, technical effort, operational costs, and cost to implement.

**Complexity:**

How technically complex a feature will be to build. This can mean the technical/development complexities, the implementation complexities, testing/UX complexities, etc.

**Risk:**

This prioritization criteria is mostly used by new products and startups who are in the process of finding their footing in the market.

# Value vs. Cost (or complexity)

We defined value as any benefit obtained by the customers and the business after a feature is built. Cost, on the other hand, refers to anything that creates a difficulty in the process of building a feature.

Cost, in this case, doesn't just come as a monetary value (man-hours per person, # of team members required full time, etc). Costs can also come as the total amount of development and implementation efforts, operational costs, technical complexity and risk factors.
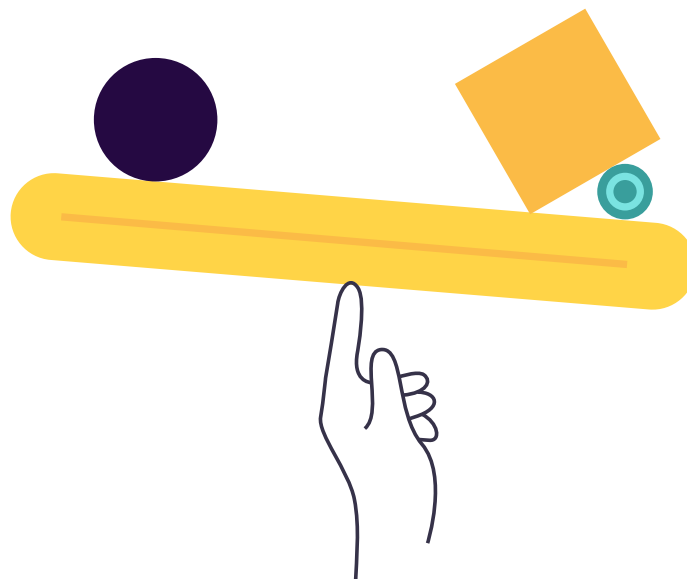
# Value vs. Risk

Development risks are any potential unforeseen problems that might arise. It's one of the hardest values to estimate and calculate because there's no way to predict what might go wrong.

When taking risk into account during the prioritization process, it results in one of two scenarios. The risks never happen, and the development team doesn't have to put any extra crisis control plans in motion. Or the risks happen, and the development team has a plan of action for surviving them and coming out victorious when the crisis ends.

Here are the types of risks a product team might want to account for during the prioritization process (as well as the types of questions they can ask for assessing each potential risk):

**Delay risks**

- What types of constraints might affect our predicted time to deliver this initiative or feature?

- Have we based our schedule estimations on as much data as possible? Or has it been mostly optimistic guesswork that doesn't account for team capacity?

- Have we accounted for every task that can affect how long it will take to build this feature?

**Cost risks**

- How might we go over budget?

- Could the development scope and requirements change over time as new research/testing findings emerge?

- Are we prepared for any unforeseen costs and have we allotted a budget for them?

**Technical risks**

- Does the team have all the tools, knowledge and inter-departmental support needed to build this initiative?

- What are some functional reasons we might not be able to build, deliver or implement a potential feature?

- Have we account for all inter-departmental dependencies for completing the deliverables? Or was the decision-making done in silos?

# RICE

Known as [Intercom](#)'s internal scoring system for prioritizing ideas, RICE allows product teams to work on the initiatives that are most likely to impact their goals.

This scoring system measures each feature or initiative against four factors: reach, impact, confidence and effort (hence the acronym RICE). Here's a breakdown of what each factor stands for and how it should be quantified:

### REACH                                    R

How many people will this feature **affect within a given time period?**

**Example:** customers per quarter, transactions per month.

### CONFIDENCE                              C

How confident are we about the impact and reach scores? How much data do we have to back up those estimates? Use a % score where:
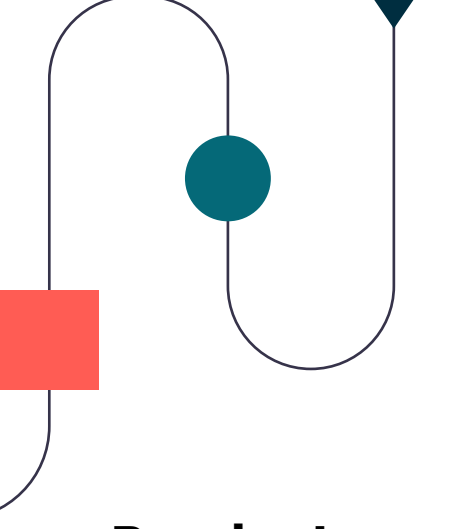
| 100% = high confidence |
| 80% = medium |
| 50% = low |

### IMPACT                                   I

How much will this impact individual users? Use a multiple choice scale:

3 = massive impact

2 = high impact

1 = medium impact

0.5 = low impact

0.25 = minimal impact

**Example:** How much will this feature affect conversion rates?

### EFFORT                                    E

How much of a time investment will this initiative require from product, design and development? Measure as persons per month (how much work one team member can do in a month).

Then, those individual numbers get turned into one overall score using a formula. This formula gives product teams a standardized number that can be applied across any type of initiative that needs to be added to the roadmap.

$$\frac{\textbf{Reach} \times \textbf{Impact} \times \textbf{Confidence}}{\textbf{Effort}} = \textbf{RICE score}$$

After running each feature by this calculation, you'll get a final RICE score that you can then use to rank the order in which you'll build the features. Here's an example:

| Idea | Reach | Impact | Confidence | Effort | Score |
|---|---|---|---|---|---|
| **Log-in to app with Face ID** <br> Authentication | 500 | 2 | 80 | 5 | 160 |
| **Budget warning: Prompt user before reching limit** <br> Notifications | 450 | 2 | 100 | 3 | 300 |
| **Auto-suggest transaction categories** <br> Budgeting | 300 | 3 | 80 | 2 | 360 |

RICE framework

**Scoring frameworks pros**

- What constitutes value or effort is flexible. For some organizations, effort could just mean development effort, in others it could be the implementation cost. A flexible prioritization framework can be used by any type of company, industry or type of product.

- It's a good tool for alignment. By encouraging teams to quantify and numerically score features, product teams can agree on which initiatives have more weight than others, leaving vague guesswork and assumptions out of the discussions.

- In companies where resources are extremely limited, a value vs effort analysis allows teams to focus only on the things that will have the biggest impact on their business and product goals.

- It's easy to use because it doesn't involve any complex formulas or models. All it requires is an agreed-upon numerical value that gets added into one overall total number.

**Scoring frameworks cons**

- Like all prioritization exercises, it's a game of estimation and guessing which leaves a lot of room for cognitive bias at the hands of the people doing the estimation. The final score for each feature might be too inflated, or not accurate enough.

- When it's time for product and development to vote on how high or low the value/effort scores should be, the disagreements can take a while to resolve.

- It can be hard to use in large teams with multiple product lines, components, and product teams that oversee each of those.

# Weighted Scorecard

This prioritization method, also known as a weighted scorecard, involves the steps of the previous methods, but with an added calculation implemented for the sake of making stakeholders choose the relative importance of each criteria.

A weighted scorecard uses a second scoring dimension: the relative value of each dimension the features are being rated on.

This relative value is a "standardized" weight of each prioritization criteria, usually adding up to 100% or 10. The added weight dimension is useful for taking into consideration the importance of each feature/initiative in relation to one another. It's a good method for creating transparency around how important each prioritization factor is to all stakeholders before the features are scored and ranked.

Here's a visualization of the weighted prioritization criteria. For this example, the criteria is customer value, impact on business goals, implementation costs and development risk. Each value has been weighted, adding up to a total priority weight of 100.

| | Customer Value | Impact on business goals | Implemen-tation costs | Dev risk | Total |
|---|---|---|---|---|---|
| Weight | 40% | 40% | 10% | 10% | 100% |
| Ideas | | Score (out of 5) | | | Priority Score |
| **Budget warning**<br>Notifications | 3 | 1 | 1 | 2 | |
| **Log-in to app with Face ID**<br>Authentication | 4 | 2 | 5 | 1 | |
| **Auto-suggest transactions**<br>Budgeting | 5 | 3 | 4 | 2 | |

Weighted scorecard
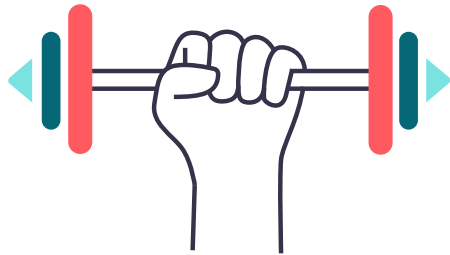
By asking stakeholders to assign a weight to each category before the features are scored, product managers are saying: "You have to decide which of these factors matters the most in terms of pushing the needle towards development, and which factors shouldn't have the same weight in the decision-making process."

The idea is that each scoring category (value, cost, impact, risk) has a different level of importance. This level of importance is then quantified as a "weight".

For the previous scorecard, this is how you'd get the final priority score:

| | Customer Value | Impact on business goals | Implemen-tation costs | Dev risk | Total |
|---|---|---|---|---|---|
| **Weight** | 40% | 40% | 10% | 10% | 100% |
| **Ideas** | | Score (out of 5) | | | Priority Score |
| **Budget warning**<br>Notifications | 3 x 40 = **120** | 1 x 40 = **40** | 1 x 10 = **10** | 2 x 10 = **20** | **190** |
| **Log-in to app with Face ID**<br>Authentication | **160** | **80** | **50** | **10** | **300** |
| **Auto-suggest transactions**<br>Budgeting | **200** | **120** | **40** | **20** | **370** |

# The Kano Model

The Kano model plots two sets of parameters along a horizontal and a vertical axis. On the horizontal axis, you have the implementation values (to what degree a customer need is met). These values can be classified into three buckets:



The Kano Model

**Must-haves or basic features:**
If you don't have these features, your customers won't even consider your product as a solution to their problem.

**Performance features:**
The more you invest in these, the higher the level of customer satisfaction will be.

**Delighters or excitement features:**
These features are pleasant surprises that the customers don't expect, but that once provided, create a delighted response.

On the vertical axis, you have the level of customer satisfaction (the satisfaction values). They range from the needs not being met on the left, all the way to the needs being fully met on the right (the implementation values). The way you get this customer insight is by developing a Kano questionnaire where you ask your customers how they'd feel with or without any given feature.

The core idea of the Kano model is that the more time you spend investing resources (time, money, effort) to create, innovate and improve the features in each of those buckets, the higher the level of customer satisfaction will be.

**Kano model pros:**

- The Kano model teaches you not to overestimate excitement features and to not underestimate must-haves.

- It can help you make better product decisions and market predictions around your features and your audience's expectations for those features.

**Kano model cons:**

- The Kano questionnaire can be time-consuming. In order to get a fair representation of all your customers, you need to carry a number of surveys that are proportionate to the number of customers you have.

- Your customers might not fully understand the features you're surveying them about.

# Kano analysis tips

- ◆ When describing features as scenarios and use cases, always validate those first to ensure it's how the customer is using your product.

- ◆ Use the Kano analysis only to analyze features and initiatives that will have a direct benefit on your users' experiences.

- ◆ The Kano model isn't a 'set it and forget it' questionnaire. Your customers' expectations evolve over time, especially within highly competitive industries with multiple products in the same space.

# Story mapping

The beauty of this product prioritization framework is in its simplicity. It also puts the focus on the user's experience, rather than on the internal opinions of your team and stakeholders.

Along a horizontal line, you create a series of sequential buckets or categories that represent each stage of the user's journey

through your product. This allows you to think about the way your customers navigate your product from signing up, to setting up their profile, to using specific features.

Along a vertical line, you then place these tasks in order of importance, from top to bottom. This allows you to prioritize the order of the features you'll work on. In some cases, the bottom part of the axis is labeled "Backlog items" for the tasks that you decide to put on hold.

Finally, you draw a line across all these stories to divide them into releases and sprints.

| Product search | Product page | Sign-up | Checkout | Story mapping |
|---|---|---|---|---|
| Sort | Reviews | Create account | Pay with PayPal | |
| Filter | Media gallery | | Pay with credit card | |
| | Descr-iption | | | |

Priority ↑

**Story mapping pros:**

- It helps you quickly—and efficiently—identify your MVP.

- It's centered around the user's experiences. You get to write user stories.

- It's collaborative. Story mapping is a group activity that involves the whole team.

**Story mapping cons:**

- It doesn't take into account external product prioritization factors like business value and complexity.

- It's mostly a technique for visualizing user journeys through the product

# Story mapping tips

◆ Your user stories are hypotheses based on estimates and research, so make sure they're worded as testable assumptions.

◆ Keep the big picture front and center by describing the user outcomes that will come from each prioritized release.

◆ Encourage discussions about feasibility/viability/desirability, but ensure there's a moderator that can keep track of those conversations.

# The MoSCoW Method

The MoSCoW method allows you to figure out what matters the most to your stakeholders and customers by classifying features into four priority buckets. MoSCoW (no relation to the city—the Os were added to make the acronym more memorable) stands for Must-Have, Should-Have, Could-Have, and Won't-Have features.

**Must-Have:** These are the features that have to be present for the product to be functional at all. They're non-negotiable and essential. If one of these requirements or features isn't present, the product cannot be launched, thus making it the most time-sensitive of all the buckets.

Example: "Users MUST log in to access their account"

**Should-Have:** These requirements are important to deliver, but they're not time sensitive.

Example: "Users SHOULD have an option to reset their password"

**Could-Have:** This is a feature that's neither essential nor important to deliver within a timeframe. They're bonuses that would greatly improve customer satisfaction, but don't have a great impact if they're left out.

Example: "Users COULD save their work directly to the cloud from our app"

**Won't-Have:** These are the least critical features, tasks or requirements (and the first to go when there are resource constraints). These are features that will be considered for future releases.

The MoSCoW model is dynamic and allows room for evolving priorities. So a feature that was considered a "Won't-Have" can one day become a must-have depending on the type of product.

**MoSCoW pros:**

- It's good for involving stakeholders without a technical background in the product prioritization process.

- Quick, easy and intuitive way of communicating priorities to the team and the customers.

- It allows you to think about resource allocation when you classify your features and requirements into each bucket.
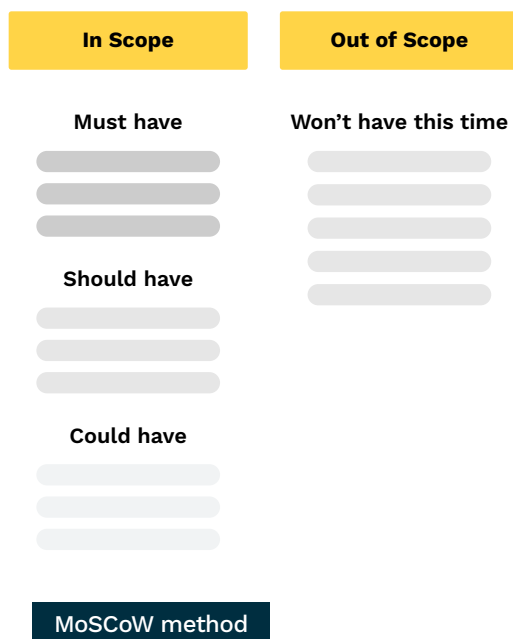
**MoSCoW cons:**

- It's tempting for teams and stakeholders to overestimate the number of Must-Have features.

- It's an exercise in formulating release criteria more than a prioritization method.

| In Scope | Out of Scope |
|----------|--------------|
| **Must have** | **Won't have this time** |
| **Should have** | |
| **Could have** | |

MoSCoW method

# MoSCoW tips

◆ The DSDM Handbook says that the ideal ratio is 60% Must Haves, and 40% between Should Have and Could Have.

◆ The MoSCoW method is great for encouraging discussions among the entire company, so the more participants you include from different departments, the better the picture.

◆ MoSCoW is all about determining and prioritizing requirements for any given initiative. The implementation details come after the exercise, so you can leave those out of the MoSCoW exercise.

# Opportunity scoring

Also known as opportunity analysis, this prioritization method comes from Anthony Ulwick's Outcome-Driven Innovation concept. His theory states that customers buy products and services to get certain jobs done. The idea is that, while customers aren't very good at coming up with the solutions to their problems, their feedback is still important. This feedback is what the product team will use to come up with the desired outcomes for a product or feature.

Opportunity scoring uses a Satisfaction and Importance graph to measure and rank opportunities. After you come up with a list of ideal outcomes, you then survey your customers to ask them the following questions:

**1**  How important is this outcome or feature? Ask your customers to rank them.

**2**  How satisfied is the customer with the existing solutions?

After you plot these answers along the chart, you should be able to see the features that matter the most to the customers (the outcomes) yet currently have low satisfaction scores within your product. These are the features you'll prioritize for your next sprint.

**Opportunity scoring pros:**

- It's a simple framework for quickly identifying the most innovative solutions to a common problem.

- It's easy to visualize and categorize on a graph.

**Opportunity scoring cons:**

- In the survey or questionnaire, customers might overestimate or underestimate the importance of a feature.

# Opportunity scoring tips

◆ Opportunity scoring is the best prioritization method for discovering which current features are undiscoverable by users, not being used to their full extent, or need improvements.

◆ This prioritization method allows product teams to rank initiatives based on the ideal outcomes a user would like to see, and how much it would cost/how many features it would take to make those outcomes happen.

Over-Served

Satisfaction

Served Right

Under-Served

Opportunity scoring

Importance
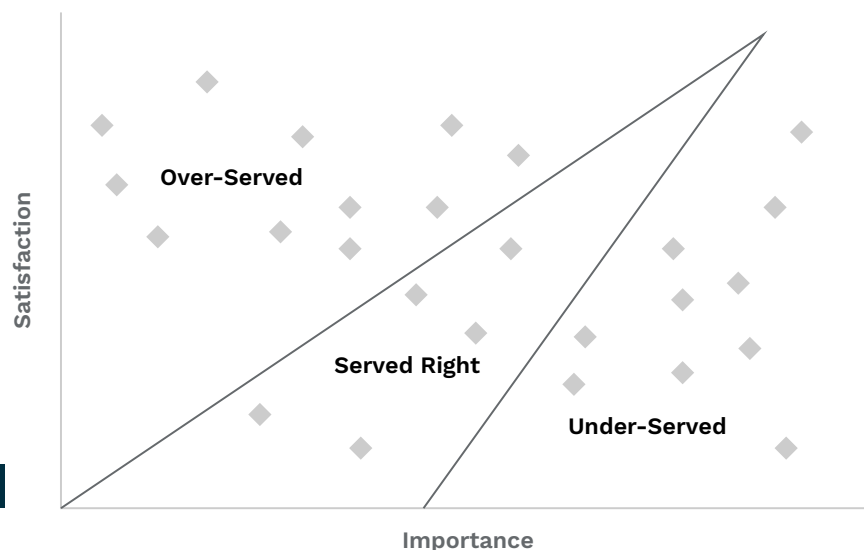
# Product Tree

Also known as pruning the product tree, this collaborative innovation game was developed by Bruce Hollman. The focus of this activity is to shape the product so it matches the customer outcomes that will bring the highest value to the company. The game aims to prune product backlog items to ensure that innovative ideas aren't being left behind.

Here's how it goes:

**1** First, draw a large tree with a few big branches on a whiteboard or a piece of paper.

The trunk of your tree represents the features your product already has.

The outermost branches represent the features that will be available in the next release.

The other branches represent the features that aren't available yet.

**2** Ask your participants (in this case, your customers) to write some potential features on sticky notes. These will be the leaves of your tree.

**3** Then, ask your customers to place their feature leaves on a branch.

By asking customers to place their desired features on the tree, you can identify the biggest clusters or branches. This will allow you to determine which areas of your product need more work, which features need to be changed, and what product feature areas can be deprioritized from all immediate future releases.

**Product tree pros:**

- It gives you a visual sense of how well balanced your product's features are.

- It's highly collaborative, allowing you to tap directly into the insight of your customers without relying on a rigid survey.

**Product tree cons:**

- This method doesn't give product managers any quantitative values for how to rank each feature, only a general visual guide.

- Features aren't separated into any sort of grouping bucket, making the exercise time-consuming.

# Product tree tips

◆ This prioritization method isn't a brainstorming exercise. As a moderator of this exercise, make sure to keep steering the conversation back to feasibility, availability and desirability limits.

◆ In trees with heavier trunks and lighter branches, ask yourself these questions: Are we accurately identifying opportunities for growth? Are we solving the most urgent problems and needs faced by our users? Are we delivering value quickly enough?

# Cost of Delay

Joshua Arnold defined Cost of Delay as "a way to communicate the impact of time on the outcomes [the company wishes] to achieve." Specifically, cost of delay allows you to ask the questions:

**1** What would this feature be worth if the product had it right now?

**2** How much would it be worth it if this feature gets made earlier?

**3** How much would it cost if it was made later than planned?

The way you assign this monetary value to each feature is by calculating how much time and team effort they will take to build. You and the team can also assign value to the features in terms of how much they will be worth after they're built.

So, let's say you have one feature that costs you $30,000 per each week that it's delayed, and it will only take three months to build. On the other hand, you have a feature that costs $10,000 per each week that it's delayed and it will take you the same amount of time to build. Within this prioritization framework, the first feature would be the one your team focuses on first.

| | Cost of Delay | Duration | CD/Duration |
|---|---|---|---|
| Feature 1 | $25k | 5 weeks | $5k/week |
| Feature 2 | $150k | 2 weeks | $75k/week |
| Feature 3 | $5k | 5 weeks | $1k/week |
| Feature 4 | $60k | 4 weeks | $15k/week |

Cost of delay

**Cost of delay pros:**

- It allows you to quantify your product backlog in terms of money.

- It helps product managers make better decisions based on the value that matters the most to the company.

- It changes the team's mindset around features from cost and efficiency metrics, to speed and value.

**Cost of delay cons:**

- The parameters for determining the monetary value of a feature are based on gut-feel and intuition. This can lead to internal disagreements regarding the arbitrary value of any given feature.

# Cost of delay tips

◆ Remember that cost isn't just about money. Other things to consider are the impact on customer satisfaction, market share and brand perception.

◆ Cost of delay is about estimating the cost using hypotheses. Knowing that, it's best to underestimate than overestimate the effort, time and cost any given feature will require in order to be built.

◆ Before diving into a Cost of Delay exercise, establish how your company will define value, both for the business and for the customers.

# Buy a feature

Buy a feature is an innovation game that can involve customers and stakeholders (it's up to you and the needs of your product). When you use it as an exercise with your customers, this method can quantifiably tell you how much a feature or an idea is worth to the people who'll end up using it.

**The game goes like this** ▶

**1** Choose a list of features, ideas, or updates and assign a monetary value to each one. This value isn't arbitrary—it should be based on how much time, money and effort each feature will cost you and the team.

**2** Put together a group of people (up to 8 customers or your own internal team).  Give them a set amount of money to "spend" on these features.

**3** Ask your participants to buy the features they like. Some customers will put all of their money on only one feature they're passionate about, others will spread their cash around multiple different features. Ask the participants to explain why they spent money on the feature they picked.

**4** Then, reorganize that list of features in order of how much money your customers were willing to spend on them.

Innovation games suggests that you price some of the features high enough that no one can buy them. This forces your customers to team up and negotiate which feature they'd be willing to pool their money on.

**Buy a feature pros:**

- If you believe in that Steve Jobs quote ("People don't know what they want until you show it to them") but you also would like to tap into the wisdom of your customers, this method is perfect.

- It replaces the stuffy, old-schooled customer questionnaire with a collaborative, fun exercise that forces your customers to rationalize why they think they need a feature

**Buy a feature cons:**

- This prioritization method can only include features that you've already decided to include in a product development roadmap—the results just tell you what features customers value the most.

- Ideally, the activity requires you to get a group of customers in one place at the same time, which can be difficult to coordinate.

# Buy a feature tips

◆ You can conduct a Buy a Feature prioritization exercises with both customers and internal teams.

◆ This prioritization method is a measure of perceived individual value and it doesn't factor in cost, impact and development effort. It should be treated as part of a whole, rather than a final prioritization method.

◆ This method isn't just for prioritizing features. It can be used for assigning the perceived value of bug fixes and product enhancements.

# Conclusion

Ideally, the prioritization process should be about getting everyone on the same page in terms of what the greater goals are for the product. Once you have a team that's holistically aligned on the strategy and the why behind the strategy, prioritization exercises become less of a headache and more of a collaborative team activity.

## Ideally, a prioritization method should do a few things:

**1** It should be a collaborative process that involves multiple team members, stakeholders, and, depending on the framework, your customers.

**2** The framework that you choose should give you the results that will drive the product strategy forward.

**3** It should motivate the team to position their prioritization reasoning in terms of how each idea contributes to the greater goals of the company.

**4** Your prioritization method should push your team to get rid of the "idea noise"—it should completely weed out the ideas that aren't worth building.

**5** In this case, worth doesn't necessarily mean revenue potential; it can also be the potential estimated effort, time, risks and other costs.

Product managers are managers of the "why"—and this includes communicating that why effectively to anyone who needs to factor in that information in their decision-making. Good product managers know how to back up their reasoning for why a feature should be next in the pipeline using relevant data and information (or estimates that aren't plagued by optimism biases).

### 🙁 Bad prioritization

- HiPPo opinions.
- Building what the highest paying customers request.
- Catching up to the competition.
- Making decisions too quickly without considering multiple data points.
- Lack of a customer feedback dashboard or repository that different teams can access.

### 🙂 Good prioritization

- Driven and informed by the product strategy.
- Based on data and driven by evidence.
- Customer-centric.
- Focused on problems to solve/Jobs to be done.
- Consolidated idea management dashboard.
- Collaborative and well-aligned.

# About

**roadmunk**

**Roadmunk was inspired by a problem our founders experienced personally: there was no simple way for product managers to build, share and align on the roadmap.**

As we got to know our users, we discovered that roadmapping is not one-size-fits-all. Companies make roadmaps of all shapes, sizes and types—it's a powerful and essential tool for alignment.

Our goal is to make strategic roadmapping quick, effective and collaborative across an entire organization.

**Start building your roadmap**

Button not working? **Click here**