



Prioritize product features: the full report

Example: a software company in Istanbul with an invoicing app. All five stages, from the guided tour.

The playbook at a glance

- 1** What are customers trying to get done?
6 of 7 ideas serve the job; 1 leaves the list
- 2** Which features matter, and how?
Fix **Reliable e-invoice sending to the tax system** first: customers expect it, and it fails today
- 3** Which ideas give the most for the effort?
Automatic late-payment reminders ranks first; test **Bank feed that matches payments to invoices** before you commit
- 4** What goes in this release?
A release that fits: Musts at 38% of 8 person-months
- 5** In what order do users get it?
Release 1 runs end to end: 4 cards across 4 steps

STAGE 1 OF 5

What are customers trying to get done?

6 of 7 ideas serve the job; 1 leaves the list

Serve it directly: **2 of 7**

Serve it: **3 of 7**

Serve it barely (go on, marked): **1 of 7**

Leave the list: **1 of 7**

What this may mean

The ideas that leave the list aren't wrong; they serve other goals, so they wait. Ranking only what serves the job keeps the next stages short and the choices clear. The idea marked barely goes on, but expect stage 2 to find it indifferent.

Investigate next

Take the 6 ideas that serve the job into stage 2, the Kano Model, to sort them by how they affect customers' satisfaction.

Directly: the idea does what "I want to" asks for. Serves it: it helps get the job done, or makes it possible.

Barely: a loose link. Doesn't serve it: it serves another goal.

Ideas that leave the list stay in your progress file, marked, so they can come back if the job changes.

Which features matter, and how?

Fix **Reliable e-invoice sending to the tax system** first: customers expect it, and it fails today

Must-be (expected): **1**

One-dimensional (more is better): **2**

Attractive (a pleasant surprise): **2**

Indifferent: **1**

Broken basics: fix first: **Reliable e-invoice sending to the tax system**

For some customers only: **Invoices in euros and dollars**

What this may mean

A basic that fails costs more satisfaction than any new feature can win back, and customers leave over basics long before they leave over missing extras. Don't rank it against the others: it goes into the release first. The other 4 ideas go on to be ranked. An idea that matters to some customers only goes on, but its reach in the ranking is those customers, not everyone.

Investigate next

Put **Reliable e-invoice sending to the tax system** into the release as a Must in stage 4, outside the ranking. Rank the 4 ideas that go on in stage 3, with RICE.

Each idea's pair of answers (how customers would feel if the product had it, and if it didn't) gives its category from the standard Kano table.

The categories are only as good as the answers behind them. The most common answers from a customer survey are the strongest; a team's best guess is a quick first look, worth checking with customers for the ideas the ranking will lean on.

Categories drift: today's pleasant surprise is next year's basic. Sort the ideas again at least once a year.

Which ideas give the most for the effort?

Automatic late-payment reminders ranks first; test **Bank feed that matches payments to invoices** before you commit

Ranked first: **Automatic late-payment reminders**

Then: **A who-owes-what dashboard**, **Bank feed that matches payments to invoices** and **Invoices in euros and dollars**

Rest on a guess (50% confidence): **Bank feed that matches payments to invoices**

Test before you commit: **Bank feed that matches payments to invoices**

Basics from stage 2, outside the ranking: 1

What this may mean

A score built on a guess can move a long way once you know more. A week of tests with a handful of customers costs far less than building the wrong thing, and shows where the idea really belongs.

Investigate next

Take the ranking into stage 4, MoSCoW, to decide what fits this release. Ideas waiting on a test start in Won't (testing first) until the test is in.

RICE score = reach x impact x confidence / effort. Reach is the customers an idea affects in one period, the same for every idea. Impact is 3, 2, 1, 0.5 or 0.25. Confidence is 100% with data, 80% with some evidence, 50% for a guess. Effort is person-months for the whole team, including design and testing.

A test is suggested for a guess that ranks in the top half, or has high impact. You can change it for any idea.

What goes in this release?

A release that fits: Musts at 38% of 8 person-months

Must have: **3 person-months of 8 (38%)**

Should have: **3 person-months**

Could have: **2 person-months: Invoices in euros and dollars, first to drop**

Planned in all: **8 person-months of 8**

Won't have this time: **3 items, including Bank feed that matches payments to invoices (testing first)**

What this may mean

The Musts leave room for surprises, and the Could's are the first to go if time runs short. A Must is something the release fails without: if it would still ship and work, it is a Should. Write the Won't list down and share it: saying no out loud is what stops the same requests coming back every week.

Investigate next

Take the release into stage 5, the user story map, to put it in an order users can walk end to end. Share the Won't list with sales and support.

Must have: the release fails without it. Should have: important, but the release still works without it. Could have: the first thing to drop. Won't have: left out this time, and said so.

The rule of thumb: Musts at about 60% of the effort or less, and the whole plan within capacity. Stage 2's basics start as Musts; ideas waiting on a test start as Won't.

In what order do users get it?

Stage 5 diagnosis

Release 1 runs end to end: 4 cards across 4 steps

Activities across the top: 4

Release 1: 4 cards: a user can walk it end to end

Release 2: 4 cards

Later: 2 cards

What this may mean

Every step has something in release 1, even if it is basic, so a user can get from start to finish on day one. Later releases add depth where release 1 is thin.

Investigate next

Ship release 1, then check that the customer's job is done better: pick one number that shows it and look at it after each release. Run the playbook again each quarter; stage 1 can be a quick check.

Read the map left to right to check the journey, and top to bottom to check the order. A release counts everything in the releases before it, and a step marked as already working counts as covered.

Splitting a big item into a basic first piece for release 1 and the rest for later is the point of the map, not a cheat.

The release as a story map

Activities run across the top, in the order a user does them. Read left to right for the journey, top to bottom for the order.

	1 Send the invoice	2 Track payment	3 Chase late payers	4 Record payment
Release 1 (week 4) End to end	E-invoice sending fixed	Due-date list, overdue first	One reminder, sent by hand	Mark as paid
Release 2 (week 8) End to end	Invoices in euros and dollars	Who-owes-what dashboard	Automatic reminder schedule	Part payments
Later	–	–	SMS reminders	Bank feed matching, if the test works

What each release lets a user do

- **Release 1 (week 4):** 4 cards, for send the invoice, track payment, chase late payers, record payment. A user can walk it from start to finish.
- **Release 2 (week 8):** 4 cards, for send the invoice, track payment, chase late payers, record payment. A user can walk it from start to finish.
- **Later:** 2 cards, for chase late payers, record payment. Not planned yet.

DECISION POINT

Ship release 1

Every step has something in it, so a user can get from start to finish on day one.
Release 2 adds depth.