Booking.com UX Writer II take-home assignment

Daniel McLeod

5 July 2025

Intro to my content design approach

When designing, I always ask myself the following 4 questions:

1. What information should the user know?
2. When should the user know this information?
3. What should be the hierarchy of this information?

   -> What's the number 1, most important thing for them to know? What comes next?
4. What's the clearest, simplest way to convey this information?

Task 2. Delete an irrigation schedule

Delete an irrigation schedule

Key choices

- Use double-confirm destructive action pattern combined with undo deletion pattern.
- Always make the consequences of deletion clear to users.
- Use plain language. Keep to Grade 8 reading level (actual level: Grade 3 on Flesch-Kincaid scale)

Audience

- Professional farmers
- Potentially recreational/hobby farmers

Considerations

- For professional farmers, creating and managing irrigation schedules is serious work, with real implications for the success of their crops and consequently their livelihood. It's essential that users know the consequences of deleting a schedule, including what will happen to any crops currently using that schedule.
- When a schedule is deleted, any crops currently using that schedule should switch to the user's preset default schedule. If the user has not defined a custom default schedule, they should switch to our app's default schedule. This will prevent any crops from going unwatered for any period of time, which would harm our users.

Key terms

- Crops – The fields or sections of field that get irrigated
- Schedules – The irrigation schedules that are assigned to crops

Benchmarks

- Lattice
- Dropbox
- NNG
- Canva

Questions I asked while designing

Question 1.
Even with a double-confirm, is it possible that users could delete a schedule by mistake? If so, do we want to save recently deleted schedules?

My answer: Originally, I thought I had put in sufficient measures to prevent users from accidentally deleting a schedule, or deleting a schedule without understanding the consequences, only to realize they want it back.

The complexity of rerouting previously deleted schedules to crops initially motivated me to choose the "permanently delete" route with extra safety measures — because if you've deleted a schedule and assigned a new schedule to crops, then you bring the original schedule back, do you reassign it to all the crops it was originally assigned to? How do you do that in a simple way?

After experimenting with alternative designs, I now prefer the undo pattern in this scenario. It just feels nicer. I think it's unlikely that users will actually go into **Schedules** > **Settings** > **Recently deleted** to restore a schedule, but I think it's very possible they'll panic after selecting **Delete** and want to see an undo button. In this case, the undo button would simply bring them back a step with all their assigned crop–schedule settings saved for them to review.

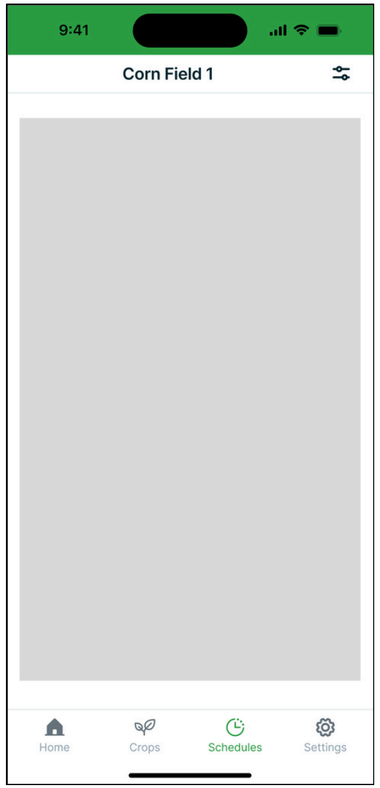
Question 2.
Is it possible that users will be too scared to select Delete from the schedule page's More options dropdown?

My answer: This is a valid concern, as an advanced user will know that some crops are currently using this schedule. This prompted me to revisit the dropdown and add explainer text, clarifying that selecting *Delete* leads to another step.

Metrics for measuring success

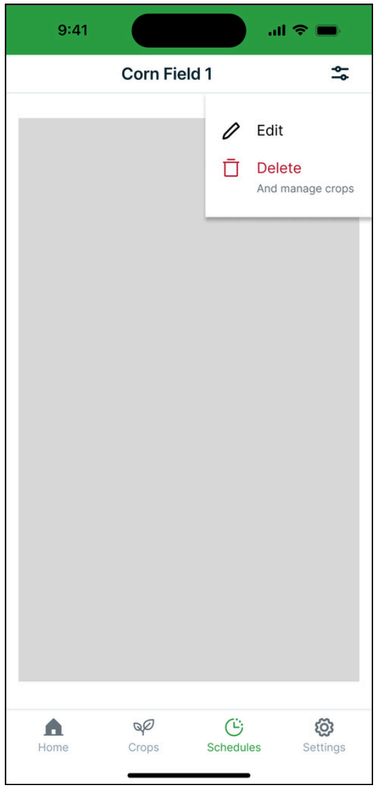
- Confirmation rate: % of users who proceed after seeing the delete confirmation modal
- Cancellation rate: % of users who back out of deletion after seeing the modal
- Undo usage rate: % of users who delete a schedule and then immediately undo
- Error reports or support tickets: Number of complaints about lost schedules or accidental deletions
- Task completion time: Time from tap to confirmation/cancellation

Official submission – Double-confirm with undo option



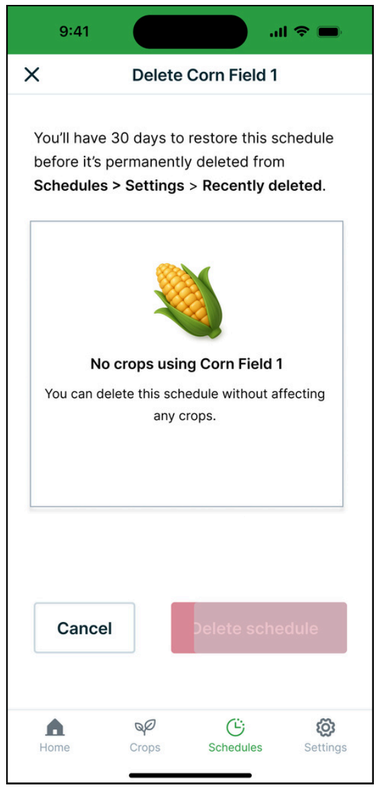
Individual schedule page

- More options dropdown in top right corner.



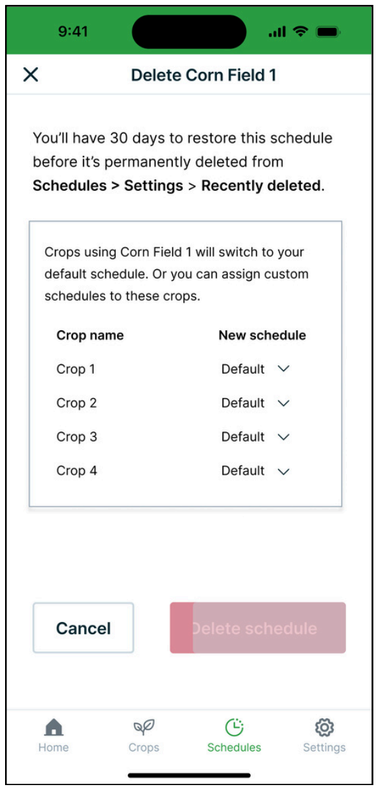
Individual schedule page

- Delete option in dropdown.
- Explainer copy beneath ‘Delete’. to inform user of next steps.



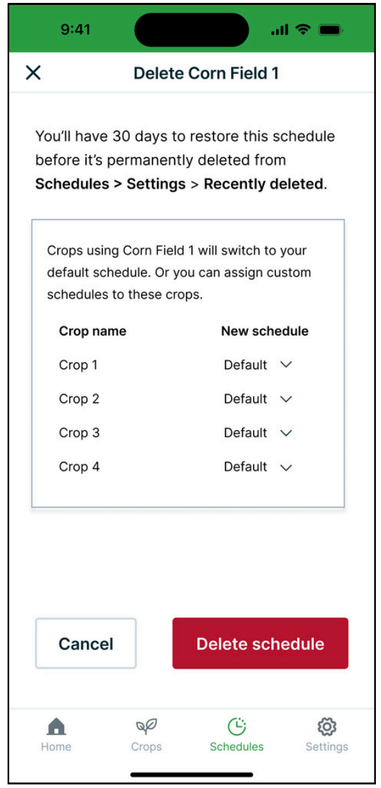
Delete schedule confirmation page

- Copy explaining that the deleted schedule can be restored from **Schedules**> **Settings** > **Recently deleted**.
- Empty state for connected crops. If no crops are using this schedule, then the user can safely delete it without further thought.



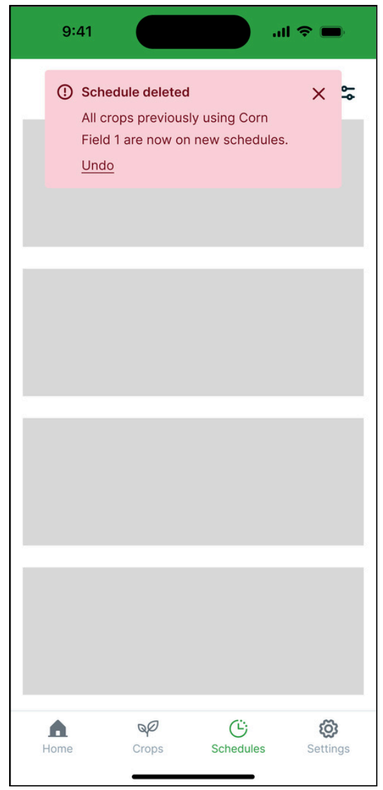
Delete schedule confirmation page

- Table showing crops currently using this schedule.
- Default schedule assigned as replacement.
- Dropdown selector to assign a different schedule.
- Primary CTA has delay loader.
- Secondary CTA spaced far apart from destructive action.



Delete schedule confirmation page

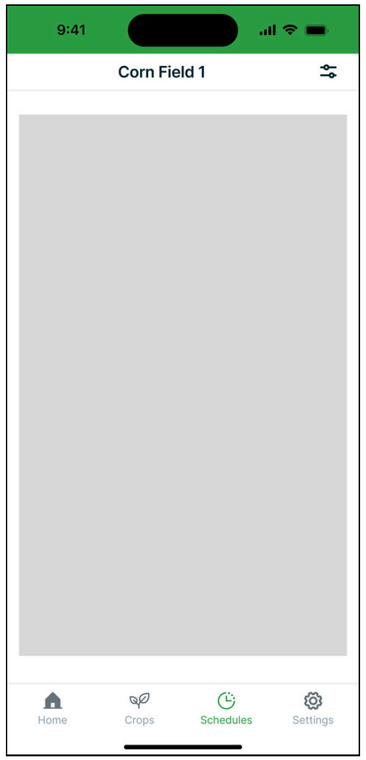
- Primary CTA is loaded.



All schedules page

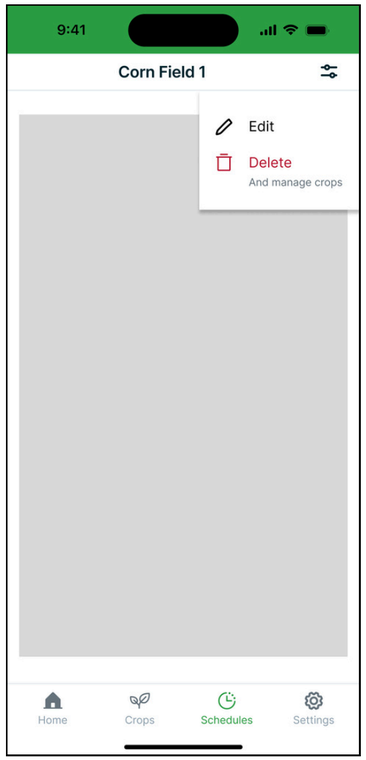
- Confirmation alert restating the consequences of deletion.
- *Undo* button takes user back to previous screen: **Delete schedule confirmation page**.
- When the user goes back, all their settings for crops and schedules are saved.

Alternative 1 – Permanently delete, with no undo option



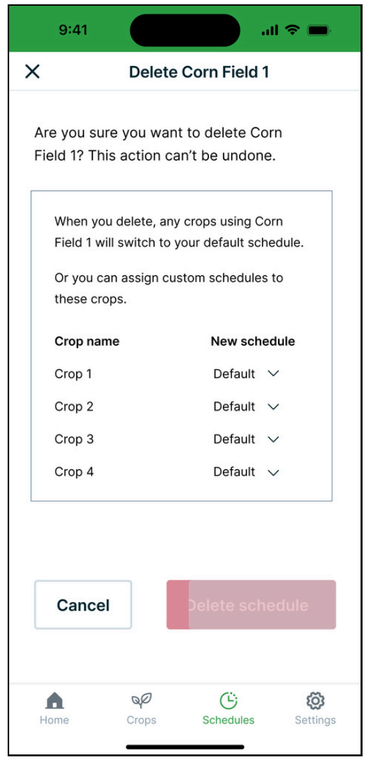
Individual schedule page

- More options dropdown in top right corner.



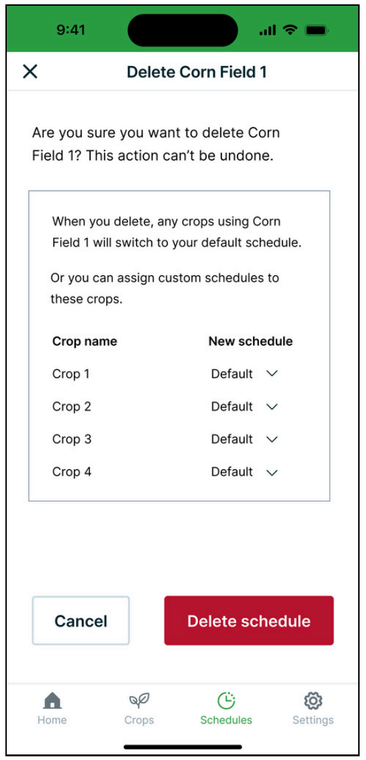
Individual schedule page

- Delete option in dropdown.
- Explainer copy beneath ‘Delete’. to inform user of next steps.



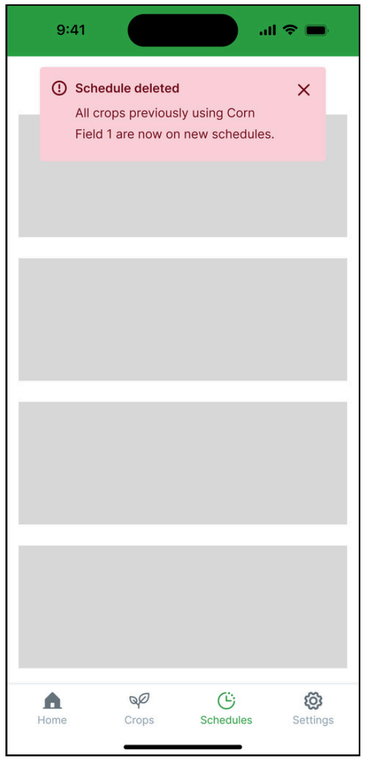
Delete schedule confirmation page

- Copy explaining that deleting is permanent and can’t be undone.
- Table showing crops currently using this schedule.
- Default schedule assigned as replacement.
- Dropdown selector to assign a different schedule.
- Primary CTA has delay loader.
- Secondary CTA spaced far apart from destructive action.



Delete schedule confirmation page

- Primary CTA is loaded.



All schedules page

- Confirmation alert restating the consequences of deletion.

Task 3. Tell users that spelling counts

Tell users that spelling counts

Cooking every day is hard enough.

Users shouldn’t have to worry about

how to spell ingredients.

When I first read this task, I knew that it wouldn’t be as straightforward as writing a simple error message. If a user spells a word wrong once, an error message may suffice. But if a user genuinely has trouble spelling — as is the case for most non-native English speakers — then I would like to offer a more comprehensive, empathetic solution. Also, now that I think about it, a lot of ingredients don’t have intuitive spelling in English. There are many loan words, and words from all over the world with different etymologies.

The idea behind an app like this is to empower users to cook. To use the ingredients and tools at their disposal to make delicious meals, easily. I want to encourage them as much as possible in this noble pursuit. Because cooking every day is hard enough. Users shouldn’t have to worry about how to spell ingredients. Ideally, if this app is well designed, our users would never even know they don’t know how to spell ingredients. They would just start typing *cin* and see *cinnamon* among the options, for example.

This is what inspired the following designs.

Tell users that spelling counts

Key choices

- Instead of telling users that spelling counts, make it so they never have to think about spelling.
- Use plain language, with exception for brand key term ‘virtual pantry’.
- Keep to Grade 8 reading level (actual level: Grade 6–7 on Flesch-Kincaid scale)

Audience

- Everyone, including people with learning difficulties and non-native English speakers
- People using the UI in other languages, so localization is a key priority
- People who cook! Both beginners and advanced

Considerations

- Predictive text to preempt spelling errors
- Speech-to-text to preempt spelling errors and increase accessibility
- Regionality, including regional names for ingredients
  - E.g., Courgette vs zucchini, aubergine vs eggplant, rocket vs arugula, etc.

Benchmarks

- Google Maps audio search
- Supercook
- Tesco recipe generator

Key terms

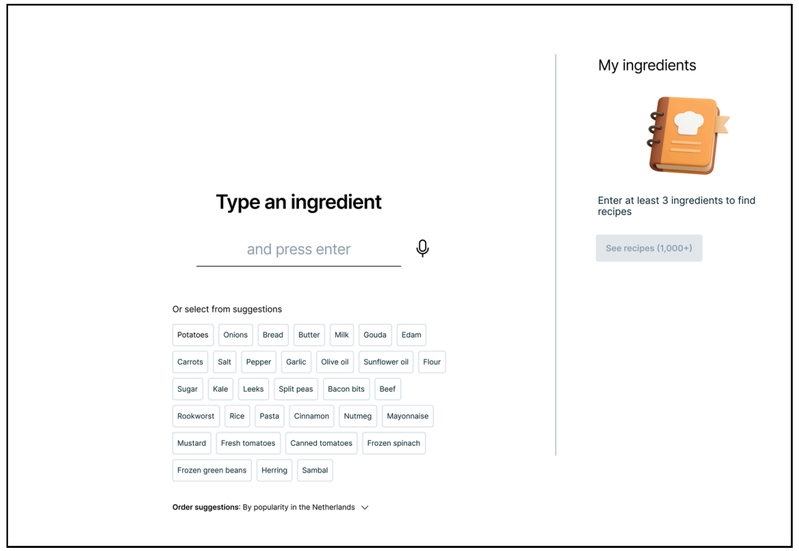
- Ingredients
- Recipes
- Suggestions
- Pantry / virtual pantry – The space where users can find any ingredient, from anywhere in the world. It’s magically thermoregulated, so it includes refrigerated and frozen goods.

Questions I asked while designing

- What if the user is dyslexic but knows perfectly well how to say broccoli?
- When typing, how does the user know to hit enter?
- In the virtual pantry, what does filtering ‘by diet’ actually do? Perhaps let’s remove this option and have only alphabetic index to start. Later, when we learn more about how users search within our pantry, we can add more filters to simplify their search.
- In the virtual pantry, do we need to add any other categories, such as frozen foods? What food categories do supermarkets use? Let’s check Albert Heijn and Tesco.
- But seriously… how do you spell broccoli?

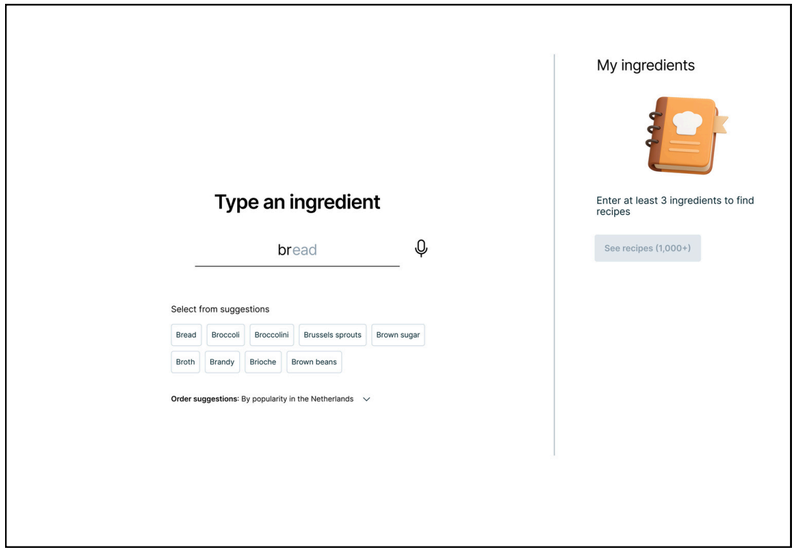
Metrics for measuring success

- Time to success: Time between entering ingredients and viewing recipe results
- Bounce rate / exit rate: Especially from the search page after entering ingredients
- User testing of both written and speech-to-text searches:
  - How easy is it for users to find their desired ingredients?
  - What challenges do users encounter when searching for ingredients?
  - Can users identify and fix errors easily?
- Track usage of speech to text feature to determine where to put developer focus:
  - E.g., Do we focus on improving speech-to-text or written auto-complete?
- Frequency of users not finding a match
- Track instances where user:
  - Clicks ‘Search ingredients’ when they can’t find a match
  - Finds the ingredient they were searching for (selects ingredient in virtual pantry)
  - Does not find what they’re looking for (closes virtual pantry without selecting something, or request new ingredient to be added to pantry)
- Requests for new ingredients:
  - Cross-check against available ingredients to determine search effectiveness
  - Assess whether we’re doing enough to address the issue of alternate names for ingredients
- User feedback or support tickets: Related to search usability or spelling frustrations



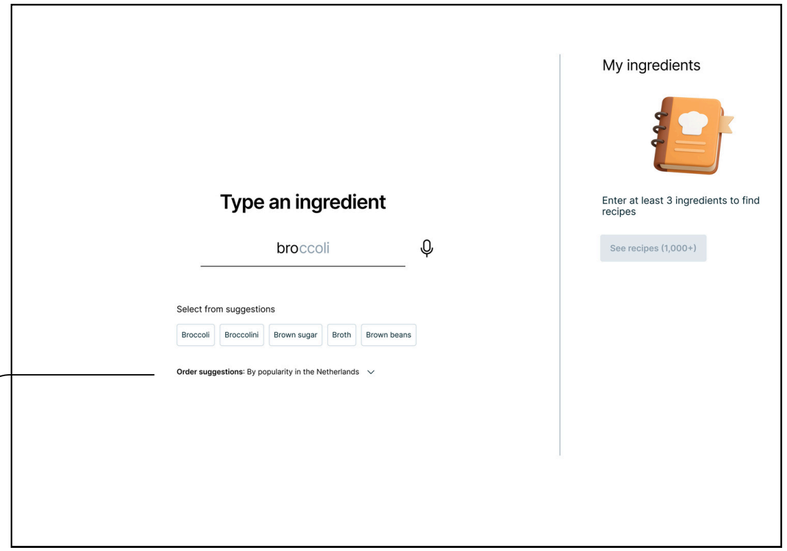
Search page

- Before a user starts typing, we show a list of most commonly used ingredients in their region. We can use the user's IP address to recognize their region initially, but they can update it if desired. The initial list I used is generated by AI, but as we collect real user data, we can update it based on our users' activity.



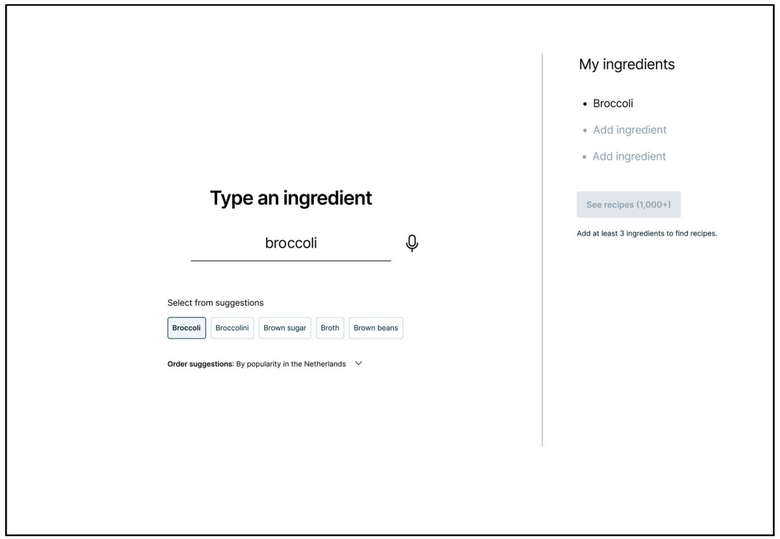
Search page

- When a user starts typing, we use predictive text to suggest ingredient names and preempt spelling errors. We don't want to use generic predictive text tools, but one tailored to the food industry, so that only possible ingredients are suggested (not random words, like tomahawk or floorboards).



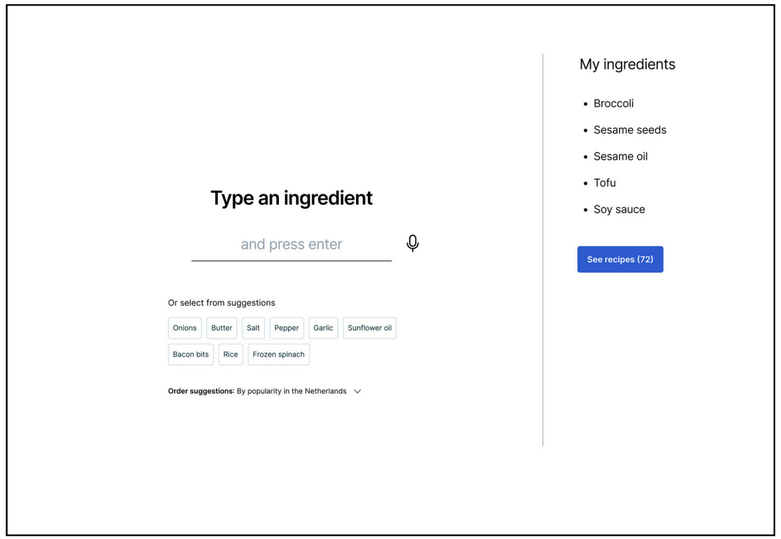
Search page

- As the user types, the list of suggestions below the text input field updates in sync with their input, refining suggested ingredients in real time.
- The default setting for showing suggested ingredients is to order by popularity in the user's region (e.g., the Netherlands), but I would also like to offer users the ability to order alphabetically, if desired.



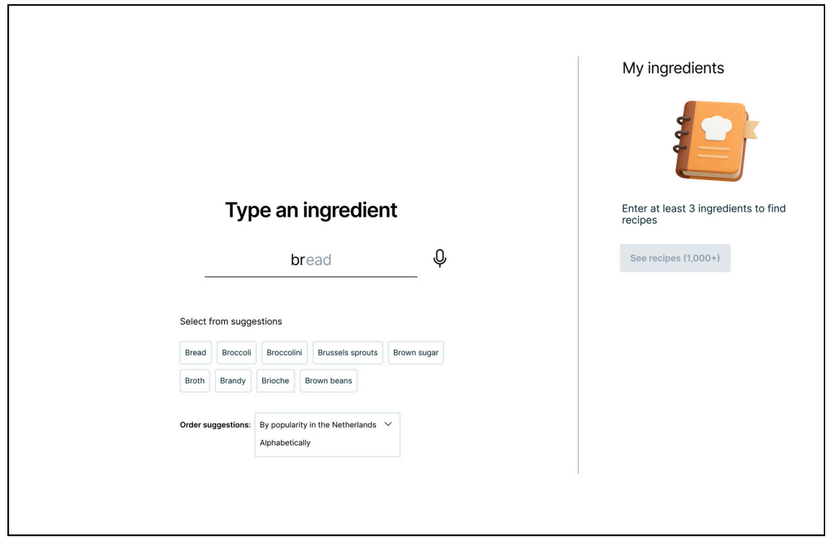
Search page

- Once the user has entered at least one ingredient, the empty state disappears.
- We show explainer copy below the CTA to remind the user that a minimum of 3 ingredients is required.
- The user can either select the suggestion pill or finish typing their ingredient and hit enter. If the user has typed the ingredient name correctly, the pill will be ‘pressed’ and the ingredient name will appear on their My ingredients list. If the user has typed it incorrectly or we don’t find a match, we will suggest the closest thing to a match from our repository — e.g., if the user enters brocolli, we will suggest broccoli and broccolini.



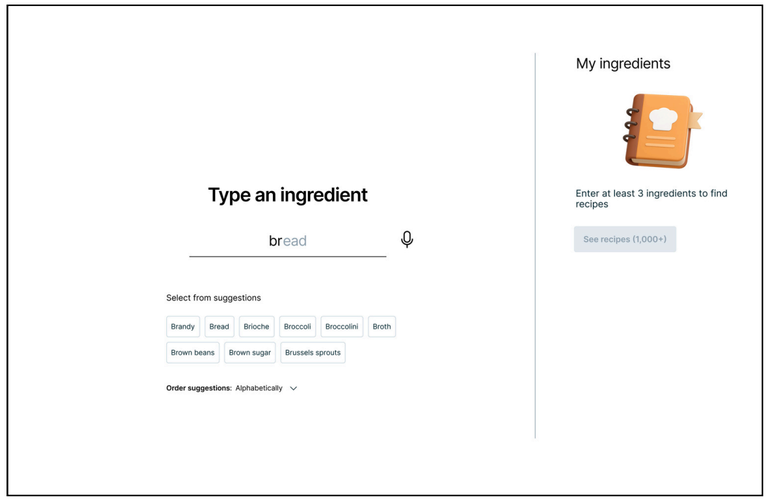
Search page

- Once the user has entered at least 3 ingredients, the primary CTA becomes active.
- We show how many recipes contain their chosen ingredients, limiting it 1,000+.
- As users enter more ingredients, we show how many recipes match their list.
- Once the user enters at least one ingredient, we can update our empty state suggestions to show foods that go with their selected ingredient.



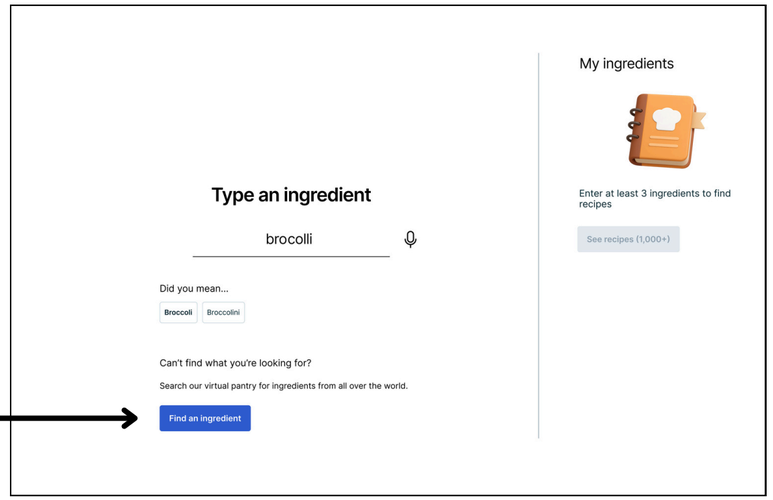
Search page

- Users can order suggestions by popularity in their region or alphabetically. This is helpful, for example, you type a broader food type such as ‘Beans’ and see a list of all types of beans in alphabetical order, so it’s easier to find what you’re looking for.



Search page

- This shows suggestions ordered alphabetically. It places *broccoli* and *broccolini* next to each other, making it easier to scan. If these were ordered by popularity, broccoli would be near the start of the list and broccolini would be near the end.



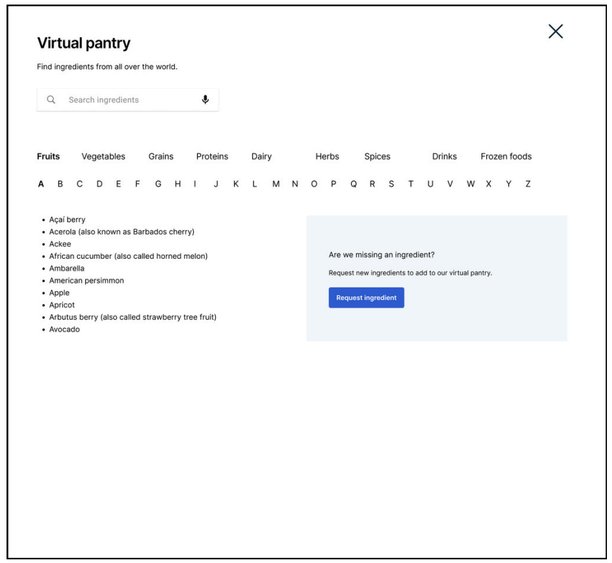
Search page

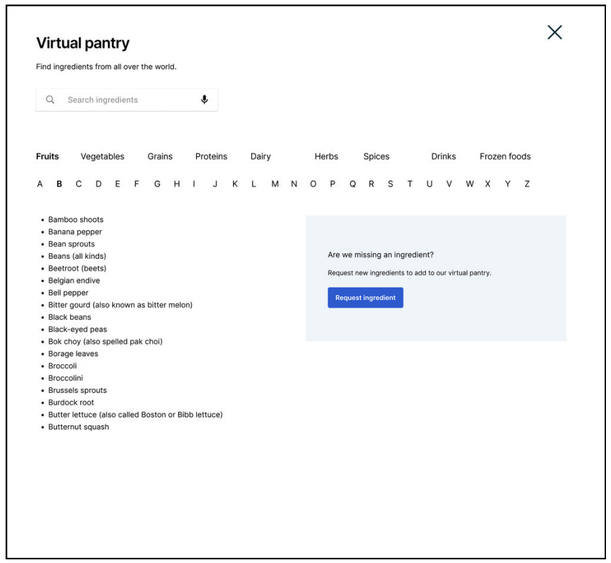
- If none of our suggestions matches what the user is looking for, we offer the option to search our virtual pantry.
- For the button, I chose to use value-driven language instead of action-driven language (e.g., *Search ingredients)*, as these tend to be more effective. The examples below are borrowed from Kinneret Yifrah's *Microcopy* course.

Don't amplify the act of proceeding.
Amplify the value of proceeding.

Examples:

A	B	Conversion	Reason
Order information	Get information	B = +38%	A emphasises the effort, B emphasises the benefit
Sign up	Sign up & Get the Best Daily Tips	B = +31%	B tells users what they'll get out of signing up
Join us	Make money flipping websites	B = +33%	B is blunt but it contains the reason



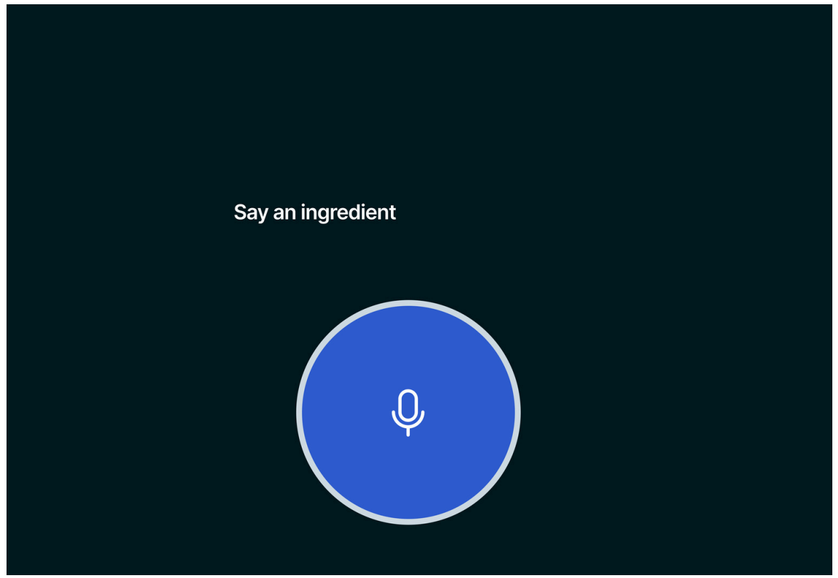


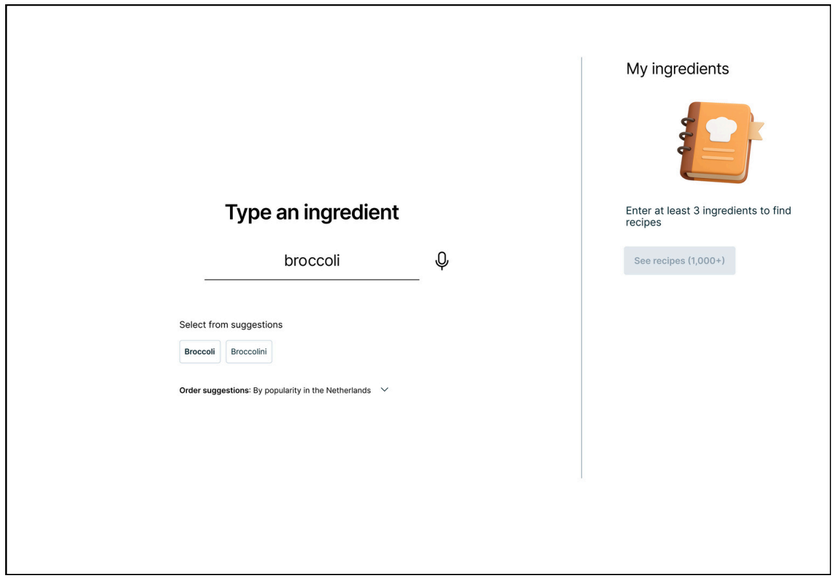
Virtual pantry

- In the virtual pantry, users can search ingredients by text input or by alphabetical index with filters.

Virtual pantry

- If a user's ingredient is not available in our virtual pantry, we offer users a means of requesting its addition.
- We process each ingredient request manually, investigate the ingredient, assess whether there are any other names it goes by and, if so, whether we already have it under another name in our pantry. We then update our pantry and suggestions based on these user insights.
- If we like, we can record users' emails when they request a new ingredient, then follow up with them once we add it to our pantry. This could create some beautiful moments of connection. People love bonding over food!





Audio search

- Users can select the microphone and say the ingredient name. This accessibility feature reduces the need for users to be good spellers. It’s perfect for people with dyslexia, for example.

Task 4. Explain non-refundable rates to users

Explain non-refundable rates

Words don’t exist in isolation. By showing the non-refundable rate alongside its counterpart, refundable rate, users can clearly understand the pros and cons of each pricing option.

Designs and words never exist in isolation. When it comes to the term *non-refundable rate*, part of the broader context involves how this rate factors into the bigger pricing options picture.

After googling what non-refundable rate means — because it wasn’t immediately obvious to me (although I did guess correctly!) — I learnt that businesses offer non-refundable rates in conjunction with refundable rates. This context is key to explaining to users the meaning and value of each rate.

Instead of explaining the *non-refundable rate* in isolation, I chose to display it next to its counterpart (refundable rate), to clearly convey to users the pros and cons of each rate. I think this is the best way to help users understand their options and choose whichever one works best for their situation.

Explain non-refundable rates

Key choices

- The term ‘non-refundable rate’ only has meaning in contrast to its counterpart. By showing both pricing options in context, we can more easily illustrate the advantages of each. We can also use a comparison table for users to effectively compare options.

Audience

- Everyone, including people with learning difficulties and non-native English speakers
- People using the UI in other languages, so localization is a key priority

Considerations

- Depending on what users understand / misunderstand, are we tied to the term ‘non-refundable rate’, or can we suggest alternatives?
- To suggest alternatives, I would need to see the bigger picture. I think for the sake of this task I will need to make some assumptions.
- Do we as a business want to nudge users towards either option? If so, which one? There are ways to make each option appear more or less desirable. Right now I have tried to give them equal pros and cons.

Assumptions

- For each listing, users can choose between:
  - Refundable rate: Higher, but you’ll get a full or partial refund depending on the details of your cancellation.
  - Non-refundable rate: Lower, but no refund if you cancel.
- This means that we don’t have listings where non-refundable rate is the only pricing option.
- We do not have any pre-existing cancellation policies that prevent me from shifting how we frame these rates. (This assumption applies to Alternative versions 1 and 2.)

Key terms

- Flexible
- Non-refundable
- Pricing options

Benchmarks

- Booking.com flights has implemented a nice ‘No refund if you cancel’ line.
  - What I like:
    - Plain language, no jargon
    - Highlights when this situation applies (if you cancel)
  - What I dislike:
    - Unclear whether this is the only option
    - Unclear what the benefit is for me
    - Can you buy additional cancellation insurance?
- Airbnb property page
- HousingAnywhere property page

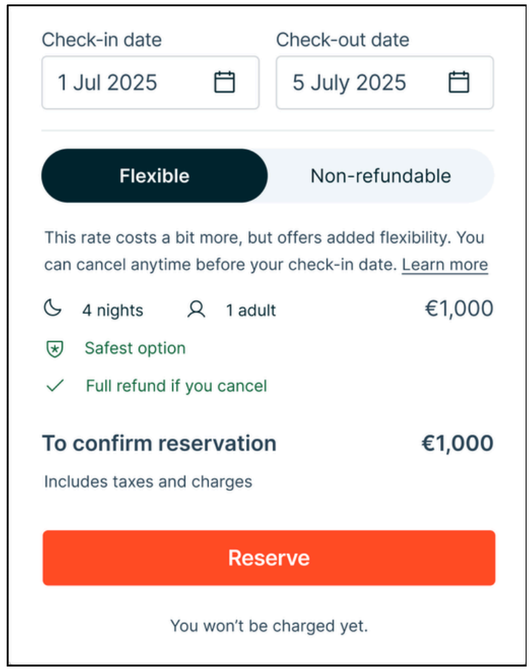
Questions I asked while designing

- What do users struggle with, precisely?
  - What is a non-refundable rate?
  - Is the whole cost non-refundable, or only a portion of it?
  - What is the benefit of a non-refundable payment? Is it lower than a refundable or partially-refundable payment?
  - Do users choose between a non-refundable rate (lower, less flexible) and a refundable rate (higher, but with full or partial refund)?
  - How does this term fit into the bigger picture?
  - What specific part of the phrase do users not understand?
    - ‘Non-refundable’?
    - ‘Rate’?
    - ‘Non-refundable rate’?
- What mental models do users bring to this scenario? Do we need to stick with ‘non-refundable rates’ as the key term, or can we reframe this as a booking’s cancellation policy, as this is effectively what it is?

Metrics for measuring success

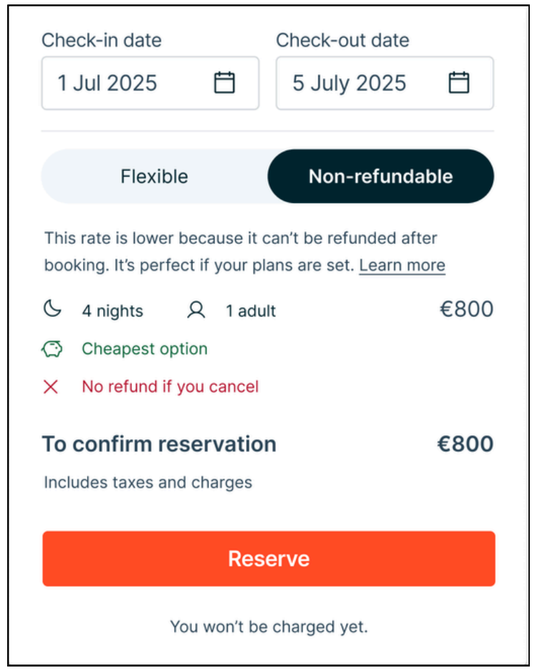
- Click-through rate on the rate or info tooltip
- Drop-off rate on the booking funnel (especially at the pricing step)
- A/B test results: comparing current wording vs. revised copy for booking conversion rate
- User comprehension scores from moderated or unmoderated testing
- Customer support queries related to cancellations or refund misunderstandings (pre/post-launch)

Official submission – Flexible and non-refundable



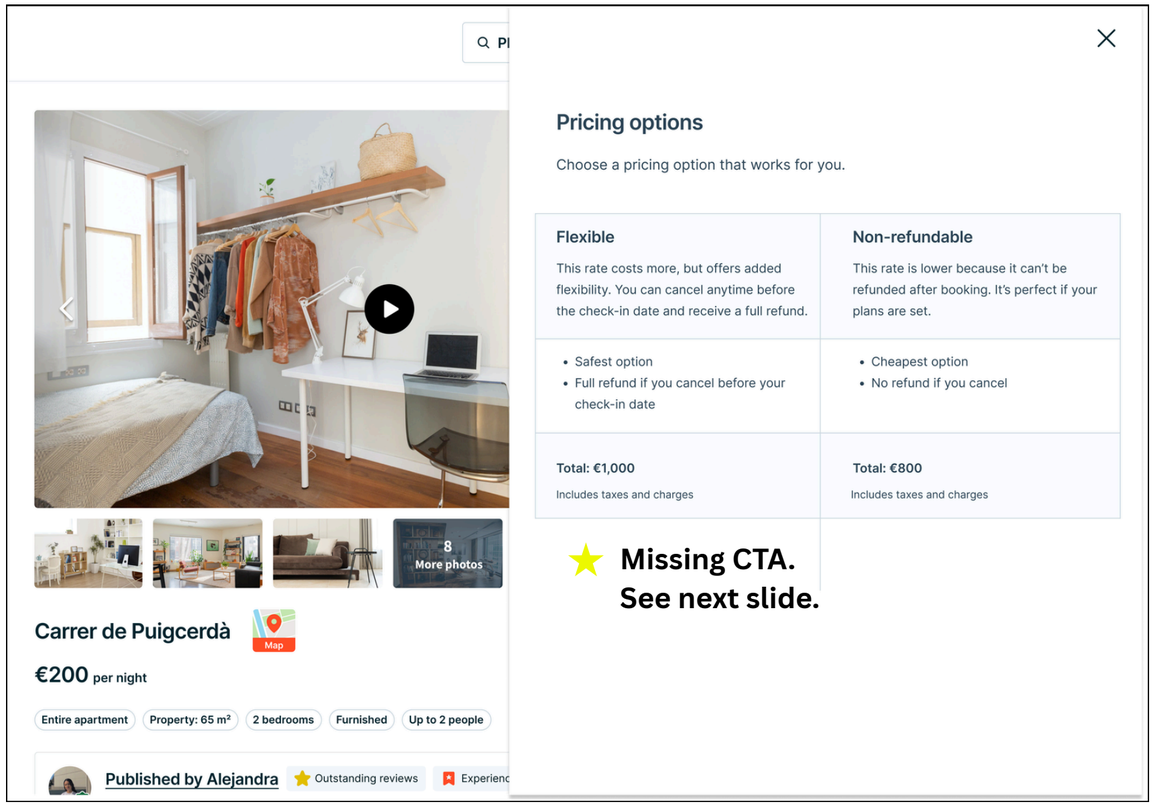
Property page — Non-refundable option selected

- In the price breakdown, the user can select pricing option: *Flexible* or *Non-refundable*.
- As they toggle between options, we show:
  - Different explainer copy for each pricing option.
  - Benefits of each cancellation type: i.e., *cheapest option* or *safest option*.
- The total price to confirm the reservation, which changes based on their cancellation policy.
- Highlight *Cheapest option* and *No refund if you cancel*, for extra visibility.



Property page — Flexible option selected

- Highlight *Flexible cancellation* and *Safest option* for extra visibility.
- It's important to tell users that they won't be charged yet. For new users, this is fairly obvious, as they haven't entered payment details so we can't charge them. But for returning users this is crucial information.



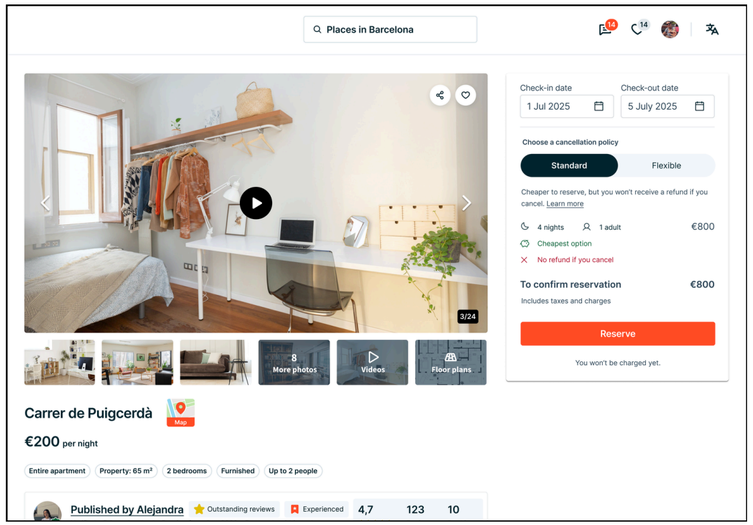
Learn more

- When the user clicks *Learn more*, we show a side-by-side comparison, so they can easily compare all the details and decide which option is best for them.
- We offer users a way to continue with either option. If a user isn't ready to continue, they can X out or click outside the panel to close.

Concerns:

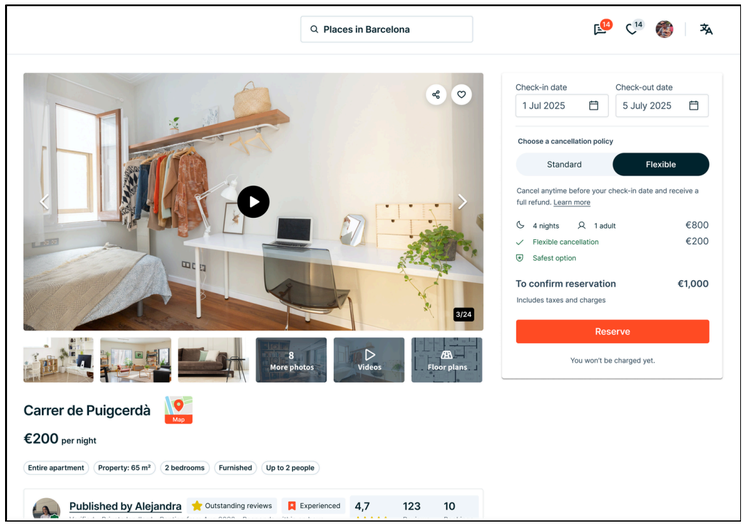
- *Continue with Flexible* and *Continue with Non-refundable* are long in EN and may pose problems for button localization. E.g.,
  - DE: Weiter mit Nicht erstattungsfähig
  - RU: Выбрать невозвратный план
- Consider *Choose flexible* and *Choose Non-refundable*, or reconsider CTAs altogether.

Alternative 1 – Cancellation policies



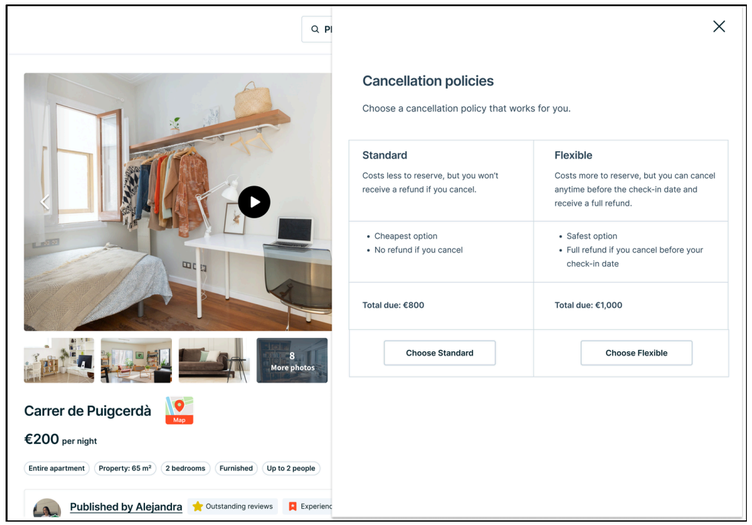
Property page — Standard cancellation selected

- In the price breakdown, the user can select their cancellation policy.
- As they toggle between *Standard* and *Flexible*, we show:
  - Different explainer copy for each cancellation type.
  - Benefits of each cancellation type: i.e., *cheapest option* or *safest option*.
- The total price to confirm the reservation, which changes based on their cancellation policy.
- Highlight *No refund if you cancel* in red, for extra visibility.



Property page — Flexible cancellation selected

- It's important to tell users that they won't be charged yet. For new users, this is fairly obvious, as they haven't entered payment details so we can't charge them. But for returning users this is crucial information.



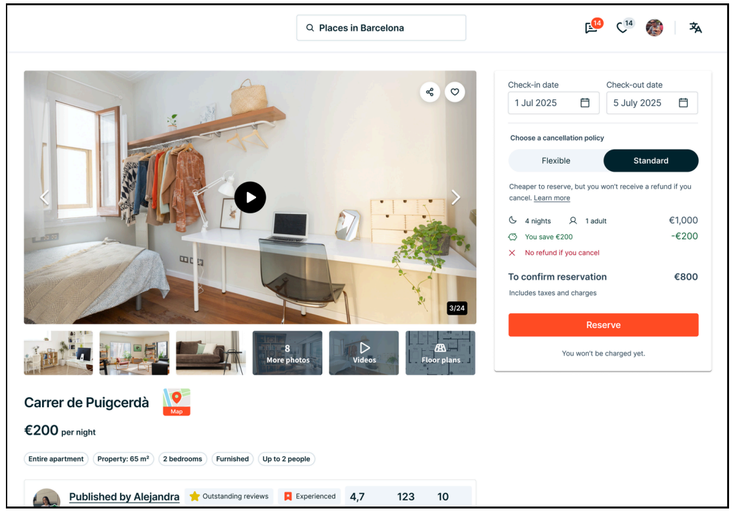
Property page side panel popover

- When user selects *Learn more*, show a side-by-side comparison on a same-page popover panel.
- This gives uses a clear overview of the pros and cons of each option.
- If users aren't ready to make a choice, they can X out or click outside the panel to close.

Alternative 2 – Change framing of Alternative 1

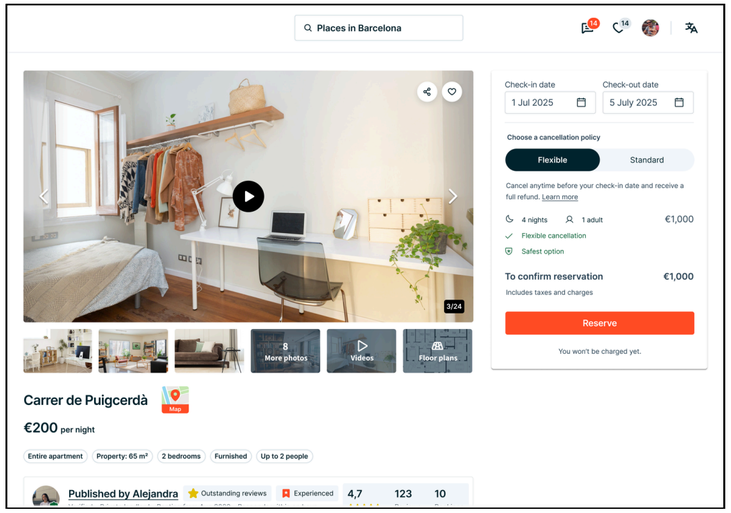
I experimented with changing the frame. In the previous version, *Standard* is the “default” option (base price) and *Flexible* costs extra.

In this version, I reversed the logic. Now *Flexible* is the base price, but users can “save money” by selecting *Standard*.



Property page — Standard cancellation selected

- Now *Flexible* is the base price, but users can “save money” by selecting *Standard*.



Property page — Flexible cancellation selected

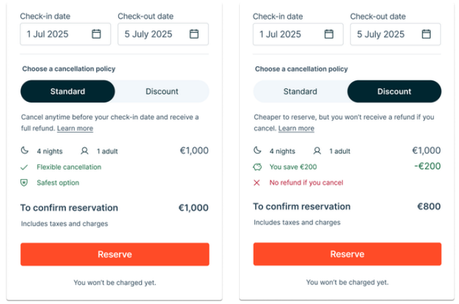
- *Flexible* is now the default price. It doesn’t feel like it costs extra to the user. Rather, *Standard* is discounted.
- For me, however, this brings into question the idea of *Standard* being standard. It’s not really the standard rate if it’s discounted.

Notes

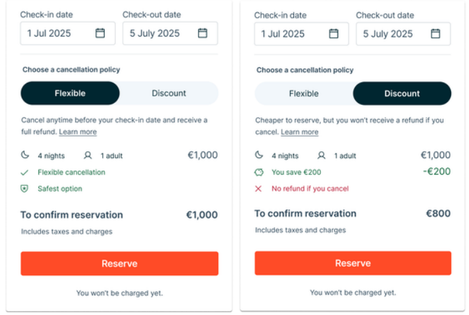
I think it would be interesting to team up with UXR to test how users respond to these two ways of framing.

IWe should also test naming. For example:

- What happens if we rename *Flexible* to *Standard*, and *Standard* to *Discount* or *Discounted*?



- Or what if we call each option what it is: *Flexible* and *Discount* or *Discounted*? This benefit-based naming may be clearest to users.



There are many directions we can go, and it’s difficult to know which one is best without validating. I’d like to discuss this with PMs, Designers, and UX Researches to explore options, co-design, and test the best ones.