

User Needs + Defining Success

Even the best AI will fail if it doesn't provide unique value to users.

This chapter covers:

Which user problems is AI uniquely positioned to solve?

How can we augment human capabilities in addition to automating tasks?

How can we ensure our reward function optimizes AI for the right thing?

Want to drive discussions, speed iteration, and avoid pitfalls? [Use the worksheet](#).

What's new when working with AI

When building any product in a human-centered way, the most important decisions you'll make are: Who are your users? What are their values? Which problem should you solve for them? How will you solve that problem? How will you know when the experience is "done"?

In this chapter, we'll help you understand which user problems are good candidates for AI and how to define success. Key considerations:

- ① **Find the intersection of user needs & AI strengths.** Solve a real problem in ways in which AI adds unique value.
- ② **Assess automation vs. augmentation.** Automate tasks that are difficult, unpleasant, or where there's a need for scale; and ideally ones where people who currently do it can agree on the "correct" way to do it. Augment tasks that people enjoy doing, that carry social capital, or where people don't agree on the "correct" way to do it.
- ③ **Design & evaluate the reward function.** The "reward function" is how an AI defines successes and failures. Deliberately design this function with a cross-functional team, optimizing for long-term user benefits by imagining the downstream effects of your product. Share this function with users when possible.

① Find the intersection of user needs & AI strengths

Like any human-centered design process, the time you spend identifying the right problem to solve is some of the most important in the entire effort. Talking to people, looking through data, and observing behaviors can shift your thinking from technology-first to people-first.

The first step is to identify real problems that people need help with. There are many ways to discover these problems and existing resources online to help you get started. We recommend looking through [IDEO's Design Kit methods section](#) for examples of how to find problems and corresponding user needs.

For our example app RUN, user needs might be:

- The user wants to get more variety in their runs so they don't get bored and quit running.
- The user wants to track their daily runs so that they can get ready for a 10k in six months.
- The user would like to meet other runners at their skill level so they can stay motivated to keep running.

You should always build and use AI in responsible ways. When deciding on which problem to solve, take a look at the [Google AI Principles](#), and [Responsible AI Practices](#) for practical steps, to ensure you're building with the greater good in mind. For starters, make sure to get input from a diverse set of users early on in your product development process. Hearing from many different points of view can help you avoid missing out on major market opportunities or creating designs that unintentionally exclude specific user groups.

Map existing workflows

Mapping the existing workflow for accomplishing a task can be a great way to find opportunities for AI to improve the experience. As you walk through how people currently complete a process, you'll better understand the necessary steps and identify aspects that could be automated or augmented. If you already have a working AI-powered product, test your assumptions with user research. Try letting people use your product (or a ["Wizard of Oz" test](#)) to automate certain aspects of the process, and see how they feel about the results.

Decide if AI adds unique value

Once you identify the aspect you want to improve, you'll need to determine which of the possible solutions require AI, which are meaningfully enhanced by AI, and which solutions don't benefit from AI or are even degraded by it.

It's important to question whether adding AI to your product will improve it. Often a rule or heuristic-based solution will work just as well, if not better, than an AI version. A simpler solution has the added benefit of being easier to build, explain, debug, and maintain. Take time to critically consider how introducing AI to your product might improve, or regress, your user experience.

To get you started, here are some situations where an AI approach is probably better than a rule-based approach, and some in which it is not.

When AI is probably better

- **Recommending different content to different users.** Such as providing personalized suggestions for movies to watch.
- **Prediction of future events.** For example, showing flight prices for a trip to Denver in late November.
- **Personalization improves the user experience.** Personalizing automated home thermostats makes homes more comfortable and the thermostats more efficient over time.
- **Natural language understanding.** Dictation software requires AI to function well for different languages and speech styles.
- **Recognition of an entire class of entities.** It's not possible to program every single face into a photo tagging app — it uses AI to recognize two photos as the same person.
- **Detection of low occurrence events that change over time.** Credit card fraud is constantly evolving and happens infrequently to individuals, but frequently across a large group. AI can learn these evolving patterns and detect new kinds of fraud as they emerge.
- **An agent or bot experience for a particular domain.** Booking a hotel follows a similar pattern for a large number of users and can be automated to expedite the process.

- **Showing dynamic content is more efficient than a predictable interface.** AI-generated suggestions from a streaming service surface new content that would be nearly impossible for a user to find otherwise.

When AI is probably not better

- **Maintaining predictability.** Sometimes the most valuable part of the core experience is its predictability, regardless of context or additional user input. For example, a “Home” or “Cancel” button is easier to use as an escape hatch when it stays in the same place.
- **Providing static or limited information.** For example, a credit card entry form is simple, standard, and doesn’t have highly varied information requirements for different users.
- **Minimizing costly errors.** If the cost of errors is very high and outweighs the benefits of a small increase in success rate, such as a navigation guide that suggests an off-road route to save a few seconds of travel time.
- **Complete transparency.** If users, customers, or developers need to understand precisely everything that happens in the code, like with Open Source Software. AI can’t always deliver that level of explainability.
- **Optimizing for high speed and low cost.** If speed of development and getting to market first is more important than anything else to the business, including the value that adding AI would provide.
- **Automating high-value tasks.** If people explicitly tell you they don’t want a task automated or augmented with AI, that’s a good task not to try to disrupt. We’ll talk more about how people value certain types of tasks below.

Key concept

Instead of asking “Can we use AI to _____?”, start exploring human-centered AI solutions by asking:

- How might we solve _____ ?
- Can AI solve this problem in a unique way?

Apply the concepts from this section in Exercise 1 in the worksheet

② Assess automation vs. augmentation

When you've found the problem you want to solve and have decided that using AI is the right approach, you'll then evaluate the different ways AI can solve the problem and help users accomplish their goals. One large consideration is if you should use AI to automate a task or to augment a person's ability to do that task themselves.

Some tasks, people would love for AI to handle, but there are many activities that people want to do themselves. In those latter cases, AI can help them perform the same tasks, but faster, more efficiently, or sometimes even more creatively. When done right, automation and augmentation work together to both simplify and improve the outcome of a long, complicated process.

When to automate

Automation is typically preferred when it allows people to avoid undesirable tasks entirely or when the time, money, or effort investment isn't worth it to them. These are usually tasks people are happy to delegate to AI as they don't require oversight, or they can be done just as well by someone (or something) else. Successful automation is often measured by the following:

- Increased efficiency
- Improved human safety
- Reduction of tedious tasks
- Enabling new experiences that weren't possible without automation

Automation is often the best option for tasks that supplement human weaknesses with AI strengths. For example, it would take a human a very long time to sort through their photo library and group pictures by subject. AI can do that quickly and easily, without constant feedback.

Consider automating experiences when:

People lack the knowledge or ability to do the task

There are many times when people would do something if they knew how, but they don't so they can't. Or they technically know how, but a machine is much better suited to the task — such as searching thousands of rows in a spreadsheet to find particular value.

There are also often temporary limitations on people, like needing to complete a task quickly, that can lead to preferences for giving up control. For example, one might save time by using the automated setting on their rice cooker when they're rushed to make dinner during the week, but make their sushi rice by hand over the weekend.

Tasks are boring, repetitive, awkward, or dangerous

There's little value in attempting to edit a document you wrote without using spell-check. It's unwise to check for a gas leak in a building using your own nose when you could use a sensor to detect the leak. In both of these situations, most people would prefer to give up control to avoid tasks that don't provide them value.

Even when you choose to automate a task, there should almost always be an option for human oversight — sometimes called “human on the loop” — and intervention if necessary. Easy options for this are allowing users to preview, test, edit, or undo any functions that your AI automates.

When to augment

When building AI-powered products, it's tempting to assume that the best thing you can do for your users is automate tasks they currently have to do manually. However, there are plenty of situations where people typically prefer for AI to augment their existing abilities and give them “superpowers” instead of automating a task away entirely.

Successful augmentation is often measured by the following:

- Increased user enjoyment of a task
- Higher levels of user control over automation
- Greater user responsibility and fulfillment
- Increased ability for user to scale their efforts
- Increased creativity

Augmentation opportunities aren't always easy to define as separate from automation, but they're usually more complicated, inherently human, and personally valuable. For example, you may use tools that automate part of designing a t-shirt, like resizing your art or finding compatible colors. The design software, in this case, augments the task of t-shirt design, and unlocks limitless silliness and ingenuity. Consider augmenting people's existing abilities when:

People enjoy the task

Not every task is a chore. If you enjoy writing music, you probably wouldn't want an AI to write entire pieces for you. If an algorithm did it for you, you won't get to participate in the creative process you love. However, using something like Magenta Studio could help you during the creative process without taking away the essential humanity of your artistic process.

Personal responsibility for the outcome is required or important

People exchange small favors all the time. Part of doing a favor for someone is the social capital you gain by giving up your time and energy. For tasks like these, people prefer to remain in control and responsible to fulfill the social obligations they take on. Other times, when there is no personal obligation, like paying tolls on roads, an automated system is typically preferred.

The stakes of the situation are high

People often want to, or have to, remain in control when the stakes are high for their role; for example pilots, doctors, or police officers. These can be physical stakes like ensuring someone gets off a tall ladder safely, emotional stakes like telling loved ones how you feel about them, or financial stakes like sharing credit card or banking information. Additionally, sometimes personal responsibility for a task is legally required. In low stakes situations, like getting song recommendations from a streaming service, people will often give up control because the prospect of discovery is more important than the low cost of error.

Specific preferences are hard to communicate

Sometimes people have a vision for how they want something done: a room decorated, a party planned, or a product designed. They can see it in their mind's eye but can't seem to do it justice in words. In these kinds of situations, people prefer staying in control so they can see their vision through. When people don't have a vision or don't have time to invest in one, they are more likely to prefer automation.

Key concept

Below are some example research questions you can ask to learn about how your users think about automation and augmentation:

- If you were helping to train a new coworker for a similar role, what would be the most important tasks you would teach them first?
- Tell me more about that action you just took, about how often do you do that?
- If you had a human assistant to work with on this task, what, if any duties would you give them to carry out?

Apply the concepts from this section in Exercise 2 [in the worksheet](#)

③ Design & evaluate the reward function

Any AI model you build or incorporate into your product is guided by a reward function, also called an “objective function”, or “loss function.” This is a mathematical formula, or set of formulas, that the AI model uses to determine “right” vs. “wrong” predictions. It determines the action or behavior your system will try to optimize for, and will be a major driver of the final user experience.





When designing your reward function, you must make a few key decisions that will dramatically affect the final experience for your users. We’ll cover those next, but remember that designing your reward function should be a collaborative process across disciplines. Your conversations should include UX, Product, and Engineering perspectives at the minimum. Throughout the process, spend time thinking about the possible outcomes, and bounce your ideas off other people. That will help reveal pitfalls where the reward function could optimize for the wrong outcomes.

Weigh false positives & negatives

Many AI models predict whether or not a given object or entity belongs to a certain category. These kind of models are called “binary classifiers”. We’ll use them as a simple example for understanding how AI’s can be right or wrong.

When binary classifiers make predictions, there are four possible outcomes:

- True positives . When the model correctly predicts a positive outcome.
- True negatives . When the model correctly predicts a negative outcome.
- False positives . When the model incorrectly predicts a positive outcome.
- False negatives . When the model incorrectly predicts a negative outcome.

| | | PREDICTION | |
|-----------|----------|---|---|
| | | Positive | Negative |
| REFERENCE | Positive |  True Positive |  False Negative |
| | Negative |  False Positive |  True Negative |

A generic confusion matrix illustrating the two kinds of successes — true positives and true negatives — and two kinds of errors — false positives and false negatives — any AI model can make.

Let's walk through an example using our example app, RUN. Suppose RUN uses an AI model to recommend runs to users. Here's how the different model outcomes would play out:

1. True positives . The model suggested a run the user liked and chose to go on.
2. True negatives . The model did not suggest a run the user would not have chosen to go on.
3. False positives . The model suggested a run to the user that they did not want to go on.
4. False negatives . The model did not suggest a run to the user that they would have wanted to go on if they knew about it.

When defining the reward function, you'll be able to "weigh" outcomes differently. Weighing the cost of false positives and false negatives is a critical decision that will shape your users' experiences. It is tempting to weigh both equally by default. However, that's not likely to match the consequences in real life for users. For example, is a false alarm worse than one that doesn't go off when there's a fire? Both are incorrect, but one is much more dangerous. On the other hand, occasionally recommending a song that a person doesn't like isn't as lethal. They can just decide to skip it. You can mitigate the negative effects of these types of errors by including confidence indicators for a certain output or result.

Consider precision & recall tradeoffs

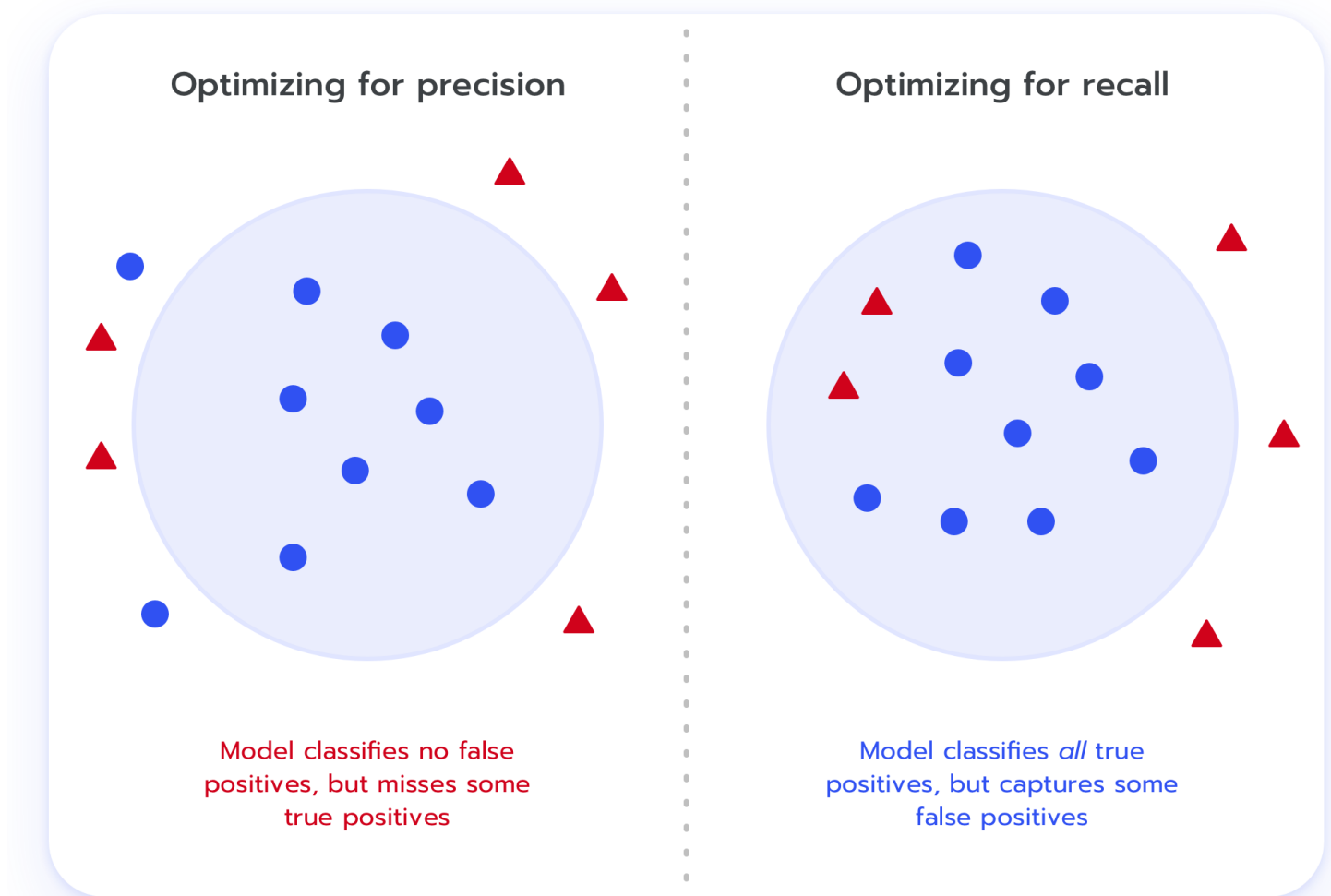
Precision and recall are the terms that describe the breadth and depth of results that your AI provides to users, and the types of errors that users see.

1. Precision refers to **the proportion of true positives correctly categorized out of all the true and false positives**. The higher the precision, the more confident you can be that any model output is correct. However, the tradeoff is that you will increase the number of false negatives by excluding possibly relevant results.

For example, if the model above was optimized for precision, it wouldn't recommend every single run that a user might choose to go on, but it would be highly confident that every run it did recommend would be accepted by the user. Users would see very few if any runs that didn't match their preferences, but they might see fewer suggestions overall.

2. Recall refers to the **proportion of true positives correctly categorized out of all the true positives and false negatives**. The higher the recall, the more confident you can be that all the relevant results are included somewhere in the output. However, the tradeoff is that you will increase the number of false positives by including possibly irrelevant results.

You can think of this as the model recommending every run a user might want to go on, and including other runs the user does not choose to go on. The user however, would always have suggestions for a run, even if those runs didn't match their preferences as well.



A diagram showing the trade-offs when optimizing for precision or recall. On the left, optimizing for precision can reduce the number of false positives but may increase the number of false negatives. On the right, optimizing for recall catches more true positives but also increases the number of false positives.

You'll need to design specifically for these tradeoffs — there's no getting around them. Where along that spectrum your product falls should be based on what your users expect and what gives them the sense of task completeness. Sometimes, seeing some lower confidence results in addition to all of the 100% results can help users trust that the system isn't missing anything. In other cases, showing lower confidence results could lead to users trusting the system less. Make sure to test the balance between precision and recall with your users.

Evaluate the reward function outcomes

The next step is evaluating your reward function. Like any definition of success, it will be tempting to make it very simple, narrow, and immediate. However, this isn't the best approach: when you apply a simple, narrow, and immediate reward function to broad audiences over time, there can be negative effects.

Here are a few considerations when evaluating your reward function:

Assess inclusivity

You'll want to make sure your reward function produces a great experience for all of your users. Being inclusive means taking the time to understand who is using your product and making sure the user experience is equitable for people from a variety of backgrounds and perspectives, and across dimensions such as race, gender, age, or body shape, among many others. [Designing AI with fairness in mind](#) from the beginning is an essential step toward building inclusively. Open source tools like [Facets](#) and the [What-If Tool](#) allow you to inspect your datasets for potential bias. There's more on this in the [Data Collection + Evaluation](#) chapter.

While the Guidebook provides some advice related to fairness, it is not an exhaustive resource on the topic. Addressing fairness in AI is an active area of research. See Google's [Responsible AI Practices](#) for our latest fairness guidance and recommended practices.

Monitor over time

You'll also want to consider the implications of your chosen reward function over time. Optimizing for something like number of shares may seem like a good idea in the short term but over enough time, bombarding users with sharing notifications could create a very noisy experience. Imagine the best individual and collective user experience on their 100th or 1,000th day using your product as well as the first. Which behaviors and experiences should you optimize for in the long run?

Imagine potential pitfalls

Second-order effects are the consequences of the consequences of a certain action. These are notoriously difficult to predict but it's still worth your time to consider them when designing your reward function. One useful question to ask is, "What would happen to our users/their friends & family/greater society if the reward function were perfectly optimized?" The result should be good. For example, optimizing to take people to the best web page for a search query is good, if it's perfect. Optimizing to keep people's attention continuously throughout the day may not provide them benefits in the long run.

Account for negative impact

As AI moves into higher stakes applications and use-cases, it becomes even more important to plan for and monitor negative impacts of your product's decisions. Even if you complete all of the thought exercises in the worksheets, you probably won't uncover every potential pitfall upfront. Instead, schedule a regular cadence for checking your impact metrics, and identifying additional potential bad outcomes and metrics to track.

It's also useful to connect potential negative outcomes with changes to the user experience you could make to address them. For example, you could set the following standards and guidance for you and your team:

- If users' average rate of rejection of smart playlists and routes goes above 20%, we should **check our ML model**.
- If over 60% of users download our app and never use it, we should **revisit our marketing strategy**.
- If users are opening the app frequently, but only completing runs 25% of the time, we'll talk to users about their experiences and potentially **revisit our notification frequency**.

As your product matures, check in your product feedback for negative impacts on stakeholders you didn't consider. If you find some stakeholders are experiencing negative effects on account of your product, talk to them to understand their situation. Based on these conversations, strategize ways to adapt your product to avoid continued negative impact.

An easy way to keep an eye on negative impacts is through social media or alert systems like [Google Alerts](#). Make sure you're listening to your users and identifying potential unintended consequences as early as possible.

Key concept

Everyone on your team should feel aligned on what both success and failure look like for your feature, and how to alert the team if something goes wrong. Here's an example framework for this:

If { **specific success metric** } for __ { your team's AI-driven feature } { drops below/goes above }__ { meaningful threshold } We will { **take a specific action** }.

Apply the concepts from this section in Exercise 4 [in the worksheet](#)

Summary

Aligning your product with user needs is step one in any successful AI product. Once you've found a need, you should evaluate whether using AI will uniquely address the need. From there, consider whether some parts of the experience should be automated or augmented. Lastly, design your reward function to create a great user experience for all your users over the long run.

- ① **Find the intersection of user needs & AI strengths.** Make sure you're solving a real problem in a way where AI is adding unique value. When deciding on which problem to solve, you should always build and use AI in responsible ways. Take a look at the Google AI Principles and Responsible AI Practices for practical steps to ensure you are building with the greater good in mind.
- ② **Assess automation vs. augmentation.** Automate tasks that are difficult or unpleasant, and ideally ones where people who do it currently can agree on the "correct" way to do it. Augment bigger processes that people enjoy doing or that carry social value.
- ③ **Design & evaluate the reward function.** The "reward function" is how an AI defines successes and failures. You'll want to deliberately design this function including optimizing for long-term user benefits by imagining the downstream effects of your product and limiting their potentially negative outcomes.

Want to drive discussions, speed iteration, and avoid pitfalls? [Use the worksheet](#)

Data Collection + Evaluation

Sourcing and evaluating the data used to train AI involve important considerations. This chapter covers:

Does our training dataset have the features and breadth to ensure our AI meets our users' needs?

Should we use an existing training dataset or develop our own?

How can we ensure that raters aren't injecting error or bias into datasets when generating labels?

Want to drive discussions, speed iteration, and avoid pitfalls? [Use the worksheet.](#)

What's new when working with AI

In order to make predictions, AI-driven products must teach their underlying machine learning model to recognize patterns and correlations in data. These data are called training data, and can be collections of images, videos, text, audio and more. You can use existing data sources or collect new data expressly to train your system. For example, you might use a database of dog images from a shelter to train an ML model to recognize common dog breeds.

The training data you source or collect, and how those data are labeled, directly determines the output of your system – and the quality of the user experience. Once you're sure that using AI is indeed the right path for your product (see [User Needs + Defining Success](#)) consider the following:

- ① **Translate user needs into data needs.** Determine the type of data needed to train your model. You'll need to consider predictive power, relevance, fairness, privacy and security.
- ② **Source your data responsibly.** Whether using pre-labeled data or collecting your own, it's critical to evaluate your data and their collection method to ensure they're appropriate for your project.
- ③ **Design for raters & labeling.** For supervised learning, having accurate data labels is crucial to getting useful output from your model. Thoughtful design of rater instructions and UI flows will help yield better quality labels and therefore better output.
- ④ **Tune your model.** Once your model is running, interpret the ML output to ensure it's aligned with product goals and user needs. If it's not, then troubleshoot: explore potential issues with your data.

① Translate user needs into data needs

Datasets that can be used to train AI models contain examples, which contain one or more features, and possibly labels.

| | | FEATURES | | | | |
|----------|-----------|------------------------|-------------|-----------|--------|--------|
| | Runner ID | Run | Runner time | Elevation | Fun | |
| EXAMPLES | AV3DE | Boston Marathon | 03:40:32 | 1,300ft | Low | LABELS |
| | X8KGF | Seattle Oktoberfest 5k | 00:35:40 | 0ft | High | |
| | BH9IU | Houston half-marathon | 02:01:18 | 200ft | Medium | |

The scope of features, the quality of the labels, and representativeness of the examples in your training dataset are all factors that affect the quality of your AI system.

The table above contains data about marathons that an app could use to train an ML model to predict how enjoyable a given race will be. Here's how examples, features and labels could affect the quality of that model:

Examples

If examples used to train the run recommendation algorithm only come from elite runners, then they would likely not be useful in creating an effective model to make predictions for a wider user base. However, they may be useful in creating a model geared towards elite runners.

Features

If the elevation gain feature was missing from the dataset, then the ML model would treat a 3.0 mile uphill run equally to a 3.0 downhill mile run, even though the human experience of these is vastly different.

Labels

Labels that reveal the subjective experience of the runners are necessary to help the system to identify the features that are most likely to result in a fun run.

When deciding which examples, features, and labels are needed to train your ML model, work through your data needs on a conceptual level, as shown in the example below.

The example shows the data needs breakdown for a product that aims to solve the user need of “I want to fit runs into my busy schedule.”

Match user needs with data needs

- **User:** Runners
- **User need:** Run more often
- **User action:** Complete runs with or while using the app
- **ML system output:** What routes to suggest and when to suggest them
- **ML system learning:** Patterns of behavior around accepting run prompts, completing runs, and improving consistency
- **Datasets needed:** Running data from the app, demographic data, physiological data, and local geographic data
- **Key features needed in dataset:** Runner demographics, time of day, run completion rate, pace, distance ran, elevation gained, heart rate
- **Key labels needed in dataset:** Runner acceptance or rejection of app suggestion, user generated feedback as to why suggestion was rejected and enjoyment of recommended runs

You can learn more about the basics of ML models in [resources](#). Once you have an idea of the type of data you will need, use [Google’s AI Principles](#) and [Responsible AI Practices](#) as a framework to work through key considerations, including the ones described below.

Manage privacy & security

As with any product, protecting user privacy and security is essential. Even in the running-related example above, the physiological and demographic data required to train this model could be considered sensitive.

There are a number of important questions that arise due to the unique nature of AI and machine learning. Below are two such questions, but you should discuss these and others with privacy and security experts on your team.

What limits exist around user consent for data use?

When collecting data, a best practice, and a legal requirement in many countries, is to give users as much control as possible over what data the system can use and how data can be used. You may need to provide users the ability to opt out or delete their account. Ensure your system is built to accommodate this.

Is there a risk of inadvertently revealing user data? What would the consequences be?

For example, though an individual's health data might be private and secure, if an AI assistant reminds the user to take medication through a home smart speaker, this could partially reveal private medical data to others who might be in the room.



Local routes

1. Elevation matters

1.2 mi

Steep hills and uneven terrain. Great for advanced runners.

Runner: "This one is a burner!"

2. Canal run

2.3 mi

Beautiful scenery and a nice leisurely incline make this route unbeatable

Runner: "Wow, such a great view"

3. Rose path

0.7 mi

This path goes right by the park. Be sure to stop and smell the roses.

Runner: "Perfect for pics"



Local routes

1. Elevation matters

1.2 mi

Steep hills and uneven terrain. Great for advanced runners.

Diane Garza: "This one is a burner!"

2. Canal run

2.3 mi

Beautiful scenery and a nice leisurely incline make this route unbeatable

Diane Garza: "Wow, such a great view"

3. Rose path

0.7 mi

This path goes right by the park. Be sure to stop and smell the roses.

Diane Garza: "Perfect for pics"

Aim for

Take extra steps to protect privacy (anonymize names for example, even if people agreed to have their name used in community reviews) when personal details (such as where people live) could be exposed as part of AI recommendations or predictions. [Learn more](#)

Avoid

Don't assume basic data policies are enough to protect personal privacy. In this case, the runner agreed to expose her name in community reviews, but because she often starts runs from the same spot, another user could infer where she lives.

Balance underfitting & overfitting

To build products to work in one context, use datasets that are expected to reliably reflect that context. For example, for a natural language understanding model meant to work for speech, it wouldn't be helpful to use words that users type into a search engine as training data — people don't type searches the same way they talk.

If your training data aren't properly suited to the context, you also increase the risk of `overfitting` or `underfitting` your training set. Overfitting means the ML model is tailored too specifically to the training data, and it can stem from a variety of causes. If an ML model has overfit the training data, it can make great predictions on the training data but performs worse on the test set or when given new data.

Models can also make poor predictions due to underfitting, where a model hasn't properly captured the complexity of the relationships among the training dataset features and therefore can't make good predictions with training data or with new data.

There are many [resources](#) that can help the software engineers and research scientists on your team with understanding the nuances of training ML models so you can avoid overfitting and underfitting. But first, involve everyone on your product team in a conceptual discussion about the examples, features, and labels that are likely required for a good training set. Then, talk about which features are likely to be most important based on user needs.

Commit to fairness

At every stage of development, human bias can be introduced into the ML model. Data is collected in the real world, from humans, and reflects their personal experiences and biases — and these patterns can be implicitly identified and amplified by the ML model.

While the guidebook provides some advice related to ML fairness, it is not an exhaustive resource on the topic. Addressing fairness in AI, and minimizing unfair bias, is an active area of research. See Google's [Responsible AI Practices](#) for recent ML fairness guidance and recommended practices.

Here are some examples of how ML systems can fail users:

- **Representational harm**, when a system amplifies or reflects negative stereotypes about particular groups

- **Opportunity denial**, when systems make predictions and decisions that have real-life consequences and lasting impacts on individuals' access to opportunities, resources, and overall quality of life
- **Disproportionate product failure**, when a product doesn't work or gives skewed outputs more frequently for certain groups of users
- **Harm by disadvantage**, when a system infers disadvantageous associations between certain demographic characteristics and user behaviors or interests

While there is no standard definition of fairness, and the fairness of your model may vary based on the situation, there are steps you can take to mitigate problematic biases in your dataset.

Use data that applies to different groups of users

Your training data should reflect the diversity and cultural context of the people who will use it. Use tools like [Facets](#) to explore your dataset and better understand its biases. In doing so, note that to properly train your model, you might need to collect data from equal proportions of different user groups that might not exist in equal proportions in the real world. For example, to have speech recognition software work equally on all users in the United States, the training dataset might need to contain 50% of data from non-native English speakers even if they are a minority of the population.

Consider bias in the data collection and evaluation process

There is no such thing as truly neutral data. Even in a simple image, the equipment and lighting used shapes the outcome. Moreover, humans are involved with data collection and evaluation, and so, as with any human endeavor, their output will include human bias. See more in the section on labeling below.

For example, say you're creating a recommendation system to recommend new health and fitness goals to users. If the intent is to set goals that are safely achievable by users with a wide range of baseline fitness levels, it's important that the training dataset includes data from a variety of user types and not just young, healthy people.

For more on the topic of fairness, see Google's Machine Learning Fairness [Overview](#) and [Crash Course](#).

Key concept

Once your team has a high-level understanding of the data your product needs, work through your own translation between specific user needs and the data needed to produce them.

Try to be as specific as possible during this step. This will have a direct impact on which user experiences your team decides to spend your resources on going forward.

Apply the concepts from this section using Exercise 1 [in the worksheet](#)

② Source your data responsibly

Once you've identified the type of training data you need, you will figure out how and where to get it. This could mean using an existing dataset, collecting your own data, or a combination of the two. Make sure that whatever you decide, you have permission to use this data and the infrastructure to keep it safe.

Use existing datasets

It may not be possible to build your dataset from scratch. As an alternative, you may need to use existing data from sources such as [Google Cloud AutoML](#), [Google Dataset Search](#), [Google AI datasets](#), or [Kaggle](#). If you're considering supervised learning, this data may be pre-labeled or you may need to add labels (see more on [labeling](#), below). Be sure to check the terms of use for the dataset and consider whether it's appropriate for your use case.

Before using an existing dataset, take the time to thoroughly explore it using tools like [Facets](#) to better understand any gaps or biases. Real data is often messy, so you should expect to spend a fair amount of time cleaning it up. During this process, you may detect issues, such as missing values, misspellings, and incorrect formatting. For more information on data preparation techniques, check out the developer guidelines on [Data Preparation](#). They can help you make sure that this data will be able to help you deliver the user experience you identified at the outset.

Build your own dataset

When creating your own dataset, it's wise to start by observing someone who is an expert in the domain your product aims to serve — for example, watching an accountant analyze financial data, or a botanist classify plants. If you can interview them as they think or work through the non-ML solution to the problem, you may be able to pick up some insights into which data they look at when making a decision or before taking an action.

You'll also want to research available datasets that seem relevant and evaluate the signals available in those datasets. You may need to combine data from multiple sources for your model to have enough information to learn.

Once you've gathered potential sources for your dataset, spend some time getting to know your data. You'll need to go through the following steps:

1. Identify your data source(s).
2. Review how often your data source(s) are refreshed.
3. Inspect the features' possible values, units, and data types.
4. Identify any outliers, and investigate whether they're actual outliers or due to errors in the data.

Understanding where your dataset came from and how it was collected will help you discover potential issues. The following are common dataset issues to look out for.

Consider formatting

Real data can be messy! A "zero" value could be an actual measured "0," or an indicator for a missing measurement. A "country" feature may contain entries in different formats, such as "US," "USA," and "United States."

While a human can spot the meaning just by looking at the data, an ML model learns better from data that is consistently formatted.

Avoid compounding errors from other ML models

If you're using the output of another ML system as an input feature to train your model, keep in mind that this is a risky data source. Errors associated with this feature will be compounded with your system's overall error, and the further you are from the original training data, the more difficult it will be to identify error sources.

There's more information on determining error sources in the chapter on [Errors + Graceful Failure](#).

Protect personally identifiable information

No matter what data you're using, it's possible that it could contain personally identifiable information. Some approaches to anonymizing data include aggregation and redaction. However, even these approaches may not be able to completely anonymize your data in all circumstances, so consider consulting an expert.

Aggregation is the process of replacing unique values with a summary value. For example, you may replace a list of a user's maximum heartbeats per minute from every day of the month with a single value: their average beats per minute or a categorical high / medium / low label.

Redaction removes some data to create a less complete picture. Such anonymization approaches aim to reduce the number of features available for identifying a single user.

Split your data

And finally, you'll need to split the data into **training** and **test** sets. The model is trained to learn from the training data, and then evaluated with the test data. Test sets are data that your model hasn't seen before — this is how you'll find out if, and how well, your model works. The split will depend on factors such as the number of examples in your dataset and the data distribution.

The training set needs to be large enough to successfully teach your model, and your test set should be large enough that you can adequately assess your model's performance. This is usually the time when developers realize that adequate data can make or break the success of a model. So take the time to determine the most efficient split percentage. A typical split of your dataset could result in: 60% for training, and 40% for testing. See [resources](#) for more details.

Collect live data from users

Once your model has been trained and your product is being used in the real world, you can start collecting data in your product to continually improve your ML model. How you collect this data has a direct impact on its quality. Data can be collected **implicitly** in the background of user activity within your app or **explicitly** when you ask users directly. There are different design considerations for each, which are covered in depth in the [Feedback + Control](#) chapter.

Key concept

ML-driven products require a lot of data to work, and this is often where product teams falter. Getting enough data to both train and test your ML model is critical to delivering a functional product. To get you started, answer the key questions below:

- If you need to **create a new dataset**, how are you planning to collect the data?
- If you have **an existing dataset**, what, if any alterations or additions need to be made for your user population?

Apply the concepts from this section using Exercise 2 [in the worksheet](#)

③ Design for raters & labeling

For supervised learning, accurate data labels are a crucial ingredient for achieving relevant ML output. Labels can be added through automated processes or by people called raters. "Raters" is a generic term that covers a wide variety of contexts, skillsets and levels of specialization. Raters could be:

- **Your users:** providing "derived" labels within your product, for example through actions like tagging photos
- **Generalists:** adding labels to a wide variety of data through crowd-sourcing tools
- **Trained subject matter experts:** using specialized tools to label things like medical images

If people understand what you're asking them to label, and why, and they have the tools to do so effectively, then they're more likely to label it correctly.

Key considerations when designing for labeling:

Ensure rater pool diversity

Think about the perspectives and potential biases of the people in your pool, how to balance these with diversity, and how those points of view could impact the quality of the labels. In some cases, providing raters with training to make them aware of unconscious bias has been effective in reducing biases.

Investigate rater context and incentives

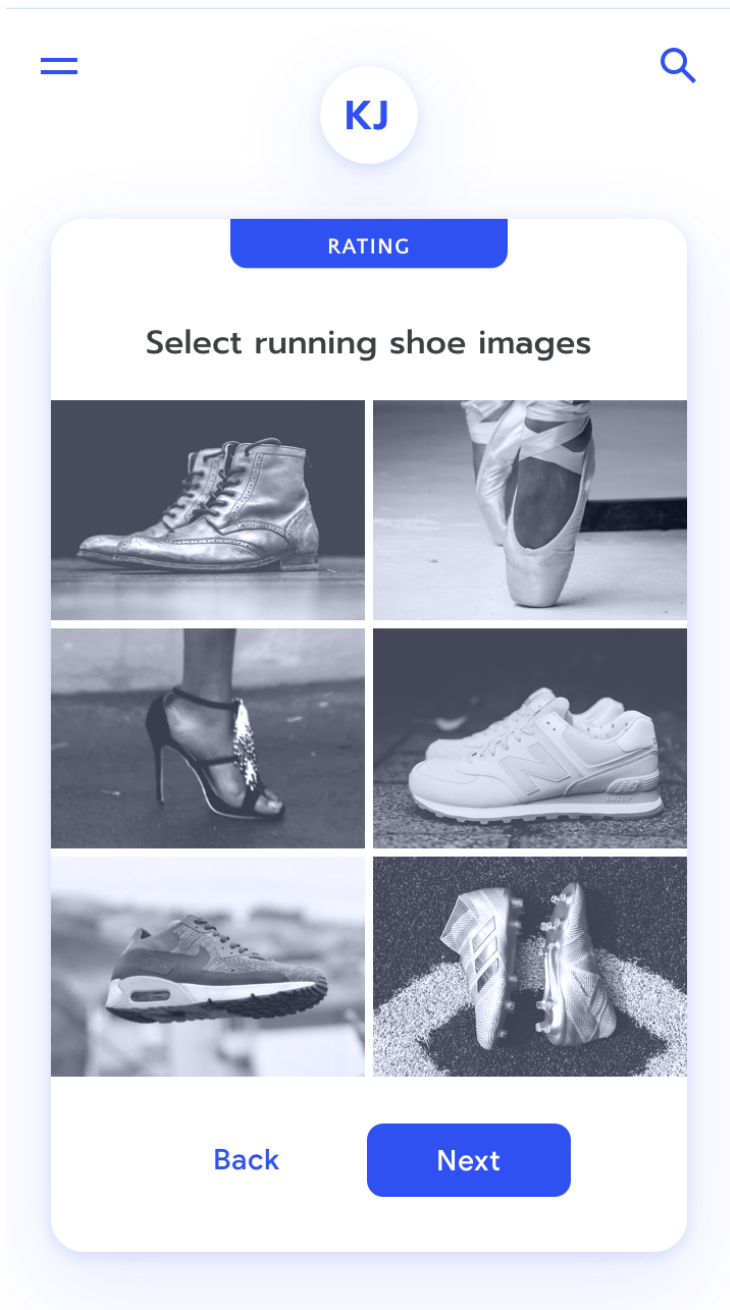
Think through the rater experience and how and why they are doing this task. There's always a risk that they might complete the task incorrectly due to issues like boredom, repetition, or poor incentive design.

Evaluate rater tools

Tools for labeling can range from in-product prompts to specialized software. When soliciting labels in-product, make sure to design the UI in a way that makes it easy for users to provide correct information. When building tools for professional raters, the article [First: Raters](#) offers some useful recommendations like the ones below:

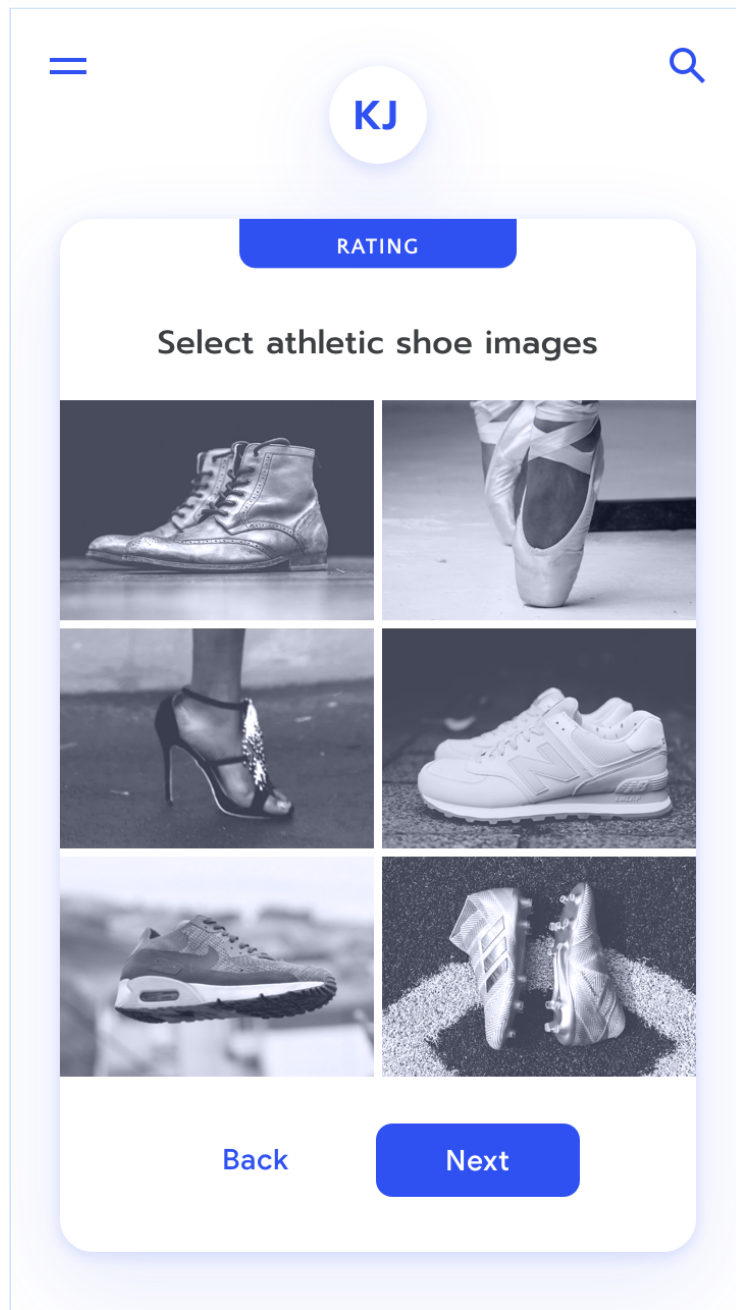
- **Use multiple shortcuts to optimize key flows.** This helps raters move fast and stay efficient
- **Provide easy access to labels.** The full set of available labels should be visible and available to raters for each item they are asked to address. It should be fast and easy in the UI to apply the labels.
- **Let raters change their minds.** Labeling can be complicated. Offer a flexible workflow and support for editing and out-of-sequence changes so that raters can seek second opinions and correct errors.
- **Auto-detect and display errors.** Make it easy to avoid accidental errors with checks and flags.

Once you've collected data from raters, you'll need to conduct statistical tests to analyze inter-rater reliability . A lack of reliability could be a sign that you have poorly-designed instructions.



Aim for

Make rater instructions as specific and simple as possible. Here, the phrase "running shoes" easily rules out the selection of other athletic shoe types like soccer cleats. [Learn more](#)



Avoid

Don't use instructions that can be interpreted multiple ways. Here, someone's subjective definition of "athletic" might or might not include dance shoes, for example.

Key concept

Before designing tools for your raters, research their needs the same way you would think about your end-users. Their motivation and ability to do their job well has a direct impact on everything else you build down the line.

- Who are your raters?
- What is their context and incentive?
- What tools are they using?

Apply the concepts from this section in Exercise 3 [in the worksheet](#)

④ Tune your model

Once your model has been trained with your training data, evaluate the output to assess whether it's addressing your target user need according to the success metrics you defined. If not, you'll need to tune it accordingly. Tuning can mean adjusting the hyperparameters of your training process, the parameters of the model or your reward function, or troubleshooting your training data.

To evaluate your model:

- Use tools like the What If tool to inspect your model and identify blindspots.
- Test, test, test on an ongoing basis.
 - In early phases of development, get in-depth qualitative feedback with a diverse set of users from your target audience to find any “red flag” issues with your training dataset or your model tuning.
 - As part of testing, ensure you've built appropriate and thoughtful mechanisms for user feedback. See more guidance for this in the Feedback + Control chapter.
 - You may need to build custom dashboards and data visualizations to monitor user experience with your system.
 - Be particularly careful to check for secondary effects that you may not have anticipated when determining your reward function, a concept covered in the User Needs + Defining Success chapter.
 - Try to tie model changes to a clear metric of the subjective user experience like customer satisfaction, or how often users accept a model's recommendations.

Once you've identified issues that need to be corrected, you'll need to map them back to specific data features and labels (or lack thereof), or model parameters. This may not be easy or straightforward. Resolving the problem could involve steps like adjusting the training data distribution, fixing a labeling issue or gathering more relevant data. Here's a hypothetical example of how a team might tackle tuning:

Let's say our running app was launching a new feature to calculate calories burned and make recommendations for mid-run changes to help users burn their target number of calories.

During beta testing, it was observed that users receiving these recommendations were far more likely than other users to quit mid-run. Moreover, users who followed the recommendations and completed the run and were less likely to return for a second run within the same week. The product manager originally assumed that this feature was a failure, but after user interviews and a deeper look at the data, it turned out that the algorithm wasn't properly weighting important data when calculating estimated calorie burn like the outside temperature and a user's weight.

After some user research, it became clear that some users were quitting because they didn't trust the calorie calculation and therefore didn't see the point in accepting the app's recommendations for mid-run changes.

The engineering team was able to re-tune the algorithm and launch a successful feature.

Key concept

Tuning is an ongoing process for adjusting your ML model in response to user feedback and issues that arise due to unforeseen circumstances. Tuning never stops, but it is especially important in the early phases of development.

- What is our plan for doing early testing of our model?
- Is our set of early beta users diverse enough to properly test our model?
- What metrics will we use to determine if our tuning is successful?

Apply the concepts from this section on tuning by exploring the [What-if tool](#). Explore more about tuning models in response to user feedback in the [Feedback + Control](#) chapter.

Summary

Data is the bedrock of any ML system. Having responsibly sourced data, from a relevant context, checked for problematic bias will help you build better systems and therefore more effectively address user needs. Key considerations for data collection and evaluation:

- ① **Translate user needs into data needs.** Think carefully as a cross-functional team about what features, labels, and examples you will need to train an effective AI model. Work systematically to break down user needs, user actions, and ML predictions into the necessary datasets. As you identify potential datasets, or formulate a plan to collect them, you'll need to be diligent about inspecting the data, identifying potential bias sources, and designing the data collection methods.
- ② **Source your data responsibly.** As part of sourcing data, you'll need to consider relevance, fairness, privacy and security. You can find more information in Google's AI Principles and Responsible AI Practices. These apply whether you are using an existing dataset or building a new training dataset.
- ③ **Design for raters & labeling.** Correctly labeled data is a crucial ingredient to an effective supervised ML system. Thoughtful consideration of your raters and the tools they'll be using will help ensure your labels are accurate.
- ④ **Tune your model.** Once you have a model, you will need to test and tune it rigorously. The tuning phase involves not only adjusting the parameters of your model, but also inspecting your data – in many cases, output errors can be traced to problems in your data.

Want to drive discussions, speed iteration, and avoid pitfalls? [Use the worksheet](#)

Mental Models

AI-powered systems can adapt over time. Prepare users for change—and help them understand how to train the system. This chapter covers:

Which aspects of AI should we explain to our users?

How should we introduce AI to the user initially—and thereafter?

What are the pros and cons of introducing our AI as human-like?

Want to drive discussions, speed iteration, and avoid pitfalls? [Use the worksheet](#).

What's new when working with AI

A mental model is a person's understanding of how something works and how their actions affect it. People form mental models for everything they interact with, including products, places, and people. Mental models help set expectations for what a product can and can't do and what kind of value people can expect to get from it. Mental models can also serve as bridges between experiences. For example, if you know how to steer a bicycle, you know something about how to steer a motorcycle.

However, users' mental models may not always match what a product can actually do. Mismatched mental models can lead to unmet expectations, frustration, misuse, and product abandonment. Often, product creators unintentionally set incorrect mental models for users by not considering the early user experience with a product or not fully explaining how the product works. Key considerations:

- ① **Set expectations for adaptation.** AI allows for more systems to adapt, optimize, and personalize for users, and probability-based user experiences have become more common over time. Building on the familiarity of existing mental models can help users feel comfortable.
- ② **Onboard in stages.** When introducing users to an AI-powered product, explain what it can do, what it can't do, how it may change, and how to improve it.
- ③ **Plan for co-learning.** People will give feedback to AI products, which will adjust the models and change how people interact with them — which will change the machine learning models further. Users' mental models will similarly change over time.
- ④ **Account for user expectations of human-like interaction.** People are more likely to have unachievable expectations for products that they assume have human-like capabilities. It's important to communicate the algorithmic nature and limits of these products to set realistic user expectations and avoid unintended deception.

① Set expectations for adaptation

Most products are highly static. You can bet that the hammer you buy today will be the same hammer tomorrow. There have also been responsive products for quite a while — products that can adapt how they respond based on user input over time. These systems keep track of whether or not an output was useful, and update how they respond going forward. Using a digital example, many streaming media services will adjust their recommendations based on your interaction with previous ones. This can in turn create an expectation that other products will adapt based on user interactions.

With AI becoming more prevalent in products and experiences, we can expect to see more experiences that change in response to users. One of the biggest opportunities for creating effective mental models of AI products is to build on existing models, while teaching users the dynamic relationship between their input and product output.

Identify existing mental models

Start by thinking about how people currently solve the problem that your product will use AI to address. That existing solution will very likely inform their initial mental model for your product. For example, if people currently label their email messages manually the assumption could be that an AI-powered email product — somehow — follows the same process that the user does. In this case, that would be: read the email, think about the meaning, context, and importance, and attach a label accordingly. Therefore, it may surprise users that the ML model could use other signals—time of day the message was sent or length of the email—to determine the label.

Key concept

To understand the context for the user's relationship to your AI product, work through some of the questions below:

- What is the user trying to do?
- What mental models might they carry over to your product?
- What is the step-by-step process that novice users currently use to accomplish the task?
- How uniform is this process between different users?

Apply the concepts from this section in Exercise 1 in the worksheet

② Onboard in stages

Onboarding is the process of helping a new user or customer get to know a product or service. The onboarding experience begins before users purchase or download your product or even visit your website, and continues indefinitely. As with any product, it's important to consider the different stages of introducing your AI and how mental models form and change along the way.

Introduce and set expectations for AI

After identifying your users' existing mental models, imagine how information your user received before their first interaction with the product — including marketing messages, ads, or manuals — has shaped their expectations. Collaborate closely with your marketing team to develop appropriate and consistent messaging.

Many products set users up for disappointment by promising that “AI magic” will help them accomplish their tasks. This kind of messaging can establish mental models that overestimate what the product can actually do. Though product developers may intend to shield users from a product's complexity, hiding how it works can set users up for confusion and broken trust. It's a tricky balance to strike between explaining specific product capabilities, which can become overly technical, intimidating, and boring, and providing a high-level mental model of your AI-powered product.



RUN

4.5 ★★★★★ (1,348,231)

RUN is a running app that adapts to your fitness levels and designs personalized workouts to help you improve your running.



Download



RUN

2.1 ★★☆☆☆ (4,651)

RUN is the only intelligent running app that uses sophisticated deep neural net machine learning to make your run smarter because we believe in ML driven workouts.



Download

Aim for

Emphasize how the app will benefit users. [Learn more](#)

Avoid

Don't emphasize the underlying technology.

Here are some messaging guidelines for setting the right expectations for your product:

- Be up front about what your product can and can't do the first time the user interacts with it, ideally in your marketing messages.
- Offer examples of how it works that clarify the value of the product.
- Let people know up front that it may need their feedback to improve over time.

- Communicate why people should continue to provide feedback, focusing on the value to them.

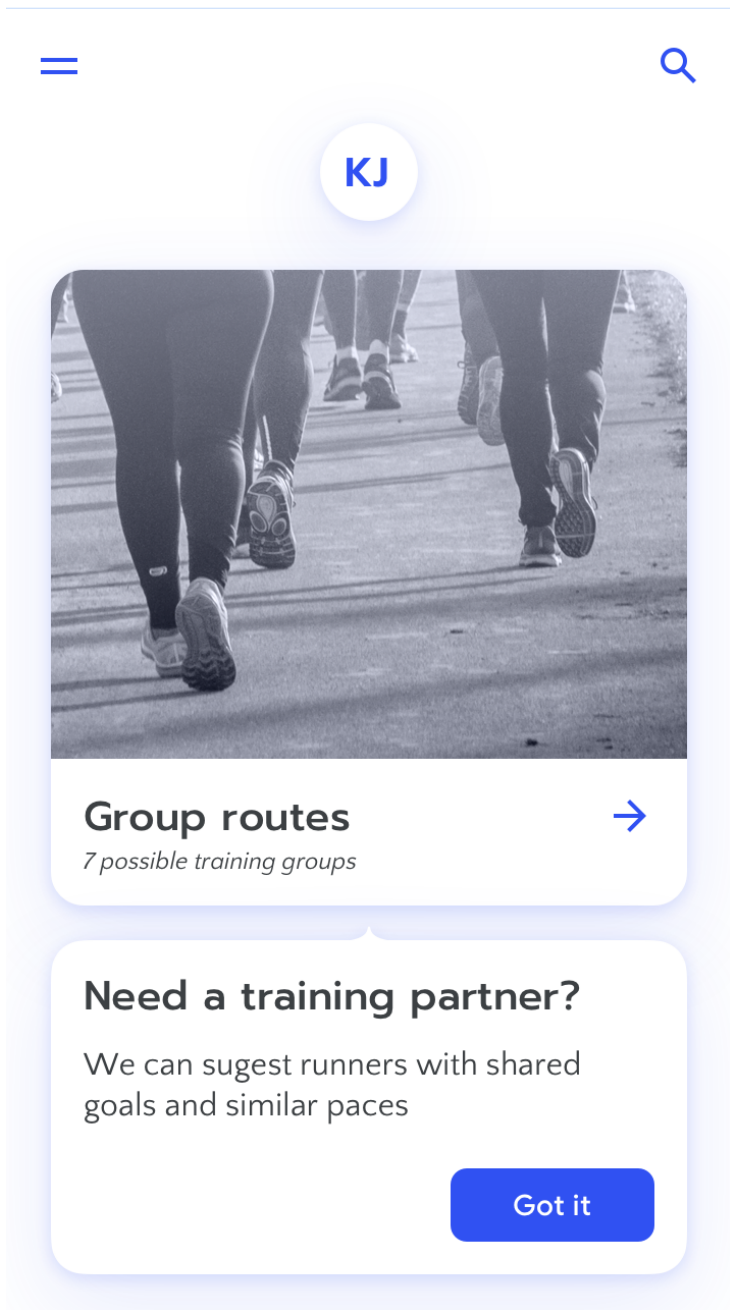
Explain the benefit, not the technology

Often as product creators, we're fascinated by the underlying technologies that make products and experiences possible. This is especially true if we've cracked a hard technical problem for the first time. However, make sure to evaluate which details users need to build a good mental model. If they're interested in understanding the underlying technology of your product, you can always provide more detail with tooltips and progressive disclosure . If you do talk about the AI, focus on how it specifically makes part of the experience better or delivers new value.

See more about explaining AI at the right level of detail in the [Explainability + Trust](#) chapter.

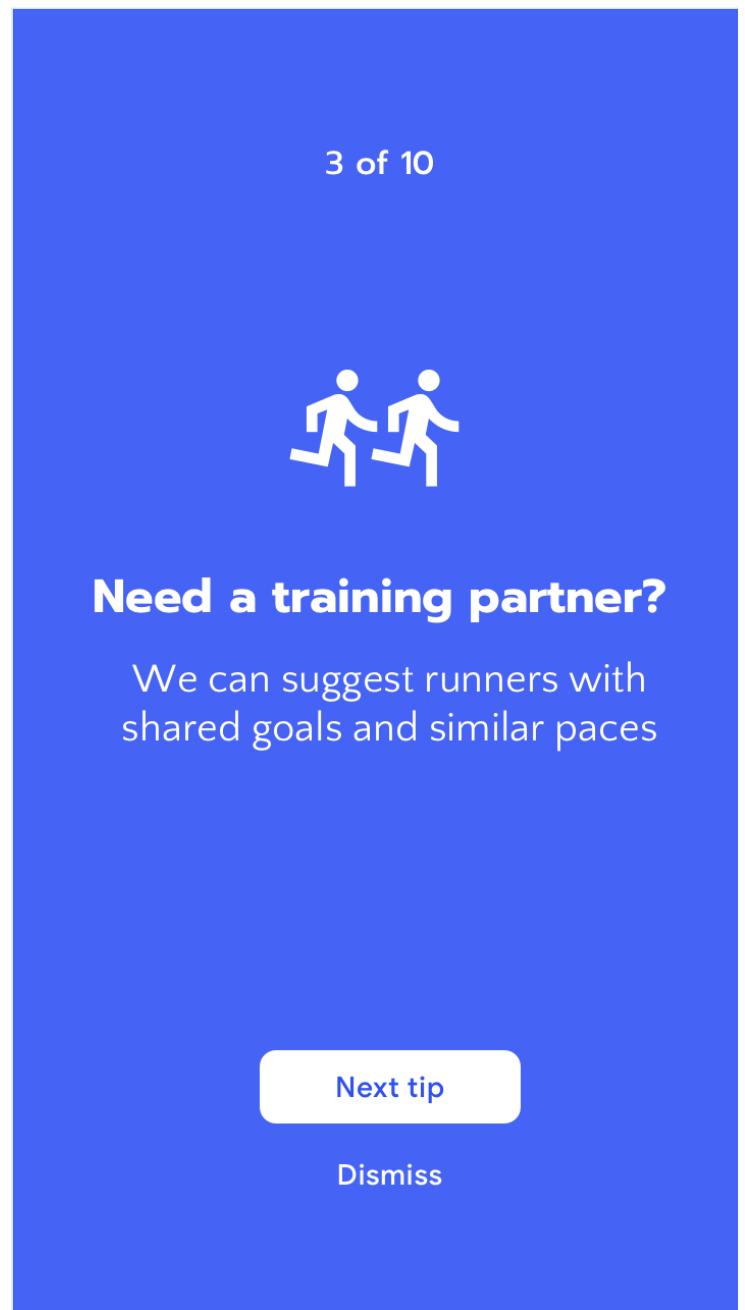
Only introduce new features when needed

As users explore the product, use relevant and actionable “inboarding” messages to help them along. Try to avoid introducing new features when users are busy doing something unrelated. This is especially important if you're updating an existing product with new AI features that change the function or user experience. People learn better when short, explicit information appears right when they need it.



Aim for

Introduce an AI-driven feature at the moment it is relevant to the user. [Learn more](#)



Avoid

Don't introduce AI-driven features as part of a long introductory list of product features.

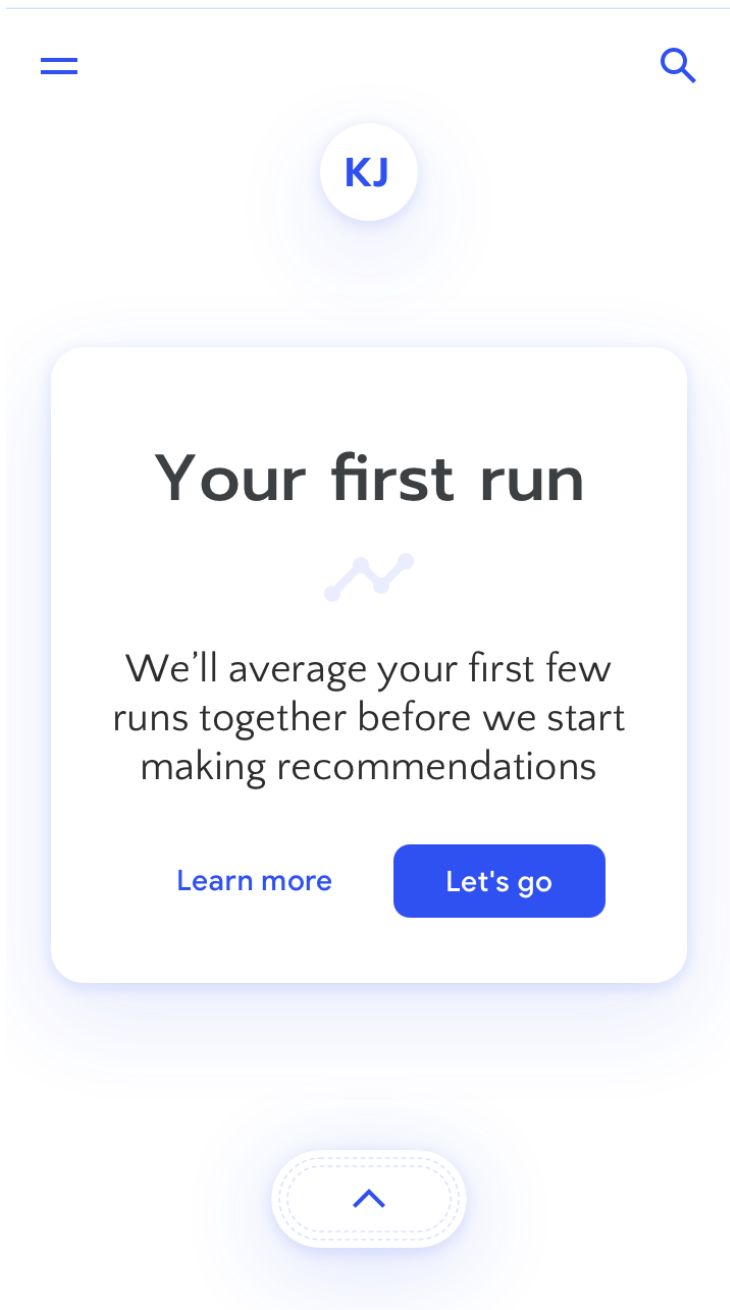
Design for experimentation

Many people learn best by tinkering with a new experience. People sometimes skip onboarding steps because they're eager to start using the system, and reading even a few screens feels like it's in the way. By keeping onboarding short, you'll let them get right to it. Suggest a low-risk or reversible action they can try right away — users are often curious about how an AI-powered feature or product will behave, so encourage that with a small, contained initial experimentation experience. For example, applying photo filters is easy to test out and undo with a tap.

One caveat is that a user's willingness or ability to spend time experimenting depends on their goal in using your product. For example, an average consumer who purchases a new smart speaker might enjoy spending time experimenting with different commands and questions. In contrast, a busy enterprise user might regard testing commands and functions as just one more chore in a busy day.

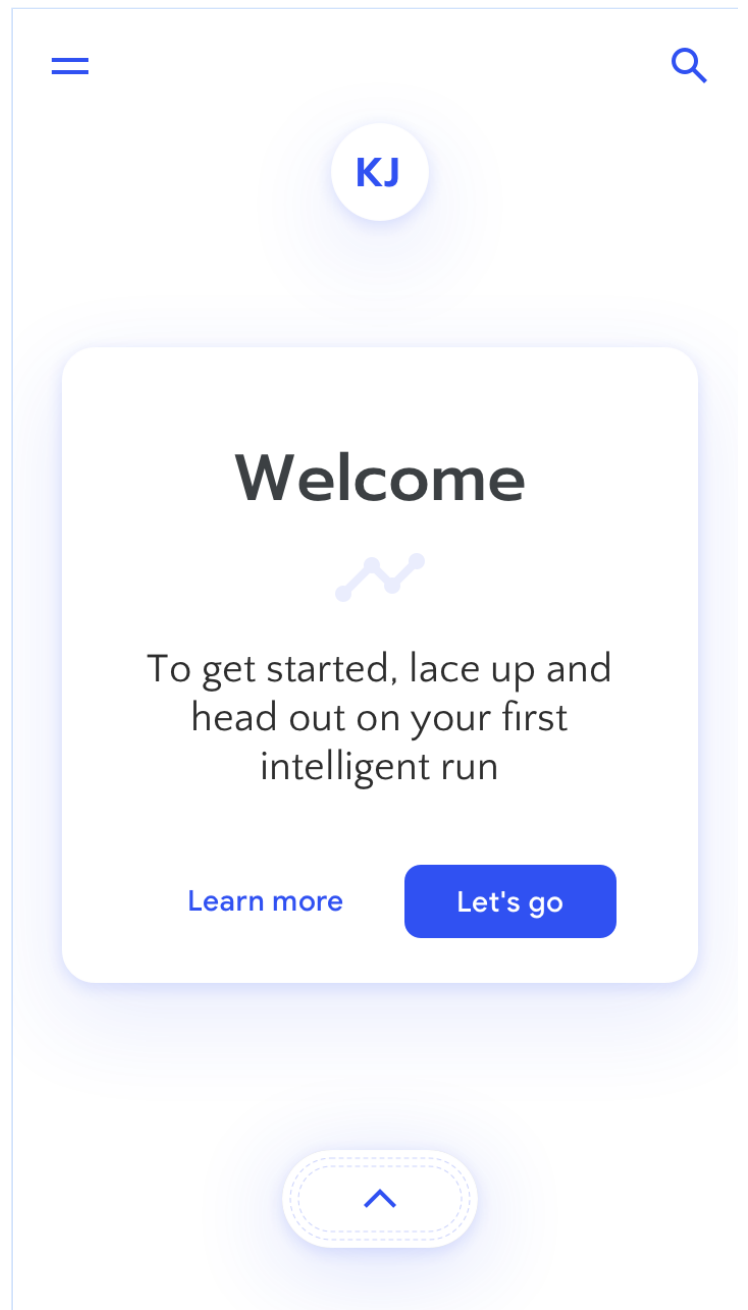
Regardless of their goals, point users towards where they can quickly understand and benefit from your product. Otherwise, they may find the boundaries of the system by experimenting in ways it isn't prepared to respond to. This can lead to errors, failure states, and potentially erosion of trust in your product.

Learn more about helping users get back on track in the [Errors + Graceful Failure](#) chapter.



Aim for

Encourage experimentation and reassure users that experimenting won't dictate their future experiences.
[Learn more](#)



Avoid

Don't assume users want the AI to start learning from the first use.

Key concept

Onboarding is all about setting up the interaction relationship between the user and your product. Here's a simple messaging framework to get you started:

This is *{ your product or feature }*,
and it'll help you by *{ core benefits }*.

Right now, it's not able to *{ primary limitations of AI }*.

Over time, it'll change to become more relevant to you.

You can help it get better by *{ user actions to teach the system }*.

Apply the concepts from this section in Exercise 2 [in the worksheet](#)

③ Plan for co-learning

Because AI-powered products can adapt and get better over time, the user experience can change. Users need to be prepared for that, and adjust their mental model as necessary.

Connect feedback with personalization

In onboarding, let users know how the feedback they provide helps the AI personalize their experience. You can tie this to the user benefit with phrasing like “you can improve your experience by giving feedback on the suggestions you receive”, and letting them know where and how to do so.

There are two ways to collect feedback:

Implicit feedback is when people’s actions while using the product help improve the AI over time. There should be a place in your product where users can see which signals are being used to what end, and this should be disclosed in your terms of service.

For example, choosing to listen to the next suggested song in a music app confirms the model’s prediction that the song is relevant to you. That fact should be part of the definition, user benefit, and terms of how the app works.

Explicit feedback is when people intentionally give feedback to improve an AI model, like picking categories of music they’re interested in. This kind of feedback can help the user feel more in control of the product. If you can, explain precisely what impact the feedback will have on your AI, and when it will take effect.

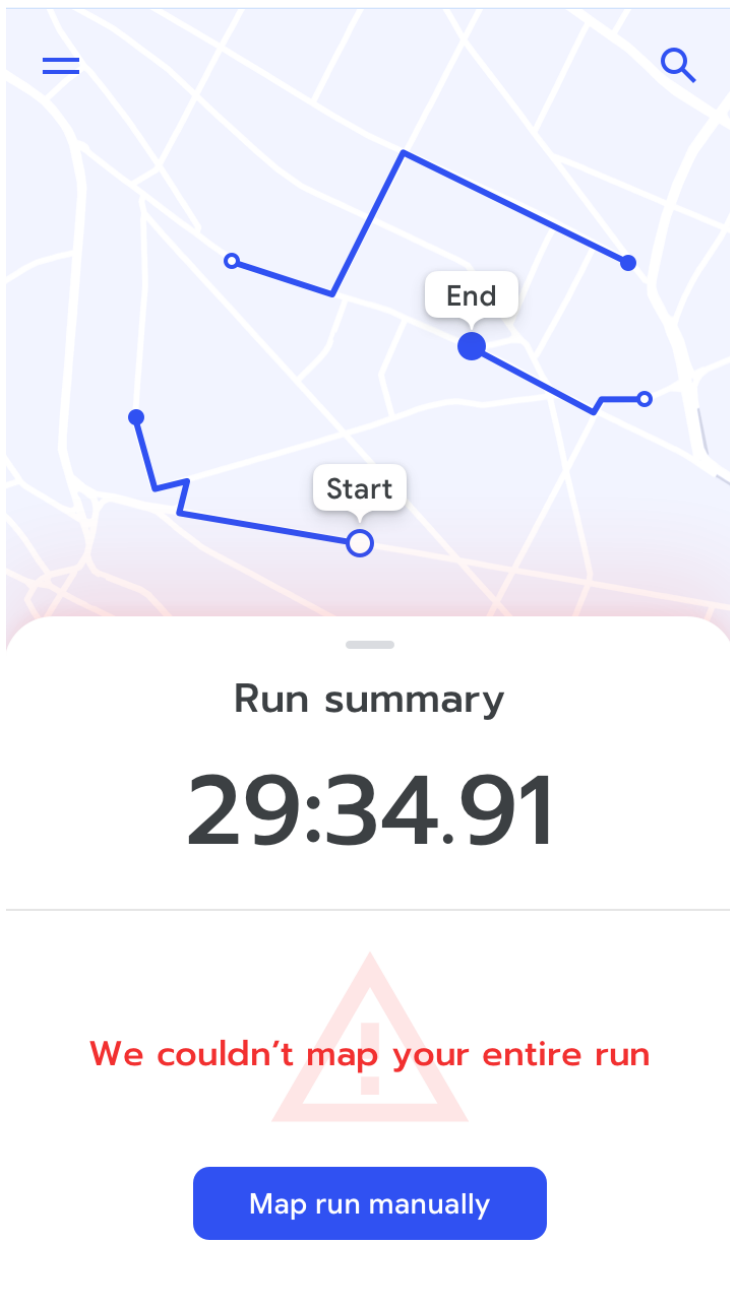
When the system collects feedback, explain how continually teaching the system benefits the user. Be clear about what information will help your AI learn and how it will improve the product output. See examples and much more detail in the [Feedback + Control](#) chapter.

Fail gracefully

The first time the system fails to meet expectations, the user will likely be disappointed. However, if the mental model includes the idea that the system learns over time, and learns better with the right input, then failure, especially the first failure the user encounters, becomes an opportunity to establish the feedback relationship. Once this relationship is set, users will see each failure not only as more forgivable, but also something that they can help fix. This buy-in can help cement the mental model of co-learning.

When your system isn't certain, or can't complete a request, make sure there's a default user experience that doesn't rely on AI. That way, the burden of educating your AI doesn't stop users from getting things done. When your product fails gracefully, it doesn't get in the user's way, and they see feedback as a way to make their objectives even easier over time, while still being able to use your product right now.

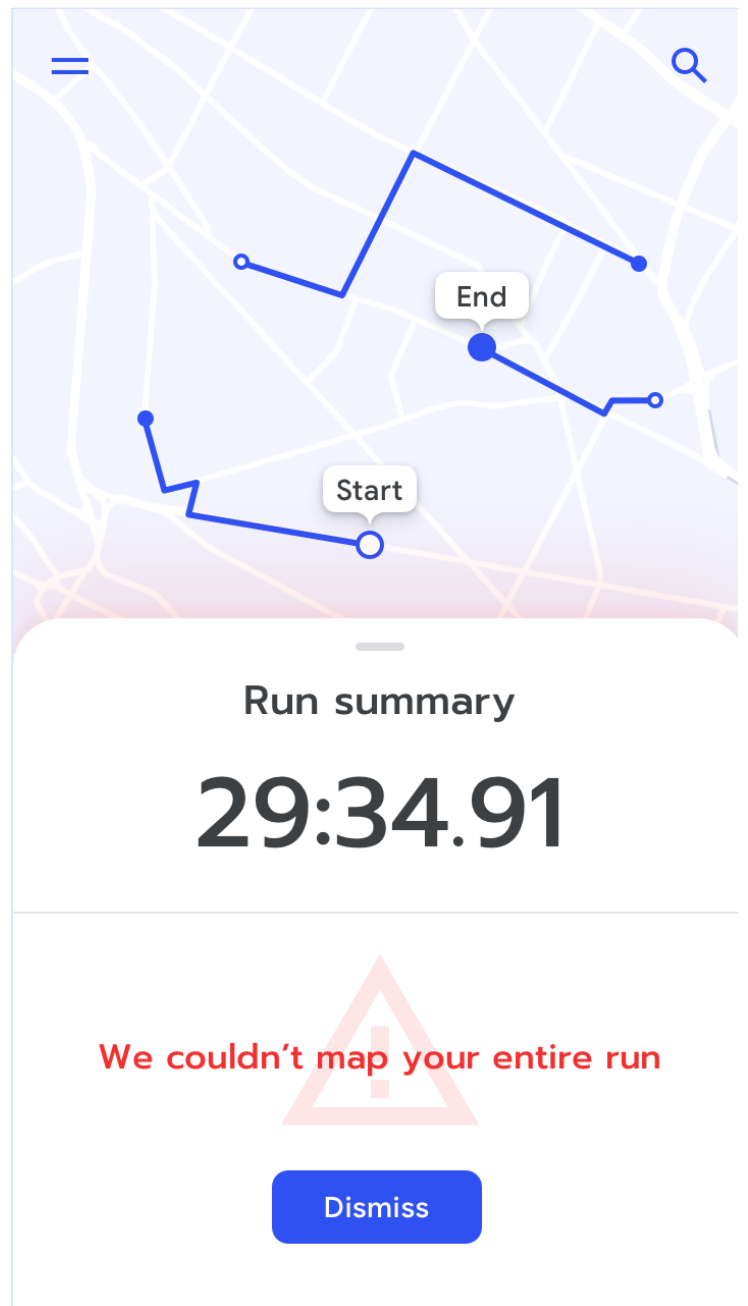
See examples and much more detail in the [Errors + Graceful Failure](#) chapter.



Aim for

Let users know an error occurred and give them a way to complete the task manually. [Learn more](#)

Remind, reinforce, and adjust



Avoid

Don't create dead-ends when an AI feature fails. These miss the opportunity to shape the user's mental model about the system's limitations.

Sometimes products become part of a user's everyday routine, so their mental models get formed and reinforced by ongoing use. But some products are only meant to be used occasionally. For these products, mental models might erode over time, so it's helpful to consider ways to reinforce them, or to remind users of the basics.

You can also help strengthen mental models across AI products by maintaining consistent messaging about the user benefits of improving AI with feedback. Over time, users may adopt a common mental model that recognizes AI solutions and their strengths and weaknesses, becoming more comfortable with what they'll get and how they can shape their experience. There are a few things you can do to increase the odds of that happening:

Keep track of user needs

Monitor how the product is being used. Reviewing your product logs can show you behavior or use trends that point to user confusion or frustration. This can help you determine when you might need to help users rebuild or adjust their mental models. If your product is only meant to be used for a short time or to achieve a specific goal, that will determine how frequently mental models should be reinforced or updated.

See more about metrics in the [User Needs + Defining Success](#) chapter.

Adapt to the evolving user journey

If how a feature works changes or improves significantly — enough that a user would notice — consider whether your users need “re-boarding” to the new experience. Re-boarding is also useful when adding new features, or if your system starts using new or different data for an existing feature.

If the system is simple and the mental model is clear and memorable, it's possible that only a little reinforcement is required. A quick user study with people who have used the product previously, but not in the last month or so, could reveal what kind of nudge, if any, might be most helpful.

④ Account for user expectations of human-like interaction

A number of products have launched in recent years that are designed to be anthropomorphic or human-like, such as Cortana, Alexa, Google Assistant, or Siri. This choice has advantages and disadvantages that should be weighed carefully. It's true that people tend to reflexively infer human characteristics from voice interfaces, and some interactions, such as conversational interfaces, are inherently human-like. However, if the algorithmic nature and limits of these products are not explicitly communicated, they can set expectations that are unrealistic and eventually lead to user disappointment, or even unintended deception.

When users confuse an AI with a human being, they can sometimes disclose more information than they would otherwise, or rely on the system more than they should, among other issues. Therefore, disclosing the algorithm-powered nature of these kinds of interfaces is a critical onboarding step. Specifically, your messages should make it extremely clear that the product is not a human, in a way that's accessible to all users regardless of age, technical literacy, education level or physical ability.

This topic is the subject of ongoing research, and these considerations are just a first step. Expect more on this topic in future editions of the Guidebook.

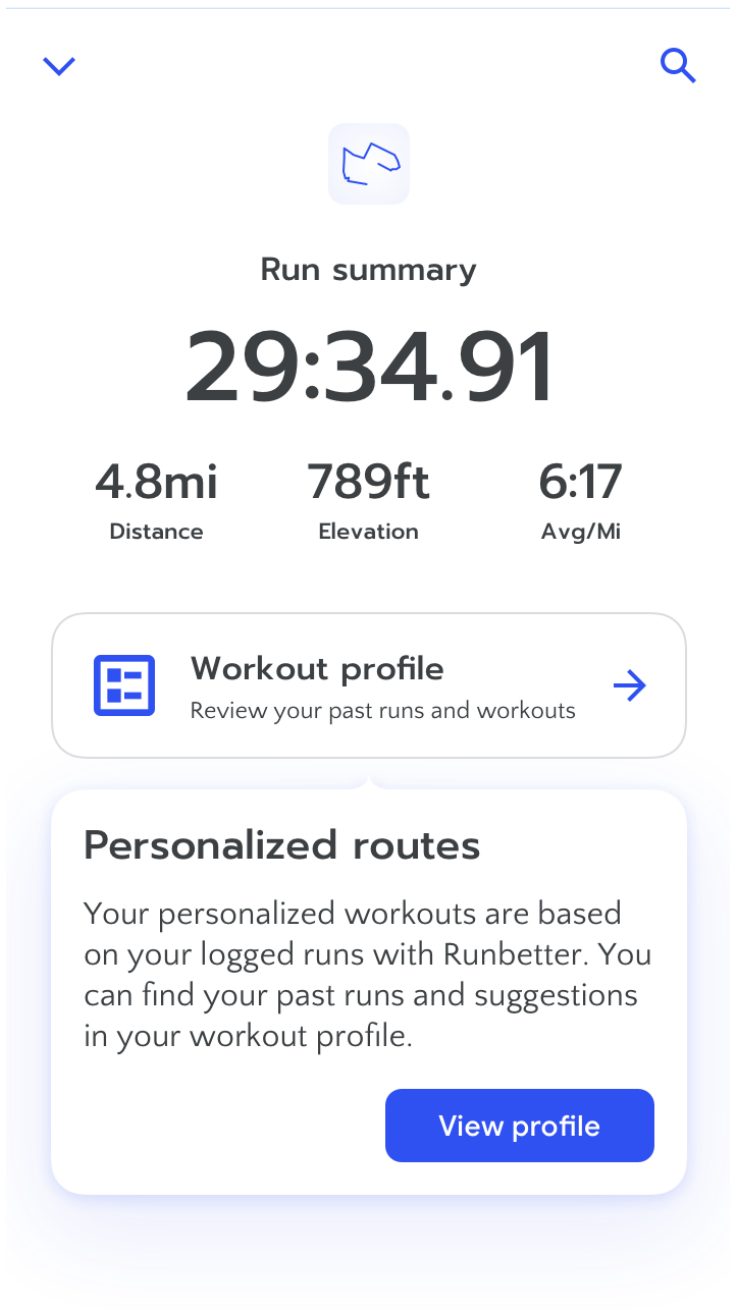
Clearly communicate AI limits and capabilities

People may struggle to form an accurate or useful mental model of an anthropomorphized AI-powered product because the way these systems execute tasks is inherently different than the way a person would. On the surface, AI-powered functionality might seem similar to the manual method but the mental model might not map precisely.

For example, “automatic photo tagging” sounds like the tool is tagging photos the same way a person would, just “automatically”. If someone hasn't used this tool before, their mental model of this process is by default, a human one. The app may be able to find and tag all the pictures of a particular friend, just as a person would, but miss some that show her from the back. This is an unexpected break: of course most people can recognize their friends from multiple angles, but this tool can't.

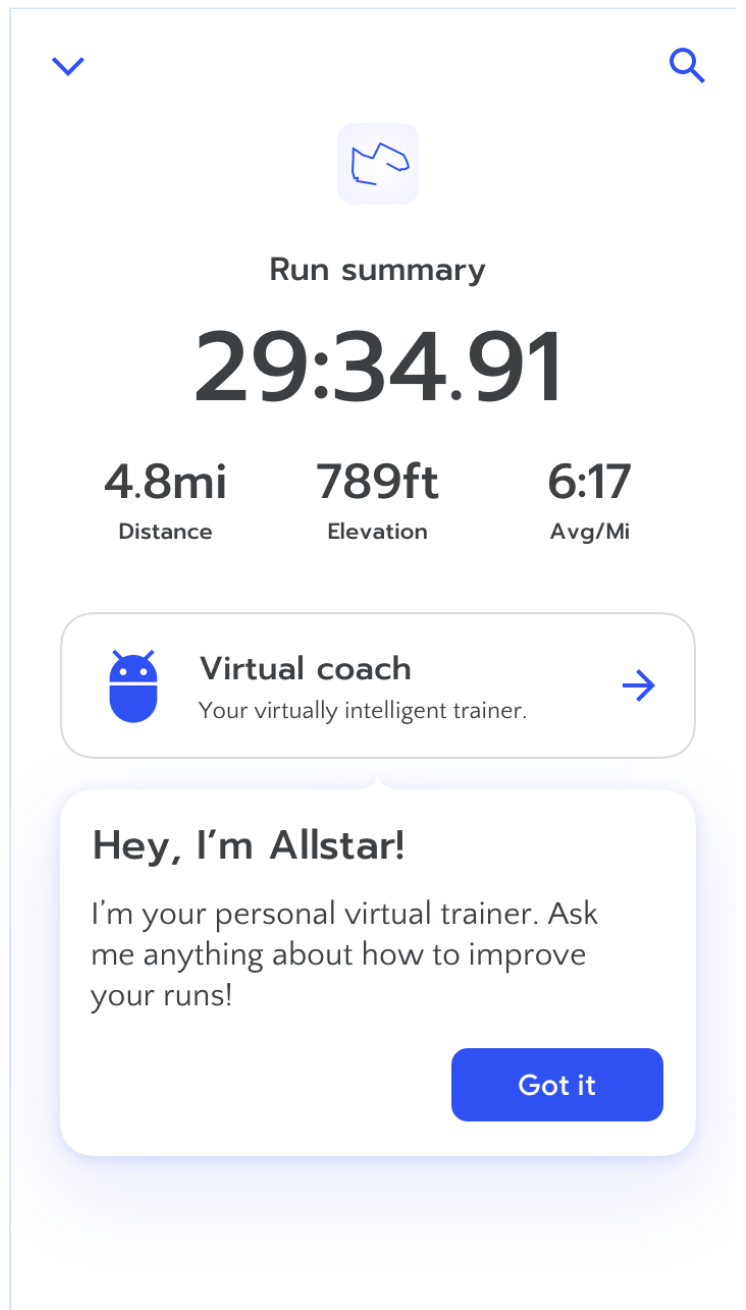
A disconnect like this doesn't necessarily mean that the product itself or the mental model is broken. Not all humans have the same abilities — sight and hearing are not a given — and the product is still doing an incredible task that humans can't do: scanning thousands of photos, identifying the subjects, and labeling them. The key is to communicate the system's limits and capabilities in a way that doesn't create or support expectations of super-human abilities. AI doesn't do anything the same way people do, so though this model is convenient, it's pretty fragile.

Often the idea of a generalized “helper AI” is easier to grasp and more inviting for users, but the risk of mistrust is high when the system's limits aren't clear. When users can't accurately map the system's abilities, they may over-trust the system at the wrong times, or miss out on the greatest value-add of all: better ways to do a task they take for granted. Choose the level of humanization based on how well your AI's capabilities match the user's perceptions of what a human can do.



Aim for

Describe AI features in terms of helping people improve while setting the right expectations for what the AI can do. [Learn more](#)



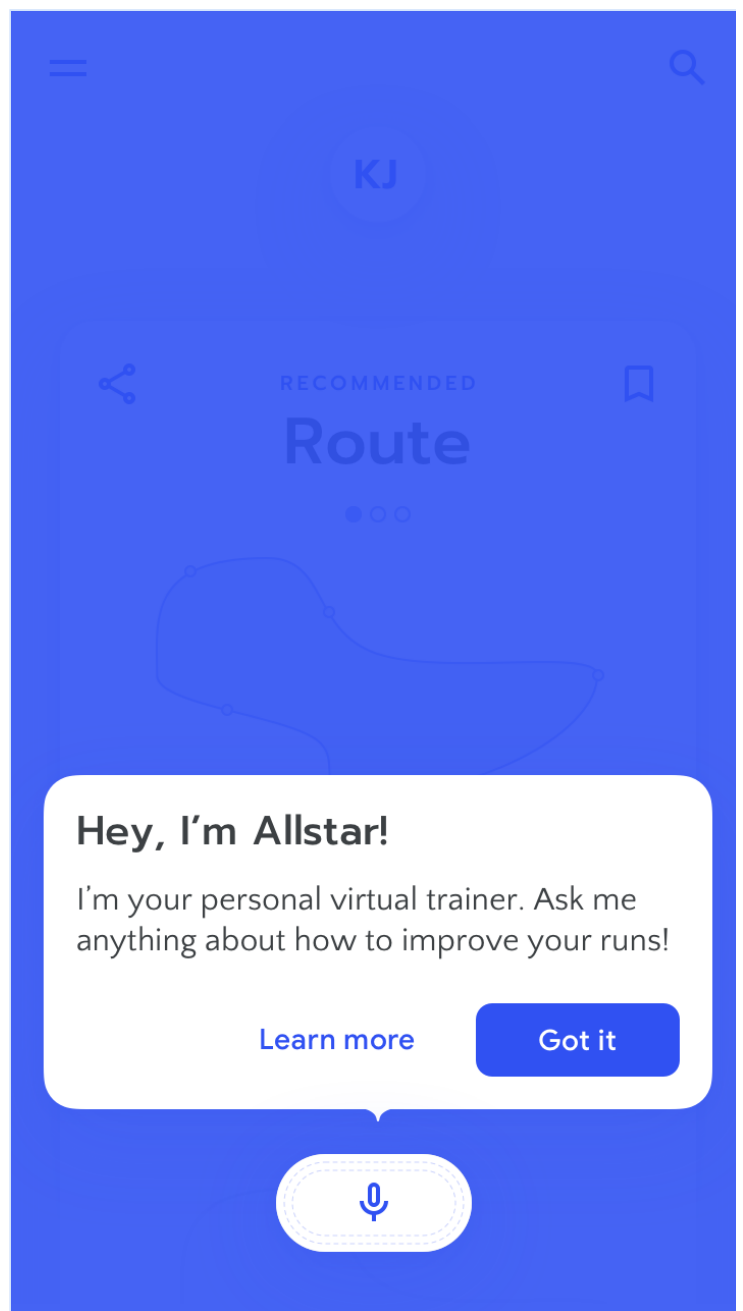
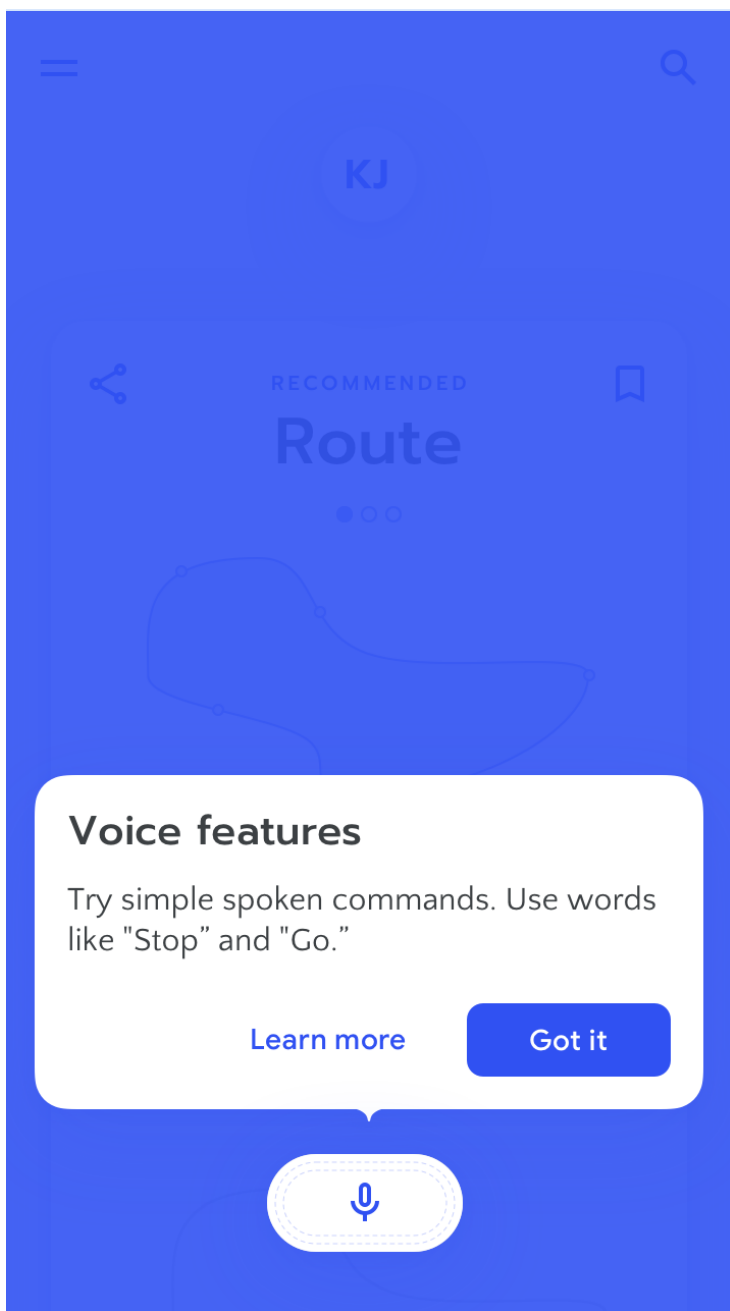
Avoid

Don't create unrealistic expectations by presenting the AI as human-like when it can actually do far less than a person.

Cue the correct interactions

Leveraging human characteristics to build mental models is particularly useful if your product interactions rely on distinctly human behaviors, such as conversation. Using the first person in chatbots and voice interactions can help people intuitively understand how to use your system. It's much easier for users to understand that they can talk to something that invites them to in a conversational way.

However, this approach has its risks. Specifically, if your conversational AI refers to itself as "I", the corresponding user mental model includes near-perfect natural language processing, which your AI may not be able to pull off yet. It's a delicate balance between cueing the right type of interaction while trying to limit the level of mismatched expectations or failures.



Aim for

Set expectations for the kind of commands the AI can understand to reinforce the right mental models. [Learn more](#)

Avoid

Don't set unrealistic expectations about what the AI can do, especially compared to humans.

Summary

Mental models for AI-driven products are influenced by multiple factors including: existing mental models for similar features or products, marketing messages from your team, onboarding and expectations setting, and the feedback relationships in your product. When you set out to help users construct the right mental models for your AI, consider the following:

- ① **Set expectations for adaptation.** Help people get the most out new AI uses by identifying and building on existing mental models. Ask yourself questions like “What is the user trying to do?”, “What mental models might already be in place?”, and “Does this product break any intuitive patterns of cause and effect?”
- ② **Onboard in stages.** Set realistic expectations early. Describe user benefits, not technology. Describe the core value initially, but introduce new features as they are used. Make it easy for users to experiment with the AI in your product.
- ③ **Plan for co-learning.** Connect feedback to personalization and adaptation to establish the relationship between user actions and the AI output. Fail gracefully to non-AI options when needed.
- ④ **Account for user expectations of human-like interaction.** Clearly communicate the algorithmic nature and limits of these products to set realistic user expectations and avoid unintended deception.

Want to drive discussions, speed iteration, and avoid pitfalls? [Use the worksheet](#)

Explainability + Trust

Explaining predictions, recommendations, and other AI output to users is critical for building trust. This chapter covers:

How much should the user trust the AI system?

What should we do if we can't show why the AI made a given prediction?

How should we show users the confidence associated with an AI prediction?

Want to drive discussions, speed iteration, and avoid pitfalls? [Use the worksheet](#).

What's new when working with AI

Because AI-driven systems are based on probability and uncertainty, the right level of explanation is key to helping users understand how the system works. Once users have clear mental models of the system's capabilities and limits, they can understand how and when to trust it to help accomplish their goals. In short, explainability and trust are inherently linked.

In this chapter, we'll discuss considerations for how and when to explain what your AI does, what data it uses to make decisions, and the confidence level of your model's output.

Key considerations for explaining AI systems:

- ① **Help users calibrate their trust.** Because AI products are based on statistics and probability, the user shouldn't trust the system completely. Rather, based on system explanations, the user should know when to trust the system's predictions and when to apply their own judgement.
- ② **Optimize for understanding.** In some cases, there may be no explicit, comprehensive explanation for the output of a complex algorithm. Even the developers of the AI may not know precisely how it works. In other cases, the reasoning behind a prediction may be knowable, but difficult to explain to users in terms they will understand.
- ③ **Manage influence on user decisions.** AI systems often generate output that the user needs to act on. If, when, and how the system calculates and shows confidence levels can be critical in informing the user's decision making and calibrating their trust.

① Help users calibrate their trust

Users shouldn't implicitly trust your AI system in all circumstances, but rather calibrate their trust correctly. There are many research examples of "algorithm aversion", where people are suspicious of software systems. Researchers have also found cases of people over-trusting an AI system to do something that it can't. Ideally, users have the appropriate level of trust given what the system can and cannot do.

For example, indicating that a prediction could be wrong may cause the user to trust that particular prediction less. However, in the long term, users may come to use or rely on your product or company more, because they're less likely to over-trust your system and be disappointed.

Articulate data sources

Every AI prediction is based on data, so data sources have to be part of your explanations. However, remember that there may be legal, fairness, and ethical considerations for collecting and communicating about data sources used in AI. We cover those in more detail in the chapter on Data Collection + Evaluation.

Sometimes users can be surprised by their own information when they see it in a new context. These moments often occur when someone sees their data used in a way that appears as if it isn't private or when they see data they didn't know the system had access to, both of which can erode trust. To avoid this, explain to users where their data is coming from and how it is being used by the AI system.

Equally important, telling users what data the model is using can help them know when they have a critical piece of information that the system does not. This knowledge can help the user avoid over-trusting the system in certain situations.

For example, say you're installing an AI-driven navigation app, and you click to accept all terms and conditions, which includes the ability for the navigation app to access data from your calendar app. Later, the navigation app alerts you to leave your home in 5 minutes in order to be on time for an appointment. If you didn't read, realize, or remember that you allowed the navigation app to access to your appointment information, then this could be very surprising.

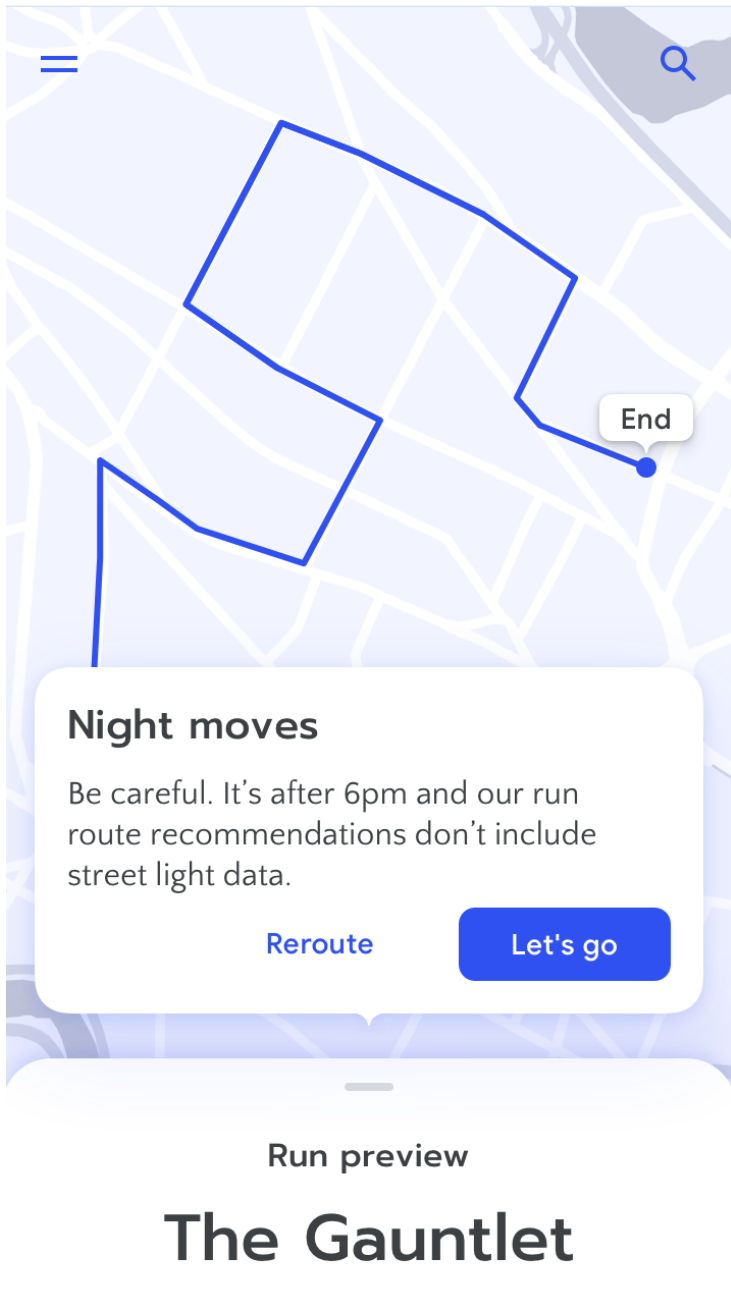
Your trust in the app's capabilities depends on your expectations for how it should work and how alerts like these are worded. For instance, you could become suspicious of the app's data sources; or, you could over-trust that it has complete access to all your schedule information. Neither of these outcomes are the right level of trust. One way to avoid this is to explain the connected data source — how the navigation app knows about the appointment — as part of the notification, and to provide the option to opt out of that kind of data sharing in the future. In fact, regulations in some countries may require such specific, contextual explanations and data controls.

Key concept

Whenever possible, the AI system should explain the following aspects about data use:

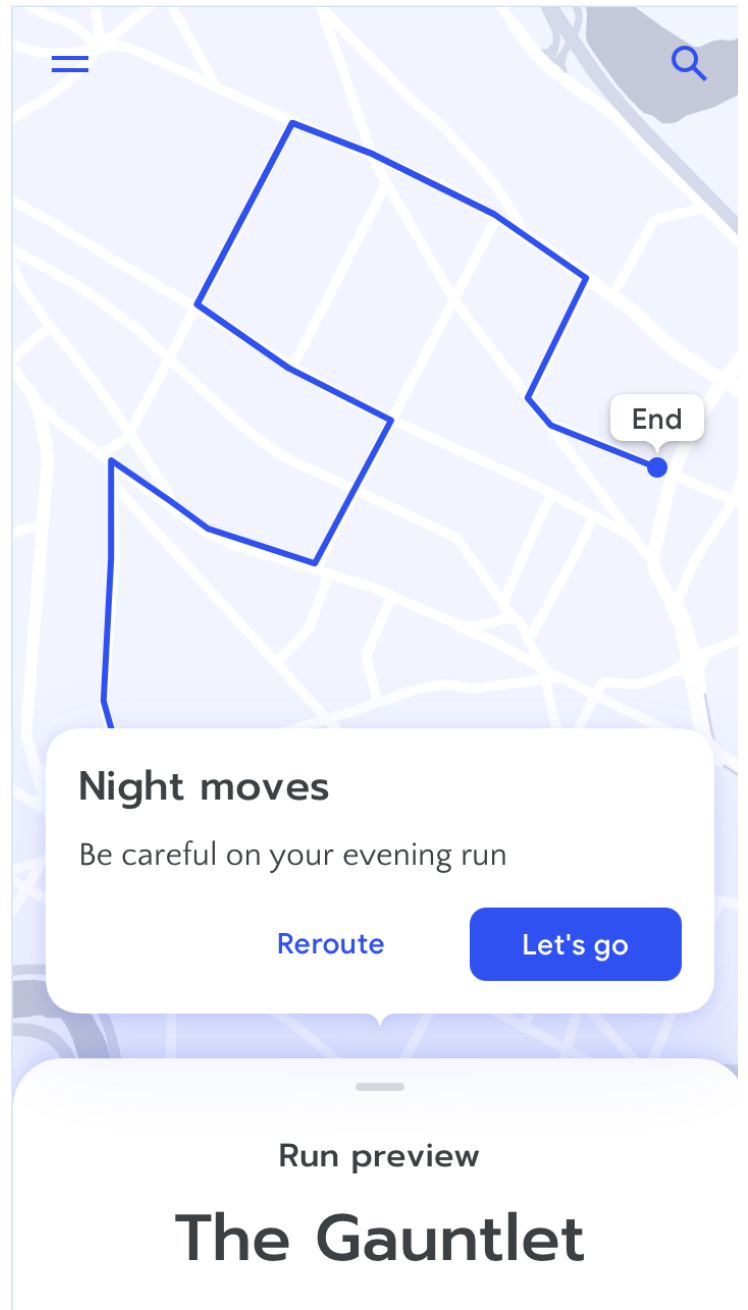
- **Scope.** Show an overview of the data being collected about an individual user, and which aspects of their data are being used for what purpose.
- **Reach.** Explain whether the system is personalized to one user or device, or if it is using aggregated data across all users.
- **Removal.** Tell users whether they can remove or reset some of the data being used.

Apply the concepts from this section in Exercise 1 [in the worksheet](#)



Aim for

Tell the user when a lack of data might mean they'll need to use their own judgment. [Learn more](#)



Avoid

Don't be afraid to admit when a lack of data could affect the quality of the AI recommendations.

Tie explanations to user actions

People learn faster when they can see a response to their actions right away, because then it's easier to identify cause and effect. This means the perfect time to show explanations is in response to a user's action. If the user takes an action and the AI system doesn't respond, or responds in an unexpected way, an explanation can go a long way in building or recovering a user's trust. On the other hand, when the system is working well, responding to users' actions is a great time to tell the user what they can do to help the system continue to be reliable.

For example, let's say a user taps on the "recommendations for me" section of an AI-driven restaurant reservation app. They only see recommendations for Italian restaurants, which they rarely visit, so they're a bit disappointed and less trusting that the app can make relevant, personalized recommendations. If however the app's recommendations include an explanation that the system only recommends restaurants within a one-block area, and the user is standing in the heart of Little Italy in New York City, then trust is likely to be maintained. The user can see how their actions — in this case asking for recommendations in a specific location — affects the system.

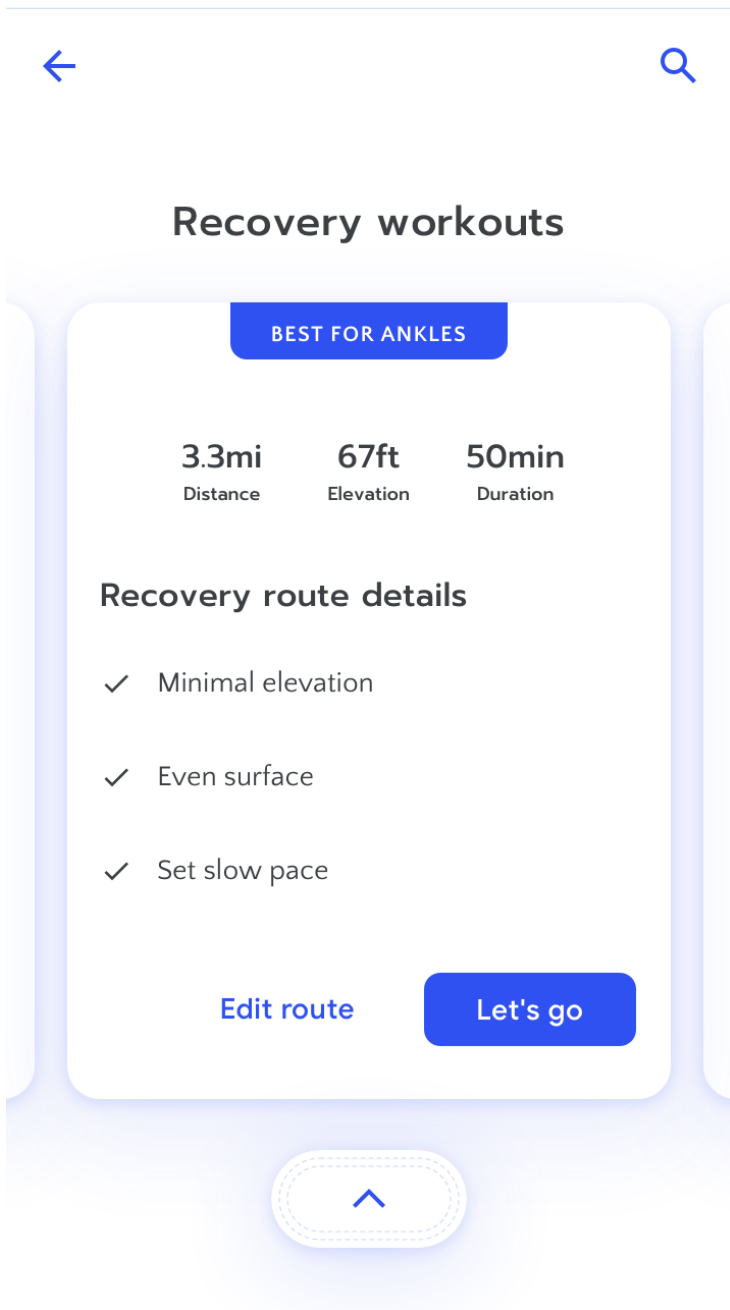
Just as you might build trust in another person through back and forth interactions that reveal their strengths and weaknesses, the user's relationship with an AI system can evolve in the same way.

When it's harder to tie explanations directly to user actions, you could use multi-modal design to show explanations. For example, if someone is using an assistant app with both visual and voice interfaces, you could leave out the explanation in the voice output but include it in the visual interface for the user to see when they have time.

Account for situational stakes

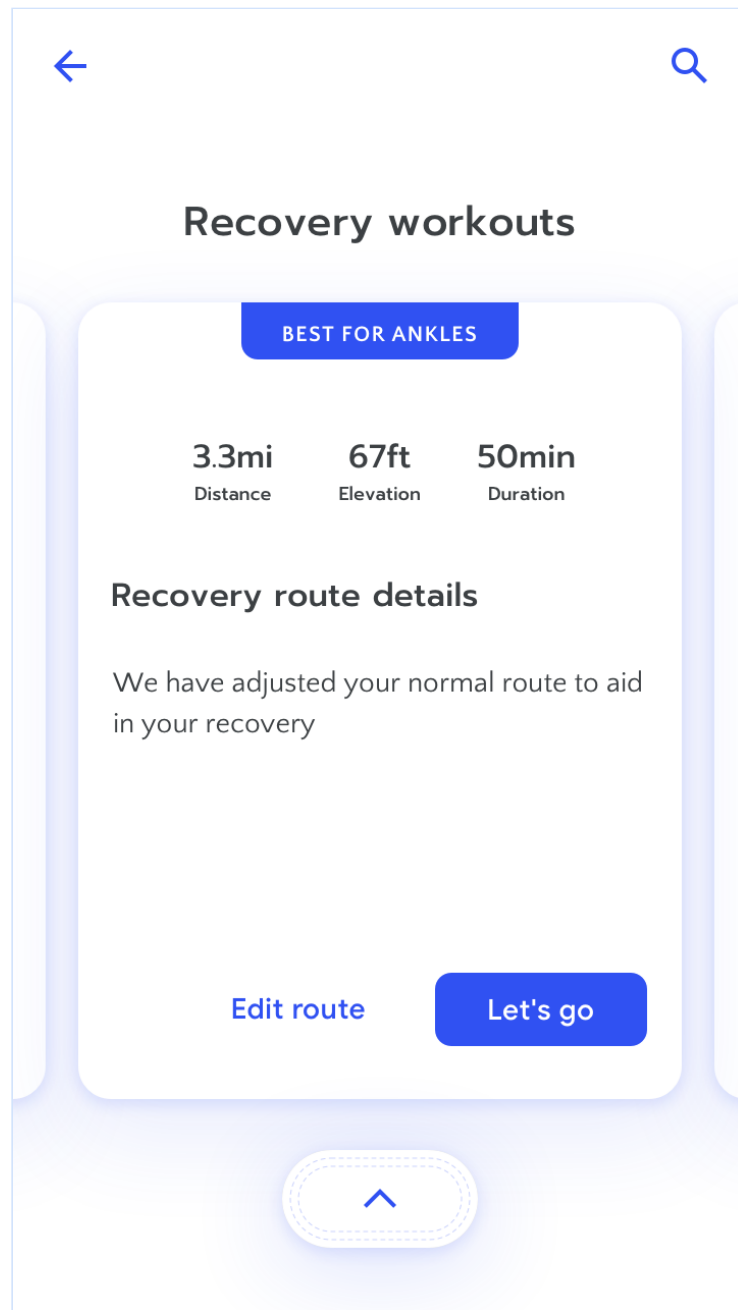
You can use explanations to encourage users to trust an output more or less depending on the situation and potential consequences. It's important to consider the risks of a user trusting a false positive, false negative, or a prediction that's off by a certain percent.

For example, for an AI-driven navigation app, it may not be necessary to explain how arrival time is calculated for a daily commute. However, if someone is trying to catch a flight (higher stakes and less frequent than a commute) they may need to cross-check the timing of the recommended route. In that case, the system could prompt them with an explanation of its limitations. For example, if a user enters the local airport as their destination, let them know that traffic data only refreshes every hour.



Aim for

Give the user details about why a prediction was made in a high stakes scenario. Here, the user is exercising after an injury and needs confidence in the app's recommendation. [Learn more](#)



Avoid

Don't say "what" without saying "why" in a high stakes scenario.

You can find detailed information about giving the user appropriate guidance in situations of failure or low-confidence predictions in the [Errors + Graceful Failure](#) chapter.

Key concept

As a team, brainstorm what kinds of interactions, results, and corresponding explanations would decrease, maintain, or inflate trust in your AI system. These should fall somewhere along a trust spectrum of “No trust” to “Too much trust”.

Here are some examples from our running app:

- A user who has never run more than 3 miles at a time receives a recommendation for a marathon training series.
- A user takes the training recommendation to their personal trainer and their trainer agrees with the app’s suggestion.
- A user follows the app’s suggestion for a recovery run, but it’s too difficult for them to complete.

Apply the concepts from this section in Exercise 2 [in the worksheet](#)

② Optimize for understanding

As described above, explanations are crucial for building calibrated trust. However, offering an explanation of an AI system can be a challenge in and of itself. Because AI is inherently probabilistic, extremely complicated, and making decisions based on multiple signals, it can limit the types of possible explanations.

Often, the rationale behind a particular AI prediction is unknown or too complex to be summarized into a simple sentence that users with limited technical knowledge can readily understand. In many cases the best approach is not to attempt to explain everything – just the aspects that impact user trust and decision-making. Even this can be hard to do, but there are lots of techniques to consider.

Explain what's important

Partial explanations clarify a key element of how the system works or expose some of the data sources used for certain predictions. Partial explanations intentionally leave out parts of the system's function that are unknown, highly complex, or simply not useful. Note that progressive disclosures can also be used together with partial explanations to give curious users more detail.

You can see some example partial explanations below for an AI-driven plant classification app.

Describe the system or explain the output

General system explanations talk about how the whole system behaves, regardless of the specific input. They can explain the types of data used, what the system is optimizing for, and how the system was trained.

Specific output explanations should explain the rationale behind a specific output for a specific user, for example, why it predicted a specific plant picture to be poison oak. Output explanations are useful because they connect explanations directly to actions and can help resolve confusion in the context of user tasks.

Data sources

Simple models such as regressions can often surface which data sources had the greatest influence on the system output. Identifying influential data sources for complex models is still a growing area of active research, but can sometimes be done. In cases where it can, the influential feature(s) can then be described for the user in a simple sentence or illustration. Another way of explaining data sources is counterfactuals, which tell the user why the AI did not make a certain decision or prediction.

Specific output

“This plant is most likely poison oak because it has XYZ features”.

“This tree field guide was created for you because you submit lots of pictures of maple and oak trees in North America”.

“This leaf is not a maple because it doesn’t have 5 points”.

General system

“This app uses color, leaf shape, and other factors to identify plants”.

Model confidence displays

Rather than stating why or how the AI came to a certain decision, model confidence displays explain how certain the AI is in its prediction, and the alternatives it considered. As most models can output n-best classifications and confidence scores, model confidence displays are often a readily-available explanation.

Specific output

N-best most-likely classifications

Most likely plant:

- Poison oak
- Maple leaf
- Blackberry leaf

Numeric confidence level

Prediction: Poison oak (80%)

General system

Numeric confidence level

This app categorizes images with 80% confidence on average.

Confidence displays help users gauge how much trust to put in the AI output. However, confidence can be displayed in many different ways, and statistical information like confidence scores can be challenging for users to understand. Because different user groups may be more or less familiar with what confidence and probability mean, it's best to test different types of displays early in the product development process.

There's more guidance about confidence displays and their role in user experiences in [Section 3](#) of this chapter.

Example-based explanations

Example-based explanations are useful in cases where it's tricky to explain the reasons behind the AI's predictions. This approach gives users examples from the model's training set that are relevant to the decision being made. Examples can help users understand surprising AI results, or intuit why the AI might have behaved the way it did. These explanations rely on human intelligence to analyze the examples and decide how much to trust the classification.

Specific output

To help the user decide whether to trust a "poison oak" classification, the system displays most-similar images of poison oak as well as most-similar images of other leaves.

General system

The AI shows sets of image examples it tends to make errors on, and examples of images it tends to perform well on.

Explanation via interaction

Another way to explain the AI and help users build mental models is by letting users experiment with the AI on-the-fly, as a way of asking “what if?”. People will often test why an algorithm behaves the way it does and find the system’s limits, for example by asking an AI voice assistant impossible questions. Be intentional about letting users engage with the AI on their own terms to both increase usability and build trust.

Specific output

A user suspects the system gave too much weight to the leaf color of a bush, which led to a mis-classification.

To test this, the user changes the lighting to yield a more uniform brightness to the bush’s leaves to see whether that changes the classification.

General system

This type of explanation can’t be used for the entire app generally. It requires a specific output to play with.

It’s important to note that developing any explanation is challenging, and will likely require multiple rounds of user testing. There’s more information on introducing AI systems to users in the chapter on [Mental Models](#).

Note special cases of absent or comprehensive explanation

In select cases, there’s no benefit to including any kind of explanation in the user interface. If the way an AI works fits a common mental model and matches user expectations for function and reliability, then there may not be anything to explain in the interaction. For example, if a cell phone camera automatically adjusts to lighting, it would be distracting to describe when and how that happens as you’re using it. It’s also wise to avoid explanations that would reveal proprietary techniques or private data. However, before abandoning explanations for these reasons, consider using partial explanations and weigh the impact on user trust.

In other situations, it makes sense, or is required by law, to give a complete explanation – one so detailed that a third party could replicate the results. For example, in software used by the government to sentence criminals, it would be reasonable to expect complete disclosure of every detail of the system. Nothing less than total accountability would be sufficient for a fair, contestable decision. Another case for complete explanation is when AI is part of open-source software that is intended to be used by others. If you are required to give a complete explanation of your model, there are additional considerations for protecting private data that may have been used to train the model.

See the [Resources](#) page for more information on legal and ethical issues regarding data handling and open-source AI.

Key concept

Think about how an explanation for each critical interaction could decrease, maintain, or increase trust. Then, decide which situations need explanations, and what kind. The best explanation is likely a partial one.

There are lots of options for providing a partial explanation, which intentionally leave out parts of the system's function that are unknown, too complex to explain, or simply not useful. Partial explanations can be:

- **General system.** Explaining how the AI system works in general terms
- **Specific output.** Explaining why the AI provided a particular output at a particular time

Apply the concepts from this section in Exercise 3 [in the worksheet](#)

③ Manage influence on user decisions

One of the most exciting opportunities for AI is being able to help people make better decisions more often. The best AI-human partnerships enable better decisions than either party could make on their own. For example, a commuter can augment their local knowledge with traffic predictions to take the best route home. A doctor could use a medical diagnosis model to supplement their historical knowledge of their patient. For this kind of collaboration to be effective, people need to know if and when to trust a system's predictions.

As described in section 2 above, model confidence indicates how certain the system is in the accuracy of its results. Displaying model confidence can sometimes help users calibrate their trust and make better decisions, but it's not always actionable. In this section, we'll discuss when and how to show the confidence levels behind a model's predictions.

Determine if you should show confidence

It's not easy to make model confidence intuitive. There's still active research around the best ways to display confidence and explain what it means so that people can actually use it in their decision making. Even if you're sure that your user has enough knowledge to properly interpret your confidence displays, consider how it will improve usability and comprehension of the system – if at all. There's always a risk that confidence displays will be distracting, or worse, misinterpreted.

Be sure to set aside lots of time to test if showing model confidence is beneficial for your users and your product or feature. You might choose not to indicate model confidence if:

- **The confidence level isn't impactful.** If it doesn't make an impact on user decision making, consider not showing it. Counterintuitively, showing more granular confidence can be confusing if the impact isn't clear – what should I do when the system is 85.8% certain vs. 87% certain?
- **Showing confidence could create mistrust.** If the confidence level could be misleading for less-savvy users, reconsider how it's displayed, or whether to display it at all. A misleadingly high confidence, for example, may cause users to blindly accept a result.

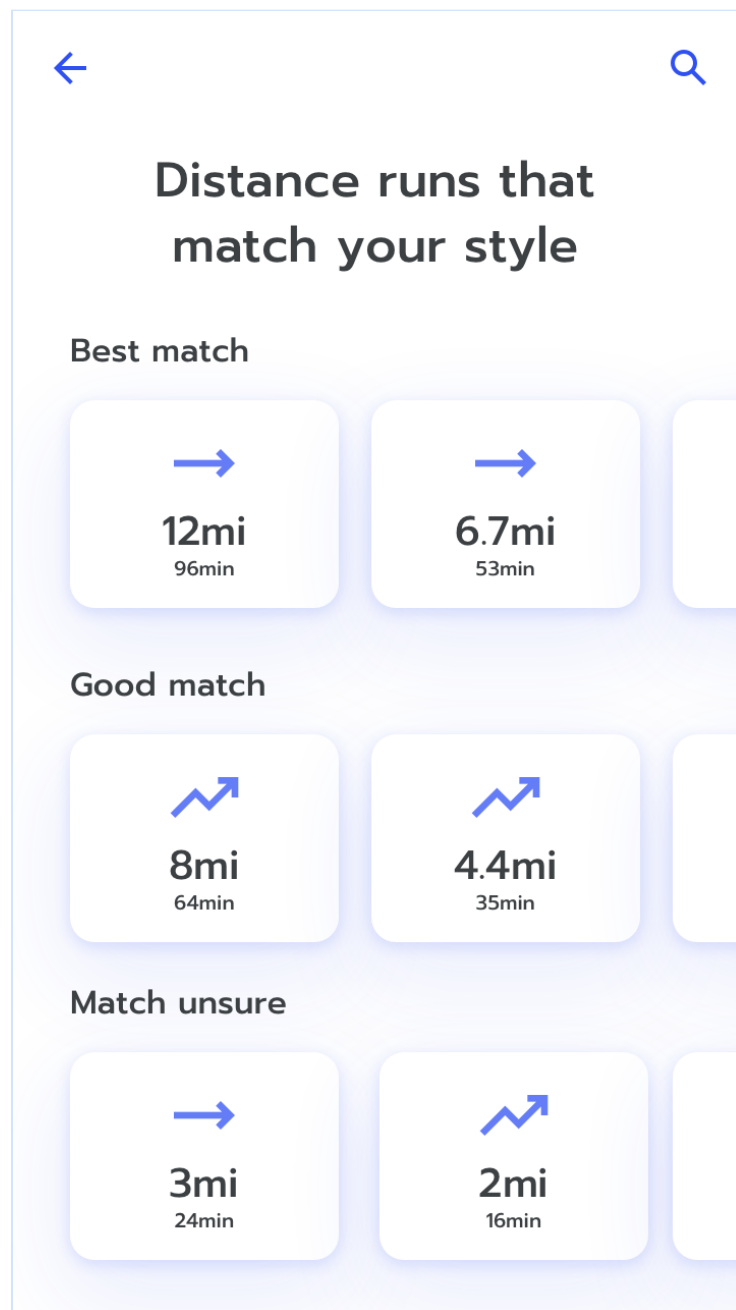
Decide how best to show model confidence

If your research confirms that displaying model confidence improves decision making, the next step is choosing an appropriate visualization. To come up with the best way to display model confidence, think about what user action this information should inform. Types of visualizations include:

Categorical

These visualizations categorize confidence values into buckets, such as High / Medium / Low and show the category rather than the numerical value. Considerations:

- Your team will determine cutoff points for the categories, so it's important to think carefully about their meaning and about how many there should be.
- Clearly indicate what action a user should take under each category of confidence.

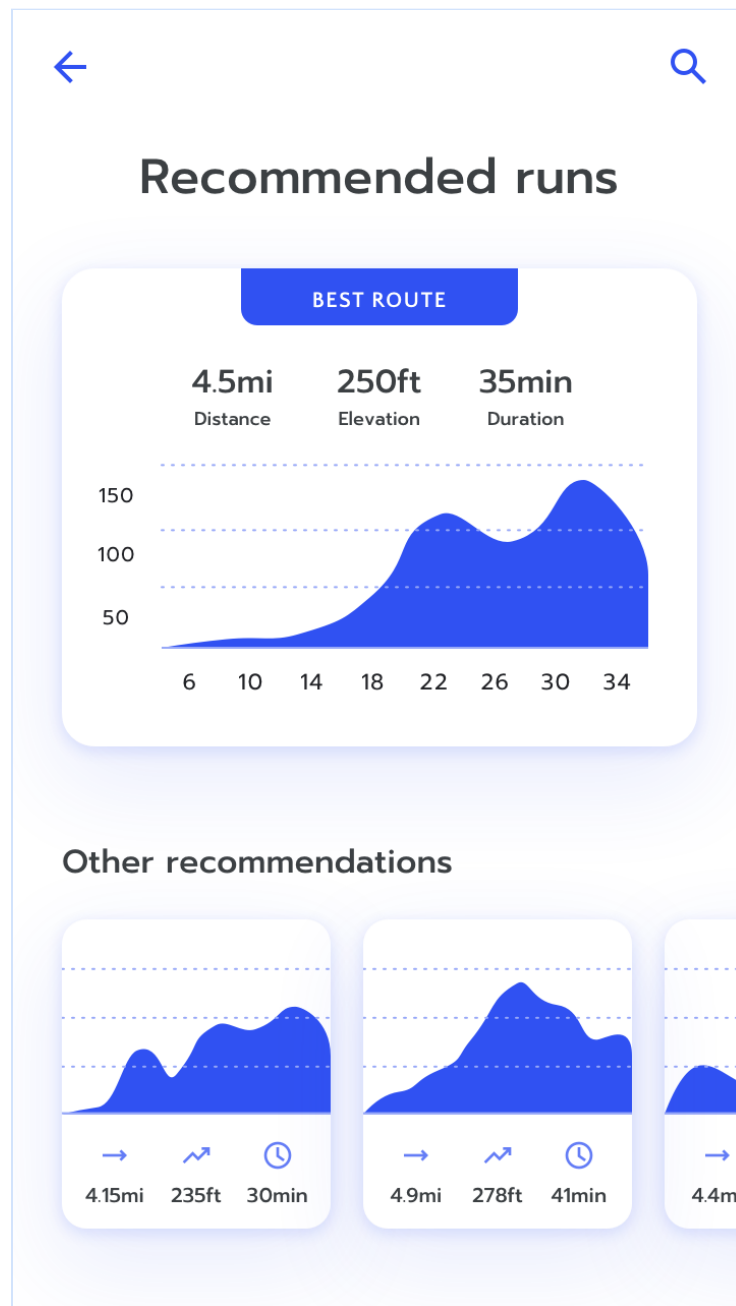


Categorical model confidence visualization

N-best alternatives

Rather than providing an explicit indicator of confidence, the system can display the N-best alternative results. For example, “This photo might be of New York, Tokyo, or Los Angeles.” Considerations:

- This approach can be especially useful in low-confidence situations. Showing multiple options prompts the user to rely on their own judgement. It also helps people build a mental model of how the system relates different options.
- Determining how many alternatives you show will require user testing and iteration.



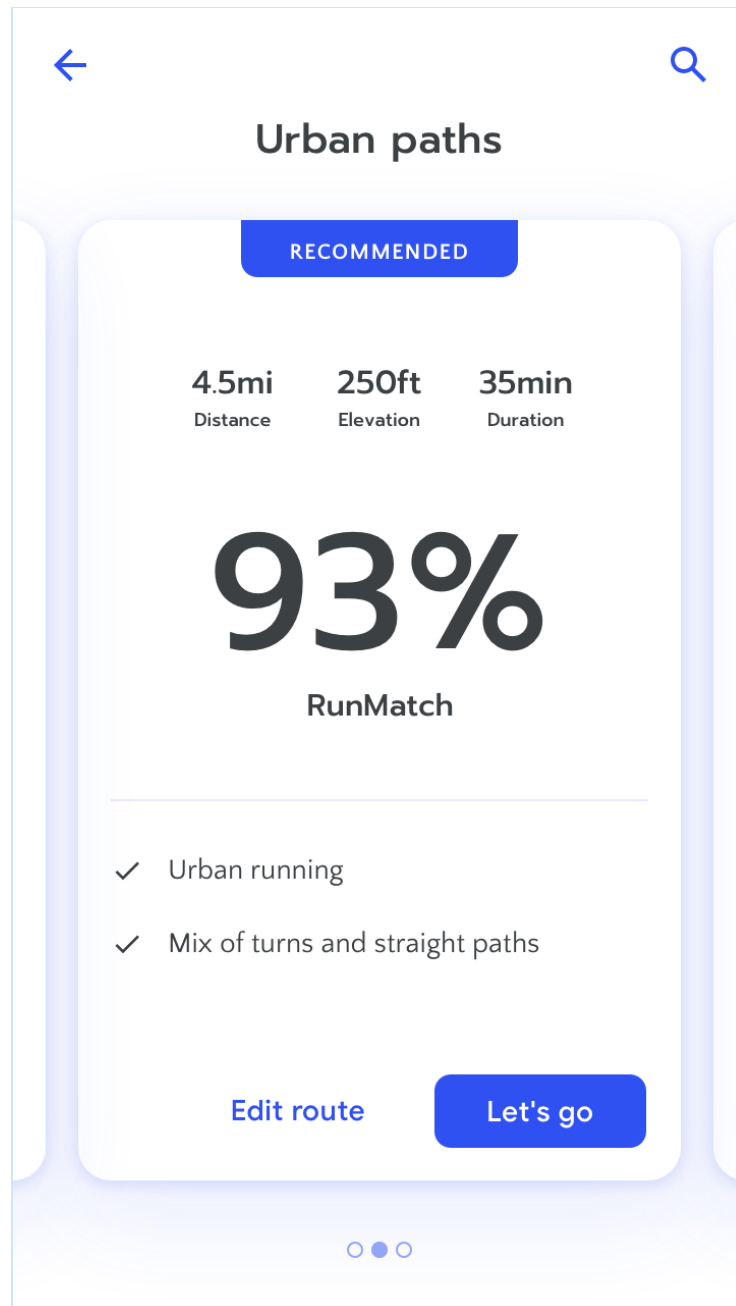
N-best model confidence visualization

Numeric

A common form of this is a simple percentage. Numeric confidence indicators are risky because they presume your users have a good baseline understanding of probability. Additional considerations:

- Make sure to give enough context for users to understand what the percentage means. Novice users may not know whether a value like 80% is low or high for a certain context, or what that means for them.

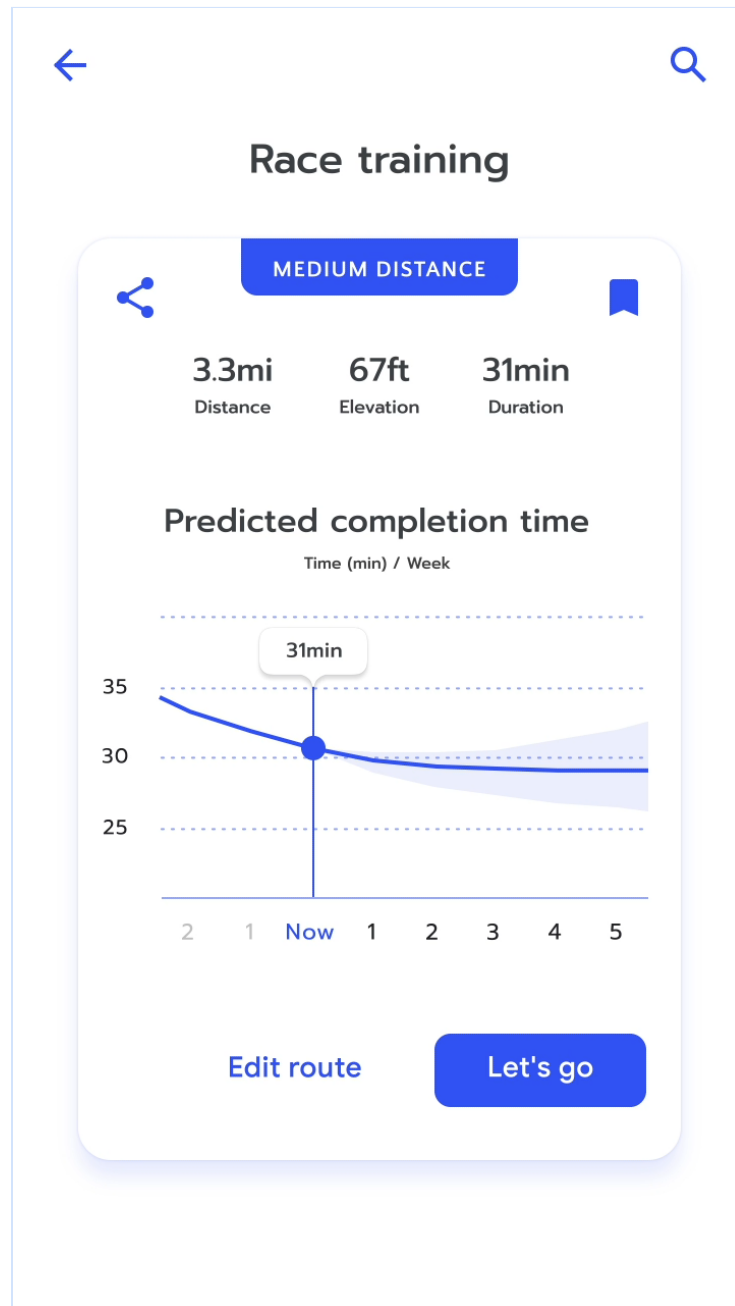
- Because most AI models will never make a prediction with 100% confidence, showing numeric model confidence might confuse users for outputs they consider to be a sure thing. For example, if a user has listened to a song multiple times, the system might still show it as a 97% match rather than a 100% match.



Numeric model confidence visualization

Data visualizations

These are graphic-based indications of certainty – for example, a financial forecast could include error bars or shaded areas indicating the range of alternative outcomes based on the system's confidence level. Keep in mind, however, that some common data visualizations are best understood by expert users in specific domains.



Data visualization of model confidence

Key concept

To assess whether or not showing model confidence increases trust and makes it easier for people to make decisions, you can conduct user research with people who reflect the diversity of your audience. Here are some examples of the types of questions you could ask:

- “On this scale, show me how trusting you are of this recommendation.”
- “What questions do you have about how the app came to this recommendation?”
- “What, if anything, would increase your trust in this recommendation?”
- “How satisfied or dissatisfied are you with explanation written here?”

Once you’re sure that displaying model confidence is needed for your AI product or feature, test and iterate to determine what is the right way to show it.

Apply the concepts from this section in Exercise 4 [in the worksheet](#)

Summary

If and how you offer explanations of the inner-workings of your AI system can profoundly influence the user's experience with your system and its usefulness in their decision-making. The three main considerations unique to AI covered in this chapter were:

- ① **Help users calibrate their trust.** The goal of the system should be for the user to trust it in some situations, but to double-check it when needed. Factors influencing calibrated trust are:
 - **Articulate data sources:** Telling the user what data are being used in the AI's prediction can help your product avoid contextual surprises and privacy suspicion and help the user know when to apply their own judgment.
 - **Tie explanations to user actions:** Showing clear cause-effect relationships between user actions and system outputs with explanations can help users develop the right level of trust over time.
 - **Account for situational stakes:** Providing detailed explanations, prompting the user to check the output in low-confidence/high-stakes situations, and revealing the rationale behind high-confidence predictions can bolster user trust.
- ② **Optimize for understanding.** In some cases, there may be no way to offer an explicit, comprehensive explanation. The calculations behind an output may be inscrutable, even to the developers of those systems. In other cases, it may be possible to surface the reasoning behind a prediction, but it may not be easy to explain to users in terms they will understand. In these cases, use partial explanations.
- ③ **Manage influence on user decisions.** When a user needs to make a decision based on model output, when and how you display model confidence can play a role in what action they take. There are multiple ways to communicate model confidence, each with its own tradeoffs and considerations.

Want to drive discussions, speed iteration, and avoid pitfalls? [Use the worksheet](#)

Feedback + Control

When users give feedback to AI products, it can greatly improve the AI performance and the user experience over time. This chapter covers:

How should the AI request and respond to user feedback?

How can we ensure our AI can interpret and use both implicit and explicit user feedback?

What's the right level of control and customization to give our users?

Want to drive discussions, speed iteration, and avoid pitfalls? [Use the worksheet](#).

What's new when working with AI

User feedback is the communication channel between your users, your product, and your team. Leveraging feedback is a powerful and scalable way to improve your technology, provide personalized content, and enhance the user experience.

For AI products, user feedback and control are critical to improving your underlying AI model's output and user experience. When users have the opportunity to offer feedback, they can play a direct role in personalizing their experiences and maximizing the benefit your product brings to them. When users have the right level of control over the system, they're more likely to trust it.

Key considerations for feedback and control mechanisms:

- ① **Align feedback with model improvement.** Clarify the differences between implicit and explicit feedback, and ask useful questions at the right level of detail.
- ② **Communicate value & time to impact.** Understand why people give feedback so you can set expectations for how and when it will improve their user experience.
- ③ **Balance control & automation.** Give users control over certain aspects of the experience and allow them to easily opt out of giving feedback.

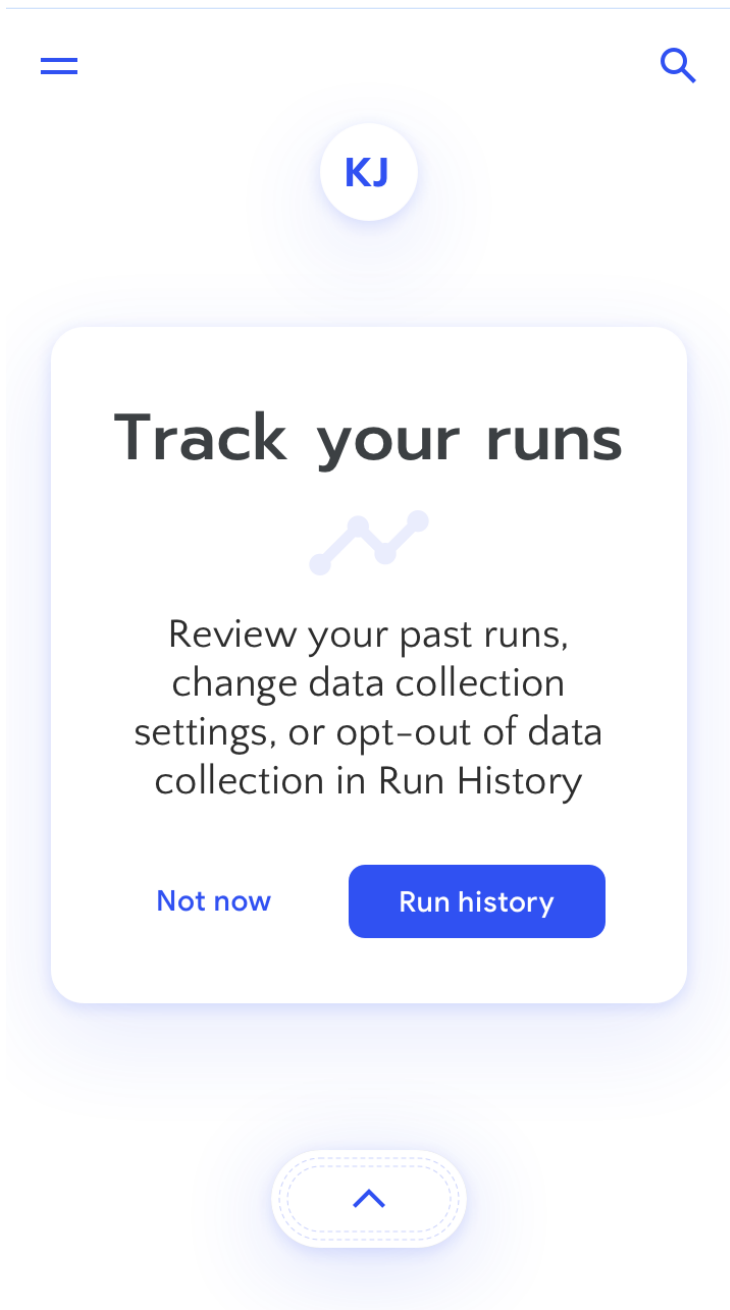
① Align feedback with model improvement

In general, there are implicit and explicit mechanisms for gathering feedback. For either type of feedback, it's important to let users know what information is being collected, what it's for, and how its use benefits them. Whenever possible, find ways to use feedback to improve your AI.

Review implicit feedback

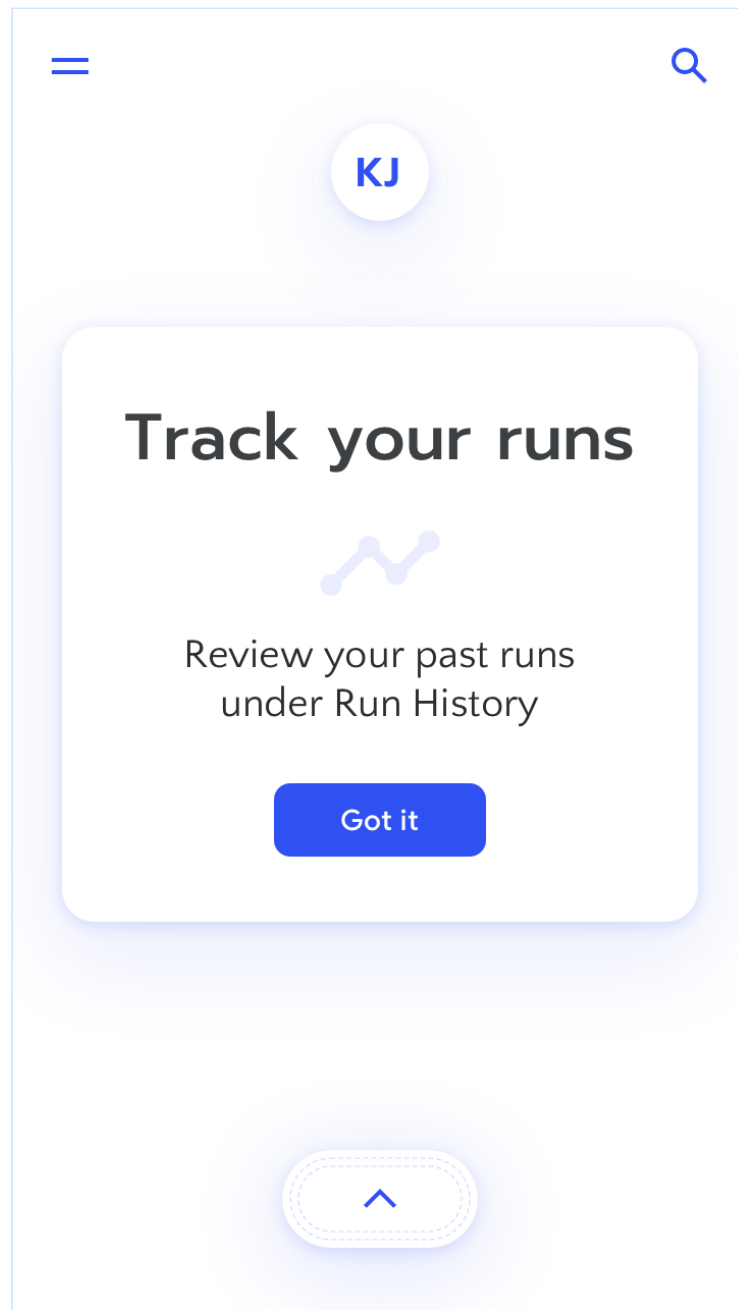
Implicit feedback is data about user behavior and interactions from your product logs. This feedback can include valuable nuggets such as the times of day when users open your app, or the number of times they accept or reject your recommendations. Often, this happens as part of regular product usage — you don't have to explicitly ask for this type of information, but you should let users know you're collecting it, and get their permission up front.

We've talked about privacy a bit in [Mental Models](#) and [Data Collection + Evaluation](#). Users aren't always aware of when their actions are being used as input or feedback, so be sure to review the considerations in the [Explainability + Trust](#) and [Mental Models](#) chapters when explaining how you'll use this data. In particular, you should allow users to opt out of certain aspects of sharing implicit feedback — like having their behavior logged — and this should be included in your terms of service.



Aim for

Let the user know where they can see their data and where they can change data-collection settings. Ideally, do this in-context. [Learn more](#)



Avoid

Don't implicitly collect data without telling people. Always provide a way to see, and ideally edit, data collected.

Collect explicit feedback

Explicit feedback is when users deliberately provide commentary on output from your AI. Often, this is qualitative, like whether or not a recommendation was helpful, or if and how a categorization — such as a photo label — was wrong. This can take many forms such as surveys, ratings, thumbs up or down, or open text fields.

The question and answer choices you provide in explicit feedback should be easy for users to understand. It's also important that your wording choices hit the appropriate voice and tone for the type of feedback you're requesting, and avoid words or references that can be interpreted as offensive. Jokes most likely aren't appropriate when asking about something serious.

Explicit feedback can be used in two ways:

1. You or your team can review user feedback for themes and make changes to the product accordingly.
2. It can be fed directly back into the AI model as a signal.

If you're building for the latter, make sure that the feedback data you receive can actually be used to improve your model. Both your system and your users should understand what the feedback means.

In most cases, options for feedback responses should be mutually exclusive and collectively exhaustive. For example, thumbs up versus a thumbs down is unambiguous ("yay" or "nay"), mutually exclusive (not "yay and nay"), and covers the full range of useful opinions ("meh" isn't very actionable). For more granular feedback, show options that match with the way that users evaluate their experiences.

Interpret dual feedback

Sometimes a single piece of feedback contains both implicit and explicit signals at the same time. For example, public 'likes' are both a way to communicate with others (explicitly) and useful data to tune a recommendations model (implicitly). Feedback like this can be confusing because there isn't always a clear link between what a user does and what a user wants from the AI model. For example, just because someone interacts with a piece of content doesn't mean they want to see more of the same. They could be trying to dismiss it — or giving in to a temporary curiosity.

In these cases, it's important to consider how you'll use the implicit signal for model tuning. Not every action can be interpreted in the same way. Similarly, if explicit feedback can have a wide interpretation, say from "this is OK" to "this is the best thing ever and all I want to see are things like this from now on", consider decreasing how it impacts tuning the model.

Design for model tuning

Ideally, what people want to give feedback on aligns with what data is useful to tune your model. However, that's not guaranteed. Find out what people expect to be able to influence by conducting user research. For example, when using a video recommender system, people may want to give feedback at a different conceptual level than the AI model understands. They may think "show me more videos about parenthood" while the model interprets "show me more videos by this creator".

Discuss with your cross-functional stakeholders all the tradeoffs of collecting or not collecting different types of feedback. Strong, unequivocal signals are great for tuning, but be thoughtful in how you surmise intent based on behavior. Sometimes, looking at interactions over a longer period of time can help you distill more accurate patterns of behavior and intent.

Key concept

List out as many events and corresponding feedback opportunities as possible that could provide data to improve the AI in your product. Cast a wide net - App Store reviews, Twitter, email, call centers, push notifications, etc. Then, systematically ask what your users and data are telling you about your product's experience.

1. What user experience is triggering this feedback opportunity?
2. What kind of content are they providing feedback on?
3. Is this feedback implicit or explicit?

Apply the concepts from this section in Exercise 1 [in the worksheet](#)

② Communicate value & time to impact

For people to take the time to give feedback, it needs to be valuable and impactful. How you communicate this is key to whether or not your users will engage. Keep in mind that “value” is often tied to motivation, so frame your feedback requests in terms of specific user benefits.

Before we get into messaging considerations for communicating the user benefit of feedback, let's start with why people give feedback in the first place.

Understand why people give feedback

There are many reasons people choose to give feedback about a product or experience. Here are a few of the canonical reasons along with the pros and cons of each.

Material rewards

Cash payments are highly motivating. Mechanical Turk is an example of this type of reward for feedback at scale.

Pros

- A direct solution to increase feedback
- May increase the volume of feedback

Cons

- Costly to run over time
- May devalue intrinsic motivations
- Biases for a subset of users
- May decrease feedback quality

Symbolic rewards

These can include status attainment, such as virtual badges, social proof and group status by projecting a self image to a community, and social capital, such as a reputation as an expert.

Pros

- Low to no cost

Cons

- Relies on users caring about how they're perceived
- Creates power imbalances in the community
- May inhibit intrinsic motivation

Personal utility

These include “quantified self” experiences including allowing users to track their progress, bookmark things for later, and explicitly training a personalized AI model – like a recommendation engine– for more relevant output later on.

Pros

- No network effects or community necessary to begin

Cons

- Privacy does not support community development
- May inhibit intrinsic motivation

Altruism

Altruistic motivations can include community building and helping other people make decisions, such as leaving a product review, as well as trying to increase fairness, like giving a conflicting opinion by disagreeing with a particular product review.

Pros

- Potential for more honest feedback based on a desire to help

Cons

- Social desirability biases may lead to extremes in feedback content
- Decrease in contributions if the opinion is already represented
- Altruism levels may vary across cultures or groups

Intrinsic motivation

Intrinsic motivation is the internal fulfillment people get from the act of expressing themselves. This includes direct enjoyment from giving feedback, the ability to vent and express opinions, and the enjoyment of community participation.

Pros

- No network effects or community needed to start
- People like to do things they enjoy

Cons

- Social desirability biases may lead to extremes in feedback content

Align perceived and actual user value

If the benefit isn't clear and specific, users may not understand why they should give feedback. They might avoid giving feedback, or if they can't avoid it, they could give meaningless responses, or feedback that ends up being harmful to your product or community.

If the user thinks their feedback is only valuable to the product developers, some might decide purposefully give bad feedback. For example, if users assume that the intent behind your "free" app is actually to collect data to sell to advertisers without telling them, this could color the feedback they give in surveys.

Ideally, users will understand the value of their feedback and will see it manifest in the product in a recognizable way. It's up to you to connect the value users think they're getting with what your AI can actually deliver.

Connect feedback to user experience changes

Simply acknowledging that you received a user's feedback can build trust, but ideally the product will also let them know what the system will do next, or how their input will influence the AI.

It's not always possible to communicate specifically when and how a user's feedback will change their individual experience. But if you do, make sure your product can deliver on the timeline you promise. If you can, let users give feedback that has an impact right away.

Here are some approaches for describing feedback timing and impact:

1. "Thanks for your feedback"

- Impact timing: None
- Scope: None

2. "Thanks! Your feedback helps us improve future run recommendations"

- Impact timing: "future" broadly
- Scope: All users' "recommendations"

3. "Thanks! We'll improve your future run recommendations"

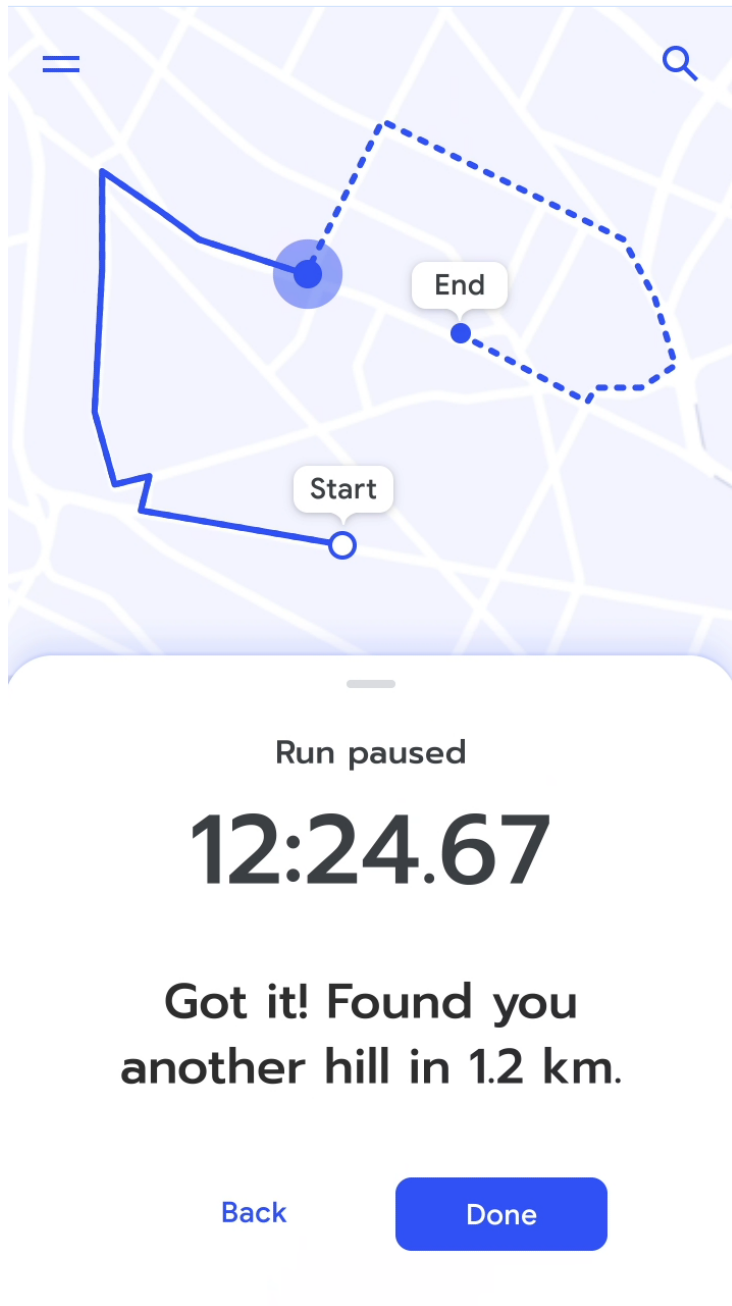
- Impact timing: "future" broadly
- Scope: Your "recommendations"

4. "Thanks! Your next run recommendation won't include hills"

- Impact timing: "next". This is a bit vague, unless there's an established cadence.
- Scope: "Hills" category

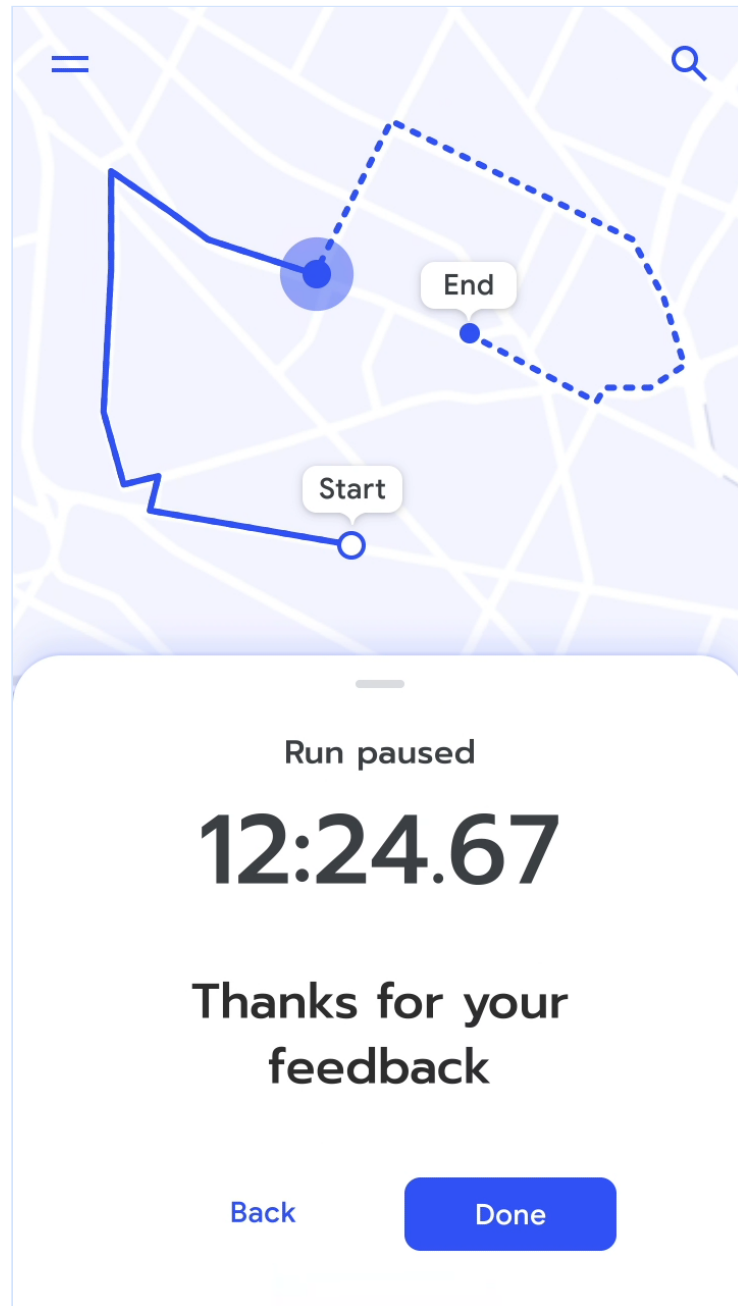
5. "Thanks. We've updated your recommendations. Take a look"

- Impact timing: “We’ve updated” implies immediately
- Scope: Demonstrated content updates



Aim for

Acknowledge user feedback and adjust immediately—or let users know when adjustments will happen. [Learn more](#)



Avoid

Don't just thank users—reveal how feedback will benefit them. They'll be more likely to give feedback again.

Set expectations for AI improvements

For many systems, even with user feedback, much of the AI output is likely to be the same as it was before. As a user provides more and more feedback, each piece will likely have less of an effect on the AI model. It's also possible the improvements you make to your model will be too subtle for your users to register right away.

Sometimes feedback isn't connected to improving the AI model at all. For example, a product may include a "show more/less of this" feedback button as part of recommendations, but that feedback may not be used to tune the AI model. It could simply be a filter on what content is shown and not have any impact on future recommendations. If this is the case, and users have a mental model of immediate tuning, this can create mismatched expectations and your users could be confused, disappointed, and frustrated.

To avoid this situation, you can use the messaging examples above to make sure the feedback scope and time to impact are clear. The user experience will be better if users know how long it will take for your model to adjust to their needs.

Even with the best data and feedback, AI model improvements are almost never possible to implement immediately. Your engineering team may need to wait to implement model updates until they have additional signals from an individual, more data from a group, or a version release.

The reality of these delays means you need to set clear expectations for when people should expect improvements to the model's performance or better output relevance from your AI-powered product. You can do this with clear design and messaging.

Key concept

When thinking about opportunities to ask for user feedback, think about how and when it will improve their experience with the AI. Ask yourself the following questions about each feedback request:

1. Do all of your user groups benefit from this feedback?
2. How might the user's level of control over the AI influence their willingness to provide feedback?
3. How will the AI change based on this feedback?
4. When will the AI change based on this feedback?

Apply the concepts from this section in Exercise 2 [in the worksheet](#)

③ Balance control & automation

For AI-driven products, there's an essential balance between automation and user control. Your product won't be perfect for every user, every time, so allow users to adapt the output to their needs, edit it, or turn it off. Their context in the real world, and their relationship with the task at hand dictates how they use your product. We talk more about whether or not to automate tasks or augment processes in the chapter on User Needs + Defining Success.

Understand when people want to maintain control

When building AI-powered products, it's tempting to assume that the most valuable product is one that automates a task that people currently do manually. This could be taking a process that currently requires seven steps to complete, and compressing it into one command. For example, imagine a music app that could generate themed song collections, so users don't have to take the time to review artists, listen to tracks, decide, and then compile the song list.

The user benefit in products like these may seem obvious, but the teams who make these products typically learn valuable lessons about how users feel about automation and control. There are some predictable situations when people prefer to remain in control of task or process — whether it has AI or not. We talk more about whether or not to automate tasks or augment processes in the User Needs + Defining Success chapter, but here's a short refresher.

People enjoy the task

Not every task is a chore. If you enjoy doing something, you probably don't want to automate the entire process.

People feel personally responsible for the outcome

For most small favors or personal exchanges, people prefer to remain in control and responsible to fulfill the social obligations they take on.

The stakes of the situation are high

People typically like to remain accountable for physical stakes such as safety or health, emotional stakes like expressing feelings, or financial stakes like sharing banking information.

Personal preferences are hard to communicate

In situations where people have a creative vision, many people prefer staying in control so they can maintain ownership and see their plan through to execution.

Understand when people will give up control

There are of course plenty of times when people feel perfectly fine giving up control, and prefer the help of an AI or automated system.

When they are unable to do a task

Often people would do something if they knew how or had time, but don't so they can't. These limitations can be temporary.

When a task is unpleasant or unsafe

Most people would prefer to give up control to avoid tasks that require a significant effort or risk for little enjoyment or gain.

Allow for opting out



When first introducing your AI, consider allowing users to test it out or turn it off. Once you've clearly explained the benefits, respect their decision not to use the feature. Keep in mind, they may decide to use it later.


For your product to augment human tasks and processes, people need to be able to control what it does based on their situation. One way to allow for this is to provide a way for users to complete their task the regular, non-automated way. As we mentioned in the [Mental Models](#) chapter, the manual method is a safe and useful fallback. Because errors and failure are critical to improving your AI, your users will definitely need a manual failsafe, especially in the early days of using your product.

Keep in mind the relative priority of your product in the daily lives of your users. The people using your product are most likely multitasking, using other products or apps, and generally getting pulled in many directions. For example, for a navigation app, suggesting a faster alternate route while someone is driving could get them home more quickly, but if their attention is split between passengers and traffic conditions, it could be dangerous to use. Remember that your product likely isn't the focus of your users' lives, so keep engagement requests strategic, minimal, and allow for easy dismissal.

Provide editability

A user's preferences may change over time, so consider how you'll give them control over what preferences they communicate to your ML model, and give them the ability to adjust it. Even if your system's previous suggestion or prediction wasn't relevant before, it may become relevant in the future. Your product should allow for people to erase or update their previous selections, or reset your ML model to the default, non-personalized version.





Run summary

29:34.91

3.4mi

Distance

789ft

Elevation

8:43

Avg/Mi

Overview

Elevation

Feedback

Workout profile

This run route doesn't seem to be a good match.
Adjust your suggestions in your workout profile.

Difficulty

Easy

Medium

Hard

Not now

Update

Aim for

Allow users to adjust their prior feedback and reset the system. [Learn more](#)

Key concept

Take time to think about your users' expectations for control over certain tasks or processes. Here's a quick checklist for you and your team to run through before setting up feedback and control mechanisms:

1. Can your AI accommodate a wide range of user abilities and preferences?
2. Does your AI deal with highly-sensitive domains, such as health, wealth, or relationships?
3. Will your AI take a long time to get to the target level of accuracy and usefulness?
4. Is your AI used in high-stakes situations, where it introduces a new mental model?
5. Are there likely changes in the user's needs that would require them to "reset" or otherwise "take over" for the model?

Apply the concepts from this section in Exercise 3 [in the worksheet](#)

Summary

When building your AI-powered feature or product, feedback and user control are critical to developing communication and trust between your user and the system, and for developing a product that fulfills your users' needs consistently over time. Feedback mechanisms partner closely with Mental Models, Explainability + Trust, and how you'll tune your AI. These three aspects are key in improving your product for your users.

Remember when working with AI, there are three new considerations for feedback and control mechanisms:

- ① **Align feedback with model improvement.** Clarifying the differences between implicit and explicit feedback, and asking the right questions at the right level of detail.
- ② **Communicate value & time to impact.** Understanding why people give feedback, and building on existing mental models to explain benefits and communicate how user feedback will change their experience, and when.
- ③ **Balance control & automation.** Helping users control the aspects of the experience they want to, as well as easily opting out of giving feedback.

Want to drive discussions, speed iteration, and avoid pitfalls? [Use the worksheet](#)

Errors + Graceful Failure

When AI makes an error or fails, things can get complicated. This chapter covers:

When do users consider low-confidence predictions to be an “error”?

How will we reliably identify sources of error in a complex AI?

Does our AI allow users to move forward after an AI failure?

Want to drive discussions, speed iteration, and avoid pitfalls? [Use the worksheet.](#)

What's new when working with AI

As users interact with your product, they'll test it in ways you can't foresee during the development process. Misunderstandings, false starts, and impropriety will happen, so designing for these cases is a core component of any user-centered product. Errors are also opportunities. They can support faster learning by experimentation, help establish correct mental models, and encourage users to provide feedback.

There are some great guides for communicating errors that still apply when creating AI-driven features and products, including these [Error Messaging Guidelines from the Nielsen Norman Group](#).

Key considerations for dealing with errors in AI-driven systems:

- ① **Define “errors” & “failure”.** What the user considers an error is deeply connected to their expectations of the AI system. For example, a recommendations system that's useful 60% of the time could be seen as a failure or a success, depending on the user and the purpose of the system. How these interactions are handled establishes or corrects mental models and calibrates user trust.
- ② **Identify error sources.** With AI systems, errors can come from many places, be harder to identify, and appear to the user and to system creators in non-intuitive ways.
- ③ **Provide paths forward from failure.** AI capabilities can change over time. Creating paths for users to take action in response to the errors they encounter encourages patience with the system, keeps the user-AI relationship going, and supports a better overall experience.

① Define “errors” & “failure”

Integrating AI into your product means establishing a relationship that changes as the user interacts with the AI system. Users will likely come to expect that interactions are curated to their tastes and behavior, and that experiences will differ from user to user.

What defines an error, then, can also differ from user to user based on their expertise, goals, mental models, and past experience using the product.

For example, if someone assumes that an AI-driven music recommendation system is only using songs indicated as “favorites” to provide recommendations, then they might consider it an error if the system recommends a genre outside of what’s in their favorites list. Someone else who assumes recommendations are based on a wide range of factors might not consider this an error. There’s more information on fostering a good user understanding of the system’s capabilities, limitations, and interaction model in the chapter on [Mental Models](#).

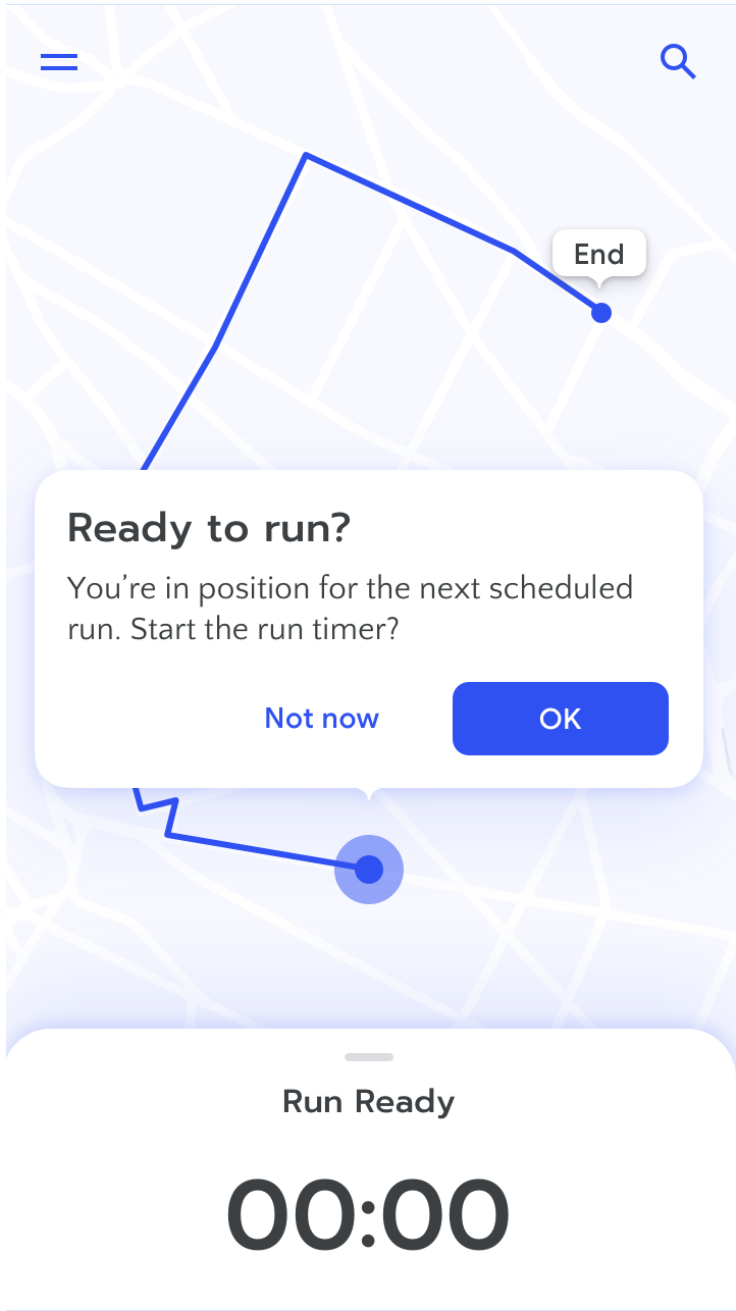
Identify user, system, and context errors

In non-AI systems, “user errors” are typically defined from the point of view of the system designers, and the user is blamed for “misuse” resulting in errors. “System errors” on the other hand, are typically defined from a user’s point of view: users blame the system designer for a product that’s not flexible enough to meet their needs. In AI systems, users can encounter a third type of error, based on the system’s assumptions about the user. These are context errors.

Context errors are instances in which the AI system becomes less useful through making incorrect assumptions about what the user wants to do at a certain time or place. As a result, users might be confused, fail at their task, or abandon the product altogether. Context can be related to individual lifestyle or preferences, or patterns can be linked to wider cultural values.

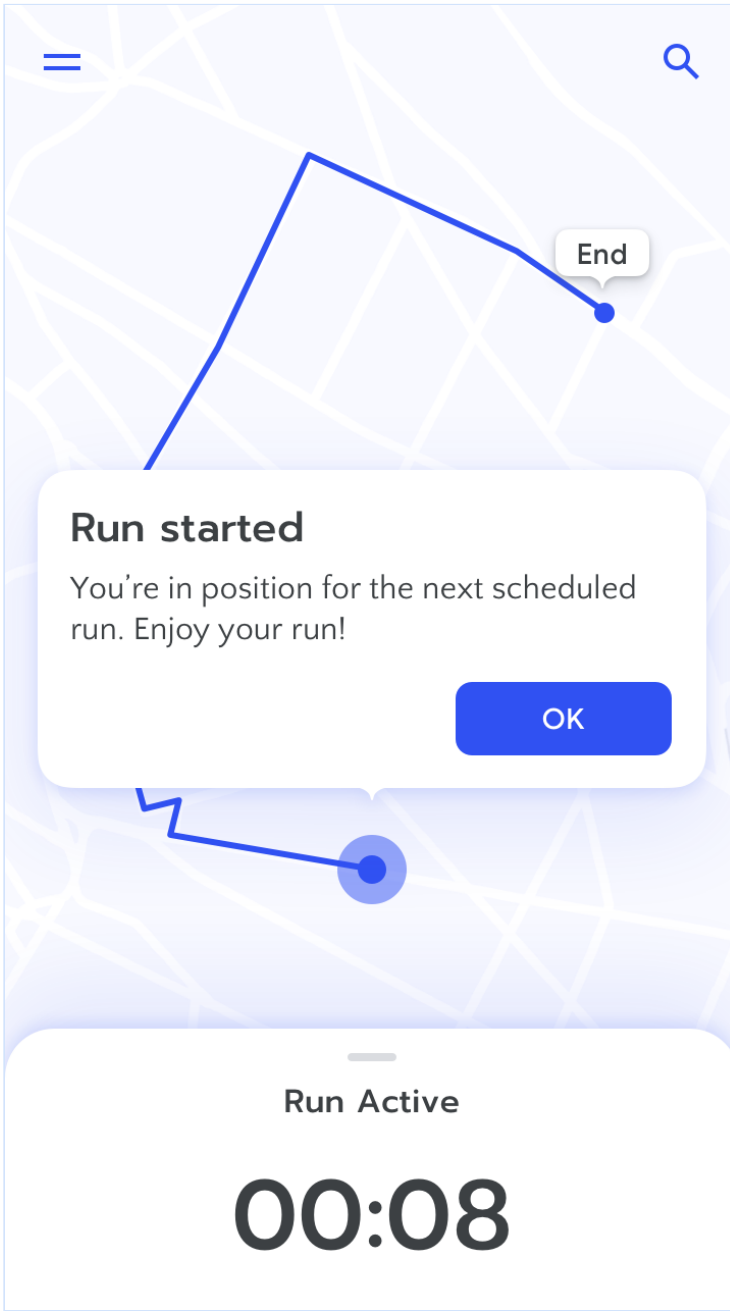
For example, if someone using a recipe suggestion app frequently rejects recommendations in the evening, this pattern should signal to the product team that there may be a persistent reason why. If the user works a night shift, they might simply prefer breakfast recommendations before they head to work. Whereas if all of the users in a certain group consistently rejected meat-based suggestions during a certain time of year, that might be linked to a cultural value or preference that your team hasn’t accounted for.

To prevent or correct for context errors, you'll need to look at the signals that the AI uses to make assumptions about the user's context and evaluate if they are incorrect, overlooked, undervalued, or overvalued.



Aim for

Provide proactive recommendations when AI has high confidence about the user's context. [Learn more](#)



Avoid

Don't act on assumptions. Doing so can lead to context errors.

Context errors and other errors can be thought of as an interaction between user expectations and system assumptions. Here's a breakdown of how user expectations and system expectations interact and cause errors.

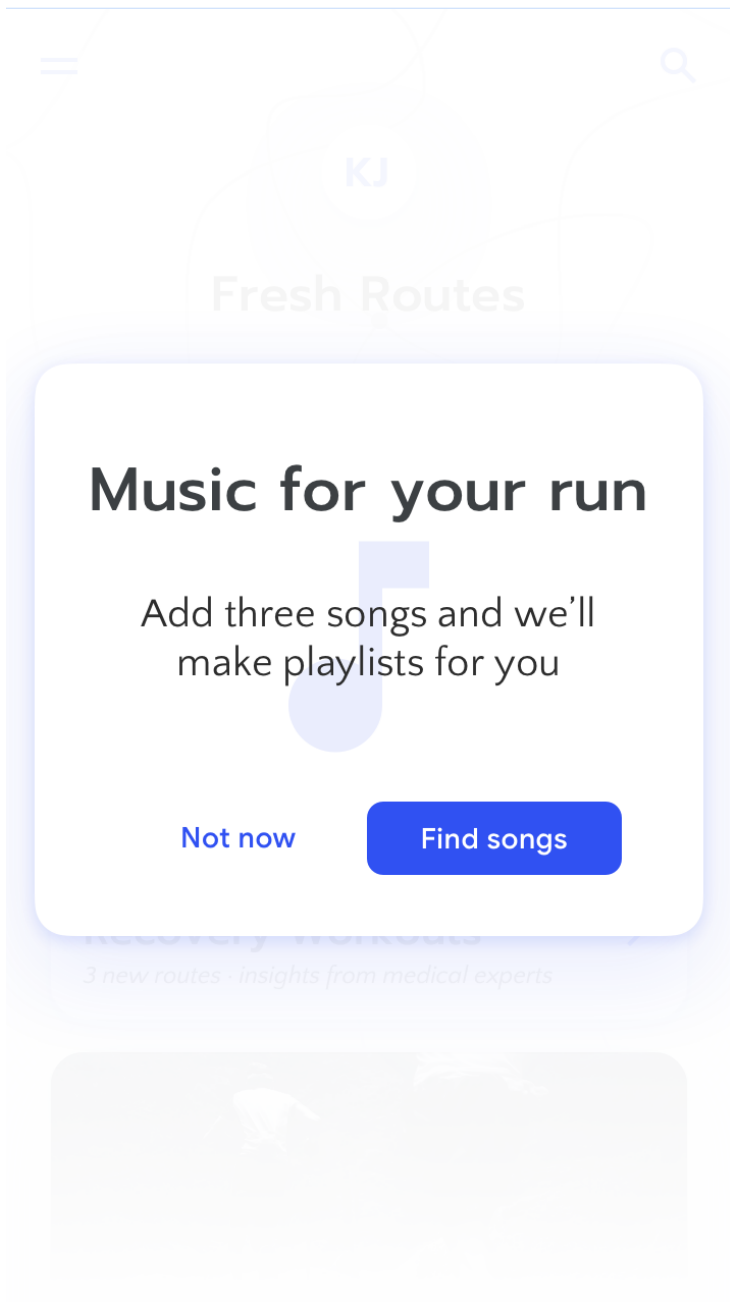
Categorize user-perceived errors

Context errors. These are often true positives : the system is "working as intended," but the user perceives an error because the actions of the system aren't well-explained, break the user's mental model, or were based on poor assumptions. For example, if a friend's flight confirmation email creates an event on your calendar, but you don't want or need it.

Addressing these errors depends on their frequency and severity. You could change the way the system functions to better align with user needs and expectations. Or, adjust your product's onboarding to establish better mental models and shift the user's perception of these situations from errors to expected behaviors.

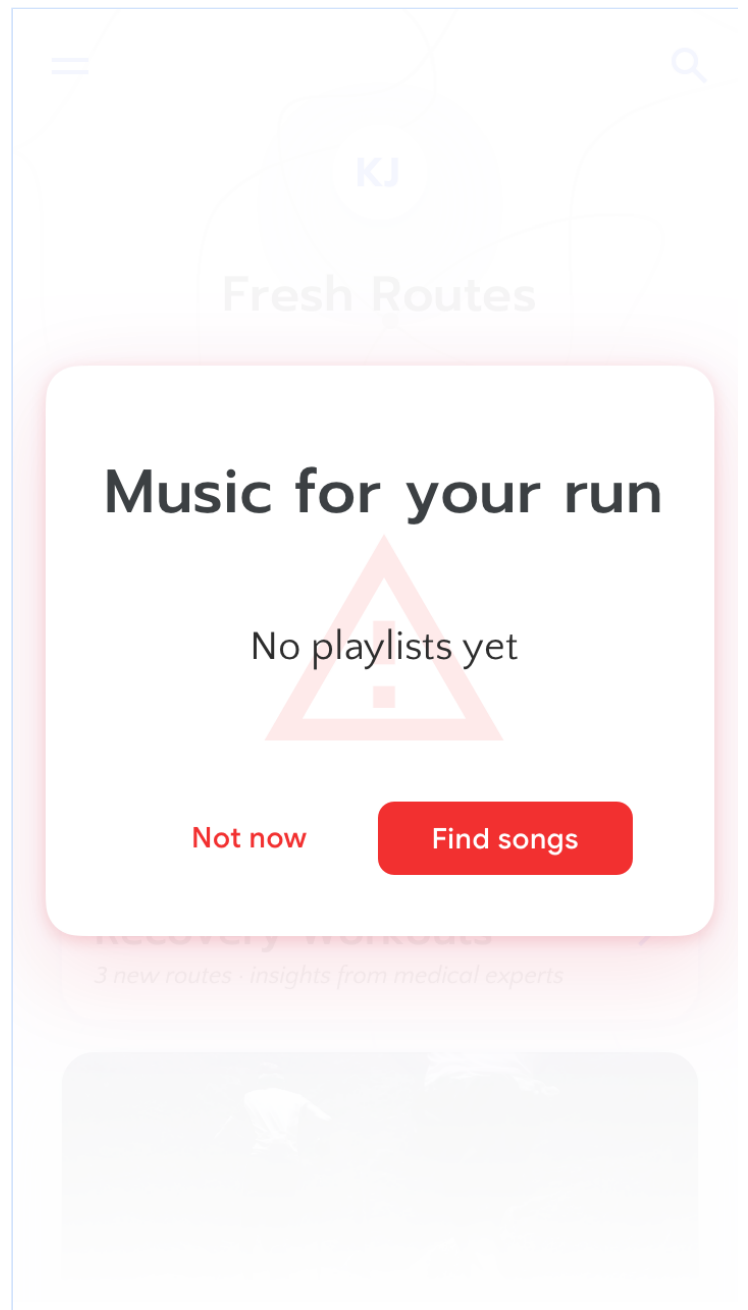
Failstates. Your system can't provide the right answer, or any answer at all due to inherent limitations to the system. These could be true negatives : the ML is correct in discerning that there's no available output for a given input, but according to users, there should be. For example, if an image recognition app can identify many different kinds of animals, but a user shows the app a photo of an animal that wasn't in the training dataset and therefore is not recognized, that's still a true negative.

Messages about user-perceived errors should inform the user as specifically as possible about the system's limitations.



Aim for

Use error states to tell the user what inputs the AI needs or how the AI works. [Learn more](#)



Avoid

Don't miss an opportunity to explain your AI system.

Diagnose errors that users don't perceive

Since these errors are invisible to the user, you don't need to worry about how to explain them in the user interface, but being aware of them could help you improve your AI.

Happy accidents. The system might flag something as a poor prediction, limitation, or error, but in rare cases it could be helpful or interesting anyway. For example, a user asks a smart speaker to take out the trash as a joke, knowing it isn't possible, and she and her family are amused at the speaker's response.

Background errors. Situations in which the system isn't working correctly, but neither the user nor the system register an error. For example, if a search engine returns an incorrect result and the user is unable to identify it as such.

These types of errors may have significant consequences if undetected for long periods of time. They also have important implications for how your systems measure failure and error rates. Because you're unlikely to detect these errors from user feedback, you'll need a dedicated quality assurance process to stress test the system and identify "unknown unknowns".

Account for timing in the user journey

Users may react differently to errors they encounter in the early days of using your product versus later, when they have more ingrained expectations of what it can and can't do. How long the user has been using the product should impact how your AI communicates errors and helps users get back on track.

For example, when a user interacts with a music recommendation system for the first time, they might not consider it an error when the initial recommendations aren't relevant to them. However, after a year of listening, liking, and adding favorites to playlists, the user might have higher expectations for the relevance of the system's recommendations and therefore consider irrelevant recommendations to be errors.

Weigh situational stakes & error risk

In some situations, AI errors and failure are inconvenient, but in others, they could have severe consequences. For example, errors in AI-suggested email responses have very different stakes than errors related to autonomous vehicle handling. This is true for non-AI systems as well, but the complexity of AI systems and the possibility of context errors requires that you give extra thought to the stakes of each situation where your AI is making decisions.

Any given scenario has inherent stakes, but the risk of errors can change depending on other contextual factors. For example, if a user is multitasking or is under time pressure while using your AI-driven product, then they have fewer mental resources available to double-check the system output.

Gauge the risk for potential errors

Lower

- User has expertise in the task
- Extremely high system confidence
- Many possible successful outcomes

Higher

- User is a novice at the task
- Reduced attention or response time (multi-tasking)
- Low system confidence
- Narrow definition of successful outcomes

Assess the stakes of the situation

Lower

- Experimentation
- Play or creativity
- Lightweight entertainment
- Non-essential recommendations

Higher

- Health, safety, or financial decisions
- Sensitive social contexts

When considering what system responses, user responses, and messaging are required in different situations, refer to the section on designing your reward function in the [User Needs + Defining Success](#) chapter. Refer also to the high-stakes scenarios and corresponding required explanations identified in the [Explainability + Trust](#) chapter.

Key concepts

Before you start designing how your system will respond to errors, try to identify errors that your users can perceive already, or that you can predict will occur. Then, sort them into the following categories.

1. **System limitation.** Your system can't provide the right answer, or any answer at all, due to inherent limitations to the system.
2. **Context.** The system is “working as intended,” but the user perceives an error because the actions of the system aren't well-explained, break the user's mental model, or were based on poor assumptions.

Also be on the lookout for **background errors**, situations in which the system isn't working correctly, but neither the user nor the system register an error.

For each situation above, ask yourself:

- How does this error impact the user?
- Are the stakes high or low for the user in this situation?

Apply the concepts from this section in Exercise 1 [in the worksheet](#)

② Identify error sources

The User Needs + Defining Success chapter illustrates how to design and evaluate the reward function , which the AI uses to optimize its output. The range and type of possible errors in your system will depend on this reward function, but generally speaking there are several sources of errors unique to AI.

Discover prediction & training data errors

These errors come from issues with your training data and the way you've tuned your machine learning model. For example, a navigation app might not have had training data for roads in a certain geographic region, limiting its capabilities.

Mislabeled or misclassified results

When the output from the system is incorrectly labeled or classified due to poor training data. For example, inconsistently-labeled berries in the training data for a plant classification app leads to misclassification of strawberries as raspberries.

Response: Allow users to give guidance or correct the data or label, which feeds back into the model to improve the dataset or alert the team to the need for additional training data.

Poor inference or incorrect model

When the machine learning model isn't sufficiently precise , despite having adequate training data. For example, a plant classification app has a well-labeled dataset that includes berries, but poor model tuning returns a large number of false positives when identifying strawberries.

Response: Allow users to give guidance, or correct the data or label, which feeds back into the AI model and helps you tune it.

Missing or incomplete data

When the user reaches the edges of what the model can do based on the data it was trained on. For example, if a user tries to use a plant classification app on a dog.

Response: Communicate what the system is supposed to do and how it works. Then, explain what it's missing or its limitations. Allow users to give feedback about the needs that the system isn't meeting.

Review the chapters on [Data Collection + Evaluation](#) and [Explainability + Trust](#) to see more information on how to identify the right data sources and how to explain them.

Predict or plan for input errors

These errors come from the user's expectation that the system will "understand" what they really mean, whether or not the AI is actually capable of doing so.

Unexpected input

When a user anticipates the auto-correction of their input into an AI system. For example, the user makes a typo and expects the system to recognize their intended spelling.

Response: Check the user's input versus a range of "expected" answers to see if they intended one of those inputs. For example, "Did you mean to search for XYZ?"

Breaking habituation

When a user has built up habitual interactions with the system's UI, but a change causes their actions to lead to a different, undesired, result.

For example, in a file storage system, the user has always accessed a folder by clicking in the top right hand region of the interface, but due to newly-implemented AI-driven dynamic design, folder locations change frequently. Clicking in that area due to muscle memory now opens the wrong folder.

Response: Implement AI in ways that don't break habituation, such as by designating a specific area of the interface for less-predictable AI output. Or, allow users to revert to, choose, or retrain a specific interaction pattern. See more suggestions in this [article on habituation from the Google Design blog](#).

Miscalibrated input

When a system improperly weights an action or choice. For example, if one search for an artist in a music app leads to endless recommendations for that artist's music.

Response: Explain how the system matches inputs to outputs and allow the user to correct the system through feedback.

Check output quality for relevance errors

Your AI-driven system won't always be able to provide the appropriate information at the right time. Irrelevance is often the source of context errors — conflicts between the system working as intended and the user's real-life needs. Unlike user input errors, or data errors, these aren't issues with the veracity of the information or how the system makes decisions. These errors are issues in how and when those results are delivered — or not.

Low confidence

When the model can't fulfill a given task due to uncertainty restraints: lack of available data, requirements for prediction accuracy, or unstable information. For example, if a flight price prediction algorithm can't accurately predict next year's prices because of changing conditions.

Response: Explain why a certain result couldn't be given and provide alternative paths forward. For example, "There's not enough data to predict prices for flights to Paris next year. Try checking again in a month".

Irrelevance

When the system output is high confidence but presented to users in a way that isn't relevant to the user's needs. For example, a user books a trip to Houston for a family funeral and their travel app recommends some "fun vacation activities".

Response: Allow the user to provide feedback to improve the system's function.

Disambiguate systems hierarchy errors

These errors arise from using multiple AI systems that may not be communicating with one another. These overlaps can result in conflicts over situational control or a user's attention. Plan ahead for how your AI system might interact with other systems, and provide users with multiple levels of control for managing output and avoiding conflicts.

Multiple systems

When a user connects your product to another system, and it isn't clear which system is in charge at a given time.

For example, a user connects a "smart thermostat" to a "smart energy meter," but the two systems have different methods of optimizing for energy efficiency. The systems send conflicting signals to one another and stop working properly.

Response: Explain the multiple systems that are connected, and allow the user to determine priority. Consider visual ways to represent the relationship between multiple AI systems in the product interface, perhaps by mapping them onto different locations.

Signal crashes

When multiple systems are monitoring a single (or similar) outputs and an event causes simultaneous alerts.

Signal crashes increase the user's mental load because the user has to parse multiple information sources to figure out what happened, and what action they need to take.

For example, imagine if a user's voice input was recognized by multiple voice-activated systems, which simultaneously respond in different ways. That would be overwhelming, and the user likely wouldn't know what to do next.

Response: Allow the user to set independent controls for your AI that don't overlap with other signals. For example, the watch word for a smart speaker to start listening could be unique. If your team can avoid creating new context errors, the system could attempt to infer which system is the one the user intended to have primacy.

Key concepts

For each error, try to determine the cause based on the resulting user experience. Ask yourself what kind of error it could be based on the examples below.

Prediction & training data errors occur when... the system's capabilities are limited by its training data.

Input errors occur when... the user anticipates the auto-correction of their input into an AI system, or their habituation is interrupted.

Relevance errors occur when... the system output appears in a time, place, or format that isn't relevant to the user's needs.

System hierarchy errors occur when... your user connects your product to another system, and it isn't clear which system is in charge.

Apply the concepts from this section in Exercise 2 [in the worksheet](#)

③ Provide paths forward from failure

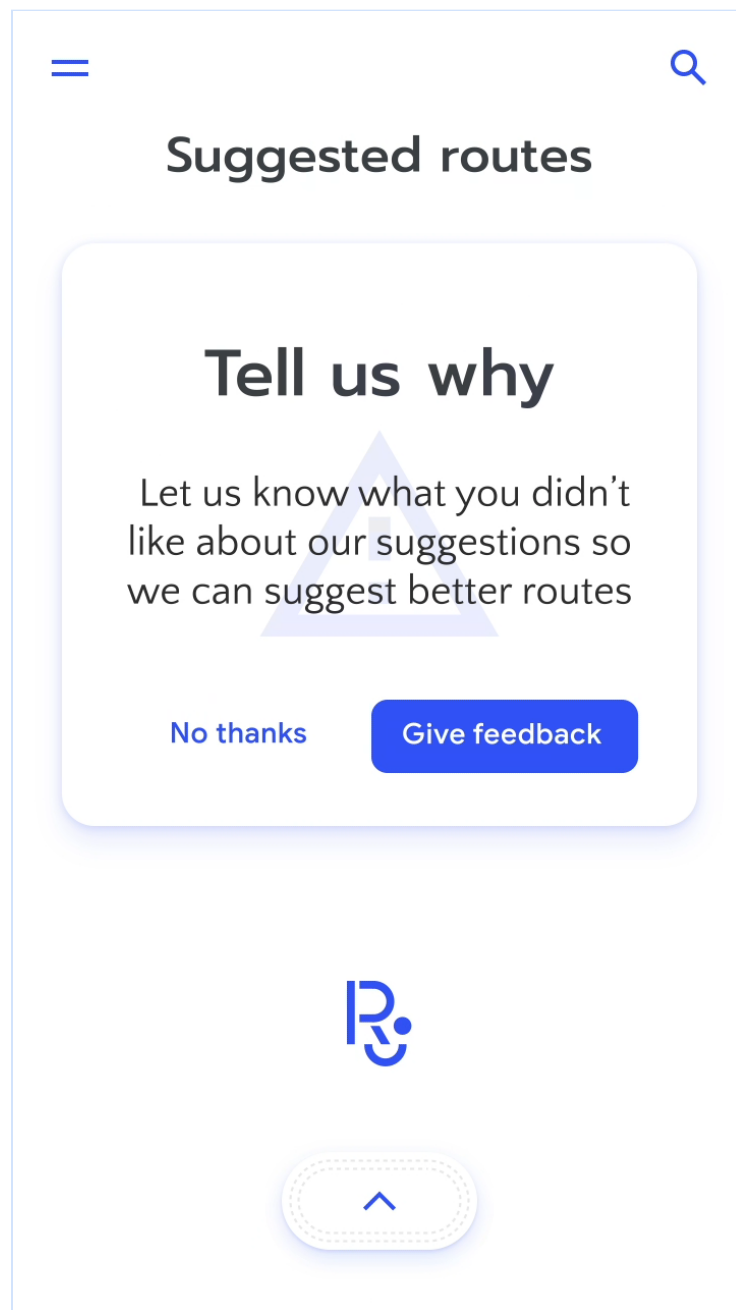
Setting reasonable expectations about the limitations of your technology helps set up good experiences with errors, but how users move forward is equally important. Focusing on what your users can do after the system fails empowers them while maintaining the usefulness of your product.

If you were a waiter in a restaurant, and a customer wanted something on the menu but the kitchen ran out, you'd have come up with an alternative option, taking into account their implied preferences. For example, "We don't have Coke, but is Pepsi OK?" In the case of higher situational stakes, there should be an additional check to make sure the alternative is indeed a safe option. For example, if the user wants a dish the restaurant doesn't have, but the closest alternative contains a common allergen, the waiter would want to think twice before making the recommendation. Your primary goal in these failstates should be to prevent undue harm to the user and help them move forward with their task.

Create opportunities for feedback

As highlighted in the above sections, many of the errors your users experience require their feedback in order to improve the system. In particular, error types that aren't easily recognized by the system itself, such as mislabeled data, rely on outside feedback to fix.

Include opportunities for users to provide feedback both when presented with an error message as well as alongside "correct" system output. For example, ask a user what they expected to happen after a system failure, and provide the option to report inappropriate suggestions.



Aim for

Ask the user for feedback if they repeatedly reject AI outputs. [Learn more](#)

Return control to the user

When an AI system fails, often the easiest path forward is to let the user take over. Whether or not giving users full “override” powers is possible and what it looks like will differ from system to system. In some cases, reverting to manual control could be risky or dangerous. For example, if a navigation system stops giving directions altogether to a user who is navigating through an unfamiliar environment during rush hour traffic.

When this AI-to-manual control transition happens, it’s your responsibility to make it easy and intuitive for users to quickly pick up where the system leaves off. That means the user must have all the information they need in order to take the reins: awareness of the situation, what they need to do next, and how to do it.

Assume subversive use

Though it’s important to write informative and actionable error messages, your team should design your product knowing that some people will intentionally abuse it. That means that in addition to mapping out potential negative impacts of your product, like in the [User Needs + Defining Success](#) chapter, you should try to make failure safe, boring, and a natural part of the product. Avoid making dangerous failures interesting, or over-explaining system vulnerabilities — that can incentivize the users to reproduce them.

For example, if every time a user marked an email in their inbox as spam, the email app explained why the system did not classify that message as spam, that could give spammers useful tips on how to avoid getting caught by the filter.

To get a really clear idea of all the potential dead-ends in your product, invest in beta-testing and pilot programs. Many products are like a maze of options and possibilities; while the builders focus on the happy path, users can get caught in the labyrinth.

Key concepts

You can use quality assurance exercises and pilots initially to find errors, but you should continue monitoring to cover any new features you launch. As a team, decide on some channels you'll monitor for new error discoveries from your users. Which of the following error report sources could work for your team?

- Reports sent to customer service
- Comments and reports sent through social media channels
- In-product metrics
- In-product surveys
- User research
 - Out-of-product surveys
 - Deep-dive interviews
 - Diary studies

As you discover each error, you'll then need to work as a team to design the correct path forward from that error.

Apply the concepts from this section in Exercise 2 in the worksheet

Summary

Learning, machine or otherwise, can't happen without making mistakes. Designing and building your system with the knowledge that errors are integral will help you create opportunities for dialogue with your users. This in turn creates more efficient pathways for error resolution, and for the users to complete their goals.

When designing your error experience, be human, not machine. Error messages may need to disclose that the system made a mistake. Address mistakes with humanity and humility, and explain the system's limits while inviting people to continue forward.

- ① **Define “errors” & “failure”.** When dealing with a probabilistic , dynamic system, a user could perceive a failure in situations where the system is working as intended. Acknowledging that a product is a work-in-progress can help encourage the adoption and feedback that designers and engineers need to continue improving the AI.
- ② **Identify error sources.** The inherent complexity of AI-powered systems can make identifying the source of an error challenging. It's important to discuss as a team how you'll discover errors and discern their sources.
- ③ **Provide paths forward from failure.** The trick isn't to avoid failure, but to find it and make it just as user-centered as the rest of your product. No matter how hard you work to ensure a well-functioning system, AI is probabilistic by nature, and like all systems, will fail at some point. When this happens, the product needs to provide ways for the user to continue their task and to help the AI improve.

Want to drive discussions, speed iteration, and avoid pitfalls? [Use the worksheet](#)