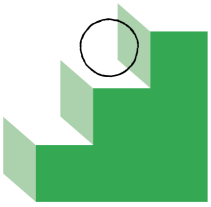
User Needs + Defining Success

Chapter worksheet



Instructions

Block out time to get as many cross-functional leads as possible together in a room to work through these exercises & checklists.

Exercises

1. Evidence of user need [multiple sessions]

Gather existing research and make a case for using AI to solve your user need.

2. Augmentation versus automation [multiple sessions]

Conduct user research to understand attitudes around automation versus augmentation.

3. Design your reward function [~1 hour]

Weigh the trade offs between precision and recall for the user experience.

4. Define success criteria [~1 hour]

Agree on how to measure if your feature is working or not, and consider the second order effects.

1. Evidence of user need

Before diving into whether or not to use AI, your team should gather user research detailing the problem you're trying to solve. The person in charge of user research should aggregate existing evidence for the team to reference in the subsequent exercises.

User research summary

List out the existing evidence you have supporting your user need. Add more rows as needed.

Date	Source	Summary of findings
2/8/26	Project Manager workflow observation	Product managers must manually translate high-level requirements into granular, assignable developer tasks
2/8/26	New hire onboarding observation	New developers spend significant time asking colleagues about codebase context which delays productivity
2/8/26	Industry research	Popular task management solutions do not support assigning ticket creation to AI services

Make a case for and against your AI feature

Meet as a team, look at the existing user research and evidence you have, and detail the user need you're trying to solve.

User Need Description (High Level): Software engineering teams need context specific AI tools to streamline workflows.

Next, write down a clear, focused statement of the user need and read through each of the statements below to identify if your user need is a potential good fit for an AI solution.

User Need Description (Project Manager): Project managers need to streamline the task creation process given new product requirements.
User Need Description (New Hire): New hires need a faster way to answer project specific questions without consuming their colleagues' time.

At the end of this exercise your team should be aligned on whether AI is a solution worth pursuing and why.

How might we **solve** the inefficiency of task creation and repository knowledge access for software teams?

Can AI solve this problem in a unique way?

AI probably better	AI probably **not** better
☐ The core experience requires recommending different content to different users.	☐ The most valuable part of the core experience is its predictability regardless of context or additional user input.
☐ The core experience requires prediction of future events.	☐ The cost of errors is very high and outweighs the benefits of a small increase in success rate.
☑ Personalization will improve the user experience.	☑ Users, customers, or developers need to understand exactly everything that happens in the code.
☑ User experience requires natural language interactions.	☐ Speed of development and getting to market first is more important than anything else, including the value using AI would provide.
☑ Need to recognize a general class of things that is too large to articulate every case.	☐ People explicitly tell you they don't want a task automated or augmented.
☐ Need to detect low occurrence events that are constantly evolving.	
☑ An agent or bot experience for a particular domain.	
☐ The user experience doesn't rely on predictability.	

We think AI **can** help solve inefficient task management and repository knowledge access for software teams, because:

- Repository Q&A requires synthesizing large codebases that exceed human working memory.
- Breaking down product requirements into actionable tasks is a problem requiring both natural language processing and project context, which RAG-enhanced AI can handle at scale.

2. Augmentation versus automation

Conduct research to understand user attitudes

If your team has a hypothesis for why AI is a good fit for your user's need, conduct user research to further validate if AI is a good solution through the lens of automation or augmentation.

If your team is light on field research for the problem space you're working in, contextual inquiries can be a great method to understand opportunities for automation or augmentation.

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

Research protocol questions

- 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?
  (Task Creation) General project management skills.
  (Q&A) Code base and company documentation understanding.
- Tell me more about that action you just took, is that an action you repeat:
  - Hourly
  - Daily (Q&A)
  - Weekly (Task Creation)
  - Monthly
  - Quarterly
  - Annually
- If you had a human assistant to work with on this task, what, if any, duties would you give them to carry out?
  (Task Creation) Product requirement and code base review.
  (Q&A) Company documentation and code base review.

If going to meet your users in context isn't feasible, you can also look into prototyping a selection of automation and augmentation solutions to understand initial user reactions.

The Triptech method is an early concept evaluation method that can be used to outline user requirements based on likes, dislikes, expectations, and concerns.

Research protocol questions

- Describe your first impression of this feature.
- How often do you encounter the following problem: [insert problem/ need statement here]?
    - Daily
    - Often (a few times a week)
    - Sometimes (a few times a month)
    - Rarely (a few times a year)
    - Never
- How important is it to address this need or problem?
    - Not at all important
    - Somewhat important
    - Moderately important
    - Very important
    - Extremely important

3. Design your reward function

Once your team has had a chance to digest your recent research on user attitudes towards automation and augmentation, meet as a team to design your AI's **reward function**. You'll revisit this exercise as you continue to iterate on your feature and uncover new insights about how your AI performs.

Use the template below to list out instances of each reward function dimension.

Reward function template

	Prediction: Positive	Prediction: Negative
Reference: Positive	True Positive	False Negative
	AI correctly generates a useful task	AI fails to generate a task that should exist
	AI correctly answers a repo question	AI says "I don't know" when answer exists in code
Reference: Negative	False Positive	True Negative
	AI generates an irrelevant/ duplicate task	AI correctly avoids generating unnecessary tasks
	AI provides incorrect information about code	AI correctly declines questions outside its knowledge

Take a look at the false positives and false negatives your team has identified.

- If your feature offers the most user benefit for **fewer false positives**, consider optimizing for **precision**.

- If your feature offers the most user benefit for **fewer false negatives**, consider optimizing for **recall**.

> Our AI model will be optimized for_____**{ precision / recall }**_____
> because_____**{ user benefit }**_____
> _____.
> We understand that the tradeoff for choosing this method means our model will _____**{ user impact }**_____
> _____
> _____.

Task Service: Optimize for Precision

Our AI model will be optimized for precision because generating irrelevant or incorrect tasks wastes PM time and erodes trust. It's better to generate fewer high-quality tasks than many low-quality ones. We understand that the tradeoff for choosing this method means our model may miss some valid tasks, however, the created tasks will be highly relevant.

Info Service: Optimize for Recall

Our AI model will be optimized for recall because missing relevant information frustrates users seeking answers. False positives (wrong info) are caught by developers; false negatives leave questions unanswered. We understand that the tradeoff for choosing this method means our model will occasionally produce irrelevant information, but it will always attempt to answer questions.

4. Define success criteria

Now that you've done the work to understand whether AI is a good fit for your user need and identified the tradeoffs of your AI's reward function, it's time to meet as a team to define success criteria for your feature. Your team may come up with multiple metrics for success by the end of this exercise.

By the end of this exercise, everyone on the team should feel aligned on what success looks like for your feature, and how to alert the team if there is evidence that your feature is failing to meet the success criteria.

Success metrics framework

Start with this template and try a few different versions:

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

Metric 1 (Version 1)

If JSON schema validation rate **for** AI-generated tickets drops below 100%, we will fix structured output formatting and validation logic**.**

Metric 2 (Version 1)

If relevance accuracy for repository Q&A responses drops below 85%, we will improve RAG retrieval prompt engineering.

Statement iteration

Take each version through this checklist:

- [ ] Is this metric meaningful for all of our users?
    - [ ] How might this metric negatively impact some of our users?
- [ ] Is this what success means for our feature on day 1?
    - [ ] What about day 1,000?

Final version

(Metric 1) If JSON schema validation rate for AI-generated tickets drops below 100% or user satisfaction falls below 85%, we will investigate the root cause and adjust prompt engineering or validation logic.

(Metric 2) If relevance accuracy for repository Q&A responses drops below 85% or user satisfaction falls below 85%, we will improve RAG retrieval logic.

Schedule regular reviews

Once you've agreed upon your success metric(s), put time on the calendar to hold your team accountable to regularly evaluate whether your feature is progressing towards and meeting your defined criteria.

Success metric review

Date:
2/12/26

Attendees
: All