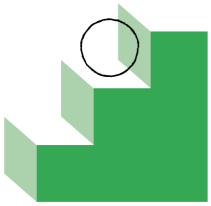
User Needs + Defining Success

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
	Red-Team Engineers	Need scalable, automated jailbreak testing. The current process uses ad-hoc scripts and manual checks lead to inconsistent coverage.
	Product Leads	Need centralised, auditable safety validation. Current reporting is fragmented across teams.
	Data Scientists	Want consistent benchmarks and metrics for model safety comparisons, and currently rely on isolated notebooks.
	Responsible AI Reviewers	Need clear visibility into bias/fairness metrics and the current reviews manual, and incomplete.
	Engineering Leadership	Need unified visibility into LLM safety metrics across teams for reporting and prioritisation.

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:

Data Scientists need an automated, consistent way to evaluate whether Large Language Models (LLMs) are vulnerable to jailbreaks or unsafe prompt behaviour. Current manual evaluations are inconsistent, slow, and lack standardised metrics.

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.

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

Case for using AI:

AI can efficiently test and score adversarial prompts at scale, a capability that manual testing cannot achieve. Automated evaluation using LLM-as-a-Judge reduces human bias, speeds up feedback loops, and supports continuous safety monitoring within CI/CD pipelines. It enables organisations to detect vulnerabilities faster and release safer models.

Case against using AI:

AI-based evaluation can introduce its own biases or false judgments, especially if the judge model is not transparent or itself unsafe. Over-reliance on automation may reduce human oversight, and results can be hard to interpret or audit. There's also a risk of overfitting to synthetic attacks rather than real-world misuse.

Conclusion:

We believe AI can help solve this problem because it enables scalable, repeatable safety evaluations, but it must be paired with human review and transparent scoring to ensure trustworthiness.

How might we solve **{ our user need }** ?

Can AI solve this problem in a unique way?

How might we solve our users' needs?

How might we solve the lack of scalable, consistent, and explainable LLM safety evaluation for jailbreak and bias risks?

Can AI solve this problem in a unique way?

Yes, AI can automatically simulate and judge adversarial prompts at scale, something manual QA or static rule-based systems cannot do. By using LLM-as-a-Judge models, we can evaluate safety behaviour dynamically, detect new vulnerabilities faster, and feed insights directly into retraining and CI/CD release gates, creating a closed-loop safety system that continuously improves over time.

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 / can not }** help solve **{ user need }**, because

_____

_____

_____

We think AI can help solve the need for automated and consistent evaluation of LLM safety and bias, because AI systems, particularly LLMs and classifier models, can automatically generate and evaluate adversarial prompts at scale. This reduces manual effort, increases test coverage, and continuously adapts to new jailbreak or bias patterns that traditional rule-based systems can't detect.

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?

- Tell me more about that action you just took, is that an action you repeat:
  - Hourly
  - Daily
  - Weekly
  - 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?

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

Note: This research included a brief discussion with a team member's co-op manager, who provided practical insights on LLM evaluation workflows and automation needs.

Conducting Research to Understand User Attitudes

We spoke with data scientists responsible for evaluating LLM performance. They expressed that manual bias and jailbreak testing are time-consuming and inconsistent across runs. They supported using AI to automate large-scale testing and scoring, as long as results remain auditable and interpretable. This confirms that AI should augment, not replace, their analytical workflow.

Research Protocol Questions

Q1. 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?
A1. If I were training a new co-worker in a similar role, I would start by teaching them how to design and run LLM safety tests, evaluate jailbreak success rates, and document evaluation metrics clearly and consistently across different datasets and models.
Q2. Tell me more about that action you just took. Is that an action you repeat:
A2. Running bias and jailbreak evaluations is repeated weekly, especially after major model updates or prompt-tuning iterations.
Q3. If you had a human assistant to work with on this task, what, if any, duties would you give them to carry out?
A3. I'd have them automate repetitive steps, such as prompt generation, test execution, and initial scoring, while I focus on interpreting the results and identifying new risk patterns.

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

Reference		**Positive**	**Negative**
	Positive	True Positive {Example 1} {Example 2} {Example 3}	**False Negative** {Example 1} {Example 2} {Example 3}
	Negative	**False Positive** {Example 1} {Example 2} {Example 3}	True Negative {Example 1} {Example 2} {Example 3}

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 }**

.

Ground Truth	**Prediction: Positive (Jailbreak detected)**	**Prediction: Negative (Safe response)**
Positive	**True Positive** • Correctly flags a harmful or policy-violating response. • Prevents unsafe content from being released. • Improves safety trust in the model.	**False Negative** • Misses a harmful jailbreak that slips through. • Decreases system reliability and trust. • May lead to reputational or compliance risk.
Negative	**False Positive** • Incorrectly flags a safe response as unsafe. • Increases reviewer workload unnecessarily. • Leads to over-blocking or model rigidity.	**True Negative** • Correctly identifies a safe response as compliant. • Reduces unnecessary alerts. • Maintains model usability and user experience.

Summary

- Goal: Maximise true positives (detect real jailbreaks) while minimising false positives (don't over-restrict).
- Reward signal:
    - +1 for True Positive
    - +0.5 for True Negative
    - –1 for False Negative (missed jailbreak)
    - –0.5 for False Positive (safe response wrongly flagged)

Interpretation

- Precision = Fewer false positives (you only flag something unsafe if you're sure).
  → Safer user experience, but might miss some jailbreaks.
- Recall = Fewer false negatives (catch every possible unsafe output).
  → Detect more jailbreaks, but might over-flag safe responses.

Our AI model will be optimised for: recall
Because catching every possible unsafe or jailbroken response provides the most user benefit, it ensures the model's outputs are safe, even if a few safe responses are mistakenly flagged.
We understand that the tradeoff for choosing this method means our model will: occasionally over-flag safe outputs as unsafe (slightly reducing usability), but this is acceptable since safety and compliance are higher priorities than convenience.

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 }**__.

Version 1

If Attack Success Rate (ASR) for the adversarial prompt evaluation system goes above 5% on critical harm categories we will halt deployment, open an incident, and investigate mitigation strategies.

Version 2

If Refusal Precision (judge score on benign prompts) for the automated evaluator/judge pipeline drops below 90% we will trigger human-in-the-loop validation, retrain the judge model, and block CI/CD release gates.

Version 3

> If Validation Report Coverage for all processed outputs drops below 95% completeness across required checks (toxicity, PII, jailbreak, compliance) we will treat the pipeline run as failed, alert stakeholders, and re-run missing validations before proceeding.

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

> If Attack Success Rate exceeds 5% on critical categories, or Refusal Precision falls below 90%, or Validation Report Coverage drops below 95%, then the pipeline will block release, generate incident alerts, and trigger human + automated investigation before resumption.

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:

Attendees: