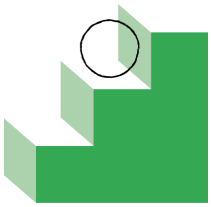
User Needs + Defining Success

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
Sep 2025	MLOps Course Requirements	Need for automated data schema generation and statistics validation in production ML pipelines. Manual schema tracking is error-prone and time-consuming.
Sep 2025	MIMIC-IV Documentation	Healthcare data requires strict patient-level splitting to prevent data leakage. Temporal splitting can leak future information into training data.
Sep 2025	Federated Learning Literature	Hospitals need to collaborate on ML without sharing patient data. Requires partitioned datasets with verified isolation.
Sep 2025	Data Engineering Best Practices	Manual data quality checks don't scale. Need automated anomaly detection with configurable thresholds.
Oct 2025	MLOps Reproducibility Studies	Data versioning is critical for ML reproducibility. DVC provides Git-like workflow for datasets.

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.

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.

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

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 / ~~cannot~~ }** help solve _____**{ user need }**_____, because

help solve automated anomaly detection and predictive quality validation, because

_____

_____

Current ML/AI capabilities that benefit HIMAS:

_____

1. Anomaly Detection: Machine learning can identify unusual patterns in data quality metrics that rule-based thresholds might miss (e.g., sudden changes in null rates, unexpected correlations between fields).

2. Adaptive Threshold Tuning: ML models can learn optimal quality thresholds from historical pipeline runs, reducing false positives while maintaining data quality.

3. Predictive Schema Evolution: AI can predict likely schema changes based on patterns in drift reports, helping teams proactively plan migrations.

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?

  First, I'd teach them about patient-level data splitting - it's critical to prevent data leakage. Then schema validation and quality checks. These are repetitive but absolutely critical for federated learning. Any mistakes here compromise the entire multi-hospital collaboration.

- 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?

Schema extraction and comparison, Statistics computation, Quality validation against thresholds

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

Reference	Prediction: Positive	Prediction: Negative
Positive	**True Positive** Pipeline continues {Example 1} Team alerted to review {Example 2} {Example 3} User Impact: Proactive issue prevention	**False Negative** Schema change missed {Example 1} Downstream pipeline breaks {Example 2} {Example 3} User Impact: HIGH - Pipeline failures
Negative	**False Positive** False drift alert {Example 1} Unnecessary investigation {Example 2} {Example 3} User Impact: Minor annoyance	**True Negative** No drift, no alert {Example 1} Pipeline continues normally {Example 2} {Example 3} User Impact: Ideal state

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 }**_____

because in healthcare federated learning, the cost of training models on corrupted data (False Negatives) is much higher than the cost of investigating false alarms (False Positives). A model trained on bad data could lead to incorrect clinical predictions, while a false alarm only requires manual review.

We understand that the tradeoff for choosing this method means our model will _____**{ user impact }**_____

_____

generate more false alarms, requiring data engineers to manually review and potentially adjust thresholds. However, this is acceptable because the alternative (missing data quality issues) could compromise patient safety in downstream clinical decision support applications.

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 pipeline success rate for HIMAS BigQuery demo DAG drops below 95% over a 7-day rolling window, we will:

Conduct root cause analysis of failures
Review and adjust quality validation thresholds if causing false failures
Fix identified infrastructure or code issues within 48 hours
Add additional error handling for identified failure modes

Version 2

If data quality validation false positive rate goes above 20% (validated through manual review), we will:

Analyze validation reports to identify overly strict thresholds
Tune thresholds based on observed data patterns
Implement adaptive threshold learning from historical data
Add context-aware validation rules for different data layers

Version 3

If schema validation coverage (percentage of tables with extracted schemas and computed statistics) drops below 100%, we will:

Investigate which tables are failing validation
Add error handling for edge cases
Implement retry logic for transient BigQuery API failures
Ensure all new tables are automatically included in validation

Statement iteration

Take each version through this checklist:

- ☐ Is this metric meaningful for all of our users?
  Yes, all users (data engineers, data scientists, researchers) depend on reliable pipeline execution
  - ☐ How might this metric negatively impact some of our users?
    Might prioritize stability over new features. Could delay experiments if we pause for investigations
- ☐ Is this what success means for our feature on day 1?
  Yes, reliability is critical from day 1
  - ☐ What about day 1,000?
    Still important, but may need to balance with other metrics (performance, features)

Final version

If pipeline success rate for HIMAS BigQuery demo DAG drops below 95% over a 7-day rolling window, we will conduct immediate root cause analysis and implement fixes within 48 hours.
Data Quality: Validation false positive rate < 20% (tuned over first month)
Coverage: 100% of tables have schemas extracted and statistics computed
Performance: Pipeline execution time < 10 minutes for demo dataset
Reproducibility: 100% of data outputs versioned with DVC
Alerting: Email alerts delivered within 5 minutes of pipeline events

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: **28th Oct 2025**

Attendees: **Yash Khare, Mohan Bhosale, Manjusha Motamarry, Raunaksingh Khalsa, and Margi Shah**