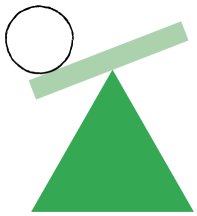
Errors + Grace Failure

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. Error audit [~1 hour]

Collect canonical error examples to define existing and potential errors and solutions.

2. Quality assurance [~30 minutes]

Prioritize how you'll test and monitor errors and reporting so you can hear from your users early and often.

1. Error audit

As a team, brainstorm what kinds of errors users could encounter. If your team has a working prototype of your feature, try to add current examples.

Use the template below to start collecting error examples so your team has a shared understanding about the different error types and solutions your model could produce.

Error	User Stakes	Error Type	Example / Description
1. Inaccurate inventory prediction	🔶 High	🧩 *System limitation*	The AI predicts that milk will last 3 more days, but it actually runs out the next day. The user's meal plan fails.
2. Recipe includes missing or expired ingredients	🔶 Medium	🧩 *Context error*	The system suggests a recipe using "tomatoes," but the user has none left. The user feels the AI "doesn't understand them."
3. Poor allergy or dietary filtering	🔴 High	🧩 *Background error*	The system suggests a meal that contains peanuts despite allergy data being set.
4. Duplicate or inconsistent alerts	🟡 Low	🧩 *System hierarchy error*	Two modules (forecasting + delivery) trigger similar notifications, confusing the user.
5. Long response time or timeout	🟢 Low	🧩 *System limitation*	The model takes too long to recommend a recipe or fails to generate results.

Error sources

Take each error identified above through these questions to determine the source of the error:

Input error signals

- [ ] Did the user anticipate the auto-correction of their input into an AI system?
- [ ] Was the user's habituation interrupted?
- [ ] Did the model improperly weigh a user action or other signal? If yes, likely a context error.

Relevance error signals

- [ ] Is the model lacking available data or requirements for prediction accuracy?
- [ ] Is the model receiving unstable or noisy data?
- [ ] Is the system output presented to users in a way that isn't relevant to the user's needs?

System hierarchy error

- [ ] Is your user connecting your product to another system, and it isn't clear which system is in charge?
- [ ] Are there multiple systems monitoring a single (or similar) output and an event causes simultaneous alerts? Signal crashes increase the user's mental load because they have to parse multiple signals to figure out what happened and what to do next.

Failure state

- [ ] Is your feature unusable as the result of multiple errors?

Error Source Type	Example / Signal in PantryPilot	Explanation
Input Error Signals	User misspells an ingredient name ("strawbery" → "strawberry"), or the system auto-corrects it incorrectly.	The user didn't anticipate that AI would modify their input. This can cause frustration or wrong ingredient detection.
Input Error Signals (Context)	AI misinterprets a user's intent (e.g., assumes "I'm out of eggs" means "add eggs to shopping list").	The model improperly weighs user actions — **context error**.

Error Source Type	Example / Signal in PantryPilot	Explanation
Relevance Error Signals	The model hasn't synced with the latest grocery data or user just restocked items.	The model is missing updated data, leading to inaccurate or irrelevant predictions.
Relevance Error Signals (Presentation)	The system displays a recipe that doesn't match the user's dietary preferences.	The output is technically valid but **not relevant** to the user's actual needs.
System Hierarchy Error	Forecasting module and grocery delivery module both send alerts for the same item.	Conflicting systems cause duplicate alerts, increasing the user's cognitive load.
Failure State	The app becomes unresponsive after multiple data sync errors.	Several small issues combine to make the feature unusable — the "graceful failure" state is lost.

Error resolution

Once you have identified the source or sources of the error, complete the sections below for each of the errors in the template with your team's plan for improving / reducing the identified error: Create as many copies as you need to cover all your identified errors.

Error rationale	**Solution type**
Why the user thinks this is an error: The user expects the system to know what's actually available in their pantry. When a suggested recipe includes an ingredient they've already run out of, it feels like the system "isn't paying attention."	☑ Feedback ☑ User control ☐ Other:

Error resolution

User path: User sees a recipe using an unavailable ingredient → clicks "Report missing ingredient" → system acknowledges the issue and offers alternative recipes based on confirmed items → user continues and completes the cooking task.
Opportunity for model improvement:

Opportunity for model improvement:
User feedback ("missing ingredient" flag) is logged for **inventory synchronization tuning**. Model retraining prioritizes pantry validation before recommending recipes, improving accuracy over time.

2. Quality assurance

Getting your feature into users' hands is essential for identifying errors that your team, as expert users, may never encounter. Meet as a team to prioritize how you want to monitor errors reported by users so that your model is being tested and criticized by your users early and often.

As you have this discussion, consider all potential sources of error reporting:

- 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, etc.)

QA template

<table>
<tr><td>Goal: Ensure PantryPilot's AI continuously improves in accuracy, personalization, and reliability through real-world user feedback and error tracking.
Focus on early detection of prediction inaccuracies, allergen filtering errors, and user dissatisfaction trends.</td><td rowspan="3">Review frequency

☐ Daily

☑ Weekly

☐ Monthly

☐ Other:</td></tr>
<tr><td>Method

• In-product metrics: Track false prediction rate, recipe satisfaction (thumbs-up/down), and feedback flags (e.g., "missing ingredient").

• Customer feedback: Analyze app store reviews and support tickets related to incorrect alerts or recipe mismatches.

• User research: Conduct short monthly interviews to validate feature trust and discover unseen errors.-</td></tr>
<tr><td>Start date: November 1, 2025

Review / End date: Ongoing</td></tr>
</table>