Errors + Grace Failure

Team 13: Intelligent Onboarding Assistant

Lakshmi Vandhanie Ganesh, Zankhana Pratik Mehta, Mithun Dineshkumar, Saran Jagadeesan Uma, Akshaj Nevgi

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	Users
1. Windows user cannot access the openAI-whisper due to missing dependencies (e.g., ffmpeg). We use this for getting meeting recording transcriptions to extract our data.	Windows users trying to use openAI-whisper.
Error type ☒ **System limitation** - Your system can't provide the right answer, or any answer at all, due to inherent limitations to the system. ☐ **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. ☐ **Background** - Situations in which the system isn't working correctly, but neither the user nor the system register an error.	**User stakes** ☒ low ☐ high

Error	Users
2. Airflow 2.10.2 installation failed because it is not compatible with Python 3.12.	Developers or data engineers using Airflow 2.10.2 with Python 3.12.
Error type ☒ **System limitation** - Your system can't provide the right answer, or any answer at all, due to inherent limitations to the system. ☐ **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. ☐ **Background** - Situations in which the system isn't working correctly, but neither the user nor the system register an error.	**User stakes** ☒ low ☐ high

Error sources

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

Error 1: Windows user cannot access openAI-whisper (resolved by installing ffmpeg)

Input error signals

- Did the user anticipate the auto-correction of their input into an AI system?

The user did not anticipate that the system would fail due to missing dependencies (ffmpeg).

- Was the user's habituation interrupted?

The user's habituation was interrupted since normally, installing a Python library should work without manual OS-level setup.

- Did the model improperly weigh a user action or other signal? If yes, likely a context error.

No improper weighing of user input — the issue stemmed from missing system-level dependencies, not model interpretation.

Relevance error signals

- Is the model lacking available data or requirements for prediction accuracy?

The system was lacking required external components (ffmpeg), which are necessary for whisper's audio processing.

- Is the model receiving unstable or noisy data?

Data itself was stable, but the model couldn't process it due to missing backend support.

- Is the system output presented to users in a way that isn't relevant to the user's needs?

Output was irrelevant (failure message) because the dependency issue wasn't clearly communicated to the user, leading to confusion about the cause.

System hierarchy error

- Is your user connecting your product to another system, and it isn't clear which system is in charge?

The user was connecting Python (and whisper) to OS-level binaries (ffmpeg), and it wasn't clear that ffmpeg was a prerequisite external system.

- 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.

No conflicting systems were running, but there was a dependency hierarchy gap — the model depended on a system the user didn't know to configure.

Failure state

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

The feature was temporarily unusable until ffmpeg was manually installed — once installed, the system functioned normally.

Error 2: Airflow 2.10.2 incompatible with Python 3.12

Input error signals

- The user did not expect Airflow installation to fail based on the Python version — a common habituation since most modern packages support 3.12.
- No model misinterpretation — rather, a version mismatch prevented successful installation.

Relevance error signals

- The system (Airflow) lacked compatibility data for Python 3.12 at the time, causing unmet dependency and build errors.
- Input (Python environment) was stable but not relevant to Airflow's requirements, hence the mismatch.

- Output errors (installation failure logs) were technically correct but not immediately helpful for users unfamiliar with version dependencies.

System hierarchy error

- The user was connecting Airflow (application-level system) to Python (runtime environment), and the hierarchical dependency between them wasn't transparent — the version requirement wasn't enforced early.

- No multiple systems in conflict, but unclear hierarchy between Python versioning and Airflow support scope caused confusion.

Failure state

- The feature (Airflow installation/execution) was completely unusable with Python 3.12, but worked once Python was downgraded to a supported version (3.10 or 3.11).

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 1: Windows user cannot access openAI-whisper (missing ffmpeg)

Error rationale	**Solution type**
Why the user thinks this is an error:The user perceives this as an error because the installation of openAI-whisper fails without a clear message explaining the dependency on ffmpeg. The user expects the package to install and run directly through pip without needing manual OS-level configuration.	☒ Feedback ☒ User control ☐ Other:

Error resolution

User path:
User installs openAI-whisper → encounters import or runtime error → searches for the issue → identifies missing ffmpeg dependency → installs ffmpeg manually → re-runs whisper successfully.

Examples: User sees errors, gives feedback, completes task; User sees error, takes over control, completes task

Opportunity for model improvement:

Improve installation feedback by automatically checking for ffmpeg and displaying a clear installation instruction (e.g., "ffmpeg not found — please install it to enable audio processing").
Example: Add a pre-install dependency validation step or auto-install ffmpeg where possible.

Example: User's feedback logged for model tuning

Error 2: Airflow 2.10.2 incompatible with Python 3.12

Error rationale	Solution type
The user considers this an error because Airflow installation fails even though Python 3.12 is the latest version. The error logs don't clearly state version incompatibility, leading users to believe something else went wrong.	☒ Feedback ☐ User control ☒ Other: Documentation clarity

Error resolution

User path:

User installs Airflow 2.10.2 on Python 3.12 → installation fails due to dependency conflicts → user researches online → discovers version limitation (supports up to 3.11) → downgrades Python → successfully installs Airflow.

Examples: User sees errors, gives feedback, completes task; User sees error, takes over control, completes task

Opportunity for model improvement:

Enhance installation scripts and documentation to explicitly flag version incompatibility upfront.

Example: Installer detects unsupported Python version and outputs a clear message — *"Airflow 2.10.2 supports Python 3.8–3.11. Please use one of these versions."*

Example: User's feedback logged for model tuning

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

Goal	**Review frequency**
To ensure seamless compatibility and installation experience for users by proactively identifying and addressing system dependency and version compatibility issues in openAI-whisper and Apache Airflow. The objective is to minimize user frustration, reduce support tickets, and enhance documentation and feedback clarity.	☐ Daily ☐ Weekly ☐ Monthly ☐ Other:
Method Collect user feedback through GitHub issue tracking, developer community forums, and customer service reports. Conduct automated dependency checks during installation to flag missing or unsupported components (e.g., ffmpeg, Python versions). Implement in-product error monitoring to capture recurring setup or environment issues. Perform periodic regression testing on supported environments to confirm cross-version stability.	

Review in-product survey data and user research findings to identify pain points related to setup and configuration.	
Start date: November 2025 Review / End date: Ongoing review with quarterly evaluations	