

Heuristic Evaluation Template

Overview & Using this Template

This template is based on [Jakob Nielsen's 10 general principles for interaction design](#). These principles, also called heuristics, define best practices for implementing good user experiences. The [STRUDEL](#) project expanded the descriptions of these principles so that they more clearly apply to scientific software products and systems.

Conducting a heuristic evaluation allows you to systematically identify usability issues in your product. By reviewing your own tool in light of Nielsen's principles, you may notice details that went unseen previously or violations of best practices you were unaware of. A heuristic evaluation like this can be applied to any user interface, meaning any digital product that a user interacts with: a website, an app, a command line tool, a software library, etc.

To use this template, first review the heuristics and their prompt questions. Then review a workflow or feature from your product carefully, noting issues in the category they fit with best. Enter one issue per cell and describe a corresponding fix in the Recommendations column. You may choose to color code issues so that they indicate their severity. Include screenshots and examples as needed for clarity.

Yellow means low severity	Orange means moderate severity	Red means high severity
---------------------------	--------------------------------	-------------------------

Example: Evaluating a Data Catalog

Heuristic #4 - Consistency and Standards

Issues	Recommendations
Empty fields are denoted in multiple ways and with different text formatting, for example: "No data available." "No data found" "- " "--" "No Tags added" "No Description" "No description"	Apply a consistent scheme for indicating a lack of information, including punctuation and capitalization. Ensure that it is clear whether the data cannot be retrieved or if it was never provided.

Example: Evaluating an AI agent with chat interface

Heuristic #9 - Help Users Recognize, Diagnose, and Recover from Errors

Issues	Recommendations
<p>Queries are not named/IDed but previous queries are discussed by the agent, making it ambiguous what is the object of discussion.</p> <p>For example: The agent says, "Let me check what went wrong with the query," and "Let me wait and check the status," but doesn't say which query is being checked.</p>	<p>Provide IDs for queries so that they can be explicitly referenced.</p>

FAQ

Isn't this subjective?

Yes, but heuristics offer rules of thumb that are known to improve usability of a given tool. If you see an instance where your tool seems to violate one of these heuristics, it is likely that other people will too. If you don't catch all the issues, what you do find can still meaningfully improve usability.

I'm not a UX expert, can I still do this?

Yes, there is no special language you need to know or outside knowledge you need to have that is not included in this template. Just describe the issues you see in your own words and provide examples to help ensure any collaborators understand as well. If you need more support or are uncertain about what recommendations to offer, reach out to the STRUDEL team at strudel@lbl.gov.

I thought my issue belonged in one category but then later I thought it belonged to another. What do I do?

Many times an issue will span multiple heuristics. Choose the option that best summarizes the problem you perceive.

Heuristic Evaluation of _____

Evaluator:

Date:

Product & version:

Task:

Summary of Evaluation

Add a high level overview here of the major issues you have identified so that others can get the gist of your findings without reading the full report. You can combine multiple issues (even those logged under different heuristics) into one theme. Include your recommendations to resolve these issues as well.

1 - Visibility of System's Current Status

The design of your product should always keep users informed about what is going on, through appropriate feedback within a reasonable amount of time.

- Does the design clearly communicate its state, including during “processing” or “thinking” periods or when a timeout error occurs?
- Is feedback presented quickly after user actions?
- When an action is cancelled, is it clear what the new system status is?

Issues	Recommendations

2 - Match Between System and the Real World

The system should reflect norms that its users are already familiar with. The design should speak the users' language. Use words, phrases, and concepts familiar to a new user, rather than internal jargon. Follow real-world conventions, making information appear in a natural and logical order.

- Will users be familiar with the terminology used in the design or commands?
- Do the design's controls follow real-world conventions and workflows?
- Do commands follow similar patterns to those used by competing or related products?

- Are the system outputs in a form the user can directly leverage for subsequent tasks?

Issues	Recommendations

3 - User Control and Freedom

Users should be able to easily perform actions and undo them. Users often perform actions by mistake. They need a clearly marked "emergency exit" to leave the unwanted action without having to go through an extended process.

- Are available options obvious to users?
- Does the design allow users to go back a step in the process?
- Are exits easily discoverable to new users?
- Can users easily cancel an action?
- Is Undo and Redo supported?

Issues	Recommendations

4 - Consistency and Standards

Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform, industry, and disciplinary conventions. Place information where users have been trained to expect it.

- Does the design follow common conventions, including information organization norms?
- Is care taken to avoid calling the same thing by multiple names?
- Are visual treatments used consistently throughout the design?
- Are your docs organized into standard sections, including information on install and dependencies up front?

Issues	Recommendations

5 - Error Prevention

Good error messages are important, but the best designs carefully prevent problems from occurring in the first place. Either eliminate error-prone conditions, or check for them and present users with a confirmation option before they commit to the action.

- Does the design prevent slips by using helpful constraints?
- Does the design warn users or introduce useful friction before they perform risky actions?
- Are appropriate defaults selected?
- Are there appropriate guardrails in place to ensure a user only works “dangerously” when it is safe to do so?
- Is input validation used when possible?

Issues	Recommendations

6 - Recognition Rather Than Recall

Minimize the user's memory load by making elements, actions, and options visible. The user should not have to remember information from one part of the interface to another. Information required to use the design (e.g. field labels or menu items) should be visible or easily retrievable when needed.

- Does the design keep important information visible, so that users do not have to memorize it?
- Does the design offer help in-context?
- Is it clear to a user where they are in the system or workflow?
- Are interactive commands used to help new users of CLI tools gain familiarity with the system and available flags?
- Are steps requiring copying and pasting from one place to another minimized?

Issues	Recommendations

7 - Flexibility and Efficiency of Use

Shortcuts — hidden from novice users — may speed up the interaction for the expert user such that the design can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.

- Does the design provide accelerators like keyboard shortcuts and touch gestures?
- Is content and functionality personalized or customized for individual users?
- Are users able to automate their workflow?

Issues	Recommendations

8 - Aesthetic and Minimalist Design

Interfaces should not contain information that is irrelevant or rarely needed. Every extra unit of information in an interface competes with the relevant units of information and diminishes their relative visibility.

- Is the visual design and content focused on the essentials?
- Have all distracting, unnecessary elements been removed?
- Are symbols and emojis used to convey meaning and not just decorate?

Issues	Recommendations

9 - Help Users Recognize, Diagnose, and Recover from Errors

Error messages should be expressed in plain language, precisely indicate the problem, and constructively suggest a solution.

- Does the design use traditional error message visuals, like bold, red text?
- Does the design offer a solution that solves the error immediately?
- Is there a way to monitor ongoing processes?
- Are log files available for reviewing issues?
- Is it clear when an issue is related to a third-party?

Issues	Recommendations

10 - Help and Documentation

Documentation helps users understand how to complete their tasks and reproduce workflows later; it is also useful for AI agents seeking information about your product.

- Is help documentation easy to search?
- Is help provided in context right at the moment when the user requires it?
- Are there examples of the most popular workflows?
- Are appropriate markdown files provided to LLMs and agents?

Issues	Recommendations