



UX Audit Checklist

Find the friction. Understand the problem.
Prioritize what matters.



UX Crumbs Framework

Goal → Journey → Evidence Audit → Prioritize → Improve

A practical checklist for reviewing a website, app, or product experience before you redesign it.

Use this resource to:

- Spot usability and interaction problems.
- Identify friction across important user journeys.
- Check content, forms, accessibility, and responsive behavior.
- Turn observations into actionable recommendations.
- Prioritize issues instead of creating a giant list of complaints.

How to Use This UX Audit

An audit is a structured review of an existing experience. It helps identify potential problems; it does not replace observing real users.

Important: A heuristic or expert audit is an evaluation method, not proof that users experience a problem. Validate high-impact findings with analytics, support data, usability testing, or other appropriate evidence.

STEP 1 — Define the audit

- What product or experience are you reviewing?
- Who is the primary user?
- What user goal or business goal matters most?
- Which journey or task is in scope?
- What device, platform, or environment is in scope?
- What evidence already exists?

STEP 2 — Walk the experience

- Complete the journey as a user would.
- Record what happens, not just what you dislike.
- Capture screenshots or links for findings.
- Note uncertainty, friction, errors, waiting, repetition, and workarounds.
- Do not redesign while auditing.

STEP 3 — Evaluate systematically

- Check visibility and system feedback.
- Check language, labels, and mental models.
- Check navigation and information architecture.
- Check forms, errors, states, and recovery.
- Check accessibility and responsive behavior.
- Check trust, clarity, and unnecessary effort.

STEP 4 — Prioritize

- How severe is the problem?
- How often is the user likely to encounter it?
- How important is the affected task?
- How strong is the evidence?
- What is the likely effort to fix it?

Audit mindset: Observe → Describe → Explain → Evidence → Prioritize.
Avoid jumping directly from “I don't like this” to “Change it.”

Audit Setup Sheet

Fill this in before you start.

Audit subject _____

Primary user _____

Primary goal _____

Journey / task _____

Platform / device _____

Audit date _____

Evidence available Analytics / research / support /
reviews / none

Choose 1–3 critical journeys

- | | |
|---|---|
| ☐ Discover / browse | ☐ Purchase / book |
| ☐ Sign up / log in | ☐ Complete a form |
| ☐ Search | ☐ Manage account |
| ☐ Compare / evaluate | ☐ Get help / recover |

Audit notes

☐	_____
☐	_____
☐	_____
☐	_____

Tip: Start with the journey that matters most, not the screen that looks most interesting.

Visibility of System Status

Use this heuristic as a prompt for investigation, not as a rigid scoring system.

What to look for: Users should be able to tell what is happening through appropriate, timely feedback.

- Does the interface show what is happening after an action?
- Are loading, saving, processing, success, and failure states clear?
- Can users tell whether an action worked?
- If something takes time, is progress or status communicated?
- Does the system acknowledge important actions?

Finding notes

- Where did you observe the issue? _____
- What is the user trying to do? _____
- What happens instead? _____
- Evidence / screenshot / data: _____
- Suggested direction: _____

Match Between System and the Real World

Use this heuristic as a prompt for investigation, not as a rigid scoring system.

What to look for: Use concepts, language, and sequences that make sense to users.

- Do labels use familiar user language rather than internal jargon?
- Are actions presented in an order users expect?
- Do icons and terminology match common conventions?
- Does the experience reflect the user's mental model?
- Are dates, units, currencies, and formats appropriate to context?

Finding notes

- Where did you observe the issue? _____
- What is the user trying to do? _____
- What happens instead? _____
- Evidence / screenshot / data: _____
- Suggested direction: _____

User Control and Freedom

Use this heuristic as a prompt for investigation, not as a rigid scoring system.

What to look for: Users need clear ways to undo, cancel, go back, or recover.

- Can users undo or reverse important actions?
- Can users cancel processes before committing?
- Is there a clear way to go back without losing work?
- Are destructive actions reversible where practical?
- Can users leave a flow without feeling trapped?

Finding notes

- Where did you observe the issue? _____
- What is the user trying to do? _____
- What happens instead? _____
- Evidence / screenshot / data: _____
- Suggested direction: _____

Consistency and Standards

Use this heuristic as a prompt for investigation, not as a rigid scoring system.

What to look for: Similar things should behave and look similar, while established conventions are respected.

- Are similar controls labeled and styled consistently?
- Do the same interactions behave the same way?
- Are platform and industry conventions followed?
- Are terminology and capitalization consistent?
- Do components maintain consistent states and behavior?

Finding notes

- Where did you observe the issue? _____
- What is the user trying to do? _____
- What happens instead? _____
- Evidence / screenshot / data: _____
- Suggested direction: _____

Error Prevention

Use this heuristic as a prompt for investigation, not as a rigid scoring system.

What to look for: Prevent problems where possible instead of relying only on error messages.

- Are risky actions clearly identified before commitment?
- Does the interface prevent invalid input when practical?
- Are constraints communicated before users make an error?
- Are defaults sensible and safe?
- Does the flow reduce avoidable mistakes and accidental actions?

Finding notes

- Where did you observe the issue? _____
- What is the user trying to do? _____
- What happens instead? _____
- Evidence / screenshot / data: _____
- Suggested direction: _____

Recognition Rather Than Recall

Use this heuristic as a prompt for investigation, not as a rigid scoring system.

What to look for: Reduce memory load by making options, information, and next steps visible or easily retrievable.

- Can users see relevant options instead of remembering them?
- Are labels and controls understandable without recalling previous screens?
- Is important context preserved across steps?
- Can users recognize what to do next?
- Are recently entered or selected values available when useful?

Finding notes

- Where did you observe the issue? _____
- What is the user trying to do? _____
- What happens instead? _____
- Evidence / screenshot / data: _____
- Suggested direction: _____

Flexibility and Efficiency of Use

Use this heuristic as a prompt for investigation, not as a rigid scoring system.

What to look for: Support new users and experienced users where appropriate.

- Can frequent users move faster?
- Are shortcuts or efficient paths available where useful?
- Can users customize or save preferences when valuable?
- Does the flow avoid unnecessary repetition?
- Are advanced options available without overwhelming beginners?

Finding notes

- Where did you observe the issue? _____
- What is the user trying to do? _____
- What happens instead? _____
- Evidence / screenshot / data: _____
- Suggested direction: _____

Aesthetic and Minimalist Design

Use this heuristic as a prompt for investigation, not as a rigid scoring system.

What to look for: Avoid unnecessary information that competes with what users need to accomplish.

- Is every major element serving a clear purpose?
- Is the primary action visually clear?
- Are secondary elements appropriately de-emphasized?
- Is important content easy to scan?
- Could removing something make the task clearer?

Finding notes

- Where did you observe the issue? _____
- What is the user trying to do? _____
- What happens instead? _____
- Evidence / screenshot / data: _____
- Suggested direction: _____

Help Users Recognize, Diagnose, and Recover from Errors

Use this heuristic as a prompt for investigation, not as a rigid scoring system.

What to look for: Errors should be understandable, specific, and actionable.

- Does the message explain what went wrong?
- Does it tell the user how to recover?
- Is the problematic field or action clearly identified?
- Is user-entered information preserved when possible?
- Are technical details translated into useful guidance?

Finding notes

- Where did you observe the issue? _____
- What is the user trying to do? _____
- What happens instead? _____
- Evidence / screenshot / data: _____
- Suggested direction: _____

Help and Documentation

Use this heuristic as a prompt for investigation, not as a rigid scoring system.

What to look for: When help is necessary, it should be easy to find, focused on the task, and actionable.

- Can users find help at the point of need?
- Does help use user language?
- Are instructions specific rather than generic?
- Does documentation explain how to complete the task?
- Can users recover without leaving the context unnecessarily?

Finding notes

- Where did you observe the issue? _____
- What is the user trying to do? _____
- What happens instead? _____
- Evidence / screenshot / data: _____
- Suggested direction: _____

Visual & Interaction Audit

A good-looking interface can still be difficult to use. Audit hierarchy, interaction, and states separately.

- Can I identify the primary action quickly?
- Is visual hierarchy aligned with task priority?
- Are headings and sections easy to scan?
- Are interactive elements visually distinguishable?
- Do buttons, links, fields, and controls look actionable?
- Are hover, focus, pressed, selected, disabled, loading, success, and error states designed?
- Is spacing helping users understand grouping and relationships?
- Are icons understandable and paired with labels where needed?
- Are animations meaningful rather than distracting?
- Does the interface avoid visual noise that competes with the task?

3-second test

Without explaining the interface, look at the screen for roughly three seconds. Then ask:

- What is this page / screen for?
- Who is it for?
- What can I do here?
- What should I do next?
- What information is most important?

If users cannot answer these questions: investigate hierarchy, labeling, navigation, content, and task clarity before simply changing colors or adding decoration.

Content, Forms & Error Audit

Many UX failures happen in language and task completion—not just visual design.

- Do headings describe what the section actually contains?
- Are buttons labeled with the outcome or action users expect?
- Is microcopy concise and understandable?
- Are required fields clearly identified?
- Are labels persistent and associated with their inputs?
- Are examples and formats shown before users make mistakes?
- Are input types appropriate for the data?
- Are defaults helpful rather than presumptive?
- Do errors appear close to the problem?
- Do error messages explain what happened and how to fix it?
- Is entered data preserved after validation errors?
- Are success and completion states clear?

Copy test

For each important piece of UI text, ask: Can the user predict what will happen before they act?

- “Submit” → describe the actual outcome.
- “Continue” → clarify where the user is going when context allows.
- “Error” → explain the problem and recovery.
- “Learn more” → identify what users will learn.
- “Confirm” → clarify what is being confirmed.

Accessibility & Inclusive UX Audit

Use this as a design-level screen. It is not a complete WCAG conformance audit.

- Is meaningful information conveyed without relying on color alone?
- Is text sufficiently distinguishable from its background?
- Are keyboard users able to reach and operate important controls?
- Is keyboard focus visible?
- Is focus order logical?
- Do form controls have clear, persistent labels?
- Do images and meaningful icons have appropriate text alternatives?
- Are headings and landmarks structured logically?
- Can users understand links and controls from their labels?
- Are error messages and recovery instructions accessible?
- Can content reflow at smaller widths and increased text sizes?
- Are motion, auto-playing media, or time limits handled appropriately?
- Can users complete important tasks without depending on a single input method?

Accessibility principle: Perceivable → Operable → Understandable → Robust. WCAG 2.2 organizes its guidance around these four principles.

Mobile & Responsive Audit

Review the experience at the sizes and conditions where people actually use it.

- Does the layout adapt without losing important content?
- Are primary actions reachable and easy to operate?
- Are touch targets appropriately sized and separated?
- Does content reflow without awkward horizontal scrolling?
- Are forms comfortable to complete on small screens?
- Does the keyboard avoid covering important fields or actions?
- Are menus and navigation understandable on mobile?
- Do images and media adapt without slowing or breaking the experience?
- Are orientation changes handled reasonably?
- Does mobile preserve the user's goal rather than simply shrinking desktop UI?

Context test

- One-handed use
- Poor network
- Bright sunlight
- Low attention
- Interruptions
- Small screen
- Keyboard open
- Returning later

Ask: “What changes when the user is not sitting calmly at a large screen with perfect connectivity?”

Trust, Performance & Recovery Audit

Users evaluate whether an experience feels reliable as they move through it.

- Does the product communicate clearly when something is loading?
- Are slow operations acknowledged rather than appearing frozen?
- Are destructive or high-consequence actions clearly signposted?
- Are payment, privacy, security, and permission decisions explained when relevant?
- Does the interface avoid surprising behavior?
- Can users recover from failed requests?
- Is retry behavior clear?
- Does the product avoid losing work after interruptions or errors where feasible?
- Are confirmations proportionate to the risk of the action?
- Does the experience create confidence at critical moments?

Recovery walkthrough

- Detect → Does the user know something happened?
- Explain → Do they understand the problem?
- Protect → Is their work/data preserved?
- Recover → Is there a clear next step?
- Resume → Can they continue without unnecessary repetition?

UX Finding Card

Use one card per issue. This keeps an audit evidence-based and actionable.

Finding #	-----	Date	-----
Screen / URL	-----	Journey	-----
Problem	What did you observe-----		
User goal	What was the user trying to accomplish-----		
Heuristic / Principle	-----	Evidence	Research / analytics / observation / none
Impact	Low / Medium / High / Critical	Frequency	Rare / Sometimes / Often / Unknown
Recommendation	What direction could reduce the problem?-----		

Write findings as: OBSERVATION → USER IMPACT → EVIDENCE → RECOMMENDATION

Example: “The checkout page hides the delivery date until after address entry. This makes it harder to compare options before committing. Validate with checkout behavior and user testing. Consider surfacing the key delivery expectation earlier.”

Prioritize the Findings

Do not treat every issue as equally important.

Dimension	Question	Score
Impact	How much does this interfere with an important user goal?	1 2 3 4 5
Frequency	How often are users likely to encounter it?	1 2 3 4 5
Evidence	How strong is the evidence that this is a real problem?	1 2 3 4 5
Effort	How difficult is the proposed improvement? (5 = easiest)	1 2 3 4 5

Simple prioritization formula

$PRIORITY = IMPACT \times FREQUENCY \times EVIDENCE \times EFFORT$

This is a practical workshop heuristic, not a universal industry standard. Adjust the dimensions to fit your product and team.

Prioritization questions

- Does this affect a critical journey?
- Could this cause abandonment, failure, confusion, or loss of trust?
- Is the issue supported by more than personal preference?
- Can we fix the root cause rather than only the visible symptom?
- Would a small change meaningfully reduce friction?
- Does the recommendation introduce a new usability or accessibility problem?

Turn the Audit Into an Action Plan

An audit becomes useful when findings lead to decisions, validation, and improvement.

Priority	Finding	Next action	Owner
1			
2			
3			
4			
5			
6			

Validation plan

- Which findings need usability testing?
- Which findings can be checked with analytics?
- Which findings need accessibility testing?
- Which findings need product / technical investigation?
- What will success look like after the change?

Before → After

■ Problem we saw:

■ Change we made:

■ What we expect to improve:

■ How we will know:

■ What we learned:

The goal is not a perfect audit.

The goal is a clearer understanding of where the experience creates unnecessary effort—and what to do next.

60-Second UX Audit Cheat Sheet

Use this when you need a fast first pass.

THE UX CRUMBS AUDIT LOOP : GOAL → JOURNEY → OBSERVE → CHECK → CAPTURE → PRIORITIZE → VALIDATE → IMPROVE

Ask these 20 questions

- Is the user's goal obvious?
- Is the next action obvious?
- Does the interface show what is happening?
- Does the language match the user's mental model?
- Can users undo, cancel, or recover?
- Are patterns and terminology consistent?
- Does the design prevent common errors?
- Can users recognize choices without remembering them?
- Is the experience efficient for repeat users?
- Is unnecessary information competing with the task?
- Are errors clear and actionable?
- Is help available when needed?
- Can users identify the primary action?
- Are states and feedback designed?
- Can the experience be used with different input methods?
- Does content reflow appropriately?
- Can users recover from interruptions or failures?
- Does the experience build trust at critical moments?
- Which issue has the greatest user impact?
- What should we validate next?

REMEMBER: An audit finds potential problems. Research and testing help you understand whether they actually affect users—and why.