User interfaces, useability and annotated mock-ups

User interface and user experience

A **user interface (UI)** is the point where a person interacts with a digital solution: screens, controls, labels, messages, sound and other inputs or outputs. **User experience (UX)** is the broader experience produced while using the solution, including how understandable, efficient, useful and reassuring it feels.

> **Key idea:** An interface is not “good” in isolation. Its quality depends on the intended users, their context, the task and the success criteria.

A visually polished screen may still provide poor UX if a user cannot find the required function, understand a label, correct an error or operate a control.

Interface or experience?

A bus app contains a search field and a *Find route* button. A commuter completes the route search quickly and feels confident that the result is current.

Recognise the UI elements and the UX evidence.

Check your answer

The search field and button are UI elements. Completing the task quickly and feeling confident about the result are aspects of the user's experience.

The five useability principles

Useability describes how well users can interact with a solution. QCAA identifies five principles.

Principle	Question to ask	Possible evidence
Accessibility	Can people with diverse abilities, needs and technologies perceive and operate it?	Keyboard access, readable contrast, text alternatives, clear language.
Effectiveness	Can users complete the intended task accurately and successfully?	Correct output, clear task path, useful search and filters.
Safety	Does the interface prevent harmful errors and support recovery?	Validation, confirmation, undo, warnings, protected destructive actions.
Utility	Does the solution provide the functions users actually need?	Required features exist and suit the context.
Learnability	How easily can a new or returning user understand what to do?	Familiar controls, consistent layout, descriptive labels and feedback.

> **Avoid one-word answers:** naming “effectiveness” does not evaluate anything. Identify interface evidence and explain its effect on a user completing a task.

Classify the main principle

Analyse each situation and identify the main useability principle affected.

1. A booking form has pale grey text on a white background.
2. A user can permanently delete all records with one accidental tap.
3. A route app has no way to request wheelchair-accessible travel.
4. A first-time user cannot tell whether a magnifying-glass icon searches or zooms.
5. A worker must convert coordinates in another app before locating each job.

Check your answer

1. **Accessibility:** low contrast makes text difficult to perceive.
2. **Safety:** the interface does not prevent or support recovery from a harmful error.

3. **Utility:** a function required by part of the intended user group is absent. Accessibility may also be relevant.
4. **Learnability:** the control's meaning is unclear to a new user.
5. **Effectiveness:** the interface delays the task and makes successful location unnecessarily difficult.

More than one principle may apply. Select the principle best supported by the stated cause and consequence.

Accessibility: remove barriers

Accessibility considers the range of people and technologies that may interact with a solution. A user might navigate with a keyboard, use a screen reader, magnify content, have reduced colour perception, experience limited dexterity or need plain language.

- Use clear text labels, not icons alone when meaning could be ambiguous.
- Provide meaningful alternatives for non-text content.
- Use sufficient contrast and readable text sizes.
- Do not communicate status using colour alone; add text or a symbol.
- Make controls operable without requiring precise pointer movement.
- Use a logical reading and focus order.

Accessibility is not a special mode added at the end. It should shape the interface and its programmed behaviour from the beginning.

Improve an inaccessible status

A task list shows overdue work only by changing its title from black to red. **Recommend** one refinement and **Justify** how it improves accessibility.

View a sample solution

Add an *Overdue* text label and warning symbol beside the title while retaining strong contrast. Users who cannot distinguish the colour change can still perceive the status, so the meaning no longer depends on colour alone.

Effectiveness and utility

These principles are related but answer different questions.

- **Utility:** Is the required tool or feature available?
- **Effectiveness:** Does the interface let the user achieve the intended result accurately and successfully?

A maintenance app might have a search feature, satisfying some utility. If it searches only meaningless file names while workers know street names, the feature is ineffective in context.

Separate the principles

A canteen app must allow students to identify meals containing allergens. It displays a list of meal names but provides no allergen information or filtering.

1. **Explain** the utility problem.
2. **Explain** how adding allergen labels and a filter could improve effectiveness.

Check your answer

1. The interface lacks the required function or information for identifying allergen-safe meals, so it has poor utility for this task.
2. Persistent allergen labels reveal relevant information and a filter reduces the list to suitable meals. Students can identify an appropriate option more accurately and directly, improving effectiveness.

Safety and error recovery

A safe interface reduces the chance and consequence of harmful user error. The design should help before, during and after an action.

Stage	Interface support	Example
Prevent	Constraints, clear labels and suitable defaults	Disable *Submit* until required fields are valid.
Detect and explain	Specific validation and feedback	“Enter a date from today onwards” beside the date field.

Stage	Interface support	Example
Recover	Confirmation, cancel, undo or version history	Allow an archived record to be restored.

> **Confirmation is selective:** asking “Are you sure?” after every harmless action creates interruption and may train users to confirm without reading. Reserve strong friction for meaningful risk.

Design safe feedback

A volunteer records a medication delivery. The current form clears immediately after *Save*, even if the network request fails.

Recommend two UI refinements that protect safety and **Justify** each one.

View a sample solution

- Keep the entered values until the server confirms success, so a network failure does not silently remove information needed to retry.
- Display a specific success message with the recipient and time, or an error message with a retry control. The volunteer can distinguish a recorded delivery from a failed request and take the correct action.

Learnability and consistent interaction

Learnability improves when the interface makes actions predictable and reduces what users must remember.

- Use familiar controls for familiar actions.
- Keep navigation, terminology and control placement consistent.
- Prefer descriptive labels such as *Save booking* over vague labels such as *Go*.
- Show current status and provide immediate feedback.

- Reveal complexity when needed instead of crowding the first view.

A new user should be able to infer the next action from the interface. Instructions can support a design, but they should not compensate for unclear controls.

Check for understanding

A three-screen form places *Next* in the bottom right on screens one and two, but labels the same action *Continue now* in the top left on screen three.

Evaluate this choice using learnability and recommend a refinement.

View a sample solution

The changed wording and position break the interaction pattern, so users must search for and reinterpret the same action. Keep the label *Next* and its placement consistent, unless the final action has a genuinely different result such as *Submit application*.

Evaluate from evidence

A complete evaluation moves from observed evidence to a supported judgment.

> **Evidence:** [specific interface feature or omission]. **Effect:** [what happens when a particular user completes a task]. **Principle and judgment:** therefore the interface has [strong/weak] [useability principle]. **Refinement:** [specific feasible change] would [expected improvement].

Worked example

A field-work app lists jobs only by internal file name and latitude/longitude. Workers know jobs by street address and often work where mobile data is unreliable.

> **Too vague:** The app is ineffective and should have a better layout.

Evaluated: The list provides file names and coordinates rather than the street information workers use, forcing them to translate each location through another service. This causes delay and may fail without connectivity, so the interface is ineffective for locating jobs. Display a cached street address and allow searching or filtering by suburb and status so workers can find the correct job

directly.

Evaluate two principles

A sports-centre kiosk asks users to enter a booking code. Its only instruction is a small placeholder reading *code*. After submission the screen does not change for ten seconds, and repeated taps create duplicate check-ins.

Evaluate the interface using two useability principles. For each, identify evidence, explain the effect and recommend a refinement.

View a sample solution

- **Learnability:** The vague placeholder does not explain the expected booking-code format, and it disappears while typing. A persistent label such as *Six-character booking code* plus an example would help first-time users understand the input.
- **Safety:** The absence of immediate processing feedback encourages repeated taps that create duplicate records. After the first tap, disable the button, show a progress status and accept only one request until success or failure is returned.

Low-fidelity mock-ups

A low-fidelity mock-up communicates layout, controls, content and interaction without requiring production styling. It should be quick to draw and easy to revise.

A useful mock-up shows:

- the screen boundary and a clear screen name
- the information users need at that point
- inputs, outputs, navigation and feedback
- important states, such as loading, validation error or success
- annotations that connect features to users, criteria and programmed behaviour.

> **Annotate decisions, not appearances:** “Blue button” describes a surface. “Persistent labelled button submits the report and then displays success or a retryable error” explains purpose and interaction.

Worked annotated mock-up

A student needs to report a broken drinking fountain. The criteria require a location, a short description, an optional photo and clear submission feedback.

A mobile form with a screen title, location selection, issue description, optional photo, submit button and status region. Numbered callouts explain the design decisions. Report a broken fountain Location Choose a building or area ▾ What is wrong? Example: water will not stop Photo (optional) Add photo Submit report Status or field-specific error appears here 1 Screen title states the task,supporting learnability. 2 Location choices use familiarplace names, preventing invalidcoordinates. 3 Persistent label remains visible;example clarifies useful detail. 4 Action label describes the result.Disable after one press whilethe request is processed. 5 Status region confirms success orgives a specific error and retry,supporting safety and recovery.

Connect the mock-up to programmed behaviour

The drawn control is only the visible part of a feature. Explain what data enters, what programmed action occurs and what output returns.

UI element	Programmed behaviour	Visible response
Location list	Loads valid locations and stores the selected identifier.	Selected place name remains visible.
Submit report	Validates required values and sends one report request.	Field error, progress state, then success or retry message.

An annotation such as “button saves data” is incomplete. Identify the data, validation, result and feedback that fulfil the criterion.

Design from criteria and user context

Translate every criterion into visible interface evidence or programmed behaviour.

Criterion	Resulting design evidence
Users can find an event by suburb.	Labelled suburb filter; result count; clear-filter control.
Users can distinguish cancelled events.	Text label and symbol, not colour alone.
Users can recover from an accidental cancellation.	Confirmation containing the event name and a time-limited undo action.

User characteristics also change the design. An outdoor worker wearing gloves may need larger controls and offline information. A young child may need concise language, images with text, audio support and immediate feedback. Do not rely on stereotypes: use characteristics provided by the scenario.

Independent practice: after-school activity check-in

A school needs a tablet interface for staff checking students into after-school activities. The tablet is used outdoors, sometimes with glare and unreliable internet. Relief staff may be using it for the first time.

Criteria

- Search for a student by name or student ID.
- Show the student's selected activity and emergency-contact status.
- Record one of three states: present, collected or absent.
- Prevent *collected* being recorded without confirmation.
- Make unsynchronised records clearly visible without using colour alone.
- Allow failed records to be retried when connectivity returns.

On paper, **Symbolise** one low-fidelity tablet screen that addresses every criterion. Add at least six numbered annotations. Each annotation must explain:

1. the interface element
2. its programmed behaviour or information
3. the criterion or useability principle it supports.

Then **Evaluate** your mock-up against two different useability principles and **Recommend** one refinement.

View a sample solution

Possible screen components

- A persistent *Search by student name or ID* label, search field and clear control.
- A result card containing student name, ID, activity and emergency-contact status.
- Large, separated *Present*, *Collected* and *Absent* controls with text labels.
- A confirmation panel for *Collected* that names the student and offers *Confirm* and *Cancel*.
- A persistent synchronisation status using a symbol plus text such as *Not synced*.
- A retry control beside failed records and a message that preserves the locally stored state.

Example annotations

1. The descriptive search label remains visible while typing, supporting first-time relief staff and learnability.
2. Search accepts either name or ID, satisfying the search criterion and improving utility.
3. The result card groups identity, activity and emergency-contact status so staff can confirm the correct student before acting, supporting safety.
4. Large separated status controls support operation in glare and reduce accidental selection.
5. The collected confirmation repeats the student's name before saving, reducing the consequence of selecting the wrong record.
6. The *Not synced* text and symbol communicate status without colour alone, supporting accessibility.
7. Retry sends the preserved local record when connectivity returns, supporting effectiveness rather than forcing re-entry.

Example evaluation

The design supports safety because staff confirm the named student before recording collection, reducing a high-consequence status error. It supports effectiveness because failed records remain visible and can be retried rather than lost during unreliable connectivity. A refinement would be to include a final filter showing all unsynchronised records, so staff can

check that none remain before ending the session.

Revision #5
Created 2026-10-06 00:40:21 UTC by Mr Napper
Updated 2026-10-06 01:32:01 UTC by Mr Napper