METHOD

Prototyping a government service with Claude

A practical workflow for UX teams working on public services

01 Why prototype with AI?

For most of the last decade, prototyping a government service meant either Sketch or Figma mockups that didn't actually work, or hand-coded HTML prototypes that did but took a week to change. The first was easy to make and easy to dismiss in research ("but it doesn't *do* anything"). The second was credible but slow, and the cost of iteration meant most teams locked in design decisions too early.

AI-assisted prototyping changes the economics. A working, clickable prototype of an eligibility checker, a multi-page form, a decision wizard, or a service confirmation flow can now be drafted in under an hour, tested in research the same week, and rewritten between two participants in the same session. That speed is the point. It moves the bottleneck back where it should be — in research and decision-making — rather than in production.

This manual is about using AI tools, principally Claude, to do that work well. It assumes you already know the fundamentals of user-centred design and government service delivery (GDS Service Manual, USWDS, or your local equivalent). It does not assume any prior coding experience.

02 What this is — and isn't — useful for

AI prototyping is well-suited to:

- Eligibility checkers and decision wizards
- Multi-step forms following the one-thing-per-page pattern
- Service confirmation pages, decision letters, and notifications
- Status-tracking pages and dashboards for caseworkers
- Conversational flows for assisted-digital exploration
- Rapid A/B variants of content or layout for testing

It is not yet suitable for:

- **Production code.** Anything you put in front of real users in a live service still needs proper engineering, security review, and accessibility audit.
- **Authoritative content.** AI will produce plausible-looking benefit thresholds, eligibility rules, and statutory language. None of it is verified. Treat all policy content as draft that subject-matter experts must confirm.

- **Identity, payment, or anything that handles sensitive personal data.** Prototypes are for testing flow and content, not for transmitting real citizen data.
- **Final accessibility.** AI-generated prototypes typically pass basic automated checks but fail real audits and real users. Use them for research, not for accessibility sign-off.

The dividing line is whether the artefact will be used to make decisions about a real person. If yes, it needs production engineering. If it's used to make decisions about your design, AI prototypes are excellent.

03 Before you start: the prototype brief

Before opening Claude, write a brief. The brief is what separates a useful prototype from a generic-looking demo. It has six parts.

Policy intent

What problem is the service solving, and for whom? One paragraph. If you can't write this from memory, the policy team hasn't briefed you well enough yet — get them in a room first.

Target user

Be specific. "A first-time applicant for housing benefit, age 24 to 35, currently in private rented accommodation, using a smartphone, English as a second language" is useful. "Citizens" is not.

The transaction

What does the user need to do, end to end? Write it as a numbered sequence at the level a colleague could read aloud. This becomes the structural backbone of the prototype.

Stage

Which GDS service phase (Discovery, Alpha, Beta, Live) is this prototype for? Discovery prototypes can be rough and provocative. Beta prototypes need to be close to production. Don't conflate them.

Hypothesis

What are you trying to learn? "Does this flow help users understand whether they're eligible?" is a research question. "Make me a benefit calculator" is not. Without a hypothesis, the prototype has no success criterion.

Constraints

What must the prototype respect? Reading age (typically grade 9 or age 14 for general public services), language, brand pattern library, the parts of the policy that are non-negotiable, and any data shape it must produce.

A good prototype brief fits on one page. If yours is longer, you are probably trying to prototype two services.

04 The workflow

A repeatable five-step loop.

1. Frame the goal in writing

Open with one sentence: "I'm prototyping X to test whether Y." This becomes the first line of your prompt to Claude. It also stops you from sliding into building the wrong thing once the screens start appearing.

2. Draft the prompt as a structured brief

Don't paste the six-part brief in raw. Convert it into a system context Claude can act on: tone, audience, content constraints, what to include, what to leave out. A prompt of 300 to 500 words usually outperforms one of 50.

3. Generate version 1 as a self-contained HTML file

Ask explicitly for a single HTML file that opens in a browser, with no external dependencies. This makes the prototype shareable, runnable on a research participant's laptop without an internet connection, and easy to version on disk.

4. Test internally before testing with users

Walk through every path with at least one colleague who didn't see the prompt. Look for plausibly-wrong policy content (the most common failure mode), missing edge cases, tone that's off, and accessibility issues a basic audit would catch.

5. Iterate with structured feedback

Don't tell Claude "make it better" or "fix the form". Tell it precisely what to change and why: "On the eligibility page, the wording 'do you receive Universal Credit' has caused confusion in research because participants weren't sure whether legacy benefits count. Rewrite to disambiguate, and add a help-text expander." Specific feedback gets specific changes. Vague feedback gets superficial rewriting.

After each round of research, capture two things separately: the changes you want to make to the prototype, and the insights you want to keep regardless of whether the prototype survives. The prototype is disposable. The insights are not.

---

05 Prompt patterns

Five patterns that cover most government prototyping work.

The eligibility checker

Multi-page checker with one question per page, a clear progress indicator, and an outcome page that gives a yes / no / maybe result with reasoning. Include eligibility rules in your prompt as a structured list, not prose. Specify what "maybe" looks like — this is where most checkers get it wrong.

> *I'm prototyping an eligibility checker for **[service]**. Users are **[audience]**. Build a single-file HTML prototype following the one-thing-per-page pattern. The eligibility rules are: **[list]**. The outcome page should clearly say "you may be eligible / you are not eligible / we need more information" with reasoning. Use plain English at reading age 9. Do not include a real submission — the final page should make clear this is a research prototype.*

The decision wizard

Like an eligibility checker but the outcome is a recommendation about which service or pathway applies. Specify the decision tree explicitly. Specify what happens when the user's situation falls between two pathways — this is where real services fail and where prototypes need to feel honest.

The form with one-thing-per-page

For longer applications. Prompt with the list of questions in order, the validation rules, the error states, and the review-and-confirm pattern. Crucially, specify the back-button behaviour: the prototype should preserve answers when the user navigates back. AI often forgets this.

> *Build a single-file HTML prototype of a **[service name]** application following the one-thing-per-page pattern. The questions are, in order: **[list]**. Each page should have a back link that preserves answers, inline error states for invalid input, and a final review-and-confirm page that lets the user edit any answer. Reading age 9. Label all sample data as fake.*

The decision letter or notification

Generate the letter content as the prototype. Ask for tone (formal but warm, citizen-facing not lawyer-facing), reading age, structure (decision, reason, what happens next, how to appeal), and accessibility-friendly formatting. Letter design is rarely treated as a UX activity but profoundly affects user experience.

The status or case-tracker page

For caseworker-facing or claimant-facing tracking. Specify the states a case can be in, what's shown at each state, and what action (if any) is available to the viewer at each state. Make explicit that the prototype should show realistic-but-fake data, and label that data as fake on every screen.

06 Pitfalls specific to government

Plausible but wrong policy interpretation

AI will confidently state benefit amounts, thresholds, and eligibility rules that sound right but are wrong by either small amounts or several years out of date. Any policy content in the prototype must be either lifted verbatim from your actual policy source, or marked clearly as "[placeholder content — confirm with policy]".

Inappropriate tone

Default AI output is friendly and reassuring. Some government contexts — a decision letter, a sanction notice, a tax demand — need authority, not warmth. Specify tone explicitly per screen if needed. The opposite failure also exists: bureaucratic stiffness in a context that needs to feel humane, like a bereavement service.

Missing the assisted-digital case

The prototype assumes a confident digital user. Most government services need to work for people being helped through them by a staff member or family member. Prototype both flows, or at minimum write a separate section testing the assisted-digital pathway.

False accessibility confidence

Generated HTML usually passes basic automated WCAG checks (alt text present, contrast ratios met, semantic elements used). It often fails real users with screen readers, cognitive impairments, or motor difficulties. Treat accessibility audit as a separate workstream, not something the prototype proves.

Realistic-looking fake data

A prototype with names, dates, and numbers that look real reads as real to a stakeholder skimming a screenshot. Mark fake data visibly — a watermark, a "PROTOTYPE — sample data" banner, made-up names that are obviously made up (avoid John Smith; try Test User One).

07 Handoff and ethics

When the prototype has served its research purpose, several things matter.

Document the prompt as an artefact

The prompt that produced the prototype is more durable than the prototype itself. Save it alongside the research findings. The next person who needs to spin up a similar prototype will thank you.

Translate insights, not artefacts, to the delivery team

Don't hand a delivery team the HTML and say "build this." Hand them the research findings, the content patterns that worked, and the flow decisions that survived testing. The production build will and should look different.

Be transparent in research that this is a prototype

Tell participants explicitly. Don't let a polished AI-generated screen create the impression that you're testing a launched service. This affects both consent (what are they participating in?) and the validity of the findings (are they reacting to the design or to the gloss?).

Do not load real citizen data into the prompt

Ever. Use synthetic data for everything, even in seemingly trivial cases. AI providers have data-handling policies that vary, your agency's data governance rules almost certainly prohibit it, and the risk of leakage is real.

Do not use AI prototypes to justify decisions about real people

The prototype is a research tool. It is not a defence in a judicial review and it is not a substitute for a properly-built service with proper accountability. Keep that line bright.