CASE STUDY · UX × AI

MedImpact Pricing Tool

Overhaul

From scattered requirements to a working, high-fidelity prototype: how Claude.ai and the MI design system replaced a slower, frame-by-frame prototyping loop.

13	~80 min	1	33
working builds	of sessions analysed	design system, as code	requirements traced to sources

Prepared for prospective employers · Prototype documentation

All client names and figures shown in the prototype are fictional or illustrative.

OVERVIEW

Contents

Executive summary . . . 3

What it is, who it serves, why it was needed . . . 4

The operating model: before and after . . . 5

Requirements, traced to their sources . . . 6

How the requirements and the design evolved . . . 9

Capturing requirements from video, audio, screenshots and notes . . . 10

Bringing the Figma-based MI design system into Claude . . . 11

Claude.ai compared with our usual Figma and Figma-agent workflow . . . 13

The UX × AI working process . . . 14

The final prototype, screen by screen . . . 15

Quality: how it was verified, and what was caught . . . 19

Results, value and next steps . . . 20

Appendix A: glossary . . . 21

Appendix B: build index . . . 21

How to read this document

Sections 1 to 3 explain what the product is and why it needed an overhaul. Sections 4 to 6 record the requirements and how they changed. Sections 7 to 10 explain the method: how Claude.ai was used, how the Figma-based MI design system was brought into it, and how this compares with our usual Figma workflow. The final sections show the finished prototype, how quality was checked, honest limits, and next steps.

SECTION 1

Executive summary

The **MedImpact Pricing Tool** is an internal web application used by underwriters to price pharmacy benefit RFPs (requests for proposal). This document records how the overhaul of that tool went from fragmented inputs (meeting recordings and transcripts, call notes, screenshots of the existing reprice workspace, brand mockups and Figma design exports) to a fully interactive, design-system-faithful HTML prototype that stakeholders could click through, test with real files and react to.

We built and iterated the prototype almost entirely in **Claude.ai**. Rather than drawing and wiring up new frames for every state, we handed Claude the same raw material a design team starts from. It returned running software: real queue logic, real claim-file validation, a real approval path. Every build then went back through the designer's review and came back as annotated screenshots that drove the next version.

13 working, numbered builds (3 precursors and v1 to v10)	**33** tracked requirements, each traced to a source	**8** recorded sessions (about 80 minutes) analysed	**63** automated checks in the largest test suite

What a hiring manager should take from this

- **Absorbing change:** halfway through, the operating model changed from four specialist roles to one underwriter-owned queue with automated intake. The prototype absorbed the pivot in a single build, and every earlier version was kept for comparison.
- **Design-system discipline:** the Figma-based MI design system was converted into reusable tokens and components, so each new screen inherited the look and late changes stayed consistent.
- **AI fluency with a verification mindset:** Claude wrote and tested the code, while the designer directed, reviewed and corrected it. This document is explicit about what the AI could not verify.

SECTION 2

What it is, who it serves, why it was needed

Project name: **MedImpact Pricing Tool** (overhaul). Type: internal enterprise web application, delivered here as an interactive prototype.

What the product does

A pharmacy benefit manager (PBM) answers prospective clients' RFPs with a priced offer. Underwriters load the prospect's claims, price them under a rate card, guarantee and rebate assumptions, review the resulting offer (total paid, effective rate, rebates, spread, guarantees) and approve it. The tool turns a raw claim file into a reviewable, approvable offer.

Who it serves

Persona	Role in the process	What they need from the tool	In the final model
Underwriter (primary user)	Reviews each RFP, fixes claim-file problems, checks the offer, approves and submits	One queue that shows what needs attention; the whole offer in one view; a clear approve path	Owns the process from first review to approval
Intake team and automated intake	Created RFP tasks and entered prospect details	Fast, complete capture of prospect and line of business	Replaced by automated creation; manual entry kept as an exception
Rebate and Specialty specialists	Completed rebate and specialty inputs; their work showed as red or green checks	To see which RFPs are waiting on them	Folded into the underwriter role; their inputs live in Phase 2 panels
Network analytics / data lake	Source of claims and RFP data	Reliable ingestion and validation of claim files	Feeds the queue; failed files route to Needs review
Product owner and leadership	Reviewed each prototype build	Something real to react to, quickly	Review each numbered version

Why the overhaul was needed

Problem in the earlier approach	Design response in the prototype
RFPs were entered by hand and passed between role-based specialists, so ownership and next steps were unclear	Automated intake into one underwriter-owned queue; manual creation kept only as a break-glass path
Claim-file problems (missing dates, quantities, unmapped columns) surfaced late or without guidance	Validation on upload, a Needs review status, highlighted rows and tooltips that say exactly what to pick
The offer was spread across drawers and tabs, so the underwriter could not see all of it at once	One-page offer with every section visible; inputs read-only in place, edited through slide-over panels
Hard to tell which RFP needs which action	Status plus a claim-file chip in every queue row; tiles that count each status
Unclear approval path	One Approve and Submit action, enabled when the record is hydrated and saved
Visual inconsistency with the MI design system	Figma-based MI design system rebuilt as tokens and components and applied everywhere

SECTION 3

The operating model: before and after

The most important design event in the project was a change in the underlying model. The first requirements described four roles sharing one queue. A later working session with the product owner replaced that with a single underwriter-owned queue fed automatically. The flow below shows the final model.

	Early model (builds 0a to v3)	Final model (v4 to v10)
Roles	Intake, Rebate, Specialty, Underwriting	One role: the underwriter
Creating an RFP	Intake role only, via a New RFP button	Automatic from network analytics, CMP / Salesforce, monitored email and folders; manual path is an exception
Showing what is outstanding	Rebate and Specialty columns with red and green checks, plus N/A	Status (New, Needs review, Ready, In progress) and a claim-file chip
Approval	Underwriter-only button, locked until every check was green	Approve and Submit once the record is hydrated and saved
Claim files	Not modelled	Upload, mapping check, Needs review, Re-check, Hydrate
Rebate, rate card, levers, exclusions	Core to the workflow	Kept, but marked Phase 2 behind one switch

Final end-to-end flow

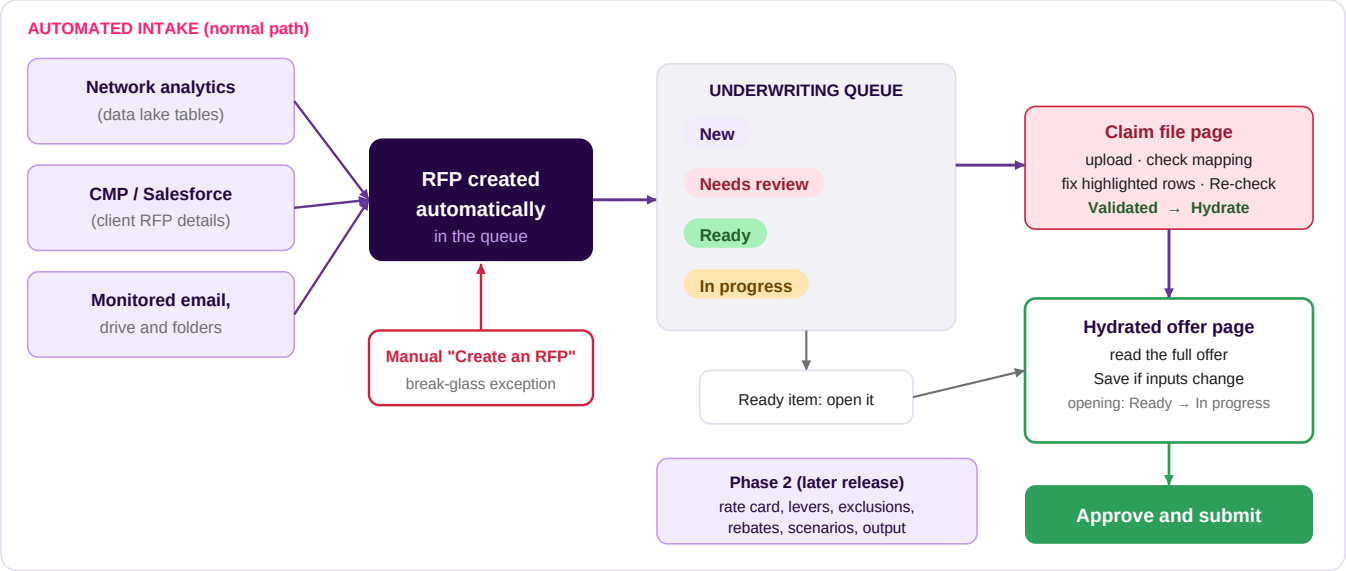


Diagram of the final flow. The queue statuses follow the call notes: New means nobody has reviewed it, Needs review is a pre-hydration claim-file problem, Ready means hydrated and waiting for review, and In progress means an underwriter has opened or changed it.

SECTION 4

Requirements, traced to their sources

Every requirement below came from a named input: the written requirements, the call notes, a recording, a screenshot or direct designer feedback. Where the version column says "superseded", the requirement was real at the time and was deliberately replaced.

Queue and status

ID	Requirement	Source	Built in
Q1	One shared queue for every RFP, owned end-to-end by the underwriter	Call notes	v4
Q2	Statuses New, Needs review, Ready, In progress, Approved; claim-file chip shown beside the status	Call notes, recording	v4
Q3	Queue tabs: In Progress, Approved, All RFPs	Designer direction	v1, kept
Q4	Search the queue from the top bar	Design export	v1
Q5	RFPs are created automatically from network analytics data, CMP / Salesforce and monitored email or folders	Call notes	v4 (simulated)
Q6	Manual New RFP kept only as a break-glass exception, visually de-emphasized	Call notes	v4
Q7	Status tiles that count New, Needs review, Ready and In progress	Design pattern	v4
Q8	Earlier model: Rebate and Specialty red/green check columns, N/A when Specialty not needed, Intake-only New RFP	Written requirements, session 1	v0a to v3, then superseded

Intake and claim files

ID	Requirement	Source	Built in
I1	Prospect, line of business and claim files on one page; no separate Ready step	Designer direction	v5
I2	Prospect fields: name, type, round, consultant, incumbent, implementation and due dates, claims dates	Recording of current tool	v4
I3	Prospect type, round and line of business as selects defaulting to "- select -"	Designer direction	v6
I4	Claim-file upload accepts any file during prototype testing	Designer direction	v5, hardened v9
I5	Add additional claim files; remove a file with a small x; green Validated pill when ready	Designer direction	v5, v9
I6	About eight required fields must map and be populated; missing key data (dates, quantities) blocks hydration	Call notes, recording	v4
I7	Mapping table with AI suggestions to confirm, confirm-not-mapped, Re-check and Undo	Recording of current tool	v4, restyled v6
I8	Rows behind each error highlighted, with a tooltip saying what to pick	Designer direction	v8
I9	Demo items in Needs review can be fixed and pass Re-check	Designer direction	v10
I10	Save incomplete work next to Back and Hydrate; keep it as Needs review instead of discarding	Call notes	v4
I11	Hydrate shows a spinner, then opens the hydrated page; progress continues in the queue if the user leaves	Designer direction, recording	v4, v5

Review and approval

ID	Requirement	Source	Built in
R1	The hydrated page shows the entire offer in the main body at all times, no hidden tabs	Designer direction	v2 onward
R2	Offer is complete against the reprice workspace: summary, ER vs guarantee, rebate guarantee, formulary, waterfall, details, RP 21, RP 22	Screenshot audit	v3
R3	Offer is read-only except fields with no panel (client guarantee per claim)	Designer direction	v2, v3
R4	Save in the same spot as Approve; Cancel appears only with changes; Save visible but disabled until something changes	Designer direction, call notes	v3, v4
R5	Approve and Submit available once the record passes claim-file checks and has no unsaved changes	Call notes	v4
R6	Editing after approval returns the RFP to In progress; opening a Ready RFP makes it In progress	Call notes	v4
R7	No breadcrumb trail	Designer direction	v3

Navigation and panels

ID	Requirement	Source	Built in
N1	RFP Queue at top of left nav; no My Favorites; no My Tasks view	Designer direction	v2
N2	Former tabs become left-nav items that open a slide-over panel (about one third of the screen) with Cancel / Save changes	Designer direction	v2
N3	A distinct, larger icon for every nav item	Designer direction	v2
N4	Rate card, levers, exclusions, rebates, scenarios and output are Phase 2; one switch hides them	Call notes	v4

Design system and quality

ID	Requirement	Source	Built in
D1	Follow the MI brand mockups, then the Leapfrog design system: nav, top bar, header card, tables, pills, buttons	Brand pack, Figma exports	brand build, v1 onward
D2	One table style everywhere; purple check icons on a lilac square like the collapse button	Designer direction	v6, v7
D3	Keyboard-reachable controls, descriptive labels on icon-only buttons, reduced-motion support	Assistant-driven	v2 onward

SECTION 5

How the requirements and the design evolved

Thirteen numbered builds were produced. Each one was kept, so a reviewer can open any version and see exactly what a change did. Most were requested and delivered inside the same working session.

Build	Name	What changed	Driver
0a	**Queue prototype**	Four roles (Intake, Rebate, Specialty, Underwriting). One queue with red/green Rebate and Specialty checks, N/A for Specialty, Intake-only New RFP, tabs per RFP, underwriter-only Approve gated on the checks.	Written requirements, recording and screenshots
0b	**Offer workspace**	Tabs removed: the whole offer on one scrolling page with every input beside the results, reprice status and Show toggles.	Seven recordings, designer: "we need to see the entire offer at all times"
0c	**Branded**	First reskin to the MI brand mockups: plum top navigation, tinted tiles, white cards, pill filters, table overhaul. Left rail removed.	Brand mockup pack
v1	**Leapfrog design**	Rebuilt in the Leapfrog design system: collapsible left nav with real logo, favorites menu, top bar, header card, in-body tabs, label-over-value fields. Logged as v1.	Figma exports (menu states, Level 3 detail screen)
v2	**Nav-driven panels**	Tabs moved into the left nav as slide-over panels with Cancel / Save changes. Offer body becomes read-only with Edit links. Favorites and My Tasks removed; new icons.	Designer direction
v3	**Complete offer**	Breadcrumbs removed; Save button beside Approve; gap analysis against the reprice workspace screenshot added Pricing Summary, ER vs Guarantee, full Rebate Guarantee, Details by chain, full RP 21 and RP 22, Show toggles and chart.	Designer screenshot audit
v4	**Single underwriter**	Roles removed. Automated intake simulation, statuses, manual wizard, claim-file mapping and validation, background hydration, Approve and Submit, Phase 2 flag.	Call notes and 9-minute recording
v5	**One-page intake**	Prospect, line of business and claim files on one page; any file accepted; multiple files with remove; spinner into the hydrated page; no Ready step.	Designer feedback
v6	**Cleaner table**	Mapping table restyled to match the hydrated-page tables; plain-text sample values; one check icon; select lists with "- select -".	Designer feedback with screenshots
v7	**Declutter**	Four captions and duplicate Needs review pills removed; + icon on "add additional claim"; check icons on a lilac square.	Annotated screenshots
v8	**Guided fixes**	Rows behind each error highlighted in red; tooltip icon says exactly what value to pick.	Designer feedback
v9	**Polish and hardening**	Green Validated pill; tooltip text wraps; fixed a defect where earlier picks blocked later uploads.	Screenshots (and a test that reproduced the defect)
v10	**Testable demo data**	Needs review demo files now carry fixable problems so a tester can pass Re-check and hydrate.	Designer feedback

SECTION 6

Capturing requirements from video, audio, screenshots and notes

The raw material was messy and multimodal. Claude reads text and images natively but cannot watch video, so we built a small pipeline instead of waiting for tidy documents.

Input	Volume	How Claude used it
Meeting transcripts and written call notes (AI-generated, human-checked)	Several documents	Read directly; treated as authoritative for decisions and wording where recordings were unclear
Recorded working sessions	7 trimmed recordings (about 70 min) plus one 9-minute screen recording	Frames pulled as contact sheets to find the key moments, then full-resolution frames of those moments; audio run through an offline speech-to-text model; results cross-checked against the notes
Screenshots of the existing reprice workspace	Multiple	Used as the ground truth for what a complete offer contains; later used for a section-by-section gap analysis
Brand mockups and Figma design exports	Dozens of images	Colour, typography, layout and component patterns extracted programmatically (see Section 7)
Designer feedback	Every round	Annotated screenshots with red circles and short instructions; each became a precise, testable change

Comparing the transcripts with the videos

When sources disagreed, the rule was: say so, pick the most recent and most explicit direction, and keep the older version available. These are the conflicts that mattered.

Conflict	Resolution
Roles: earlier requirement had four roles; call notes remove three of them	Followed the newer call notes; older builds preserved for comparison
"Remove the statuses" (spoken) versus four statuses (call notes)	Kept the four statuses, shown mainly through a claim-file chip, and flagged it for confirmation
Check Mapping as its own step (on screen) versus folded into Claim Files (spoken)	Merged them, then later merged all intake into one page
Save only when something changed (earlier request) versus an explicit Save (call notes)	Save is always visible but disabled until something changes
Several claim files (spoken) versus one file per line (on-screen text)	Built multiple files with remove, following the later designer direction
Anyone can create an RFP (earlier session) versus Intake only (written)	Followed the written requirement until the call notes made creation automatic
Audio passages that were unintelligible	Not guessed at; flagged as unknown and decided from the notes

SECTION 7

Bringing the Figma-based MI design system into Claude

The goal was a prototype that looks and behaves like the real product, not a generic approximation. The MI design system lives in Figma, so the bridge was the design exports.

1 Export	The designer exported the relevant Figma frames as images: the expanded and collapsed menu with hover and active states, the member-search screen, the group-details overview and the Level 3 detail screen. A separate brand mockup pack supplied the earlier colour language.
2 Extract tokens	Claude analysed the images programmatically, sampling pixel colours from specific regions to derive exact values for navigation, active states, accents, rules, badges and status colours, then measured spacing and sizing patterns.
3 Recognise components	From the frames Claude identified the reusable parts: collapsible left navigation with an icon chip and pink active bar, top bar with search, breadcrumb, header card with a meta line, light-weight title and pills, tabs with a pink underline, label-over-value fields, lilac-header tables, status pills, filled and outlined buttons.
4 Encode as code	Tokens became CSS custom properties and the components became shared render functions, so every new screen reuses the same building blocks. The logo was cropped from the export and embedded as an asset.
5 Review against Figma	The designer compared each build with the Figma frames and sent back annotated screenshots. Corrections (for example "same icon for both, with hover text" or "table should match the hydrated page") were applied to the shared components, not to one screen.

Design tokens extracted from the Figma exports

Nav / titles
#240644

Active item
#320C5B

Accent / active tab
#FF1E74

Icon chip
#C59CF1

Buttons / links
#613590

Page background
#F2F1F5

Lilac (headers, badges)
#F2EBFB

Rules
#E5D7F8

Status green
#A7F0BA

Label grey
#6F6F6F

Typography. The Figma frames appear to use a Roboto-style sans-serif (the font file itself was not supplied), with light-weight page titles and bold grey labels over regular values. The prototype uses Roboto first with a system-font fallback so it renders on any machine.

Fidelity note. Claude works from image exports and cannot see the rendered page, so fidelity was verified the way a designer would verify it: by comparing the running prototype to the Figma frames. We therefore describe the result as high-fidelity, and credit the designer’s review for closing the last gaps.

Component inventory: from Figma pattern to working HTML

Component	Pattern taken from the Figma exports	How it works in the prototype
Left navigation	Plum panel, icon chips, pink active bar, collapse control, hover state	Collapses to icons; items open slide-over panels; disabled until an RFP is open
Top bar	White bar with outlined search field, utility icons, avatar	Search filters the queue live
Header card	Meta line, light-weight title, pills, dates row, summary boxes	Shows status, source, last update, headline numbers and the Save and Approve actions
Tabs	Plain text with pink underline for the active tab	Queue views: In Progress, Approved, All RFPs
Label-over-value fields	Grey bold label above the value with a hairline rule	Read-only offer inputs; editable twins in panels and the intake page
Tables	Lilac header, bordered cells, right-aligned numbers, shaded rows	Same table everywhere, including the mapping table and its red action rows
Pills and chips	Rounded status pills, uppercase badges	Status, claim-file state and Validated
Buttons	Filled purple primary, outlined secondary	Disabled states explain themselves in the footer or tooltip
Icon on a lilac square	The collapse button	Reused for the confirm check in the mapping table, at the designer’s request
Slide-over panel	New pattern composed from existing tokens	Opens from the nav over about one third of the screen; Cancel / Save changes; blocks leaving with unsaved edits

Why this made late changes fast

Because the design system lives in shared tokens and components, a request such as "make every table look like the hydrated-page table" changed one component and every screen followed. This is the practical advantage of rebuilding the system as code rather than re-drawing it per screen.

SECTION 8

Claude.ai compared with our usual Figma and Figma-agent workflow

Our usual approach to rapid prototyping starts in Figma: components, auto-layout frames, prototype links and, increasingly, Figma's AI-assisted tools. That approach is strong for visual exploration and for stewarding the design system. This project used Claude.ai for a different job: turning an approved design system plus changing requirements into **testable behaviour**.

Dimension	Typical Figma or Figma-agent prototype	Claude.ai-built HTML prototype (this project)
Interaction depth	Linked frames and variants; flows follow pre-scripted paths	Real state: statuses change as work happens; files upload; mapping validates; hydration runs; approval locks and unlocks
Rules and data	Hard-coded sample content; each edge case needs more frames	Rules written once (required fields, status derivation, scaling of sample data, edit-after-approval reopens)
Handling a requirements pivot	Re-draw the affected frames and re-link flows	Change the rule once and the whole product reflows; the four-roles-to-one-underwriter pivot was a single build
Design-system fidelity	Native: components and variables live in the file	Rebuilt from exports as tokens and components, verified against Figma by the designer
Testing	Usability sessions on click paths	The same, plus automated checks of rules and flows before every hand-off, and real files can be uploaded
Stakeholder review	Share a link; comment on frames	One HTML file that runs in any browser; reviewers can try their own inputs
Developer hand-off	Specs, variables and dev mode	Readable HTML, CSS and JavaScript with tokens, states and accessibility attributes; prototype-grade, not production code
Cost of change	Low for visual change, higher for behavioural change	Low for both; every change shipped as a new numbered build
Weak points	—	Cannot see rendered pixels; depends on exports; no live link to the Figma file; needs designer verification

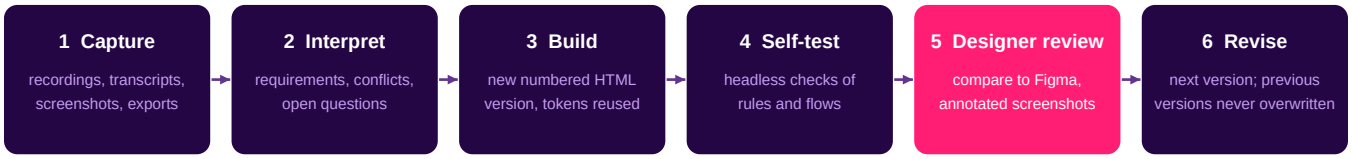
Our position

Figma remains the source of truth for design. Claude.ai did not replace it. It replaced the slowest part of prototyping: hand-assembling every state and flow so people can finally test the thing. The best results came from a hybrid: Figma defines the system, Claude turns the system and the requirements into something that behaves, and the designer closes the loop.

SECTION 9

The UX × AI working process

The workflow was a tight loop that repeated for every build. The designer stayed in the role of art director and reviewer; Claude acted as analyst, front-end engineer and tester.



Most revisions were delivered within the same working session they were requested.

Ground rules that kept quality high

- **One numbered version per change.** Nothing was overwritten, so every decision is reviewable and reversible.
- **Say what is uncertain.** Each hand-off listed assumptions, discrepancies between sources and anything not built.
- **Test before showing.** Every build was run through automated checks in a headless browser before it was handed over.
- **Ask only when blocked.** Ambiguity was resolved with a stated assumption and flagged, not with a stream of questions.
- **Fix the component, not the screen.** Corrections were applied to shared parts so the whole product improved at once.

Direction patterns that worked

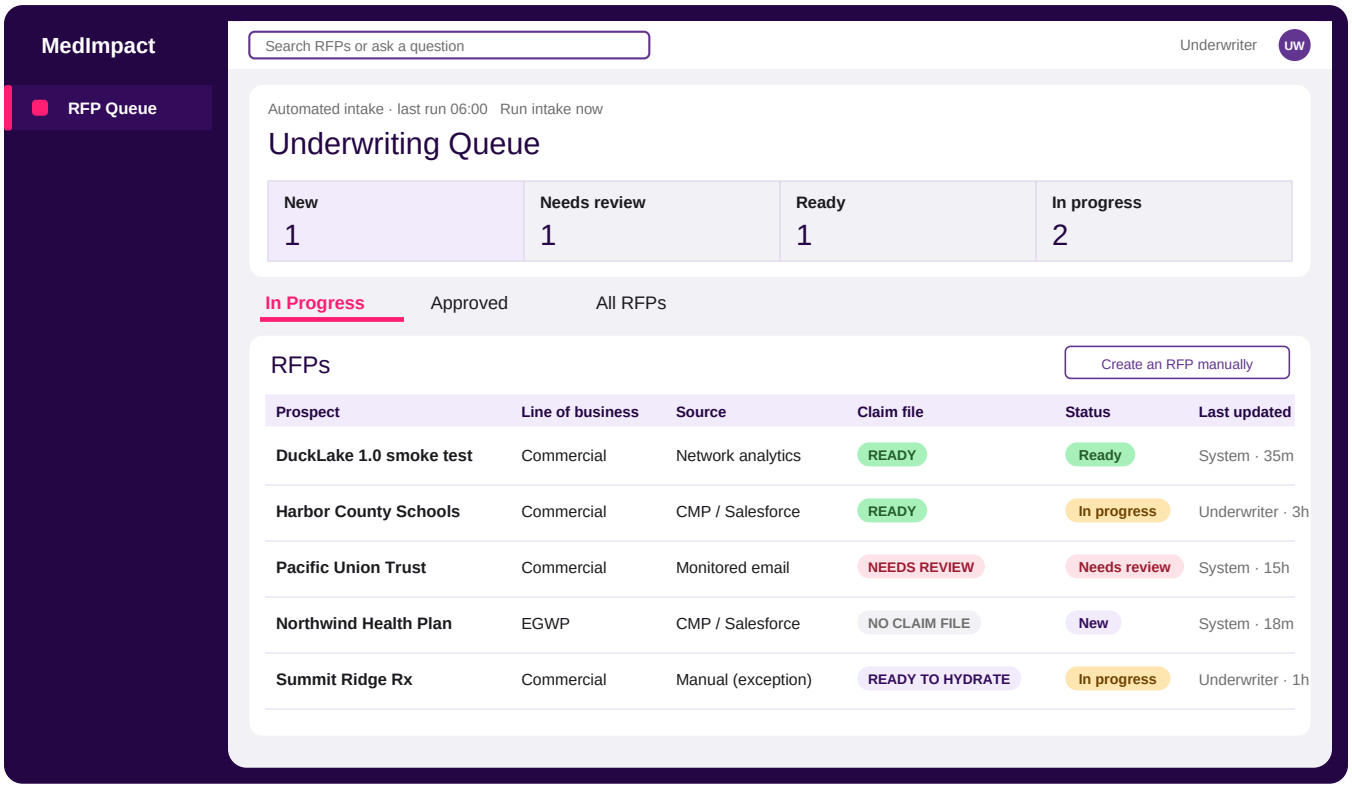
Pattern	Example of the direction given (paraphrased)	What it produced
Precise, testable instruction	"Make the same icon for both, add alt text that says what it is on hover."	One shared check icon with different hover text for each action
Annotated screenshot	A screenshot with red circles: "we don’t need Needs review in three places."	The duplicate pills and headings removed; only the header status kept
Ask for a gap analysis	"Make sure we have all the sections in the screenshot’s offer. Feels like we may be missing something."	A section-by-section audit and five missing sections added
Compare sources	"Read the transcripts. Compare the transcripts with the videos."	A conflict list with a recorded resolution for each
Version discipline	"Let’s log it" and "call it pricing-tool-v2."	A clean version trail from v1 to v10
Reality check from the user	A screenshot of a problem the tests had not caught	Root-cause fixes and new regression tests

SECTION 10

The final prototype, screen by screen

The drawings below are schematics of the final screens, drawn for this document. They show structure and interaction, not pixels. The prototype itself is a set of single-file HTML pages that run in any browser.

1 Underwriting Queue

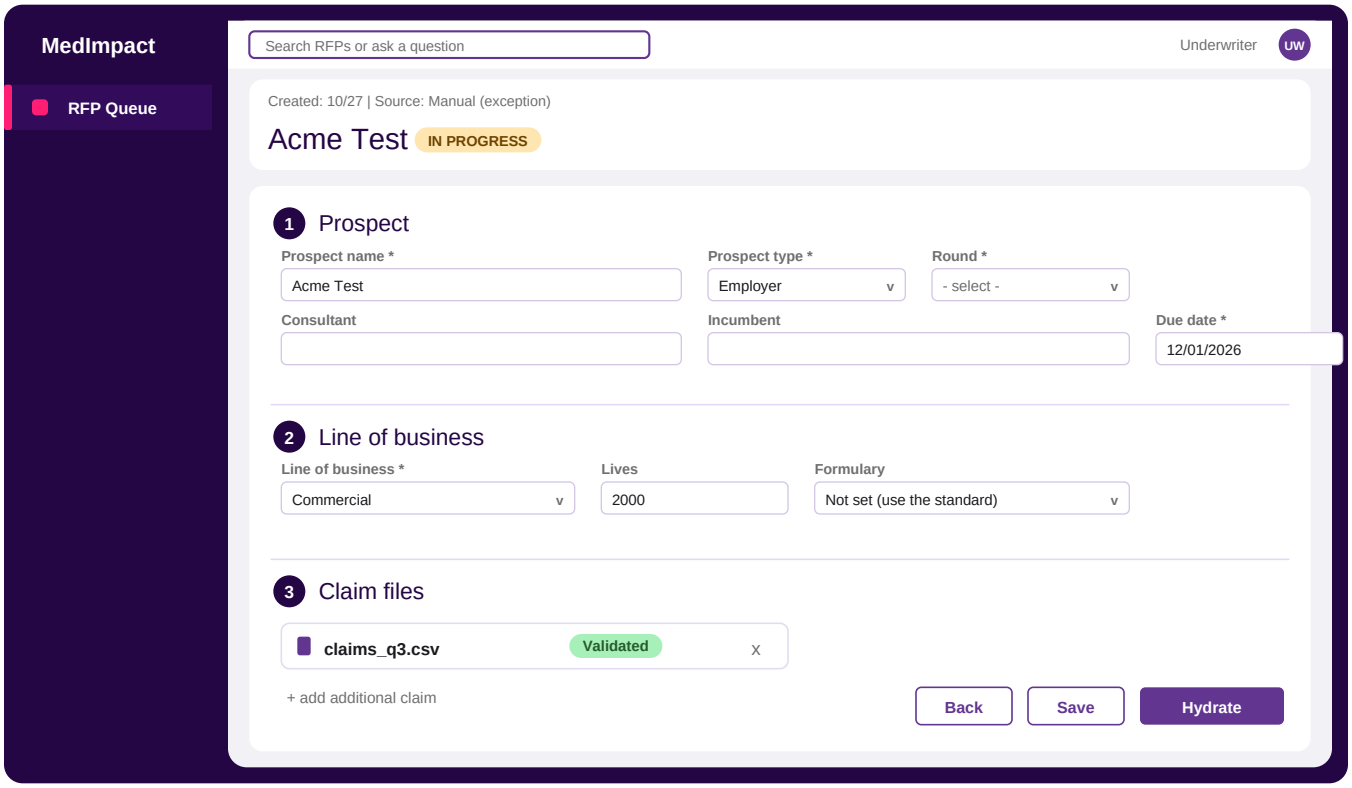


One queue, one owner. RFPs arrive automatically and each row says where it came from, what state its claim file is in and what status it has.

What to notice

- The "Automated intake" line and "Run intake now" simulate the feed, so a reviewer can watch new RFPs arrive and land in the right status.
- Tiles count New, Needs review, Ready and In progress. Tabs split In Progress, Approved and All RFPs, and the search box filters live.
- The claim-file chip reads No claim file, Needs review, Ready to hydrate, Hydrating n% or Ready, so the next action is obvious without opening the RFP.
- The manual path ("Create an RFP manually") is a small outlined button on purpose: it is a break-glass exception, not the front door.
- Opening a Ready RFP moves it to In progress; an RFP in Needs review opens straight onto its claim files.

2 One-page intake and claim files

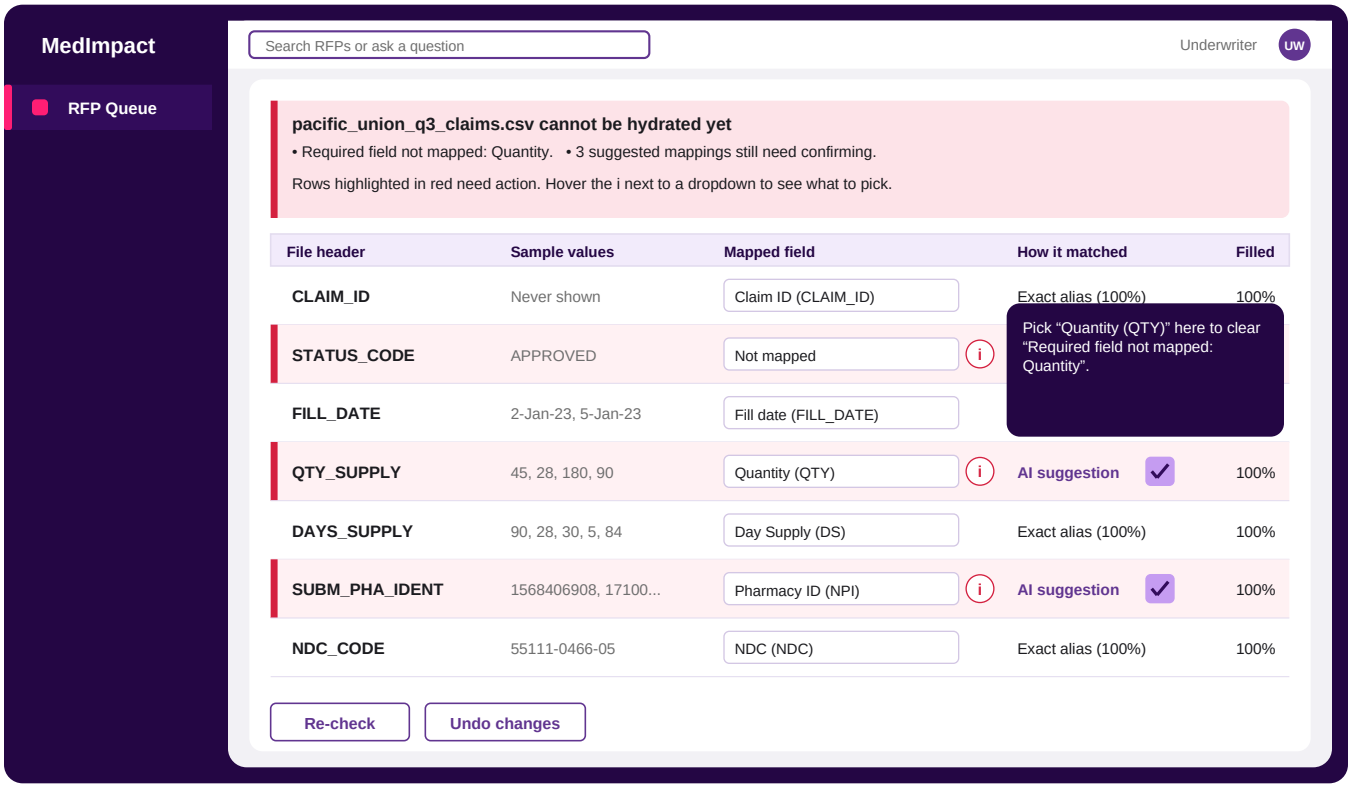


Prospect, line of business and claim files share one page, replacing a six-step wizard. There is no separate Ready step: the hydrated page itself shows readiness.

What to notice

- Dropdowns default to "- select -". Lives and formulary appear once a line of business is chosen and may wait until known.
- Before upload there is one primary button, "Upload Claim File". After upload, a file line shows the name, a green Validated pill and a small x to remove it.
- "Add additional claim" is deliberately a quiet secondary link: it is rarely needed. During testing any file type is accepted.
- Back, Save and Hydrate sit together. Save keeps incomplete work in the queue instead of discarding it. Hydrate stays disabled, with a plain-language reason beside it, until everything required is present.
- Hydrate shows a spinner and then opens the hydrated page automatically. Leaving mid-way is safe: the queue shows live progress.

3 Fixing a claim file that needs review

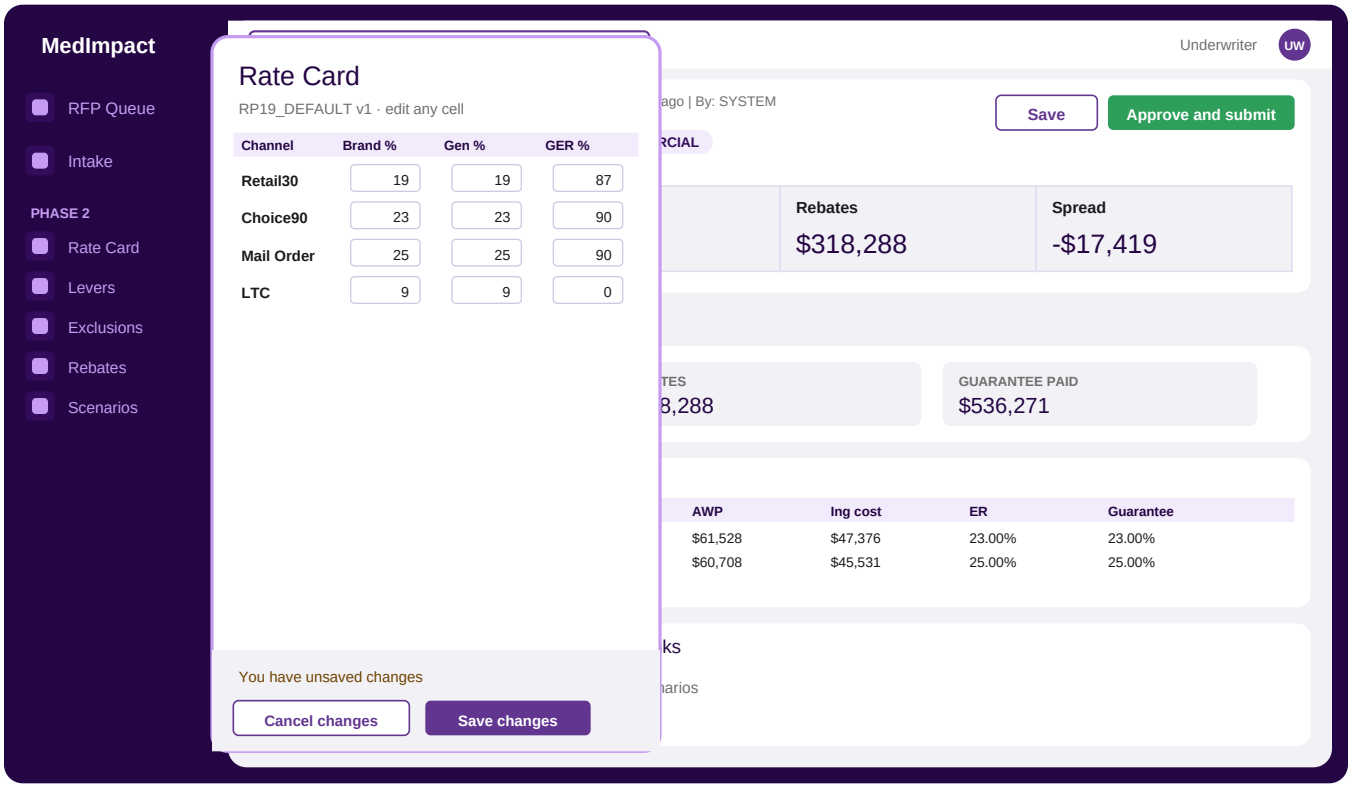


When a claim file cannot be hydrated, the page does not just say so: it shows the rows behind each problem and what to do about them.

What to notice

- Red rows are the ones an error points to. A required field with no column flags the likely source column; an unconfirmed suggestion flags itself; blank data flags its column.
- The i beside a dropdown opens a tooltip that says which value will clear the error ("Pick Quantity here to clear Required field not mapped: Quantity"). Blank-data rows say that no mapping can fix it and a corrected file is needed.
- One purple check icon on a lilac square confirms either a suggested mapping or a "not mapped" column, with different hover text for each. The cell text then changes to confirmed.
- Re-check re-runs the checks and Undo returns to the starting state, which makes the flow repeatable in a test session.
- When every problem is cleared, the red box and the table disappear, the file gains the Validated pill and Hydrate enables.

4 The hydrated offer, with a slide-over panel



The underwriter sees the entire offer in the main body. Inputs are read-only in place; changing one opens a panel from the left nav, over about one third of the screen.

What to notice

- Header: source, last update, status, headline numbers, Save and Approve and Submit together at the top right.
- Sections follow the real reprice workspace: Pricing Summary, ER vs Guarantee, Rebate Guarantee, Rebates by Formulary, Waterfall, Details, RP 21 Summary and RP 22 Exclusions.
- Only the client guarantee per claim can be edited in the body; changing it reveals Save and Cancel. Approve stays disabled until the edit is saved.
- Panels edit a draft. Cancel changes discards it; Save changes applies it, closes the panel and updates the offer. Leaving with unsaved edits is blocked.
- Phase 2 items (rate card, levers, exclusions, rebates, scenarios, output) sit under their own nav heading and can be hidden with one switch.

SECTION 11

Quality: how it was verified, and what was caught

After every build Claude ran automated checks in a headless browser: it opened the page, clicked through the flows, filled in fields, uploaded files and asserted the results. The largest suite had 63 checks. This caught real defects before hand-off, but it cannot judge appearance, so the designer’s review caught the rest.

Defect	Found by	Fix
Seed data was created before the pricing constants it depended on, which would have stopped the page loading	Automated check	Reordered initialisation
A helper function was deleted during a rewrite, so "Keep editing" and "Discard" in the unsaved-changes dialog would have failed	Automated check	Restored and covered by a test
The Hydrate button did not enable while typing in text fields	Code review while writing the tests	Footer state now updates live
Mappings picked on one file silently applied to later files, so a new upload was wrongly blocked ("fgg.docx: NDC not mapped")	Designer screenshot, then reproduced in a test	Uploads ignore earlier picks; regression test added
Tooltip text ran out of its box because table cells forbade wrapping	Designer screenshot	Fixed width with wrapping inside the tooltip
The demo file in Needs review could never pass Re-check (blank dates cannot be fixed by mapping)	Designer screenshot	Demo files now carry fixable problems

Verification philosophy

Automation checks behaviour; people check judgement. Three of the six defects above were found only because the designer looked at the result and sent a screenshot. Treating the AI’s output as work to be reviewed, with tests and a human in the loop, is what made the speed safe.

Scope and honest limits

- This is a prototype. Pricing numbers are illustrative: they scale a sample claims file, and lever effects are placeholders rather than real pricing logic.
- Claim-file "AI suggestions" are simulated with simple alias rules. Real parsing and mapping would come from the data platform.
- There is no back end. Data lives in the browser session; refreshing resets it.
- Claude cannot see rendered pixels. Visual fidelity relies on the designer’s review against the Figma frames.
- Some audio passages in the recordings were unintelligible; decisions in those places came from the written notes.

SECTION 12

Results, value and next steps

What the approach delivered

- **A working product to react to, not a picture of one.** Stakeholders reviewed real queue behaviour, real validation and a real approval path in a browser.
- **A requirements pivot absorbed cleanly.** Moving from four roles to one underwriter and from manual to automated intake took one build, with the old version preserved.
- **Consistency at speed.** One set of tokens and components kept thirteen builds visually coherent and made late corrections cheap.
- **Traceability.** Every requirement links to its source; every build links to its driver; every disagreement between sources has a recorded resolution.
- **Reusable artifacts.** The tokens, component inventory, mapping rules, test scripts and version log can all seed the production build.

Skills this project demonstrates

Skill	Evidence in this project
Systems thinking in UX	Turned roles, statuses, claim files and approval into one coherent workflow and flow diagram
Design-system stewardship	Extracted tokens and components from Figma exports and applied them consistently across ten versions
AI direction and critique	Precise instructions, annotated feedback, gap analyses, source-conflict resolution
Research synthesis	Combined transcripts, recordings, screenshots and notes into a traced requirement set
Quality mindset	Automated checks every round, plus a candid account of what slipped through
Communication	Flagged assumptions and limits in every hand-off, and wrote this documentation for a non-specialist audience

Recommended next steps

- Run moderated usability sessions with underwriters on the current build, using their own claim files.
- Connect real claim parsing, mapping suggestions and the pricing engine; replace placeholder lever effects.
- Sync the extracted tokens back into Figma variables so design and prototype share one source.
- Complete the Phase 2 scope (rate card, levers, exclusions, rebates, scenarios, output) against the same components.
- Run an accessibility audit and responsive review, then hand the component inventory to engineering.

APPENDIX

Appendix A: glossary

Term	Meaning
PBM	Pharmacy benefit manager: administers prescription drug benefits for employers and health plans
RFP	Request for proposal: a prospective client asks PBMs to bid, and the underwriter prices the bid
Claim file	The prospect’s historical prescription claims, used as the basis for pricing
Mapping	Matching each column in a claim file to the field the tool expects (for example Quantity, NDC, Fill date)
Hydration	Loading the validated claims into the data lake and building the model that drives the offer
Needs review	A pre-hydration state: the claim file has a problem that the underwriter must resolve
AWP	Average wholesale price, the list price from which discounts are applied
ER (effective rate)	One minus ingredient cost divided by AWP: the overall discount achieved
Guarantee	The discount rate or rebate amount the PBM promises the client
Rebate guarantee	Rebate dollars divided by eligible brand claims, compared with the client guarantee per claim
Formulary	The list of covered drugs and tiers that determines which claims earn rebates
Waterfall	Total paid shown step by step as each lever is applied
Levers	Adjustments the underwriter can apply, such as shifting claims between channels
Phase 2	Rate card, levers, exclusions, rebates, scenarios and output; later than the first release

APPENDIX

Appendix B: build index

File	What it is
mi-reprice-queue-prototype.html	Build 0a: four-role queue with Rebate and Specialty checks
mi-reprice-offer-workspace.html	Build 0b: one-page offer, no tabs
mi-reprice-branded.html	Build 0c: first brand reskin
pricing-tool-v1.html	Leapfrog design with in-body tabs
pricing-tool-v2.html to v3.html	Slide-over panels; complete offer with body Save
pricing-tool-v4.html	Single-underwriter model with automated intake, claim-file validation and hydration
pricing-tool-v5.html to v10.html	One-page intake, table clean-up, guided fixes, hardening, testable demo data