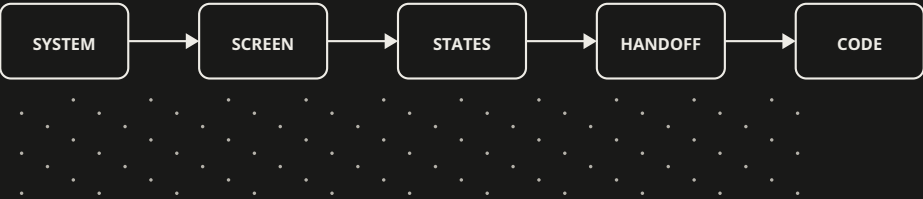

CLAUDE DESIGN / FIELD PRACTICE

Claude Design Field Guide

From a living design system to a production-ready handoff



Michael Long

October 5, 2026 | Version 1.3.0

Orientation

Use this guide to extend an existing product through Claude Design, Claude Code, and Figma. The recurring example is **Velori**, a personal-robot store. A shopper chooses a finish, face, expression color, and accessory, then checks the configuration before returning to edit.

The assignment starts with a working configurator. Preserve its identity, product photographs, choices, captions, and selection behavior. Add one bounded review journey. The public package includes a completed React answer key, frozen reference, reusable components, prompts, and local validation scripts. Historical native packets and their metadata remain private and are excluded from the download. Follow Chapters 2 and 3 to define and validate your source system before creating a new review screen.

Claude Design supports interactive prototypes and a Claude Code handoff. Claude Code also provides a `/design` command for artboards. Choose the surface that can produce the required result, then test its interactions and handoff. See the [Design getting-started guide](#) and [Claude Code command reference](#).

Start the sample

Download the [Velori sample](#), extract it, then run these commands from its `velori-store` folder:

```
cd velori-store
npm ci
npm run dev
```

Open <http://127.0.0.1:4197>. Use Node.js 22 and npm. The first install requires the npm registry. Product assets ship in the folder, and the store runs without CLEAR, a Claude account, or a Figma account. If the port is occupied, choose another port and use the address Vite prints.

Select **Graphite**, **Simple**, **Amber**, and **Carry bag**. Open accessory details, browse Charging dock, and return. Continue to review, then edit. Reload. The robot, caption, controls, and summary should still agree.

Figure 1
Configure overview

VELORI | Personal robotics

1 Configure2 Review

YOUR EVERYDAY COMPANION

View accessory details



Velori, made for your home.

Graphite shell · Simple face · Amber · Carry bag

THE DETAILS MAKE IT YOURS

Make Velori yours.

A finish, a face, a little personality.

Shell finish

Chalk

Pebble

Graphite

Face display

Friendly

Simple

Face expression color

Amber

Accessories

Dock

Carry bag

None

A padded carry bag keeps Velori protected on the move.

Continue

Figure 1 (continued)
Configure controls detail

THE DETAILS MAKE IT YOURS

Make Velori yours.

A finish, a face, a little personality.

Shell finish



Chalk



Pebble



Graphite



Face display



Friendly



Simple



Face expression color



Amber



Accessories

Dock



Carry bag

None

A padded carry bag keeps Velori protected on the move.

Continue

iv

Figure 1 (continued)
Review summary

YOUR VELORI

Review configuration.

Check your choices.

Shell finish

Graphite

Face display

Simple

Face expression color

Amber

Accessories

Carry bag

Edit configuration

← Back

Public React sample. Graphite, Simple, Amber and Carry bag appear in the controls, configured photograph, caption and Review summary.

The completed review screen is new sample behavior. For the exercise, provide the configurator and accessory details as the baseline and ask Claude Design to recreate the review journey. Keep the worked sample beside your result for comparison. Payment, inventory, accounts, shipping, and order submission remain outside the brief.

What you will produce

- 1 A preservation brief with a named configuration fixture.
- 2 A React system imported into your chosen Claude Design workspace.
- 3 A generated review journey that retains the selected values.
- 4 An actual handoff that runs from a new local folder.
- 5 A bounded React implementation with navigation and persistence checks.
- 6 A native Figma working copy with component and variable identity verified.
- 7 A review record that separates the app, native tools, candidate package, and publication.

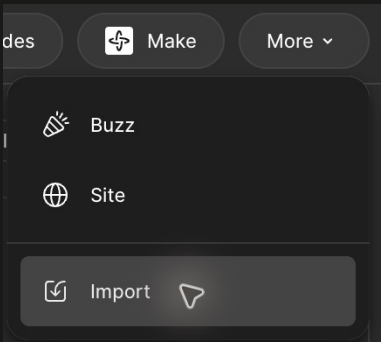
Choose the destination first

Confirm the intended account, organization, selected system, and project before uploading. Claude Code login and Design read/write authorization are separate steps when `/design-login` requests authorization. Use the same intended account for that browser flow. If access fails, check the active organization and Design eligibility before retrying the upload. The [Design admin guide](#) describes organization settings.

Download the [Velori native Figma file](#) and its [companion manifest](#) separately from the React ZIP. Import the `.fig` file into your own Figma Drafts and record the new file URL and companion SHA-256. Preserve the supplied source pages and reference frames; create a separate page for your work. The completed Store handbook exercise page is an answer key for comparison, not initial generation context.

Figure 2

Import control



Import completed

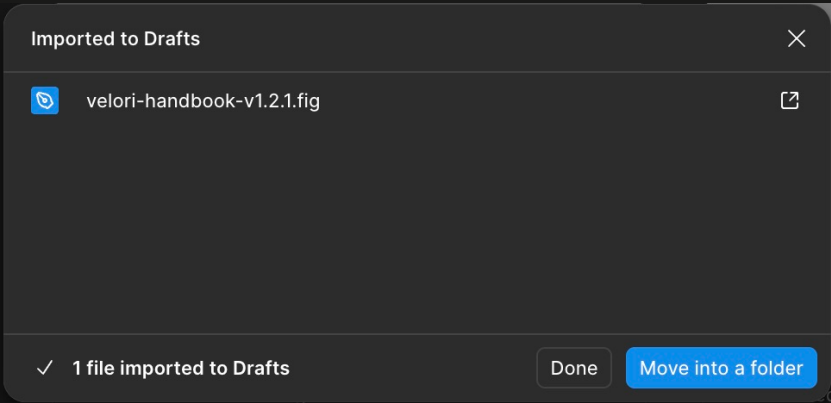
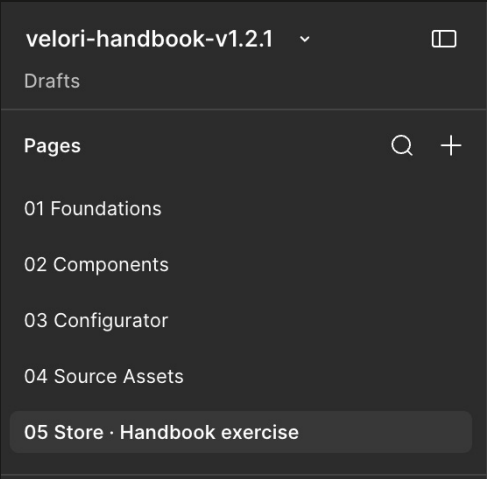


Figure 2 (continued)
Imported Drafts pages

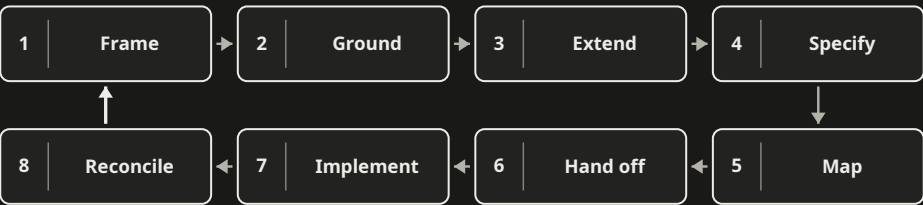


Figma. More opens the Import control. The completion dialog confirms import into Drafts; the imported file retains the supplied source pages and separate exercise page.

For the variable-driven native journey, use a paid Figma Design plan or eligible Education plan with edit access. Starter can import the file, but variables in prototypes require those plan features. Confirm this access before starting the Figma work.

The Velori sample source, documentation, frozen reference, imagery, and authored native Figma companion use the MIT license. Keep the copyright and third-party notices when redistributing them. Arimo and Lora are font references; no font software is included in the companion. Captured Claude Design source and runtime remain in the private review archive and outside these downloads.

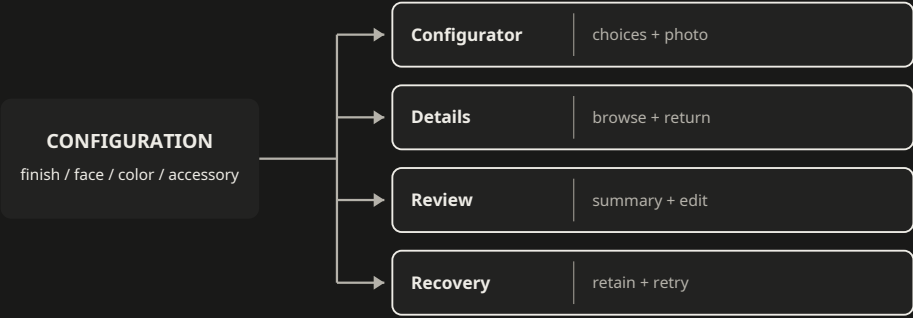
The workflow



Rendered evidence and drift restart the loop

Stage	Working tool	Observable result
Frame	Product brief and local app	Preserved choices and bounded extension
Ground	Claude Code and Claude Design	Imported components and photos load
Extend	Claude Design	Configure, details, review, and edit agree
Specify	Design and team review	Navigation, focus, reload, and recovery defined
Map	Design and engineering	Components and state ownership identified
Hand off	Claude Design to Claude Code	Captured source and dependencies run locally
Implement	Claude Code	React change passes the journey checks
Reconcile	Figma	Editable frames and native prototype agree

The configurator, accessory details, review, and image recovery share the same four configuration values. Browsing an accessory must not change the selected accessory. An unavailable photo must not erase the shopper's choices.



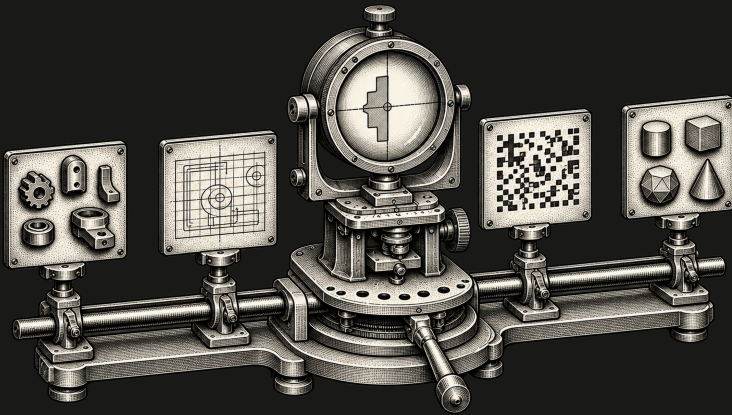
Contents

The chapters move from source authority to rendered proof. Worksheets and appendices remain part of the same operating package.

Orientation	ii	9. Hand off to Claude Code	44
1. Decide what is authoritative	1	10. Implement in Figma and reconcile	50
2. Prepare the Velori system	5	11. Guided capstone	57
3. Validate the imported system	13	Appendix A: Prompt sequence	62
4. Extend the existing screen	21	Appendix B: Handoff record	62
5. Connect the adjacent surfaces	26	Appendix C: Package map	63
6. Specify states and transitions	30	Further reading	64
7. Prove keyboard and responsive behavior	35	Changelog	65
8. Run the design evidence gate	41		

Decide what is authoritative

Preserve a working product before asking an assistant to extend it.



A design-system request can refer to component anatomy, React behavior, tokens, screenshots, or product rules. Each source answers a different question. Declare which source governs each decision before generation.

Run your working copy

From the extracted package, run `npm ci` and `npm run dev`. Open <http://127.0.0.1:4197>. Use Node.js 22 and a free local port. Choose Graphite, Simple, Amber, and Carry bag, follow details -> review -> edit, and reload before changing the app. The README provides reset and keyboard instructions. The assignment adds review to the supplied configurator; the completed sample is the expected result. Commerce remains outside the brief.

Establish the source order

- 1 The consumer application governs identity, imagery, configuration values, and captions.
- 2 The React components govern props, selection behavior, keyboard interaction, and rendering.
- 3 Tokens govern color, typography, spacing, and control dimensions.
- 4 The Figma working copy governs native anatomy, variants, variables, and prototype wiring.
- 5 The approved brief governs the new review journey.

The source snapshot and manifest retain copied file hashes, source commit, exclusions, and the copy date. The reference is frozen. New implementation belongs beside it, so a later correction cannot quietly rewrite the starting point.

Define the clean baseline

Preserve the 1024 × 768 composition, finish and face choices, expression colors, accessory photographs, captions, and configuration persistence. The baseline uses a dropdown for expression and one accessory choice.

Normalize the two teaching violations: Continue uses Ink with white text, 12px corners, and 48px height. Accessory cards have 24px gaps. Keep the purple Continue button and 8px accessory gaps as separately named fixtures. Color chips and independent accessory checkboxes remain follow-up exercises.

The package uses its own storage namespace. The package does not import the source owner's fixes, proposals, review state, or saved configuration. Reset changes only the sample key.

Name the fixture

Value	Default	Review fixture
Finish	Chalk	Graphite
Face	Friendly	Simple
Expression	White	Amber
Accessory	Dock	Carry bag

Use the non-default fixture in every review. A screen that always displays the defaults can appear correct while ignoring the shopper's choices.

CLAUDE DESIGN / PRESERVE THE CONFIGURATOR

Inspect the supplied Velori configurator and component library. Record the choices, asset composition, captions, storage behavior, and reusable exports. Preserve the existing identity and 1024 × 768 reference layout. Use the documented expression dropdown and one accessory choice. Describe the bounded change needed for Continue to open Review configuration, with edit and Back retaining the four

CLAUDE DESIGN / PRESERVE THE CONFIGURATOR

values. Do not add payment, stock, accounts, shipping, or order submission.

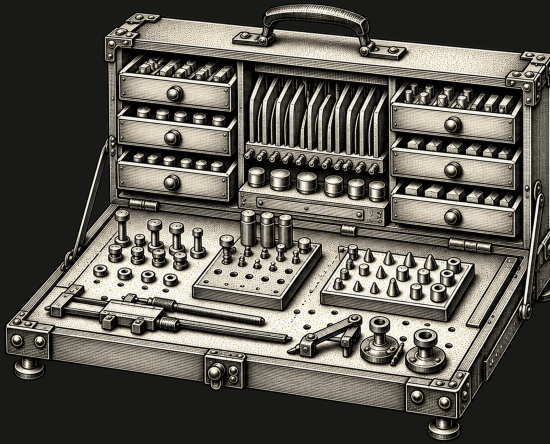
Chapter checkpoint

Record the source revision, working package, default and review fixtures, allowed changes, and preserved behavior. Keep any unresolved disagreement between code, tokens, and Figma in the review record.

Velori exercise and author synthesis. Primary setup references are listed in Further reading.

Prepare the Velori system

Define the source rules before importing the React library.



Prepare a reusable source system before asking Claude Design to generate a screen. A collection of working components can still leave type, spacing, layout, and usage rules undefined. Claude Code can help inspect and document those rules. The source preparation and checks below are this handbook's recommended procedure.

Velori uses the React codebase route described in Claude's [design-system setup guidance](#). `/design-sync` converts and imports that source. The asset-based setup route is an alternative for other inputs; the Velori exercise does not require both routes or manual entry of every design value.

Separate the starting screen from the answer key

Start an extracted copy using its README. Keep a second, unchanged copy for comparison. The public package contains a completed React review screen for comparison. It contains no historical native Code packet or native project pin. The answer key does not complete your new native exercise.

Supply the configurator, expression menu, accessory details, and unavailable-image configurator as the generation baseline. Use the 13 exports in `src/library.jsx` through `.design-sync/host-entry.mjs`. The application barrel `src/index.js` also exposes the completed Review screen and is not the import entry. Exclude `src/ReviewConfiguration.jsx`, completed answer-key guidance, historical native mappings, and `reference/` from the import and initial generation context. After the system passes Chapter 3, ask Design to create a new review screen, then compare it with the answer key.

Inspect the source package

Input	What it establishes
reference/index.html and reference/design-system.js	Frozen composition and documented consumer rules
src/library.jsx and src/index.js	Named components, props, controlled values, and interactions
tokens.json and src/tokens.css	Existing named colors, shapes, sizes, and font families
src/velori.css	Applied type, spacing, layout, and state rules, including literal values
/#specimens and reference/	Component states and screen comparators
public/assets/velori/ and its manifest	Product photos, masks, layers, captions, and asset identity
.design-sync/conventions.md and previews/	Import usage guidance and preview cases
Imported Figma working copy	Native anatomy, variants, variables, and font correspondence

Read docs/inventory.md and provenance/source-freeze.json before changing the source. Record the source revision and manifest hash. Keep the frozen reference unchanged. Use code for executable behavior, documented consumer rules for the normalized baseline, and Figma for native correspondence. Record disagreements before choosing a value.

Define rules and component use

Inventory every color role, type role, spacing value, radius, control size, layout dimension, and interaction state used by the starting screens. Include CSS literals: the token files alone do not define the complete system. For each rule, record its value, source path and selector or export, intended use, and whether it is observed, inferred, proposed, or unresolved.

Document each component's anatomy, valid props, states, composition constraints, and usage. A prop type can tell you that accessory accepts bag; usage guidance should also tell you that browsing bag details does not change the selected accessory. The app owns configuration and navigation. Controlled components receive values and callbacks.

Export	Responsibility
Brand, StoreHeader	Product identity and navigation shell
PrimaryButton, VeloriIcon	Reusable actions and icon roles
FinishChoice, FaceChoice	Controlled finish and face choices
ExpressionSelect	Dropdown, selected value, open state, and keyboard behavior
AccessoryChoice, ChoiceGroup	One accessory choice and grouped controls
SelectionMark	Selected-state marker inside a positioned choice tile
ProductPreview	Photo, finish, face, expression, caption, and recovery
AccessoryPhoto	Carry bag or Charging dock detail imagery
ConfigurationSummary	The four chosen values

Three finishes, two faces, four expression colors, and three accessory values produce 72 configurations. Preserve those values. Color chips and independent accessory checkboxes remain extension exercises.

CLAUDE CODE / PREPARE THE SOURCE SYSTEM

Work in this extracted Velori package. Inspect reference/index.html, reference/design-system.js, src/library.jsx, src/index.js, src/main.jsx for starting-screen copy and navigation, src/model.mjs for labels and captions, tokens.json, src/tokens.css, src/velori.css, docs/inventory.md, the component previews, the asset manifest, and the identified Figma working file. Prepare docs/system-definition.md and docs/source-conflicts.md before uploading. Inventory colors, typography, spacing, layout, component anatomy, valid props, states, and usage. Include CSS literals. Cite the source path and selector, export, or Figma identity for each rule. Separate directly observed

CLAUDE CODE / PREPARE THE SOURCE SYSTEM

rules, inferred compositions, proposed additions, and unresolved conflicts. Preserve existing names and consumer behavior. Explain which useful values need named tokens or usage guidance; make source changes only after the choices are settled. Preserve the frozen reference, product imagery, dropdown, one-accessory rule, and 72 configurations. Normalize Continue to Ink/white, 12px corners, and 48px height, with 24px accessory gaps. Keep purple Continue and 8px gaps only as named teaching fixtures. Record Arial/Georgia to Arimo/Lora mapping and weight or measurement differences. Project exact configure and accessory-details wording, caption construction, and details routing into docs/native-baseline.md. Map original source references to their imported bundle, CSS, and component guidance paths. Exclude ReviewConfiguration, completed Review implementation, and the historical handoff from the baseline system. Stop before upload if an unresolved decision affects the starting screens. Do not invent a component or silently accept a proposed pattern.

Review the two documents and any proposed source diff. Resolve required conflicts before importing. Project their accepted starting-screen rules into docs/native-baseline.md; include that document and the bounded product brief in the configured guidelines. Include exact configure and accessory-details copy, caption construction, and details routing, including the no-accessory case. Record both the original source identities and the paths actually present in the imported bundle. Page-level copy is not supplied by component prop types. Keep completed Review answers, historical mappings, and the frozen comparison source outside the uploaded context. Verify the actual converted guidance inventory, not only the local files. The React package uses Arial and Georgia; the native Figma system uses Arimo and Lora. Record the differing type weights and measurements. A font substitution does not establish pixel parity.

Open /#specimens and compare the prepared source with the frozen 1024 × 768 reference. Inspect default, selected, focus, open, loading, and unavailable states where applicable. Compare

`/?fixture=purple-continue` and `/?fixture=accessory-gap-8` with the accepted baseline. Run `npm run check`, `npm run build`, and the README's browser validation after any source change.

Import the prepared draft

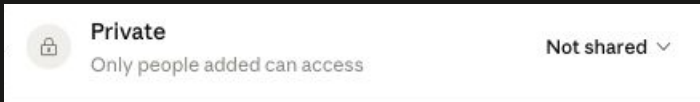
Open Claude Code in the extracted folder. Check the installed `/design-sync` and `/design-login` commands. Complete separate Design authorization when requested. In Claude Design, verify the intended account and organization in the switcher. Record whether you will create a new system or update an explicitly chosen system, and record its destination before uploading. The reader configuration has no owner project pinned.

This walkthrough uses the standalone `claude.ai/design` project surface. If that address redirects to conversation artifacts, use the visible **Visit standalone homepage** link before continuing. Inspect the returned system there before creating a project. Conversation `design-system` artifacts are a separate route outside this capstone; their inventory does not establish that a standalone system exists. If the installed skill cannot target the chosen surface, stop and resolve the destination before creating anything.

New-system creation may require its own approval. The inspected native tool reports that a new system initially uses organization-wide visibility. Check the intended visibility before creation and inspect the Share menu before uploading files. Stop if the selected destination cannot meet the agreed access boundary.

Figure 3
Design system destination



Figure 3 (continued)**Private before upload**

Claude Design, October 5 walkthrough. The example destination is Velori Handbook v1.2.0. Before upload, Share shows Private and Not shared.

Type `/design-sync yourself` at the Claude Code prompt and submit the native command. Asking Claude to run it in a pasted message does not invoke the command. After it opens, supply the import instruction below with the library, prepared tokens and CSS, conventions, previews, and assets. Preserve the local conversion report and the returned upload result. Inspect the returned system URL in the intended organization.

CLAUDE CODE / IMPORT THE PREPARED VELORI DRAFT

Import the prepared Velori source system into the recorded destination using the native command already opened. Use `.design-sync/config.json`, `src/library.jsx`, `src/tokens.css`, `src/velori.css`, `docs/native-baseline.md` derived from the reviewed system definition and resolved source-conflict decisions, conventions, previews, and packaged assets. Exclude the frozen reference, ReviewConfiguration, completed answer-key guidance, historical handoff and mappings, `node_modules`, validation outputs, and private owner state from upload. Confirm the account, organization, new or existing system, and destination before sending files. Check all 13 baseline exports and their usage guidance. Use the supplied declared source adapter so assets and their hash inventory exist before native validation and capture. Include `assets/**` in the upload scope, review obsolete asset paths, and verify the product files reached the recorded destination. Recapture asset-dependent previews whenever the asset inventory changes. Preserve the conversion report, file inventory, returned result, and system URL. Stop on a rejected upload or unresolved source conflict. Keep the system private; do not publish it or change the organization default.

The supplied source adapter copies the product assets during conversion, before the native driver validates and captures cards. It records their hashes in `guidelines/velori-runtime-assets.md`, which participates in the native upload fingerprint. `node .design-sync/package-assets.mjs ds-bundle` can repeat that same copy. Include `assets/**` in the approved upload scope and review obsolete asset paths before deletion. Verify the files in the returned inventory. The native skill's default path list omits this sample-specific directory.

An asset-only change can trigger upload without invalidating component grades. Recapture and visually inspect `ProductPreview` and `AccessoryPhoto` whenever that asset inventory changes. If a capture reports a missing SVG mask, inspect its response type: the mask must be served as `image/svg+xml`. Preserve and document a local `capture-helper` correction before repeating the native validation; do not replace the product image or accept a failure placeholder as a successful preview.

The import adapter resolves product assets relative to the compiled system script. Inspect those paths after conversion. A preview at the local origin root can work while the same code fails beneath a hosted project path.

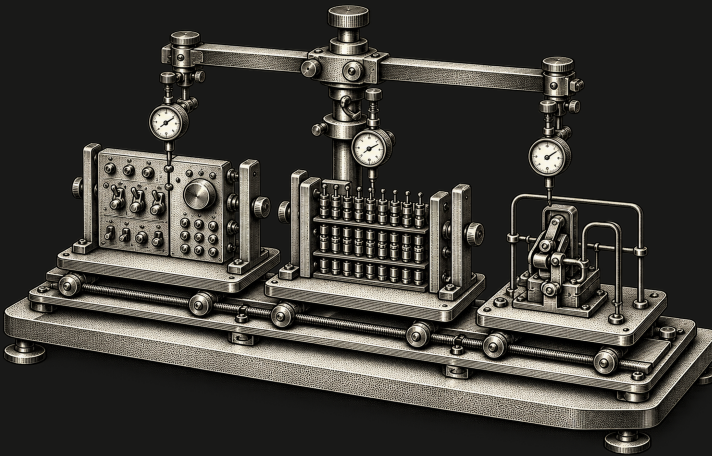
Chapter checkpoint

Keep the source definition, resolved conflict record, component usage, source diff, local checks, and actual import receipt together. Proceed to native inspection only when the destination matches and the accepted source describes the starting screens.

Velori exercise and author synthesis. Primary setup references are listed in [Further reading](#).

Validate the imported system

Prove source reuse and correction before extending the store.



Open the imported Velori system in Claude Design. Inspect its component cards, usage guidance, type and spacing rules, and photographs. Select that exact system in a new project. Recreate known screens before generating Review configuration.

Figure 4
Imported component inventory

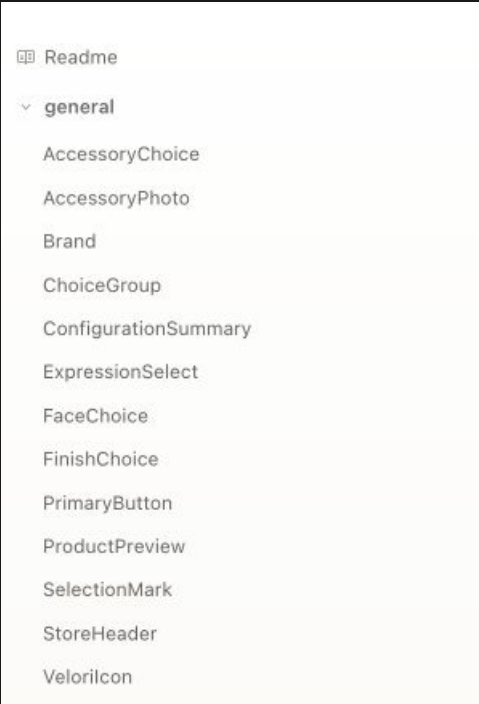
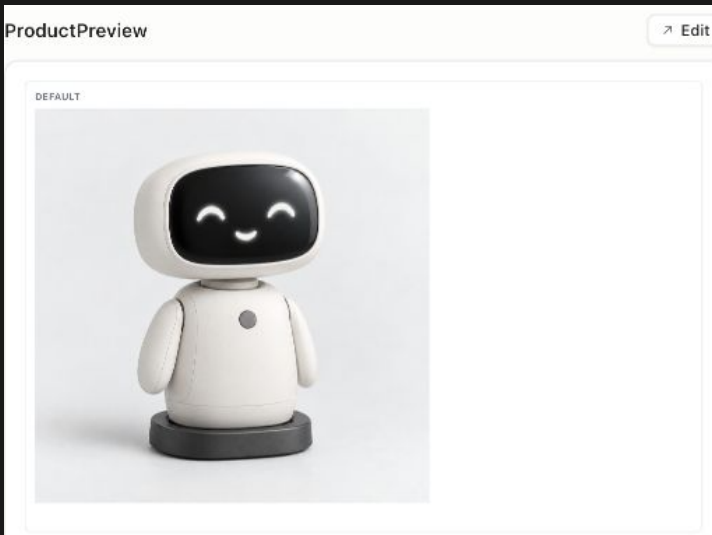


Figure 4 (continued)**Rendered ProductPreview****Opened usage guidance****Usage notes for Claude**

ProductPreview from velori-store-handbook-sample. Use via `window.Velori.ProductPreview` (bundle loaded from the root `_ds_bundle.js`).

Claude Design. Inspect the imported component inventory, rendered ProductPreview and opened usage guidance. The preview uses its own component example, separate from the Amber journey fixture.

Generate the known screens

Use the frozen reference and accepted source definition as comparators at 1024×768 . Generate the default configurator with Chalk, Friendly, White, and Dock, then change to Graphite, Simple, Amber, and Carry bag. Open the expression menu and both accessory-detail screens. Browsing Charging dock must leave Carry bag selected. Return to the configurator and inspect its caption and selected controls.

Render the unavailable-image configurator with the same four non-default values retained. Check that recovery preserves them. Keep

Review configuration out of these tests; it is the new composition introduced in Chapter 4.

CLAUDE DESIGN / RECREATE KNOWN VELORI SCREENS

Use the selected Velori system and accepted source definition. Create one responsive interactive store with one shared configuration and one mounted screen at a time. Use fixture controls or routes to inspect the required states while keeping component IDs unique. Preserve the supplied storage and focus behavior. Recreate the configurator at 1024 × 768 with Chalk, Friendly, White, and Dock, then Graphite, Simple, Amber, and Carry bag. Show the expression menu, Carry bag details, Charging dock details, and unavailable-image configurator. Use the actual named components, controlled props, tokens, and product assets. Keep selection, photograph, mask, caption, and labels consistent. Browsing another accessory must not change selection. Preserve dropdown keyboard behavior. Identify missing or unverifiable source rules before substituting them. Do not add Review configuration yet.

Inspect the actual interactive result. Check pointer and keyboard selection, menu cancellation, details return, asset loading, and image recovery. Save screenshots and the generated source or component references. A matching static photograph does not establish controlled state.

Sample ten elements for fidelity

Element	Source to inspect
Brand and navigation	Brand, StoreHeader, identity and icon rules
Page heading	Accepted type role and applied CSS
Continue	PrimaryButton, Ink/white, 12px radius, 48px height
Finish choice	FinishChoice anatomy and selected state
Face choice	FaceChoice anatomy and selected state
Expression dropdown	ExpressionSelect anatomy, menu, focus, and keyboard use
Accessory choice	AccessoryChoice, one value, 24px gaps
Product imagery	ProductPreview, original assets, masks, and composition
Caption	Configuration-derived copy and accepted caption type
Screen layout	Accepted columns, spacing, alignment, and narrow behavior

For each element, save its source identity or rule reference and classify the result as **exact reuse**, **valid composition**, **cosmetic resemblance**, **invented replacement**, or **unverifiable**. Exact reuse points to the actual source component or token. A valid composition assembles accepted components and rules without replacing them. Appearance alone cannot establish either classification.

CLAUDE DESIGN / AUDIT SOURCE CONFORMANCE

Audit these ten elements: brand/navigation, heading, Continue, finish, face, expression dropdown, single accessory choice, product imagery, caption, and layout. For each, cite the actual component/export and token or accepted CSS rule used, with its size, type, spacing, radius, and color where applicable. Classify it as exact reuse, valid composition, cosmetic resemblance, invented replacement, or unverifiable. Separate observed source evidence from inference. Flag altered props, substituted imagery, literal values that disagree with the definition, and missing states. Do not infer reuse from a screenshot. Stop before the review extension if any element lacks accepted source evidence.

Inspect the cited evidence yourself. Pass only exact reuse and valid composition, or a specifically reviewed exception with its reason recorded. Correct an invented replacement or unresolved rule at its source, rerun the import where necessary, and repeat the affected known-screen test.

Check that a correction persists

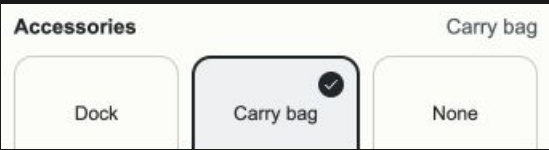
Use an isolated generated test screen and one named teaching fixture. Supply either purple Continue or 8px accessory gaps as the deliberate defect. Ask Design to restore the accepted rule. Keep the source system's accepted PrimaryButton and 24px gap rule unchanged; fixture styling must not become a new system variant.

CLAUDE DESIGN / CORRECT AND REGENERATE

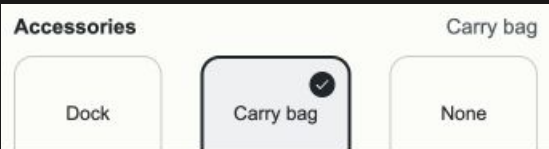
In this isolated test screen, correct the named purple-continue or accessory-gap-8 fixture using the accepted Velori source system. Continue must use PrimaryButton with Ink/white, 12px corners, and 48px height. Accessory choices must have 24px gaps. Identify the source rule used for the correction. Then generate a second configurator state with Pebble, Friendly, Mint, and no accessory. Confirm that the corrected rule persists there and that the fixture has not become an accepted variant. Record any unresolved replacement before proceeding.

Save the before, corrected, and second-generation results with the rule references. If the second generation repeats the defect, improve the source usage guidance or correct the generation context, resync when the source changes, and rerun the test. Keep the unsuccessful attempt in the review record.

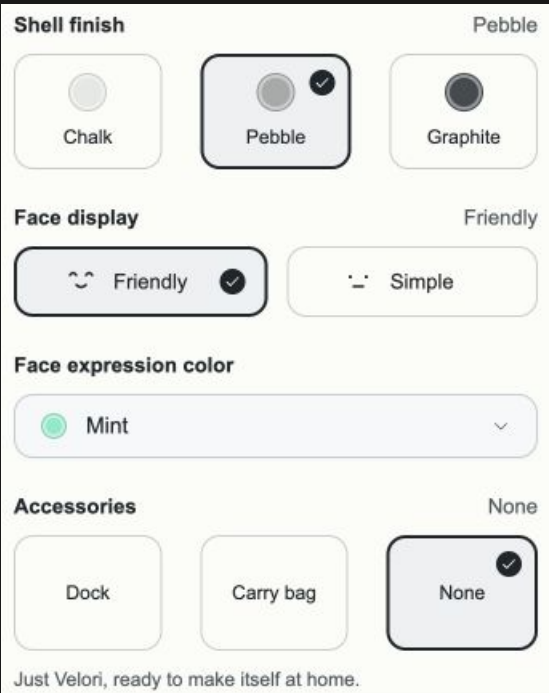
Figure 5
Deliberate 8px gap



Corrected 24px gap



Independent generation



Claude Design teaching fixture. The accessory gap changes from 8px to the accepted 24px. An independent generation retains the 24px rule with Pebble, Friendly, Mint and None; that screen omits storage and details.

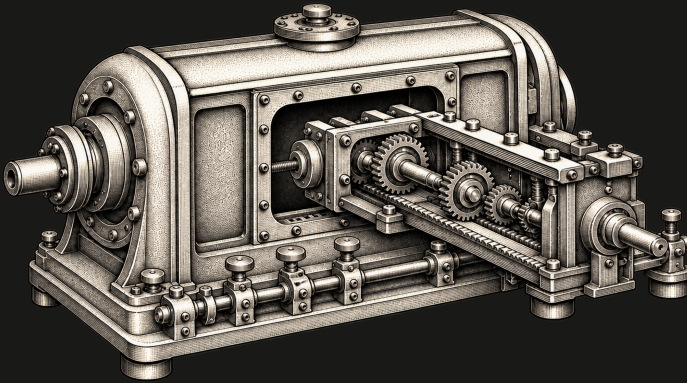
Chapter checkpoint

Proceed to Review configuration only after the actual selected system renders the known screens, the ten-element sample passes, and the correction survives another generation. Record accepted exceptions and font differences. Keep source completeness, conversion, upload, native rendering, and system publication as separate results.

Velori exercise and author synthesis. Primary setup references are listed in Further reading.

Extend the existing screen

Make Continue open one useful review step.



The shopper needs to check the chosen robot and accessories before returning to edit. Continue opens **Review configuration**. The review screen displays the chosen robot and all four values, with **Edit configuration** and **Back** returning to the configurator.

Generate the bounded change

CLAUDE DESIGN / ADD REVIEW CONFIGURATION

Use the selected Velori system and supplied configurator as the baseline. Make Continue open Review configuration. Show the chosen product photo and finish, face, expression color, and single accessory. Add Edit configuration and Back. Both return without losing values. Support pointer and keyboard navigation. Focus the Review heading when it opens; when Edit configuration, Back, or browser Back returns to the configurator, restore focus to Continue. Preserve reload persistence and an unavailable-image state that retains the summary. Reuse the system components and tokens. Identify the review screen as new sample behavior. Keep payment, inventory, accounts, shipping, and order submission outside the result.

Review the first diff

Inspect the current and proposed configurator beside each other. The extension should not replace the brand, photographs, choice vocabulary, dropdown, or one-accessory rule. The review column can change composition, but its values must come from the same configuration object.

Use Graphite, Simple, Amber, and Carry bag. A review screen with a hard-coded default photo fails even when its summary is correct. A correct photo with a stale summary also fails.

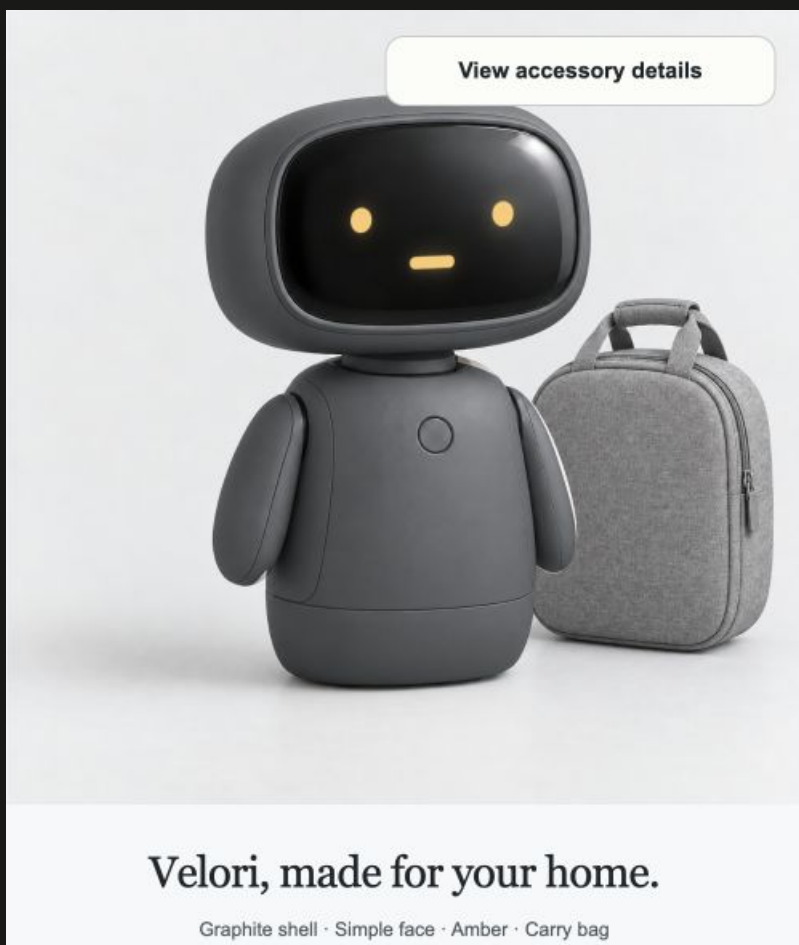
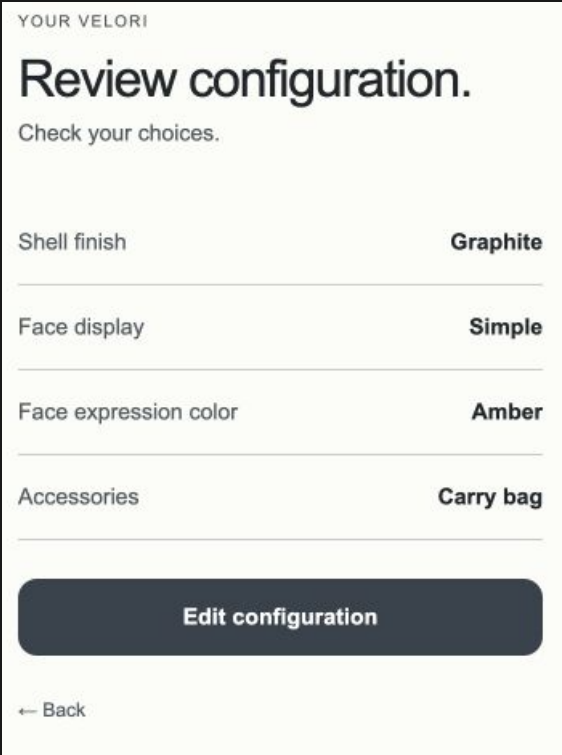
Figure 6**Native Configure photo and caption**

Figure 6 (continued)
Generated native Review



Claude Design, October 5 walkthrough. The generated Review retains Graphite, Simple, Amber and Carry bag from Configure, with Edit configuration and Back available.

Keep navigation explicit

Continue creates a review route. Edit and Back return to configure. Heading focus identifies the newly opened screen; returning restores focus to the control that led there. Browser history should behave predictably when the shopper moves through details and review.

Do not add an order action to fill the space. Review is the last step of this exercise, and editing is its useful next action.

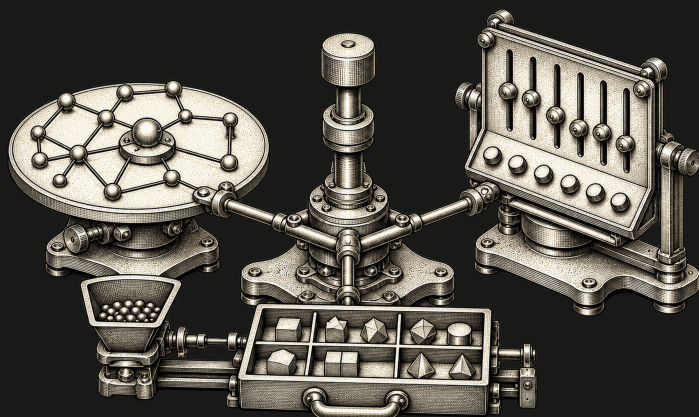
Chapter checkpoint

The generated result completes configure -> review -> edit with a non-default configuration. The selected system and new product composition remain identifiable.

Velori exercise and author synthesis. Primary setup references are listed in Further reading.

Connect the adjacent surfaces

Details, review, and editing keep the same choices.



Accessory details answer a product question without changing the configuration. Review answers whether the shopper has chosen the intended combination. Both depend on the configurator's existing values.

Browse without selecting

Open Carry bag details when Carry bag is selected. Browse Charging dock. The detail photograph and title should change, while the selected accessory stays Carry bag. Returning to configure restores the original selection and caption.

The detail view describes the accessory being browsed. The selected accessory remains in the parent configuration and appears unchanged when you return. A details tab must not silently commit a selection.

Figure 7
Browse Charging dock

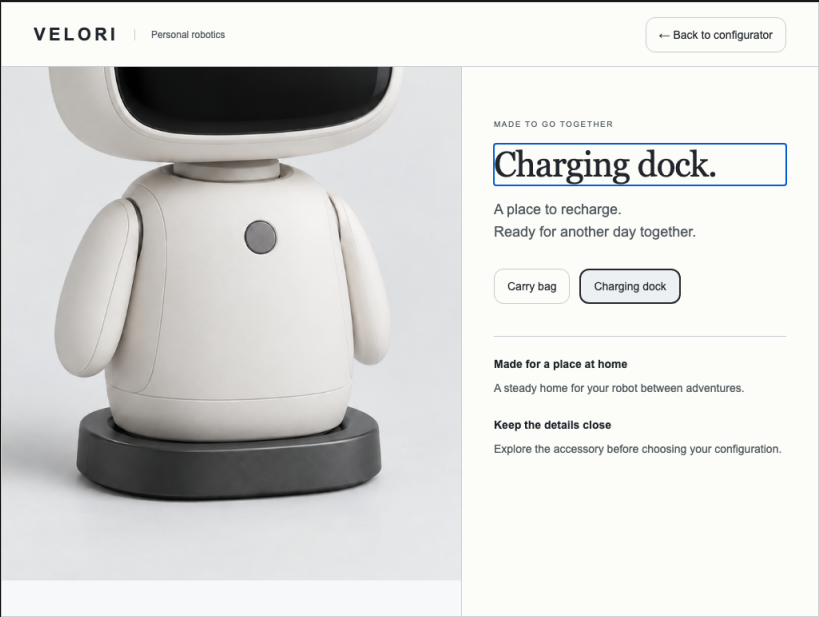
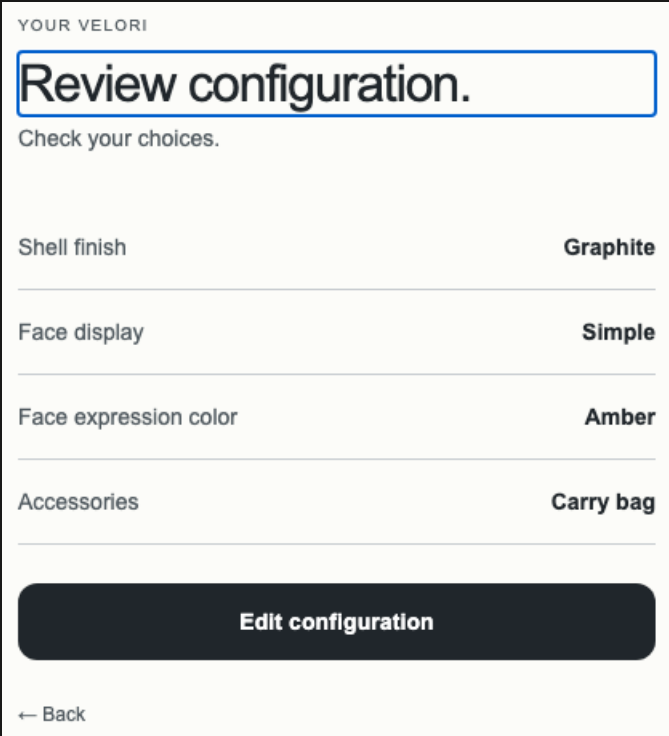


Figure 7 (continued)
Review after returning



Public React sample. Browsing Charging dock changes the detail view while Carry bag remains selected. After returning to Configure and choosing Continue, Review still lists Carry bag.

Make all surfaces agree

Surface	Read from shared configuration	Local view state
Configurator	Four values, photo, caption, marks	Open expression menu
Accessory details	Parent configuration retained unchanged	Accessory being browsed
Review	Photo and four summary values	Review route
Image recovery	Configuration and summary	Loading, failure, retry

A missing image changes rendering state. Image failure does not change the configuration. A details tab changes the browsed accessory. Browsing does not change the selected accessory.

CLAUDE DESIGN / AUDIT THE COMPLETE JOURNEY

Select Graphite, Simple, Amber, and Carry bag. Open accessory details, browse Charging dock, and return. Continue to review, then edit. Inspect photographs, captions, selected marks, and summaries after each transition. Keep Carry bag selected while browsing Dock. Show the unavailable-image state on review with the four values still visible. Report any surface that reads stale or independent values.

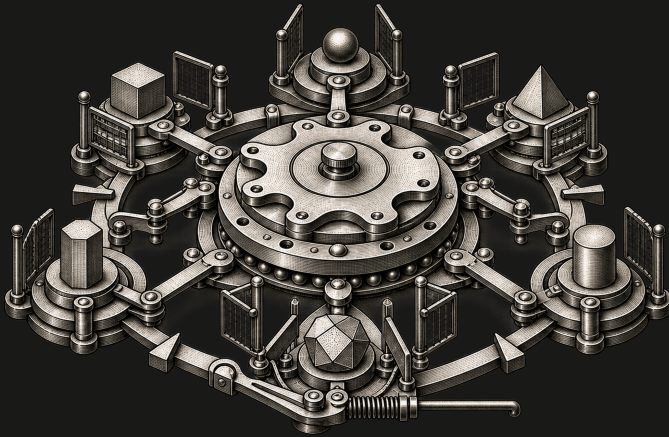
Chapter checkpoint

The full configure -> accessory details -> review -> edit journey retains the fixture. Each screen has a clear return path and owns only its view-specific state.

Velori exercise and author synthesis. Primary setup references are listed in Further reading.

Specify states and transitions

Define loading, failure, persistence, and recovery before handoff.



The normal journey is only part of the contract. Photographs load asynchronously. The shopper can change choices rapidly. Browser storage can be unavailable. Reload can occur on review.

State matrix

State	Visible behavior	Values retained	Recovery
Loading product	Loading treatment with current caption	All four	Wait or keep editing
Product ready	Correct photo, mask, caption	All four	Continue or edit
Image unavailable	Plain explanation and available summary	All four	Retry actual failed load
Simulated unavailable	Teaching placeholder	All four	Clear teaching simulation
Storage unavailable	Notice below the store	Current visit	Continue using the app
Reload with storage	Route and saved values restored	All four	Resume or edit
Reset	Default choices	Defaults only	Begin a fresh journey

Teaching controls belong outside the shopper's review column. They help demonstrate states without making reset or image simulation part of the product flow.

Transition contract

For every transition, name the trigger, values read, values changed, visible result, focus destination, and return path. For Continue, the trigger is the action, the four values stay unchanged, Review configuration opens, and its heading receives focus. Edit returns to configure and restores focus to Continue.

Avoid stale images

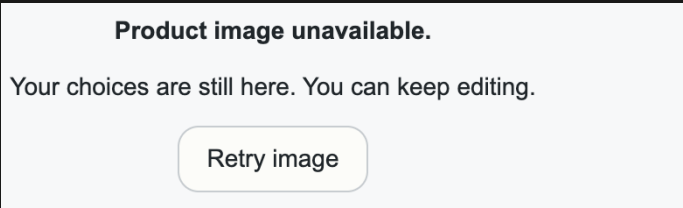
Rapidly change finish, face, expression, and accessory. The final photograph and caption must describe the last configuration. A previous image load must not replace the newer result. An unavailable image must retain its caption and summary while retrying.

CLAUDE CODE / VERIFY RECOVERY

Test rapid changes and a genuinely failed product-image request. Confirm the final image and caption agree, a failed image retains choices, and Retry reloads the asset after it becomes available. Test reload on review, blocked browser storage, and reset isolation. Keep unrelated storage values intact. Record actual results rather than adding untested behavior claims.

Figure 8

Image unavailable and Retry

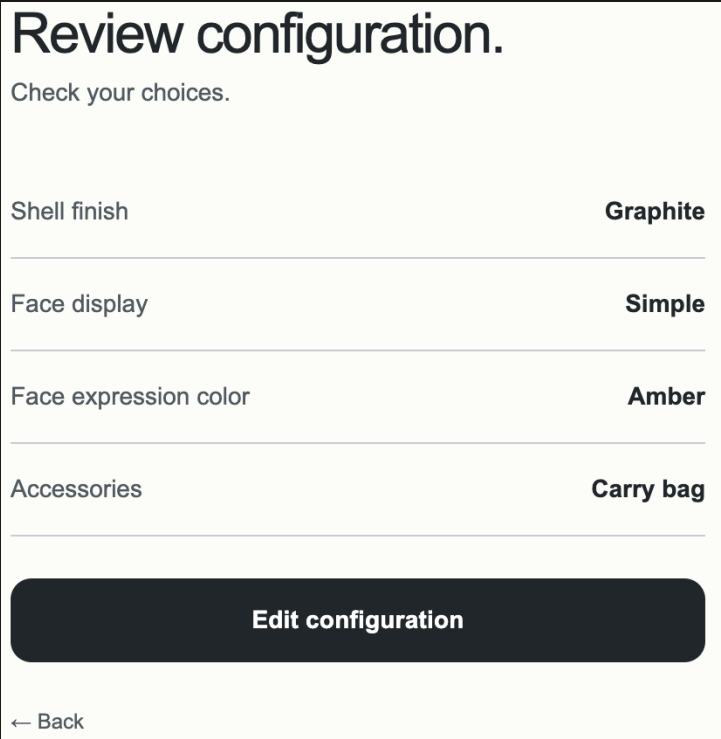


Retained caption during failure

Graphite shell · Simple face · Amber · Carry bag

Figure 8 (continued)**Recovered photograph**

Figure 8 (continued)
Retained Review summary



Public React sample. Blocking the actual required shell-image request produces the real unavailable state and Retry image action. After unblocking and retrying, the photograph returns and Review still lists Graphite, Simple, Amber and Carry bag.

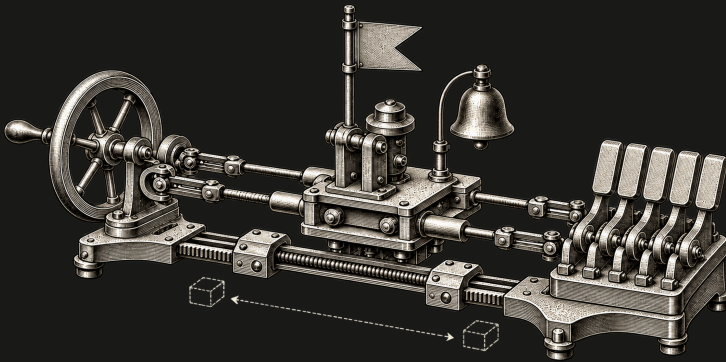
Chapter checkpoint

The handoff includes normal, loading, unavailable, rapid-change, reload, blocked-storage, and reset behavior. The configuration has one owner throughout.

Velori exercise and author synthesis. Primary setup references are listed in Further reading.

Prove keyboard and responsive behavior

Check the actual controls and return paths at the reference and narrow widths.



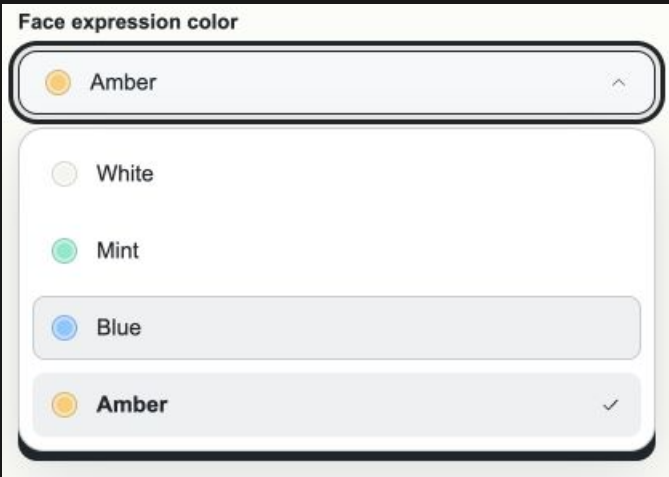
Test the real journey with the keyboard. Labels, focus movement, selection marks, and return paths must be observable. A promise of accessibility does not establish those behaviors.

Operate the dropdown

Tab to the expression trigger. Enter or Space opens the menu. Arrows move through options; Home and End reach the first and last. Enter commits. Escape cancels the pending change. Typing an option's name moves to it. Tab closes the menu, and the committed result returns focus to the trigger.

Figure 9

Keyboard-open expression menu



Public React sample. Amber is committed and checked while Blue is the active option in the keyboard-open menu. The real trigger focus outline remains visible.

Finish, face, and accessory choices respond to Enter or Space. A visible selected mark supplements their pressed state. Preserve the names and values when introducing a follow-up control pattern.

Check focus through the journey

- 1 Open details with the keyboard and confirm heading focus.
- 2 Return and confirm focus on the accessory-details control.
- 3 Activate Continue and confirm focus on Review configuration.
- 4 Activate Edit and confirm focus returns to Continue.
- 5 Use Browser Back and confirm the route and values remain coherent.

Automated focus checks support this contract. A screen-reader audit remains a separate check if the team requires a claim of complete assistive-technology support.

Compare viewports

At 1024 × 768, compare the product image bounds, composition, control column, and captions against the frozen reference. The normalized Continue and accessory gaps are intentional differences.

At 390px width, product and controls stack without horizontal overflow. Test the review and detail screens too. A dropdown or return action hidden outside the viewport fails the journey.

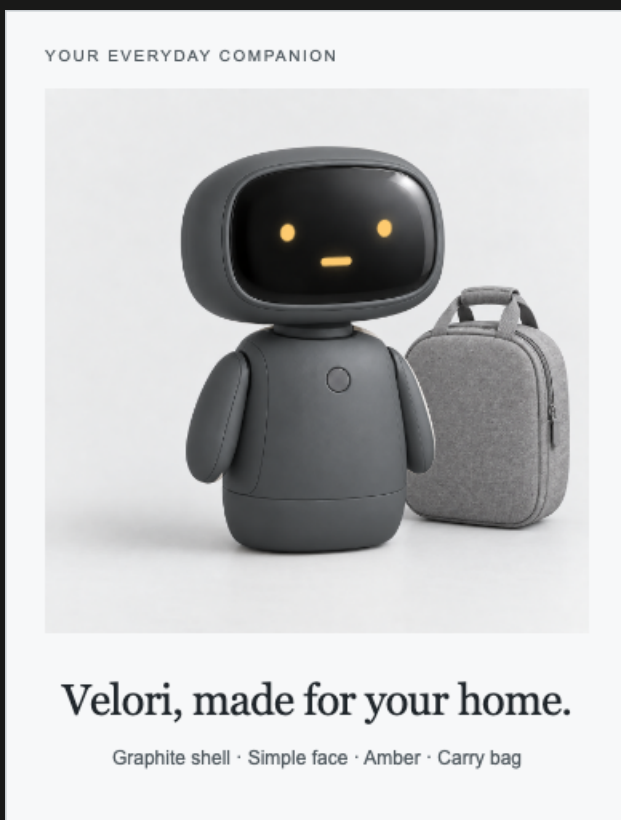
Figure 10**390px product and caption**

Figure 10 (continued)

390px summary

YOUR VELORI

Review configuration.

Check your choices.

Shell finish

Graphite

Face display

Simple

Face expression color

Amber

Accessories

Carry bag

390px Edit and Back

Edit configuration

← Back

Public React sample at 390px. The product and caption stack above the four-value Review summary. Edit configuration and Back remain reachable.

The consumer uses Arial and Georgia. The native Figma system uses Arimo and Lora. Record that mapping instead of presenting the renderers as typographically identical.

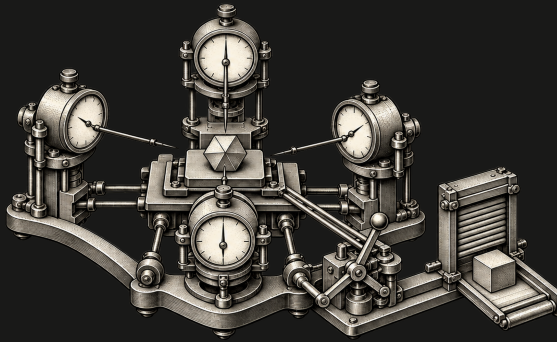
Chapter checkpoint

Keyboard choices, dropdown cancellation, heading and return focus, reference composition, and narrow layout have recorded results. Broader accessibility claims match the checks actually performed.

Velori exercise and author synthesis. Primary setup references are listed in Further reading.

Run the design evidence gate

Review component use, behavior, and handoff evidence together.



Review the extension against the brief, components, tested journey, and actual artifacts. Keep a small disposition table so an engineer can identify reuse and new behavior.

Element	Disposition	Governing source
Brand and store header	Reuse	Velori exports
Finish, face, expression, accessory	Reuse	Controlled component props
Product preview and recovery	Reuse	Asset and image-state contract
Review summary	Reuse	ConfigurationSummary
Review column	Product composition	Generated result and bounded brief
Navigation and persistence	Application behavior	Route and state contract
Native Figma review	Native composition	Linked components and variables

Challenge the result

Use non-default choices and ask whether each surface reads the same values. Inspect a real image failure. Reload at review. Check the browser history. Compare native component identity rather than relying on a screenshot.

CLAUDE DESIGN / REVIEW AGAINST EVIDENCE

Audit the Velori journey against the preserved baseline and bounded review brief. Name each reused component, product composition, changed behavior, and unresolved issue. Test the non-default fixture, detail browsing, review, edit, keyboard return, reload, and unavailable images. Keep conclusions tied to the rendered result and actual handoff. Do not infer interactive behavior from an artboard or PNG.

Record the decision

Use High, Medium, and Low for findings. A High issue blocks the intended journey or loses configuration. A Medium issue affects a required control, return path, or handoff. A Low issue is a local polish defect with a working journey.

Approve only the bounded result that was inspected. A successful local build does not establish native generation. An accepted upload does not establish rendered photos. A generated screen does not establish a portable export.

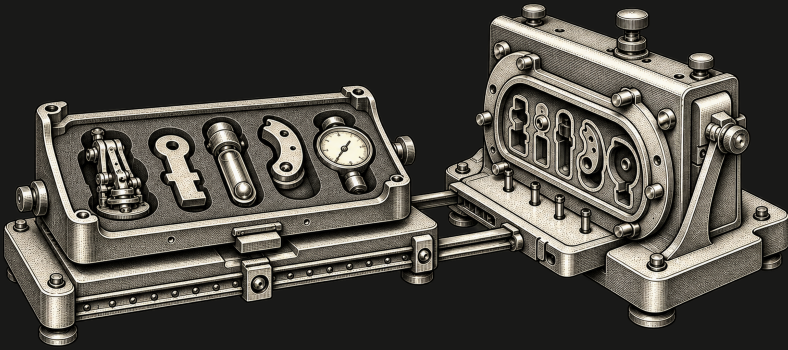
Chapter checkpoint

No unresolved issue blocks the handoff. The review record identifies the actual project, selected system, inspected journey, accepted findings, and remaining limits. Capture the files and record the implemented change in Chapter 09.

Velori exercise and author synthesis. Primary setup references are listed in Further reading.

Hand off to Claude Code

Preserve the actual source and make it run outside the design project.



Keep the generated source, dependencies, assets, accepted decisions, and expected journey together. Test the handoff from an extracted folder before calling it portable.

Capture your generated result

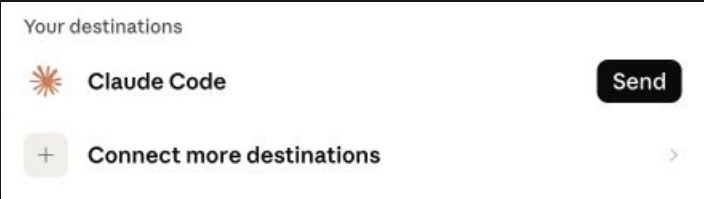
Open the new generated project in the intended organization. Choose **Share -> More formats and apps**. Under **Your destinations**, choose **Claude Code**, then **Send**. In **Hand off to Claude Code**, select **Local agent** and **Copy prompt**. Paste the copied prompt into Claude Code working in your extracted package.

The copied prompt identifies the project URL, selected generated HTML and imported files for the authenticated `claude_design` MCP. Complete `/design-login` in the intended account only if authorization is requested. Confirm the source and dependency reads succeed before starting adaptation. Save the copied prompt, actual returned files and read receipts unchanged in `handoff/new-run/captured/`, with project URL, selected file, handoff method, date, hashes and dependency origins in `handoff/new-run/HANDOFF.json`.

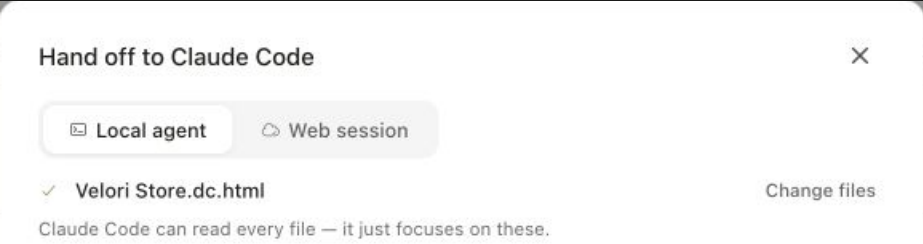
A binary read can return truncated data even when the request succeeds. Preserve that read receipt and obtain the full project archive through **Share -> Project HTML -> Project archive -> Export**. Use the archive as a companion to the copied handoff prompt and MCP reads. Preserve its original bytes and extracted entries unchanged, hash both, and compare complete overlapping files with the MCP reads. Stop until every required dependency has complete bytes; do not substitute assets from the sample or a historical packet. See [Design's export workflow](#).

Figure 11

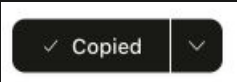
Send to Claude Code



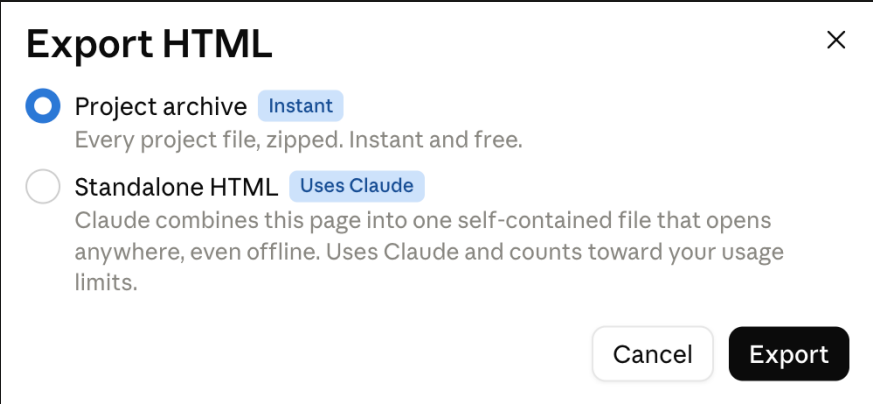
Local agent and selected file



Copied after the action



Project archive export



Claude Design. Send opens the Local agent handoff for the generated Velori Store file. Copied records the completed copy action; Project HTML provides the Project archive export path.

Ask Code to inspect the returned source before implementation. Include support.js, referenced system bundles and styles, runtime dependencies, and product assets. A copied prompt starts the import; successful file reads establish the capture. Preserve captured bytes and make any host adaptation beside them. Stop for unresolved failed reads or missing source or dependencies. The local packet must resolve its private project references before it can pass independent validation.

CLAUDE CODE / PREPARE THE NEW CODE PACKET

Use the copied native handoff prompt to read the recorded Velori project through the authenticated claude_design MCP. Confirm the intended account and complete /design-login only if authorization is requested. Read the selected generated HTML, support.js, every referenced system bundle and stylesheet, and all referenced assets. Preserve the copied prompt, returned files and read receipts unchanged in handoff/new-run/captured/, and record their hashes, project URL, selected file, handoff method and dependency origins in handoff/new-run/HANDOFF.json. Treat a truncated binary response or an incomplete decode as a failed read, and preserve its receipt. Obtain full bytes through Share -> Project HTML -> Project archive -> Export. Preserve the original archive and extracted entries unchanged, hash both, and compare complete overlapping files with the MCP reads. Do not fill missing files from the sample or a historical packet. Stop until every required source file and dependency has complete bytes. After capture is complete, create a separate runnable adaptation in handoff/new-run/ with package.json, lockfile, local assets, README, a dev script serving localhost:4206, and a validate script. Do not reuse the historical packet as the new handoff. Inspect the received runtime for globals, CDN URLs and resource mappings; provision required dependencies locally and document every URL or asset-route mapping separately from captured source. Validation must target this new packet and test all 72 configurations, configure -> details -> review -> edit, dropdown keyboard behavior, Back, reload persistence, rapid changes, unavailable-image recovery, and storage independence. Install Playwright as a development dependency for browser checks. Run without external HTTPS requests after installation. Preserve the sample application.

Review the packet's README and `handoff/new-run/HANDOFF.json`, then install and start it from that folder:

```
npm ci
npx playwright install chromium
npm run dev
```

Open <http://127.0.0.1:4206>. Leave that server running and run `npm run validate` in another terminal in the same folder. Keep the test output and screenshots. These scripts are outputs of the packet-preparation step; stop if they are missing or target the historical packet. Only a passing new packet establishes your new handoff.

Use the sample as an answer key

The public sample includes a bounded React Review configuration column adapted from the native Review source accepted on October 5, 2026, SHA-256 03b16ac0f4fea25202384a1f355badd71dcf70db20b0b976b68f0f46811150a. The existing app retains configuration, routing, persistence, focus, and image recovery. Run the sample from the extracted `velori-store` folder using the README. Captured Claude Design source and runtime remain private and outside the public downloads. Capture and validate your own new packet; the answer key does not verify your generation.

Keep the adaptations visible

If the received source needs a localhost adapter, inspect `support.js` and the selected system runtime for globals, CDN URLs, resource mappings and asset paths. Provision the required runtime dependencies locally and map the received system and asset routes beside the captured source. Preserve captured bytes and record hashes, dependency origins and every host adaptation in `handoff/new-run/HANDOFF.json`.

For MCP capture or a separately downloaded archive, inspect the full runtime and asset closure before treating the result as portable. Keep unsuccessful reads or downloads in the review record and document the working setup path.

Bring the bounded change into React

`src/ReviewConfiguration.jsx` adapts the generated review column. It reuses `ConfigurationSummary` and the primary action while the parent application retains routing, state, persistence, and focus. The shopper's column omits prototype inspection controls.

CLAUDE CODE / IMPLEMENT THE EXPORTED REVIEW

Inspect `handoff/new-run/captured/` and `HANDOFF.json` from the passing new packet. Adapt only the bounded Review configuration column into the React sample. Preserve named exports, tokens, photos, all four values, the expression dropdown, and one accessory choice. Keep routing, persistence, focus, and image recovery in their existing owners. Test `configure -> details -> review -> edit` with Graphite, Simple, Amber, and Carry bag. Record the source-to-implementation mapping and deviations. Do not edit `CLEAR` or `publish`.

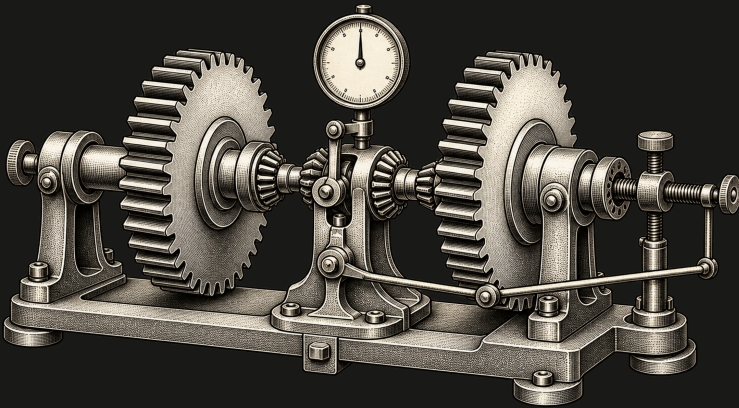
Chapter checkpoint

The packet runs outside the native project, and the implemented review can be traced to its actual source. Dependencies and host changes are disclosed. A PNG, PDF, or artboard is appearance evidence, not the interactive Code handoff.

Velori exercise and author synthesis. Primary setup references are listed in Further reading.

Implement in Figma and reconcile

Verify editable components, variables, and the native prototype.



Figma is part of this exercise. Confirm access to variables in prototypes on a paid or eligible Education plan. Use your imported working copy and keep editable frames for configure, the open expression menu, accessory details, review, and unavailable review. Preserve the supplied source pages and reference frames, and build the extension on a separate working page.

Keep native identity

Before editing, verify local main components, internal instance links, variable collections and modes, image resources, and available fonts in your imported copy. Stop if a required external library or resource is inaccessible. Use linked instances for the existing Velori components. Bind finish, face, expression, accessory, captions, and summaries to shared variables. Record the component and variable IDs from your actual file. A flattened screenshot or a collection of look-alike rectangles does not preserve native reuse.

Configure and Review compose the product photograph, face and expression from the four selected values. Accessory details use the original Chalk Carry bag and Charging dock reference photographs. AccessoryPhoto receives the browsed accessory only; browsing those views preserves the shopper's configuration. Preserve that source behavior when reconciling the native file.

Navigate by page and frame names from the companion manifest. Preserve **01 Foundations**, **02 Components**, **03 Configurator**, and **04 Source Assets**, including **Configure · White** on **03 Configurator**. The completed **05 Store · Handbook exercise** page contains Configure, Expression color menu, Carry bag details, Charging dock details, Review configuration, and Review · Image unavailable. Use those frames for comparison after your generation. Imported files have their own identities; record the IDs you inspect rather than relying on author-file node IDs.

FIGMA / BUILD AND INSPECT THE NATIVE STORE

Use the identified Velori working file. Preserve the original foundation, components, configurator, and source assets. Add editable native frames for configure, expression menu, Carry bag details, Charging dock details, review, and unavailable review. Reuse existing component instances and shared variables. Wire Graphite, Simple, Amber, and Carry bag through details, review, and edit. Confirm selected states, configured product images, captions, and summaries update. Keep the original Chalk reference photos in accessory details, matching AccessoryPhoto; browsing Dock must preserve the selected Carry bag. Record component and variable identities and test the actual prototype.

Test the prototype

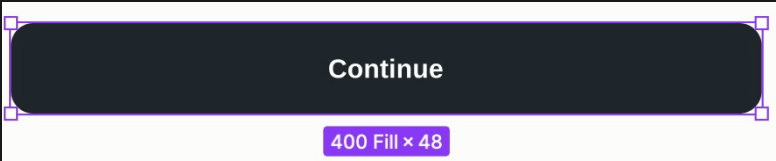
Select the non-default fixture. Follow both detail views, return, continue to review, show the unavailable image, and return to edit. Check the native values after every transition. Inspect representative bindings across all 72 combinations, not only the default frame.

The supplied prototype also includes keyboard shortcuts for its demonstration: C opens the expression menu, Escape dismisses it, Enter activates Continue or Edit, and I opens the unavailable-image review. Edit or Back returns to the configurator. These prototype shortcuts do not establish the browser dropdown's full keyboard model.

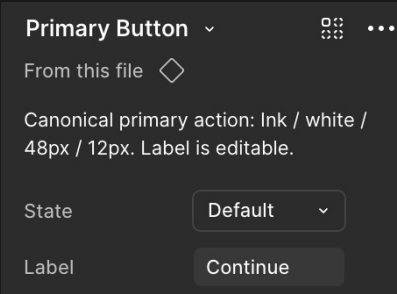
Native prototype variables support the configured visit. Browser reload persistence belongs to the React and Code apps and is tested there. Record that difference when reconciling behavior.

Figure 12

Selected Continue instance



Linked Primary Button



Bound height variable

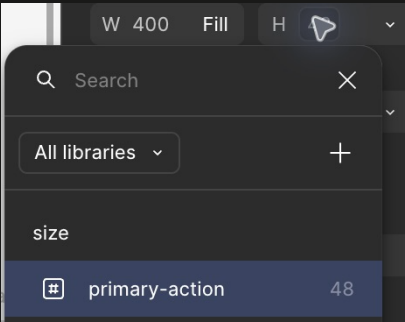
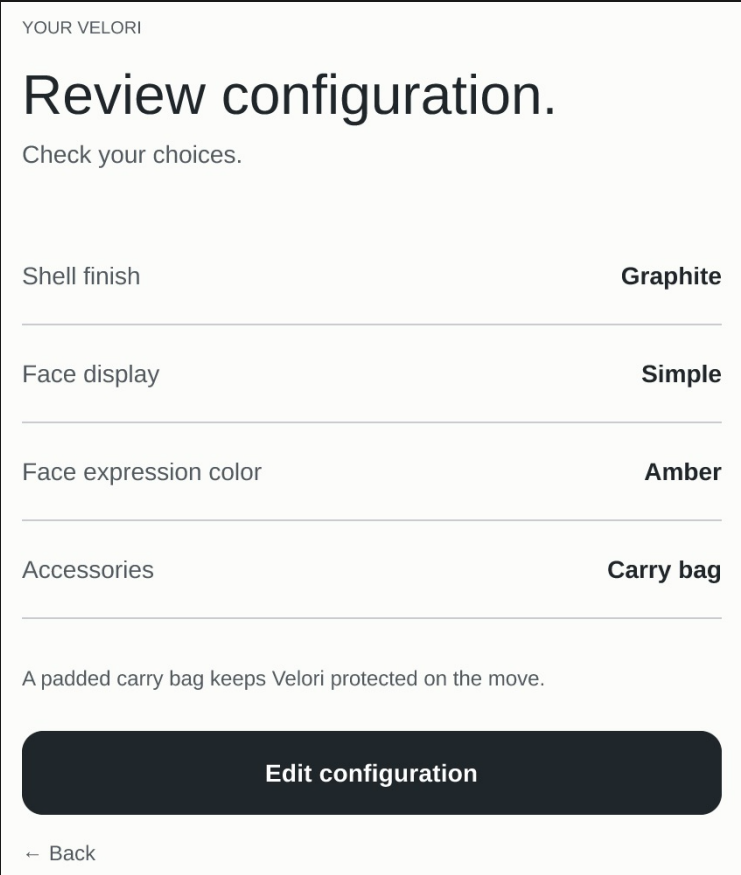


Figure 12 (continued)**Imported Figma product**

Figure 12 (continued)

Imported Figma Present Review



Imported Figma working copy. Inspect linked component identity and a representative variable binding, then compare Present Review with Graphite, Simple, Amber and Carry bag. Prototype variables cover the configured visit.

Reconcile the result

Compare the reference and extracted screens at 1024 × 768. Record the font mapping and intentional normalized controls. Resolve unexpected drift in the owning source, then repeat the affected check. Keep rendering differences and product behavior differences distinct.

The [Figma write-to-canvas guidance](#) describes native editing. Do not treat a Claude image export as evidence that component and variable identities survived in Figma.

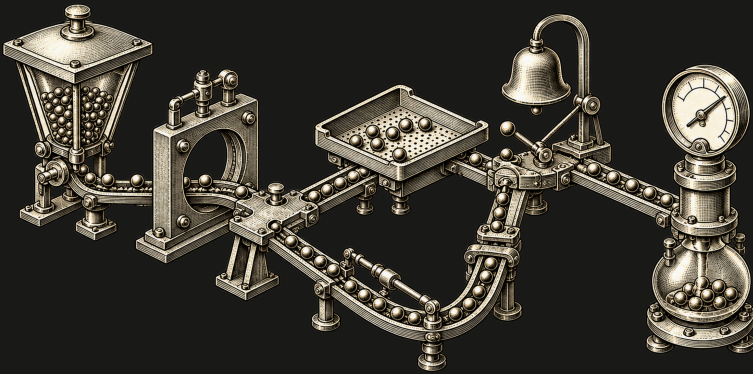
Chapter checkpoint

Native editable frames, instance identities, variable bindings, and prototype transitions have returned evidence. The browser and Figma behavior differences are explicit.

Velori exercise and author synthesis. Primary setup references are listed in Further reading.

Guided capstone

Complete one bounded store journey across the real tools.



Work from an unchanged extracted copy. Keep the completed sample as the expected result. The capstone recreates Review configuration from the existing Velori configurator and details, then implements and verifies it through the native tools.

Phase 1: Frame and ground

Start the package, inspect the frozen reference, record the default and review fixtures, and confirm the preservation boundary. Identify your Claude account, organization, system destination, and Figma working copy before uploading.

Phase 2: Define, import, and validate

Follow Chapters 2 and 3 in order. Prepare the source definition and conflict record, including CSS literals, usage guidance, and Figma font correspondence. Resolve required decisions, run local checks, and import the prepared draft into the recorded destination. Inspect actual component cards and recreate the known configurator, menu, details, and unavailable-image screens. Save the ten-element fidelity sample and correction-persistence test. Stop for an unresolved source rule or a failed native check.

Phase 3: Generate the extension

After the system passes, create Review configuration using the configurator and details as the baseline. Exclude the completed review answer key from generation context. Test the non-default fixture, return paths, keyboard use, reload, and unavailable imagery.

Phase 4: Capture and implement

Preserve the actual generated source or handoff, including its dependencies. Run it in a new folder. Adapt the review column into the React sample without replacing state ownership or existing controls. Keep unsuccessful exports in the review record.

Phase 5: Build the native Figma journey

Use linked native components and shared variables. Add the editable frames beside the reference. Test configure -> details -> review -> edit with Graphite, Simple, Amber, and Carry bag. Confirm that configure/review captions and configured images update together. Details retain the original Chalk accessory-reference photographs.

Phase 6: Verify the reader package

```
npm run check
npm run build
npx playwright install chromium
npm run dev
```

Leave Vite running. In another terminal:

```
npm run validate:browser
```

Validate the newly received packet in a separate folder as described in Chapter 9. Run `npm run validate` only after the new packet has the captured inputs and the documented validation script. The public sample has no historical handoff validator; its local app checks cannot certify a new native export.

Check all 72 combinations, rapid changes, dropdown keyboard behavior, reload, real image failure, reset isolation, reference composition, and narrow layout. Preserve the test output and

screenshots. Keep a fresh extracted copy for the final install and journey check.

Completion record

Result	Required evidence
Local app	Clean install, build, complete journey, 72 configurations
Claude Design	Source definition, resolved conflicts, intended destination, accepted import, known screens, fidelity sample, correction persistence, new review generation
Code handoff	Actual source and dependencies run independently; bounded change implemented
Figma	Working file, editable frames, identities, bindings, native journey
Handbook candidate	Reader instructions match the package and tested behavior
Publication	Separate release authorization and established redistribution terms

Answer key

A passing result preserves the store and adds one review step. The photograph, controls, caption, and summary agree. Browsing details does not select an accessory. Edit, Back, reload, and image recovery preserve the four values. The dropdown keeps its documented keyboard behavior.

The native result uses the selected Velori system. Its handoff includes executable source and dependencies. The React adaptation composes existing exports. The Figma result uses linked components and shared variables, with the actual prototype tested. The package runs without CLEAR or private owner state.

Do not award completion for screenshots alone, a successful upload without inspection, an export that cannot start, or local tests substituted

for an inaccessible native step. Leave an inaccessible required step unfinished and name the practical next action in the review record.

Velori exercise and author synthesis. Primary setup references are listed in Further reading.

Appendix A: Prompt sequence

Use the prompts in order: preserve the configurator; prepare the source system; import the prepared draft; recreate known screens; audit ten elements for source conformance; correct a named fixture and regenerate; add Review configuration; audit the complete journey; check recovery; review against evidence; prepare and validate the new Code packet; implement the captured change; reconcile native Figma. The package's docs/prompts.md supplies copyable versions.

Appendix B: Handoff record

Record the source definition and conflict decisions, ten-element fidelity sample, correction and second-generation evidence, feature, account and organization, selected system and project URL, Figma working file, source revision, captured files and hashes, host adaptations, component map, route and state inventory, accepted copy, expected journey, keyboard behavior, validation results, deviations, and open decisions. Use docs/handoff-template.md.

Keep credentials and authorization material out of the reader package. Reader configuration should name a destination only after the reader chooses one.

Appendix C: Package map

Path	Contents
README.md	First-run, journey, validation, and reset instructions
src/	App, reusable library, model, tokens, and review adaptation
public/assets/velori/	Product photographs, masks, layers, and manifest
reference/	Frozen consumer reference and extraction manifest
.design-sync/	Reader import configuration, adapter, conventions, previews
docs/	Brief, baseline import rules, source definition, decisions, prompts, walkthrough, expected results, notices
provenance/	Frozen consumer-source identity; private native records are excluded
provenance.json	Package file hashes and exclusions
scripts/	Asset, browser, and public-package checks; model tests live in src/

The [native Figma companion](#) and its [manifest](#) are separate downloads beside the React ZIP. Import the file into your own Drafts; it is not bundled in the ZIP.

Data Pipeline Studio remains the historical exercise in `Local_Coding_Exercise.md`. Its released editions and artifacts are preserved. Velori is the current sample for this guide.

Further reading

- 1 Anthropic. [Get started with Claude Design](#). Reviewed October 4, 2026.
- 2 Anthropic. [Set up your design system in Claude Design](#). Reviewed October 4, 2026.
- 3 Anthropic. [Claude Code commands](#). Reviewed October 4, 2026.
- 4 Anthropic. [Design admin guide](#). Use for current organization setup.
- 5 Figma. [Write to canvas](#). Reviewed October 4, 2026.
- 6 Velori reader materials. The React ZIP contains MIT source and assets, frozen reference, token and asset manifests, exercise materials, and validation scripts. The separate native Figma file and companion manifest supply a reader-owned working copy. Completed React and Figma screens are comparison material. Captured Claude Design packets and metadata stay private. Verify each new native run separately.

Changelog

Compared with version **1.2.1**, published **October 5, 2026**. Previous Daytime PDF SHA-256: 43302892dd7a961dba1154027a8a42da72e716b21cb7534e9f9cf2d7a52b2272.

Addition: Orientation and Chapters 2-7, 9 and 10. Added screenshots for setup, system inspection, correction, the store journey, recovery, keyboard use, responsive layout, Code handoff, and Figma review. Readers can locate the controls and compare their results with the worked example.

[Velori React sample](#), [native Figma companion](#), and [Claude Design workflow](#).