DateFix Demo

How It Was Built — A Complete Breakdown

For Emily's Portfolio & Learning Reference

This document walks through every step of building the DateFix interactive product demo — from generating the initial images with AI, to extracting and masking assets in Photoshop, to assembling everything into an interactive web experience with Claude Code. It covers the full creative and technical pipeline so you can understand (and reproduce) the process.

The full DateFix product lineup — Original, Ginger, Turmeric, and Cinnamon

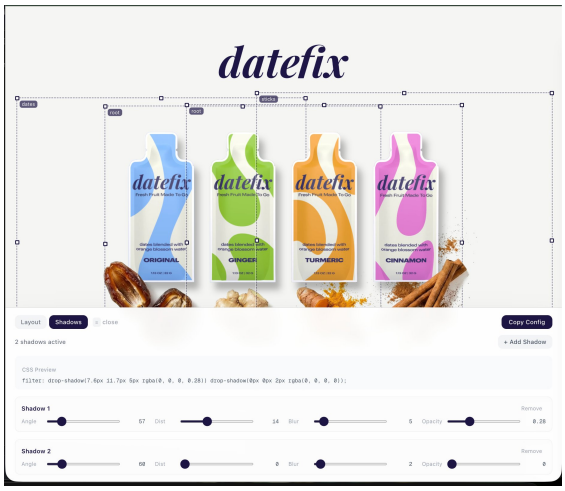
What's Inside

1. Overview

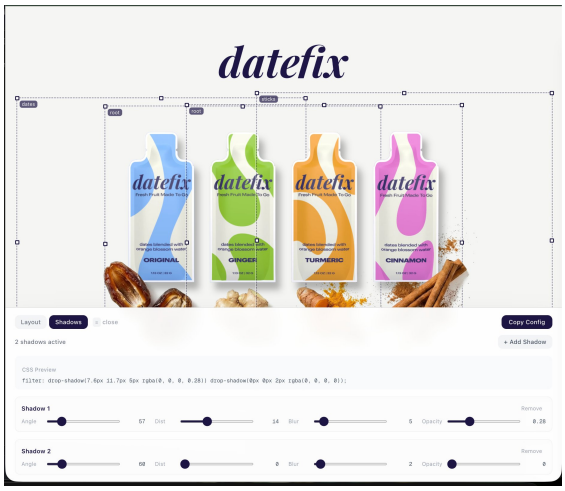
The DateFix Demo is an interactive web page that showcases the DateFix product line — a series of date-based fruit gel pouches in four flavors: Original, Ginger, Turmeric, and Cinnamon. The tagline is **"Fresh Fruit Made To Go"** and each pouch contains dates blended with orange blossom water.

The goal was to create a polished, portfolio-quality brand demo that shows off all four flavors with real product photography aesthetics, interactive hover states, and informational popups — all built entirely from AI-generated imagery and assembled through a combination of Photoshop, code, and Claude AI assistance.

The Four Flavors



Original (Blue)



Ginger (Green)

2. Image Generation

The entire project started with AI image generation using **Nano Banana**. This is the tool that was used to create the original product photography-style images of the DateFix pouches.

Initial Generation — Portrait (1080×1920)

The first generation was done at a **1080 × 1920 portrait** orientation. This produced a tall image showing all four DateFix pouches lined up with their corresponding ingredients (dates, ginger root, turmeric root, and cinnamon sticks) arranged below each packet. The portrait orientation was useful as a starting reference but wasn't ideal for the final web layout.

Original portrait generation (1080×1920) — used as the initial source

Re-generation — Landscape / Horizontal

The image was then **regenerated in horizontal / landscape orientation** to better suit a web UI layout. This wider composition became the primary base image for extracting all the individual assets. It gave more breathing room between each packet and ingredient, making the extraction process cleaner.

Horizontal re-generation — this became the base image for all asset extraction

3. Asset Extraction in Photoshop

This is where the bulk of the manual work happened. Each element — every packet and every ingredient — needed to be isolated on its own transparent layer so it could be positioned independently in the web demo.

Masking Techniques Used

1. Manual Selection

For areas where the edges were clean and well-defined, manual selection tools (like the Pen tool or Lasso tool) were used to cut elements out. This is the most precise but also most time-consuming approach.

2. Select Color Range

Photoshop's **Select** → **Color Range** tool was used to quickly select the white/light background behind the packets and ingredients. This tool lets you click on a color in the image and it selects all similar colors across the canvas. By adjusting the Fuzziness slider, you control how much of a range of colors gets selected. This was especially effective because the product shots had clean white backgrounds.

3. Blend If (Blending Options)

This was the key technique for getting **perfect masks**. In Photoshop's **Layer Style** → **Blending Options**, there are "Blend If" sliders for both the current layer and the underlying layer. By adjusting the **"This Layer"** and **"Underlying Layer"** white point sliders, you can make white pixels disappear without destructively editing the image.

The technique: Hold **Alt/Option** and drag one half of the white slider to split it, creating a smooth feathered transition instead of a hard cutoff. This blends the white background away seamlessly. Combined with Select Color Range, this gave near-perfect cutouts with minimal manual cleanup.

What Was Extracted

Asset	Description	Notes
Packets (×4)	Each flavor pouch on transparent bg	Original, Ginger, Turmeric, Cinnamon
Dates	Two split dates (for Original)	Bottom layer + top overlap layer
Ginger	Ginger root (for Ginger)	Single layer, no overlap

Turmeric	Turmeric root with powder (for Turmeric)	Single layer, no overlap
Cinnamon	Cinnamon sticks with powder (for Cinnamon)	Bottom layer + top overlap layer

Extracted Ingredient Assets

Here are examples of the extracted ingredient layers. Notice they're on black/transparent backgrounds — in the actual project files these are PNGs with transparent backgrounds:

Dates (bottom layer)

Ginger root

Turmeric root

Cinnamon sticks

4. Asset Preparation

Consistent Canvas Sizing

This step is **critical** and easy to overlook. Every single asset — all four packets and all the ingredient layers — needed to be exported at the **exact same canvas size and aspect ratio**.

Why? Because in the web demo, each image is positioned and scaled relative to a common coordinate system. If one packet image is 500×800 and another is 600×900, they won't align properly even if you scale them to the same display size — the content will be offset within the canvas. By keeping all canvases identical, when you overlay them in CSS at the same position, everything lines up pixel-perfectly.

The Overlap Layer Trick

For the **Dates** and **Cinnamon** flavors, the ingredient partially overlaps the packet in the original photo. To recreate this depth in the web demo, the ingredient was split into **two separate layers**:

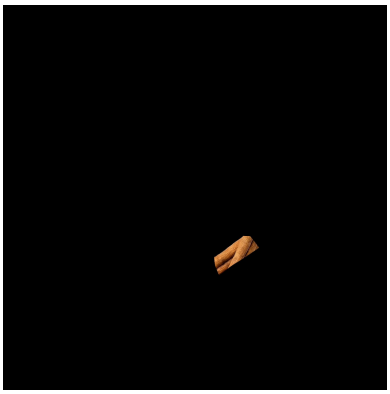
Bottom layer: The full ingredient, positioned *behind* the packet in CSS (lower z-index). This is the main body of the dates/cinnamon that sits below the pouch.

Top layer: Just the overlapping portion of the ingredient, cut out and placed on its own layer, positioned *in front of* the packet in CSS (higher z-index). This creates the illusion that the ingredient wraps around the packet — part behind, part in front.

> *Both layers are the same canvas size and aligned to the same position, so when they're stacked in the browser with the packet sandwiched between them, the depth illusion is seamless.*

Top Overlap Layers

Top date layer — just the part overlapping the packet



Top cinnamon layer — just the overlapping cinnamon sticks

5. Project Setup

With all assets prepared, it was time to set up the actual web project.

Folder Structure & GitHub

A new folder called **datefix-demo** was created and connected to a GitHub repository. All the extracted PNG assets were placed inside an assets folder within the project. This makes them accessible to the web code and also version-controlled.

Briefing Claude Code

The project was built with **Claude Code** (Anthropic's AI coding assistant). The process started by giving Claude a detailed brief of what the demo should look like and how it should function. This included uploading all the assets and describing the desired layout, interactions, and visual style.

A planning conversation was used to define the project scope (see the ROADMAP.md that was generated). The key was being specific about the visual goals — referencing the original generated images and explaining exactly how the packets and ingredients should be layered.

Brand Assets Collected

Several additional brand elements were gathered to make the demo feel authentic:

Logo: A high-resolution screenshot of the DateFix logo was taken directly from the official site, then cropped and cleaned up for use in the demo header.

Brand Colors: Each packet's signature color was sampled using the eyedropper tool, giving exact hex values for the blue (Original), green (Ginger), orange (Turmeric), and pink (Cinnamon). These colors were used throughout the UI for hover states, shadows, and flavor popups.

Flavor	Color	Used For
Original	Light Blue (#89B4D4)	Packet accent, shadow, popup
Ginger	Green (#7CB342)	Packet accent, shadow, popup
Turmeric	Orange (#F09819)	Packet accent, shadow, popup
Cinnamon	Pink (#E879A8)	Packet accent, shadow, popup

6. Building the UI

This was the longest phase — taking all the individual assets and assembling them into a cohesive, interactive web page. Here's how it was approached:

Layer Structure in CSS

The key architectural decision was recreating Photoshop's layer system in the browser. Each flavor "slot" consists of multiple stacked elements:

1. Color overlay layer (furthest back) — A duplicate of the packet image with a full color overlay matching the flavor's brand color. This sits behind the actual packet and becomes visible as a glow/shadow effect during hover interactions.

2. Bottom ingredient layer — The main ingredient image (dates, ginger, turmeric, or cinnamon) positioned behind the packet.

3. Packet layer — The actual product pouch image.

4. Top ingredient layer (only for dates & cinnamon) — The overlap portion that sits in front of the packet, completing the depth illusion.

Alignment Challenges

Getting everything to line up correctly was one of the most time-consuming parts of the build. Because each asset is a separate image file being positioned with CSS, even small misalignments between the packet and its ingredient layers become immediately visible.

The initial approach of just setting CSS positions manually was too imprecise. This led to building custom debug tools (covered in Section 7) to visually adjust everything in real-time.

Reference Image Overlay

A clever technique was used for alignment: the original horizontal AI-generated image (showing all four packets together) was placed as a **semi-transparent overlay** on the web page. By reducing its opacity (to around 30%), it served as a visual guide — like tracing paper. The individual packet and ingredient layers could then be adjusted until they matched the positions in the reference image exactly.

> *The debug menu included controls for toggling this reference image on/off and adjusting its opacity, size, and vertical position.*

7. The Custom Debug Menu

One of the most interesting parts of this build was the custom debug interface that was created specifically for this project. It's accessible by pressing the **equals key (=)** on the keyboard when viewing the DateFix demo site.

The custom debug menu — showing layout controls, shadow settings, and per-flavor adjustments

What the Debug Menu Contains

Layout Tab

Controls for the overall layout of all four packets together. This includes the gap between packets and global positioning offsets. These let you adjust how close together or spread apart the four flavors are displayed.

Reference Image Controls

A checkbox to show/hide the reference overlay image, with sliders for its opacity, size (scale), and vertical offset (Ref Y). This was essential during the alignment phase — you'd turn on the reference, lower the opacity, and then tweak each element until it matched.

Per-Flavor Controls

Each flavor (Original, Ginger, Turmeric, Cinnamon) has its own set of sliders with a color-coded dot indicator. For each flavor, you can adjust the **Scale** (size of the ingredient), **X position** (horizontal offset), and **Y position** (vertical offset) of the ingredient layer relative to the packet.

Shadows Tab

This was modeled after Photoshop's Drop Shadow dialog. It lets you visually create and fine-tune drop shadows for the packets — adjusting properties like offset, blur, spread, and color — all in real-time on the live site. This is much faster than writing CSS box-shadow values by hand and refreshing the page repeatedly.

Copy Config Button

Once the shadows and positioning looked right visually, the **"Copy Config"** button in the top-right corner would copy all the current configuration values to the clipboard as a structured object. This configuration would then be fed back into Claude Code to hard-code the final values into the site's source code. This workflow — visual tweaking → copy config → paste into code — was the core iteration loop for getting everything pixel-perfect.

8. Photoshop-Style Transform Controls

At a certain point, the slider-based debug menu wasn't precise enough for fine-tuning. So **Photoshop-style transform controls** were built directly into the web page.

These are the familiar bounding-box handles you see in Photoshop when you select a layer — small squares at the corners and edges that you can drag to resize, and the ability to click and drag the element to reposition it.

How They Worked

Each packet and ingredient layer got its own set of transform handles. You could click on any element to select it (showing its bounding box), then drag the corner handles to scale it up or down, or drag the center to reposition it. This made it dramatically faster to get precise placement, especially for the ingredient layers that needed to sit at very specific positions relative to their packets.

Like the shadow controls, these transforms were also exportable — the final positions and scales could be copied out and hard-coded into the production version of the site.

> *This is a great example of building custom tooling to solve a specific problem. Rather than fighting with CSS values in code, you build a visual tool that lets you work the way you naturally think (dragging and resizing), then export the result back into code.*

9. Interactivity

Hover Effect — Packet Rise

When you hover over any of the four packets, it **rises up** with a smooth CSS transition. This is a simple but effective interaction that gives the product cards a tactile, app-like feel. The transform is applied to the entire packet group (packet image + color overlay + ingredient layers) so everything moves together.

Color Overlay Glow

Behind each packet sits a **color overlay duplicate** — a copy of the packet image that's been tinted entirely in the flavor's brand color (blue for Original, green for Ginger, etc.). This acts as a colored "glow" or shadow behind the packet. On hover, this can become more visible, adding to the lifted, glowing effect.

Flavor Popup / Info Card

When you **click** on a packet, a small informational popup appears. This popup displays:

- **The flavor name** (e.g., "Cinnamon")
- **The product description** (e.g., "dates blended with orange blossom water")
- **A brief flavor description** — a short sentence about each specific variety

These popups are styled to match the brand aesthetic and provide just enough information to give context without overwhelming the visual presentation. It adds a layer of interactivity that makes the demo feel like a real product page rather than a static image.

Reactive Drop Shadows

The drop shadows on each packet aren't just static CSS shadows — they were carefully tuned using the Photoshop-inspired shadow debug panel. Each packet has custom shadow values (offset, blur, spread, color) that were visually dialed in and then hard-coded. The shadows respond to hover state, enhancing the feeling of the packet lifting off the surface.

10. Summary — The Full Pipeline

Here's the complete pipeline from start to finish, condensed into one view:

1. Generate

Use Nano Banana to generate product photography images at 1080×1920 (portrait), then regenerate in horizontal/landscape for the web layout base.

2. Extract

Open in Photoshop. Use manual selection, Select Color Range, and Blend If (Blending Options) to mask each element onto its own transparent layer.

3. Split Overlaps

For ingredients that overlap packets (dates, cinnamon), cut the overlapping portion onto a separate "top" layer aligned to the same canvas.

4. Standardize

Make every asset the exact same canvas size and aspect ratio. Export all as PNGs with transparency.

5. Collect Brand

Screenshot the logo from the official site. Color-pick each packet's brand color with the eyedropper tool.

6. Set Up Project

Create folder, connect to GitHub, drop in all assets. Brief Claude Code with the full visual and interaction spec.

7. Build Layers

Assemble in HTML/CSS: color overlay → bottom ingredient → packet → top ingredient, using z-index for depth.

8. Build Debug Tools

Create the custom debug menu (= key) with layout sliders, shadow controls (Photoshop-style), reference image overlay, and per-flavor position/scale adjustments.

9. Align Visually

Use the reference overlay at low opacity as a guide. Adjust each element with the debug sliders and transform handles until everything matches.

10. Copy Config

Use the Copy Config button to export all visual settings, feed them back into Claude Code to hard-code into the source.

11. Add Interactivity

Implement hover rise effect, color glow, click-to-open flavor popups, and reactive drop shadows.

12. Polish & Ship

Remove debug tools from production, final QA, push to GitHub.

Key Takeaways

AI as starting point, not final product. The AI-generated images were a foundation — they still required significant Photoshop work to extract, mask, and prepare individual assets.

Build your own tools. The debug menu and transform controls were custom-built for this specific project. When standard tools aren't enough, building your own workflow tools saves enormous time and produces better results.

Visual iteration → code export. The Copy Config workflow (visually adjust → export values → hard-code) is a powerful pattern. You work visually where your eyes can judge quality, then programmatically lock in the results.

Layer thinking translates. Photoshop's mental model of layers, z-index, and blending maps directly to how CSS stacking works. Understanding one helps you master the other.

Consistent canvas sizes matter. This tiny detail — making every asset the same dimensions — saved hours of alignment headaches. Always think about how assets will be composited together.