

Build your own virtual closet

The full build guide // @hinnazeejah

This is the companion to the reel. The full prompt I used is in here, along with the stack, the build order, and every single thing that broke along the way.

The honest version: the app is one prompt. The work is in the photos and in the five or six problems that show up after the first build runs. Most of this guide is about those.

You can have something working in an afternoon. Read the photo rules before you take a single picture, because that is the part that decides whether this looks polished or looks rough.

What you are building

A web app where you:

- Upload or photograph one clothing item at a time
- Have the background stripped off it automatically, in the browser
- See everything you own in a wardrobe sidebar, filterable by tops and bottoms
- Drag pieces onto a mannequin to layer a full outfit
- Resize, rotate, clear, and save looks into an outfits gallery
- Come back the next day and find all of it still there

That is version one. It is deliberately small. Resist adding weather integration and an AI stylist before the basics work, because that is how projects die half finished.

The stack

- Frontend: React or Next.js with Tailwind CSS and Lucide icons
- Background removal: [@imgly/background-removal](#), which runs client side so nothing leaves the browser
- Canvas: HTML5 drag and drop, or Fabric.js if you want resize and rotate handles
- Storage: IndexedDB for images, with localStorage only for small settings

The storage choice matters more than it sounds like it does. There is a whole section on it below.

Before you start

What you need

- A Claude account
- Node and npm installed
- Your phone camera
- A white sheet, a plain wall, or a bed with plain bedding

- About two hours, most of which is photographing clothes

You do not need frontend experience. That is the entire point of this build.

The photo rules

Read this part twice. Background removal is only as good as what you feed it.

- One item per photo. Not an outfit, not a stack, one thing
- Lay it flat on a plain surface and shoot straight down from above
- Natural light near a window. No overhead lamp, because shadows get read as part of the garment
- Shoot from roughly the same height every time so items end up a consistent scale
- Do not shoot white clothing on a white sheet. Use a contrasting surface or the edges dissolve
- Smooth the fabric out. Wrinkles become strange silhouettes once the background is gone

The fastest way to make this project look bad is good code and bad photos. Spend the extra twenty minutes here.

The prompt

This is the full prompt. Paste it as-is, or swap the color palette if pink is not your thing.

Build a full-stack Virtual Closet web application with a pink aesthetic and an interactive styling canvas.

Core features and functionality

1. One-time photo upload and automatic background removal
 - Provide an image uploader where users can upload or take photos of individual clothing items.
 - Automatically process uploaded photos to remove the background (using @imgly/background-removal, or a rembg backend integration) so only the garment remains, with a transparent background.
 - Categorize items into Shirts/Tops and Pants/Bottoms.
2. Digital wardrobe inventory
 - Display uploaded items in organized tabs: Shirts, Pants, All Items.
 - Allow users to view, delete, or tag items with colors or custom labels.
3. Interactive "virtual stylist" canvas
 - A central styling area featuring an interactive mannequin or figure overlay.
 - Users can drag, drop, or select items from their wardrobe onto the figure to layer shirts over pants and compose full outfits.
 - Controls to resize, rotate, clear, or save current outfit combinations to an "Outfits Gallery".

UI and styling requirements

- Color palette: soft pastel pinks (#FFB6C1, #FFF0F5), vibrant rose accents (#FF69B4), clean white (#FFFFFF), and

- soft charcoal for high-contrast text.
- Vibe: modern, aesthetic, clean, and playful.
- Layout: responsive, with a side panel for the wardrobe library and a large central canvas for the styling studio.

Tech stack suggestions

- Frontend: React / Next.js with Tailwind CSS, Lucide icons, and React HTML5 drag and drop (or Fabric.js / HTML canvas for the styling canvas).
- Background removal: @imgly/background-removal for client-side background separation.
- Storage: IndexedDB for persistent local storage of images and saved outfits.

Please set up the project structure, install necessary dependencies, implement background removal processing with loading feedback, build the wardrobe manager, and implement the interactive mannequin styling canvas.

Why this prompt works

Three things, and you can reuse all of them on any build:

- It names the library. "Remove the background" gets you a guess. Naming @imgly/background-removal gets you a specific, working implementation
- It specifies the layout before the features. Side panel plus central canvas is a shape, and giving the shape up front stops you from getting a generic page
- It asks for loading feedback explicitly. Background removal takes a few seconds per image, and without that line you get an interface that looks frozen

Build it in phases

Do not paste the prompt, get a whole app back, and then try to debug all of it at once. Run it, then work through these four checkpoints in order.

Phase 1 // Upload and display

Get one photo to upload and appear in the sidebar. Ignore background removal and dragging for now. You are only proving the upload path works and the item renders.

Phase 2 // Background removal

Test the background strip on your hardest item first, something patterned or light colored, not your easiest one. If it handles the hard one it handles everything. Watch for the loading state actually appearing.

Phase 3 // The canvas

This is the fussiest part and where most of your debugging time goes. Get a top to land on the mannequin and stay put. Then add bottoms and check the layering, because tops need to sit above bottoms. Then wire up resize and rotate.

Phase 4 // Persistence

Make the wardrobe survive a refresh, then make saved outfits survive a refresh. If you close the tab and your closet is empty, you have a prototype, not an app.

The part that actually matters

Storage. Specifically, choosing IndexedDB over localStorage.

This sounds like a footnote and it is the whole project. localStorage caps out around 5MB and stores strings only, so every image has to be base64 encoded, which inflates it by roughly a third. Two or three garment photos and you have blown the entire budget. The app then fails silently or throws a quota error the first time a real user adds their fourth item.

IndexedDB has no practical size limit for this and stores binary blobs directly, no encoding overhead. Same app, completely different ceiling.

When you explain this project to anyone, lead with the storage decision. Background removal is the part that looks impressive and it is the part you did the least work on. The storage call is the engineering.

Where it breaks

The list of things that went wrong, so you can skip a few hours:

- The background removal model downloads on first use and it is not small, so your very first removal takes noticeably longer than every one after it. Users think it is broken. This is exactly why the prompt asks for loading feedback
- White or very light garments photographed on a white background come back with pieces missing
- Busy patterns confuse the cut around the edges
- HTML5 drag and drop does not fire on touch screens, so an app that works on your laptop does nothing on your phone. Pointer events are the fix
- Photos straight off a phone camera run several megabytes each. Resize before storing or everything gets slow
- Items photographed at different distances end up wildly different sizes on the mannequin
- Layer order flips and your pants render on top of your shirt

Every one of these is fixable by describing the problem back precisely and asking for a fix. Noticing the problem and describing it accurately is most of what the job is anyway.

How you know you are done

- You uploaded at least ten real items, not three test images
- You closed the tab, reopened it, and everything was still there
- You built an outfit, saved it, and loaded it back from the gallery
- It works on your phone, not just your laptop
- You can name the two things that broke and what you did about them
- You actually used it once to decide what to wear

Stretch goals

Only after version one works and you have used it for a few days:

- Filtering by the color and custom tags you already built
- An outfit history, so you can see what you wore and when
- A "what have I not worn in a month" view, which is the genuinely useful feature
- Shoes and outerwear as additional layers
- Shareable outfit links

Resume bullet template

Fill in your own real numbers. Run the thing and count. Do not invent them.

```
Built a full-stack virtual wardrobe app in React with  
client-side image background removal, a layered drag-and-drop  
styling canvas, and IndexedDB persistence for binary image  
storage; catalogued [n] garments with [x]% clean extraction  
across varied fabric and lighting conditions.
```

If you finish the stretch goals, swap the second half for the filtering and wear-history features instead.

The shipping checklist

A project nobody can see is not a project.

- A README with a GIF at the top showing an item being uploaded, cut out, and dragged on. Recruiters give a repo about eight seconds
- At least ten real garments in every screenshot. An empty wardrobe sidebar reads as unfinished
- One command to run it
- A note in the README about why you chose IndexedDB over localStorage. That paragraph is what gets quoted back to you in an interview
- Three sentences at the bottom on what you would do differently. Most underrated section in any repo
- No API keys, no tokens, nothing personal in your git history. Check before you push, not after

One last thing

This works as a portfolio piece for one reason, and it is not that it is cute. It is that you had a problem in your own life, you scoped it small, and you shipped something you actually use.

That is the whole lesson. Nobody is getting hired off another tutorial notebook. Build the thing that annoys you.

Go build something <3

Made by @hinnaazeejah · Hinna