Product Requirements Document: Live Canvas Preview Architecture

1. Executive Summary & Motivation

Objective: Replace the current CSS-based preview engine in the `BackdropPositioning` component with a high-performance **HTML5 Live Canvas Preview**.

Motivation: Currently, LuxSnap uses CSS transforms (`top`, `left`, `transform: translate`) to simulate product positioning. However, the final output generation uses the HTML5 Canvas API (`src/lib/canvas-utils.ts`). This architectural discrepancy causes:

1. **WYSIWYG Failures:** Users position an item in the CSS preview, but the final compiled image renders it slightly differently due to different scaling algorithms between CSS and Canvas.
2. **Effect Inconsistency:** Complex canvas operations like "Gradient Depth of Field" and "Cloudinary Padding" are difficult to replicate exactly in CSS, leading to preview inaccuracies.

The Solution: We will implement a "Single Source of Truth" architecture. The exact same mathematical functions used to generate the final high-res batch output (`computeCompositeLayout`) will be used to render the low-res live preview.

2. Architectural Strategy

The new system will operate on a **Scene Graph** concept using two canvas layers to ensure 60fps performance during drag operations.

The "Offscreen Buffer" Strategy

Rendering the backdrop blur (Depth of Field) is computationally expensive (approx. 50-100ms). We cannot re-render this every time the user drags the mouse.

1. **Layer 1 (Cached Buffer):** An offscreen canvas that renders the Backdrop + Gaussian Blur + Gradient Mask. This only updates when the user changes the backdrop or toggles the blur switch.
2. **Layer 2 (Active Canvas):** The visible canvas. On every frame (drag event), it simply clears, draws Layer 1 (as an image), and then draws the Subject on top.

Unified Coordinate System

Instead of CSS percentages, we will use a **Normalized Coordinate System (0.0 - 1.0)**.

- `x=0.5, y=1.0`: Subject centered horizontally, anchored at the bottom of the frame.
- **Interaction:** Mouse movements on the canvas will be converted from **Screen Pixels** -> **Normalized Vectors** -> **State Updates**.

---

3. Technical Specifications & Deep Dive

A. Modified Module: `src/lib/canvas-utils.ts`

We need to expose internal logic that is currently locked inside the batch processor so the preview component can use it.

Changes Required:

1. **Export `applyDepthOfField`:** Currently, this is a local helper function. It must be exported so the preview component can generate the exact same "Long Ramp" gradient blur as the final output.
2. **Verify `computeCompositeLayout`:** Ensure this function is pure and accepts dimensions that can be scaled. It will be the engine driving the preview positioning.

B. Overhauled Component: `src/components/BackdropPositioning.tsx`

This file requires a total rewrite of the render logic.

State Management Changes:

- Remove: `getPreviewStyles()` and all CSS transform logic.
- Add: `useRef<HTMLCanvasElement>` for the visible canvas.
- Add: `useRef<HTMLCanvasElement>` for the offscreen "Backdrop Buffer".
- Add: `useRef<HTMLImageElement>` for persistent storage of loaded images (to prevent reloading on every render).

Rendering Pipeline (UseEffect Hook):

1. **Initialization:** Load `backdropUrl` and `subjectUrl` (previewCutout or live Cloudinary URL) into `HTMLImageElement` refs.
2. **Buffer Step:**
    - Check if `backdrop` or `blurBackground` changed.
    - If yes, draw backdrop to `offscreenCanvas`.
    - If `blurBackground` is true, call `applyDepthOfField(offscreenContext, ...)`.
3. **Composite Step:**
    - Get `canvas.width` / `canvas.height` (Display Space).

    - Call `computeCompositeLayout` using Display Space dimensions.
    - `ctx.drawImage(offscreenCanvas, 0, 0)` (Draw background).
    - `ctx.drawImage(subject, layout.rect.x, layout.rect.y...)` (Draw subject).

Interaction Logic (Drag Handlers):

- **Mouse Down:** Set `isDragging = true`.
- **Mouse Move:**
    - Calculate delta Y in pixels.
    - Convert pixel delta to percentage of Canvas Height.
    - Update `placement.y` state (clamped 0-1).
    - *Note:* `placement.x` remains fixed at 0.5 (Center) per current business rules.