AI REIMAGINE PROJECT · FILE 04

Prototype Build & Wireframes

FitSense - a working AI fit-advisor on the Myntra PDP

Prepared by Abhishek Katkar · June 2026

This is a working prototype, not a static mock. FitSense runs on a Myntra-faithful product page, calls a live LLM through a server function, parses a strict JSON contract, and handles every state - including the low-confidence and AI-unavailable fallbacks that File 3 specified. Every screen below is a screenshot of the running app.

1. What Was Built

- **A brand-faithful Myntra PDP** for the Imsa Moda Men Henley Neck T-shirt, with the size selector and the size-chart modal (chest / front-length).
- **The FitSense card** in the size area - the hybrid model from File 2: an always-on, explained, confidence-scored recommendation, plus an optional "Refine my fit" expansion (fit preference + optional chest measurement).
- **A real backend AI call** that reasons over the garment's size chart and the user's fit profile and returns { recommendedSize, confidence, reason }.
- **Honest confidence + graceful degradation** per File 3: Low confidence at cold-start, and a fallback to the static size chart when the AI is unavailable.
- **Demo controls** (returning user / new user cold-start / simulate AI error) so every state can be exercised and evidenced.

2. Tool Stack

Layer	Tool	Role
Build platform	Lovable	Prompt-driven full-stack build and preview hosting.
Frontend	React + TanStack Start + Vite	PDP UI, size-selector state, and the FitSense card (loading / result / fallback).
Backend	TanStack Start server function	Server-side LLM call and strict-JSON parsing (key kept off the client).
AI	Lovable AI Gateway -> Google Gemini	Reasons over garment + profile; produces size, confidence, and reason.
Styling	Myntra design tokens	Assistant font, #FF3F6C pink, #282C3F text - brand-faithful look.

3. System Architecture

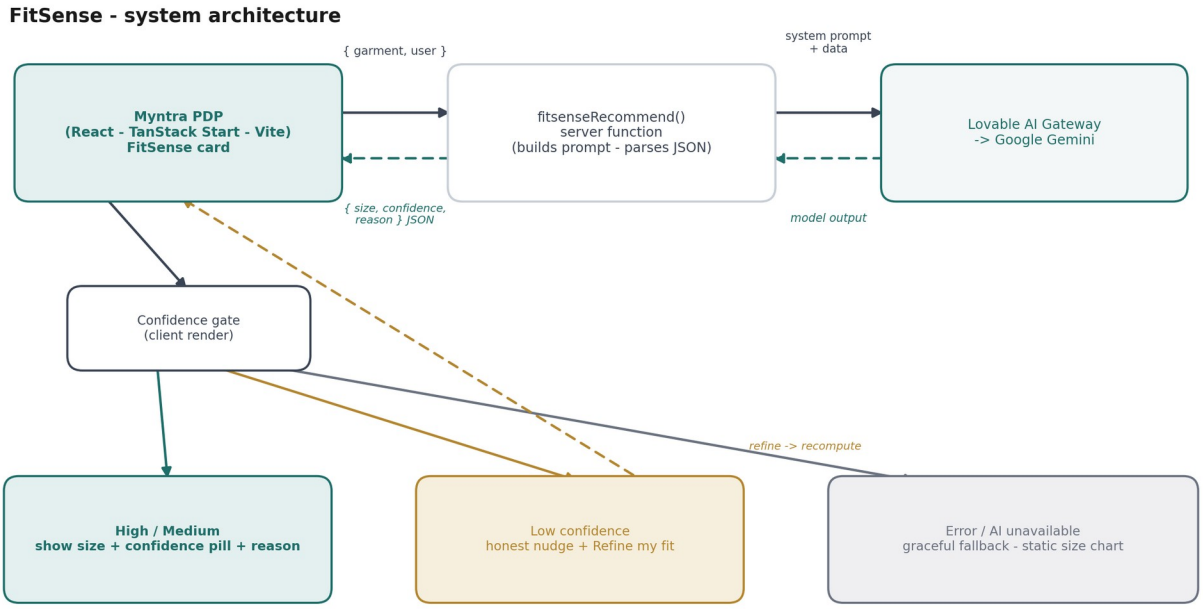

Figure 4.1 - On PDP load the frontend sends { garment, user } to the server function, which calls the AI gateway, parses strict JSON, and returns the recommendation. A client-side confidence gate then renders one of three outcomes.

On load, the PDP sends `{ garment, user }` to the `fitsenseRecommend` server function. The function injects the system prompt, calls the Lovable AI Gateway (Google Gemini), parses the model's strict-JSON reply, and returns `{ recommendedSize, confidence, reason }`. The client confidence gate renders a High/Medium recommendation, a Low-confidence honest nudge with "Refine my fit," or - if the call fails - a graceful fallback to the static size chart.

4. Backend Wiring

The recommendation is generated server-side. The system prompt enforces the confidence-honesty rules from File 3 (Low at cold-start, no invented garment attributes). The core of `src/lib/fitsense.functions.ts`:

```
// src/lib/fitsense.functions.ts  -  TanStack Start server function

const SYSTEM_PROMPT = `You are FitSense, a size-fit advisor on a product
  page. Recommend the best size with an HONEST confidence level and a
  one-line reason. Return STRICT JSON only:
    recommendedSize, confidence ('High'|'Medium'|'Low'), reason, evidenceUsed.
  - Ground the reason ONLY in provided data; never invent attributes.
  - 'High' only with strong, consistent signal; 'Low' when inputs are thin
    (cold-start) - still give a best guess and invite the user to refine.`;

export const fitsenseRecommend = createServerFn({ method: 'POST' })
```

```
.handler(async ({ data }) => {
  const res = await fetch('https://ai.gateway.lovable.dev/v1/chat/completions', {
    method: 'POST',
    headers: { 'Lovable-API-Key': process.env.LOVABLE_API_KEY },
    body: JSON.stringify({
      model: 'google/gemini-3-flash-preview',
      messages: [
        { role: 'system', content: SYSTEM_PROMPT },
        { role: 'user',   content: JSON.stringify({ garment, user }) },
      ],
    }),
  });
  const json = await res.json();
  return JSON.parse(stripFences(json.choices[0].message.content));
  // -> { recommendedSize, confidence, reason, evidenceUsed }
});
```

The API key is read from the environment and never reaches the client. The strict-JSON contract is what lets the UI stay simple: the card just reads `confidence` and `reason` and renders the right state.

5. Screens & States

Each screen is a screenshot of the running prototype.

5.1 FitSense on the Myntra PDP

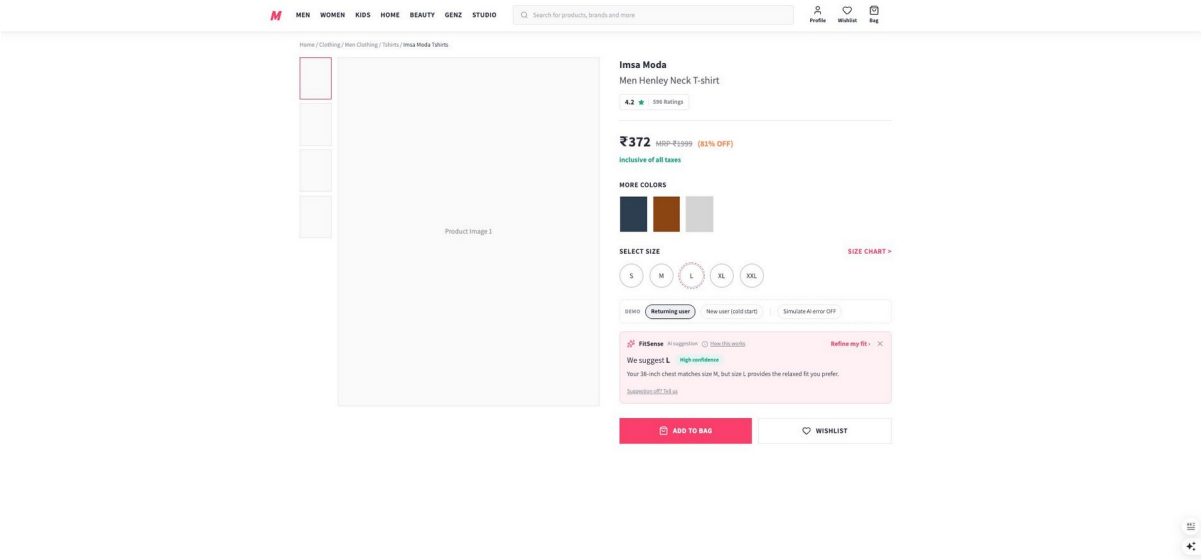

Figure 4.2 - The card is native to the page: an AI-suggestion label, a confidence pill, and a one-line reason grounded in the garment and the user's profile (here, refined to High confidence).

5.2 AI processing (loading)

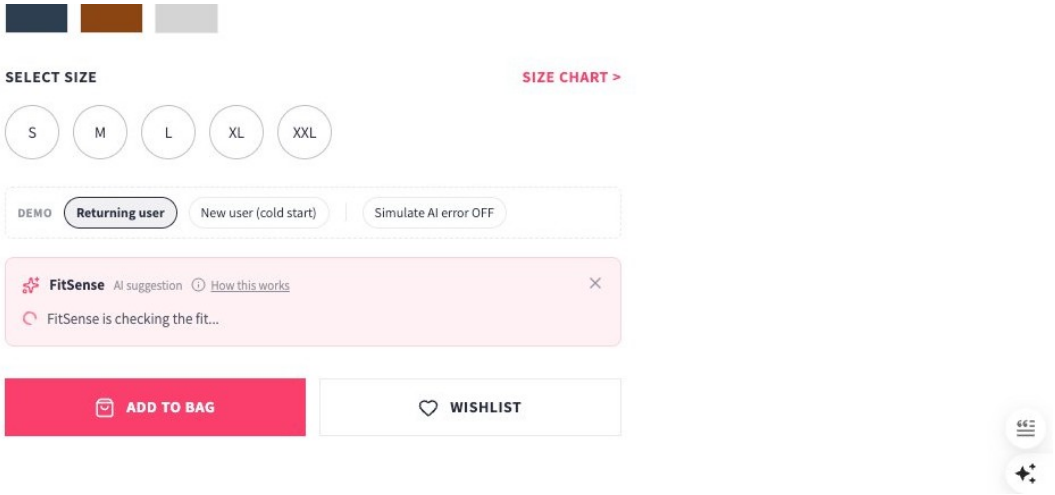

Figure 4.3 - While the server function calls the model, the card shows a live loading state.

5.3 Refine my fit (input)

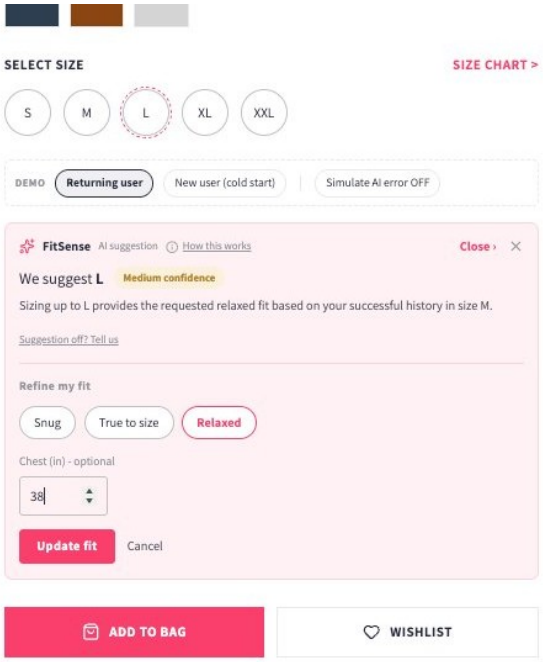

Figure 4.4 - The optional expansion collects a fit preference and an optional chest measurement, then recomputes. Adding a measurement typically lifts confidence from Medium to High.

5.4 Cold-start - Low confidence (honest fallback)

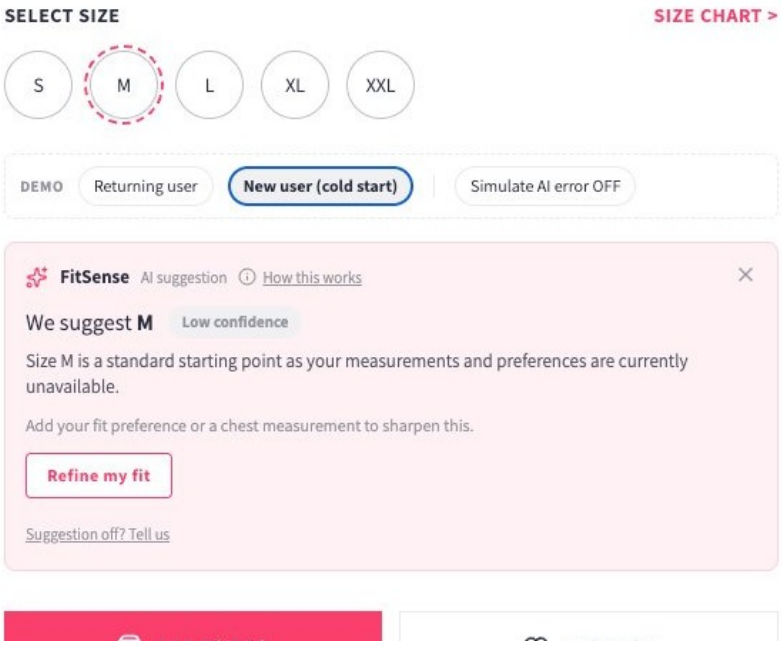

Figure 4.5 - A new user with no history or measurements gets a Low-confidence label, an honest nudge, and a prominent Refine button - never a bluffed certainty.

5.5 Graceful degradation (AI unavailable)

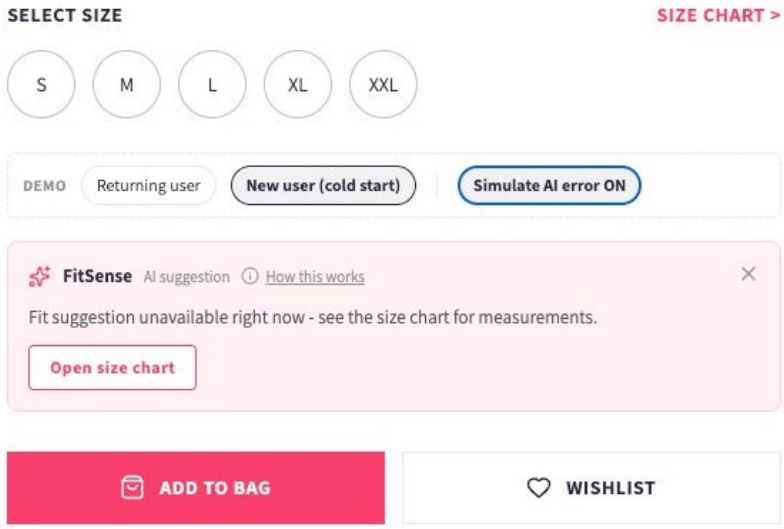

Figure 4.6 - When the AI call fails, the card degrades to a fallback that points to the size chart; the PDP never blocks.

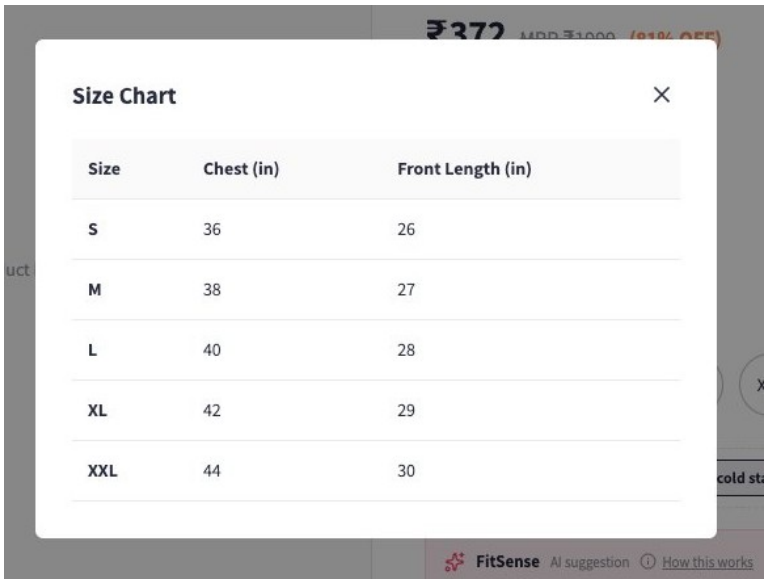

Figure 4.7 - The fallback button opens the existing size chart, so the user can still self-size.

5.6 Size chart

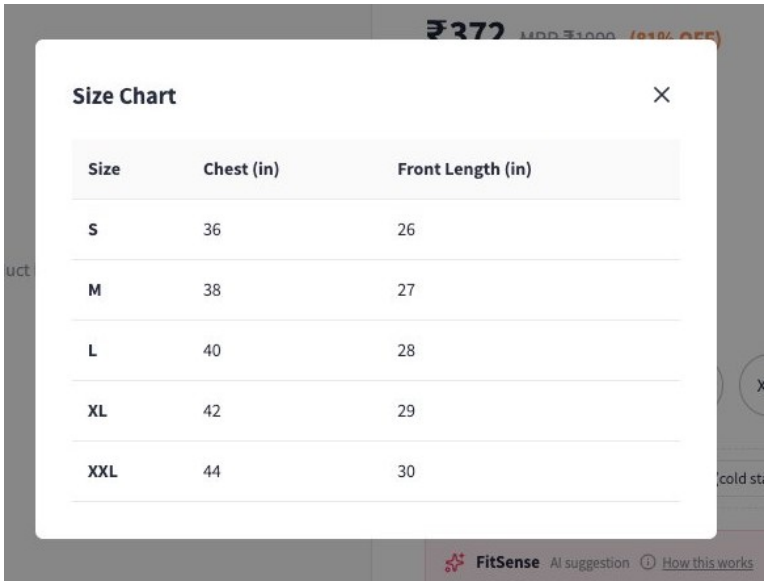

Figure 4.8 - The garment's size chart (chest / front-length), available throughout.

5.7 Transparency - “How this works”

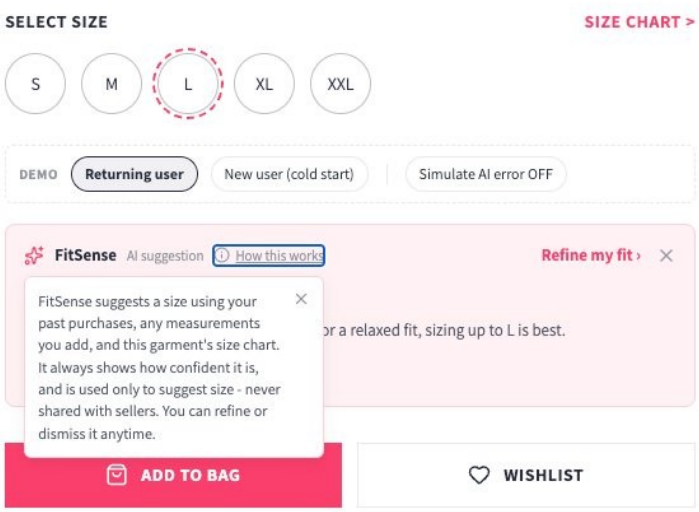

Figure 4.9 - A plain-language explainer of what data is used and that it is never shared with sellers, plus a flag control and a dismiss option (per File 3).

6. UX Rationale

- **Hybrid by design.** The passive explained recommendation reaches every shopper; the optional refine lets motivated users add a signal and earn higher confidence - reach and accuracy without forcing effort.
- **Confidence honesty is the spine.** The pill is always shown, and at cold-start the system says Low and invites refinement rather than bluffing. That is the core differentiator from today’s silent “we recommend L.”
- **Never blocks a purchase.** If the AI is down, the card degrades to the size chart - the feature fails safe.
- **Brand-native.** FitSense is styled to read as a first-party Myntra capability, not a bolt-on, which is what earns user trust at the size-decision moment.
- **Maps directly to F1.** Garment-aware (reads the size chart), explainable (one-line reason), confidence-scored, and useful at cold-start - exactly the gaps the existing purchase-history recommendation leaves open.

7. How This Maps to the Build Requirements

Every required state is evidenced above: entry (4.2), AI processing (4.3), AI output (4.2), refine input (4.4), low-confidence fallback (4.5), AI-unavailable fallback (4.6 - 4.7), and transparency (4.9), plus the backend wiring (Section 4). Files 5 and 6 build on this: File 5 defines how recommendation accuracy, confidence calibration, and cohort lift are measured; File 6 is the panel defence.