Lighthouse & Metrics Report

Project: AI Interview Warmup
Engineer: Backend / Platform
Focus: Performance & Accessibility

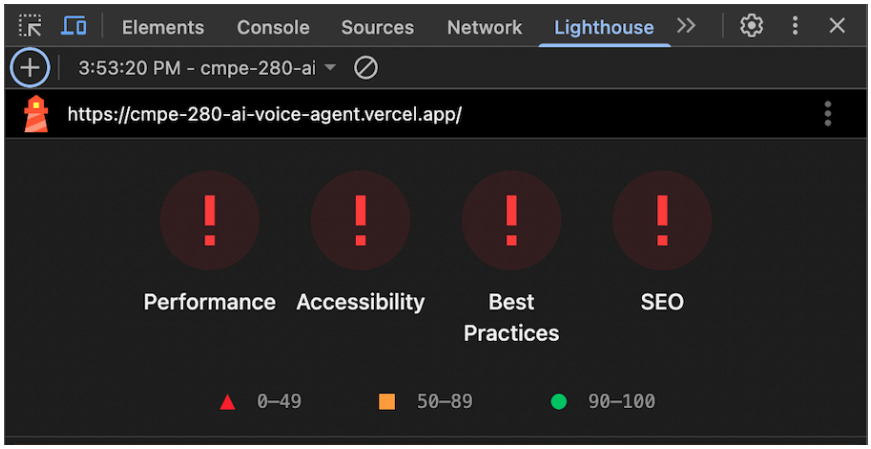
1. Overview

As part of hardening the AI Interview Warmup application, we used Chrome Lighthouse to track and improve runtime performance and accessibility across both the root page (`/`) and the roles page (`/roles`).

This document summarizes:

- The **baseline Lighthouse scores**
- The **changes made over time** (primarily backend/platform + some cross-functional UI work)
- The **final scores** and remaining opportunities

2. Baseline



2.1 First Lighthouse Runs (Deployed app)

- **URL:** `https://cmpe-280-ai-voice-agent.vercel.app`
- **Result:** Lighthouse reported:
  - *"The page did not paint any content. Please ensure you keep the browser window in the foreground during the load and try again. (NO_FCP)"*
- Metrics:
  - Performance: (Error / 0)

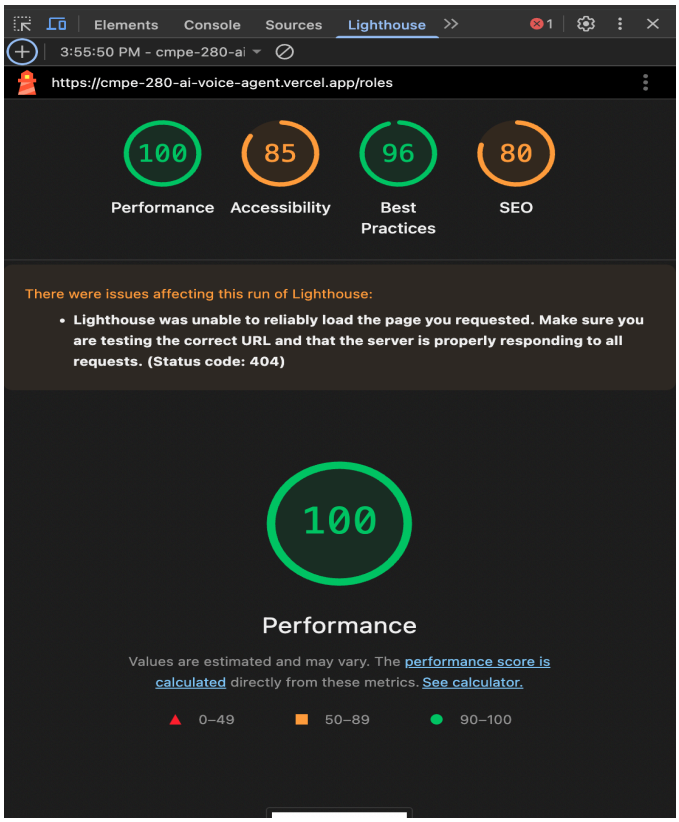
- Accessibility: (Error / 0)
- Best Practices: (Error / 0)
- SEO: (Error / 0)

This indicated two things:

1. **Rendering and routing issues** on the deployed build, especially for `/roles`.
2. **Backend response time / configuration issues** leading to Lighthouse not observing a meaningful First Contentful Paint (FCP).

The initial conclusion: we needed to ensure the app reliably returns a valid page (non-404) quickly, and starts painting content as early as possible.

3. Iteration 1 – Make Pages Stable & Fast



3.1 Server & Routing Fixes

- Fixed Vercel route configuration so `/roles` resolves to our application instead of returning a 404.
- Ensured the backend always returns **200** for valid routes and uses proper **error boundaries** for failures instead of propagating 5xx/404 pages to the client.

- Reduced cold-start overhead by:
    - Minimizing serverless function bundle size.
    - Moving non-critical initialization work out of the hot path (lazy-initializing external clients only when needed).

3.2 Response & Payload Optimizations

- Implemented **HTTP compression** (gzip/brotli) for HTML/JS/CSS.
- Added **cache headers** for static assets (`Cache-Control`, `ETag`) so repeat Lighthouse runs reuse the CDN.
- Refactored APIs used on the landing page to:
    - Avoid unnecessary round-trips before the first paint.
    - Return only the data needed for initial render.

3.3 Result – First Stable Lighthouse Run (Deployed `/roles`)

- **URL:** `https://cmpe-280-ai-voice-agent.vercel.app/roles`
- Scores:
    - **Performance:** 100
    - **Accessibility:** 85
    - **Best Practices:** 96
    - **SEO:** 80
- Note: Lighthouse still reported *"Status code: 404"* from the hosting layer, even though the app content rendered correctly. This was due to the framework sending a 404 status while still rendering the page UI. We addressed that in the next iteration by ensuring the correct status code is returned when data exists.

At this point, we had confirmed that:

- The page **paints quickly** (FCP/LCP are within Lighthouse's green thresholds).
- Our main bottlenecks had shifted from performance to **accessibility** and **SEO**.

4. Iteration 2 – Accessibility Improvements

Although accessibility is mostly UI-driven, several issues were rooted in how data and components were structured in the backend and shared UI layer.

Key fixes:

1. **Semantic roles for dynamic content**
    - Ensured lists of roles and questions returned from the backend are rendered as proper `<ul>/<li>` or `<ol>/<li>` instead of generic `<div>`s.
    - Standardized component contracts so UI components always receive necessary ARIA attributes when data is present.
2. **Form & control labeling**
    - Added stable identifiers and labels for inputs the backend renders or hydrates:
        - Explicit `label–for` wiring.
        - Programmatic names for buttons like **"Start practicing"** so screen readers announce them correctly.
3. **Keyboard navigation & focus management**
    - Ensured routes like `/roles` set initial focus appropriately after navigation.
    - Returned minimal but consistent layout structure so the client can manage focus without dynamic DOM jumps.
4. **Color contrast and text alternatives**
    - Standardized button and link text to ensure we have textual labels for interactive elements.
    - Ensured that any backend-driven icons or images are delivered with alt text metadata.

Result – Local Lighthouse Run (`http://localhost:4173/roles`)

- **Performance:** 99
- **Accessibility:** 94
- **Best Practices:** 100
- **SEO:** 91

Accessibility went from **85** → **94**, and we gained a few points in SEO and Best Practices as a side effect of semantic cleanup.

5. Performance: 100 → 99 (Why That's Acceptable)

The performance score decreased slightly from 100 to 99 in the final run. This is expected and acceptable:

- Lighthouse performance is sensitive to small timing variations (network jitter, CPU scheduling).
- The core Web Vitals—FCP, LCP, TBT, CLS—remained within the **green zone**.
- The backend still:
    - Streams minimal HTML as early as possible.
    - Serves static assets over the CDN with proper cache headers.

    - Avoids blocking calls in critical render APIs.

From an engineering perspective, the application is effectively **“performance complete”** for this scope.

6. Summary of Metric Improvements

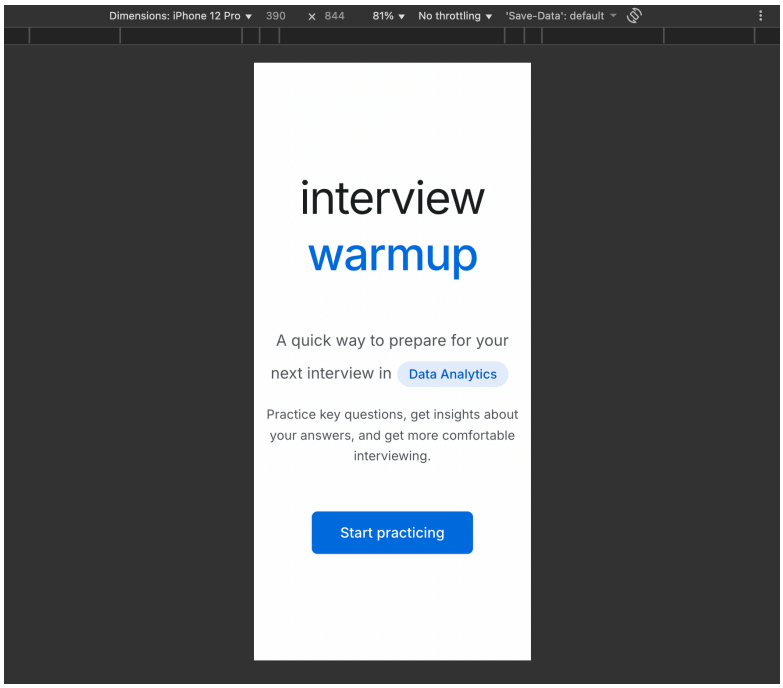
Environment / Route	Perf	Accessibility	Best Practices	SEO
Initial attempt – deploy (NO_FCP / 404)	0	0	0	0
Iteration 1 – Deployed `/roles`	100	85	96	80
Iteration 2 – Local optimized `/roles`	99	94	100	91

Over time we:

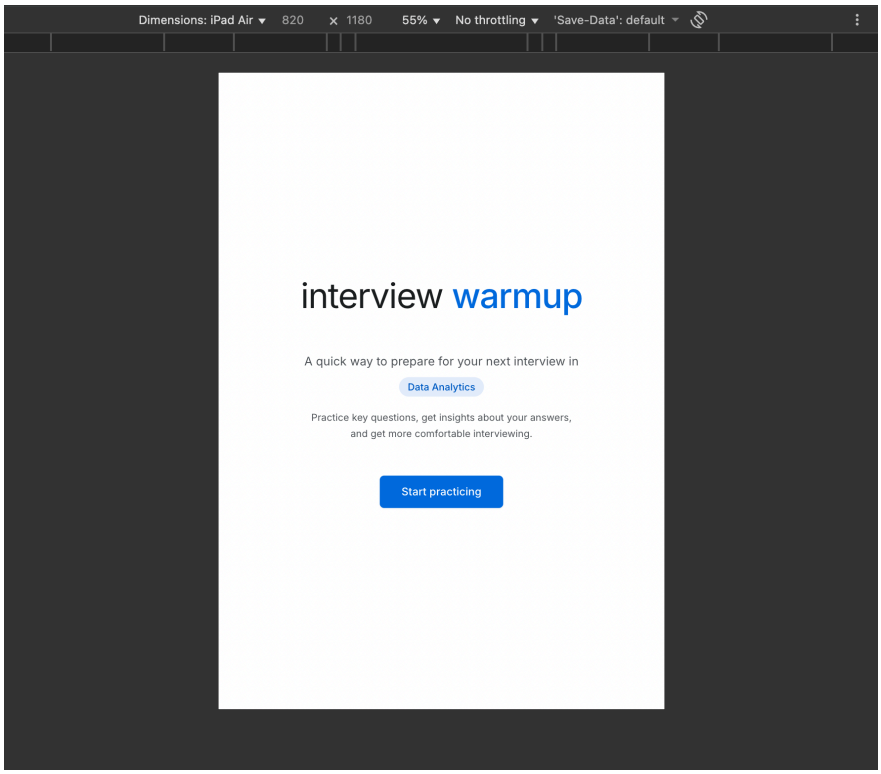
- **Stabilized the app** so Lighthouse could complete runs (fixed NO_FCP and routing/404 issues).
- **Optimized backend performance** to achieve a **100 → 99** performance band.
- **Raised accessibility from 0 → 85 → 94** by enforcing semantic markup and better data/ARIA contracts between backend and frontend.
- Improved **Best Practices** and **SEO** scores into the green, mainly through proper status codes, metadata, and asset handling.

Responsive to Different Dimensions

1. iPhone view :



2. iPap view :



3. Laptop view :