SourceFlow

Site Audit Report

https://www.gravitasgroup.co.uk/

Generated: 01 Jul 2026 · Detailed Report

SCORES

	Mobile	Desktop
Overall	81	84
Performance	27	37
SEO	100	100
UX / Best Practices	100	100
Accessibility	97	97

CORE WEB VITALS

	Mobile	Desktop
Largest Contentful Paint (LCP)	18.2 s	4.3 s
First Contentful Paint (FCP)	7.4 s	1.3 s
Interaction to Next Paint (INP)	120 ms	61 ms
Cumulative Layout Shift (CLS)	0.02	0.012
Total Blocking Time (TBT)	2,120 ms	2,220 ms
Speed Index	20.1 s	3.5 s

RESOURCE DIAGNOSTICS

Unused CSS / JS	4 files — save ~405 KiB
Large Assets (> 500 KB)	1 flagged

Unused CSS / JS files:

250.0 KiB https://cdn.sourceflow.co.uk/formio/formio.full.min.js
70.4 KiB https://www.googletagmanager.com/gtag/js?id=G-WVTVJKJG0G
58.3 KiB https://www.gravitasgroup.co.uk/_next/static/chunks/pages/_app-0d721b65169f1d63.js
26.3 KiB https://www.gravitasgroup.co.uk/_next/static/css/dafcdc4262ffbf10.css

PAGESPEED INSIGHTS — ISSUES

FAIL **Render-blocking requests**

Est savings of 780 ms

Requests are blocking the page's initial render, which may delay LCP. can move these network requests out of the critical path.

FAIL **LCP breakdown**

Each . Ideally, most of the LCP time should be spent on loading the resources, not within delays.

FAIL **LCP request discovery**

by making the LCP image discoverable from the HTML immediately, and avoiding lazy-loading

FAIL **Use efficient cache lifetimes**

Est savings of 433 KiB

A long cache lifetime can speed up repeat visits to your page. .

WARN **Improve image delivery**

Est savings of 21 KiB

Reducing the download time of images can improve the perceived load time of the page and LCP.

WARN **Legacy JavaScript**

Est savings of 28 KiB

Polyfills and transforms enable older browsers to use new JavaScript features. However, many aren't necessary for modern browsers. Consider modifying your JavaScript build process to not transpile features, unless you know you must support

WARN **Font display**

Est savings of 20 ms

Consider setting to swap or optional to ensure text is consistently visible. swap can be further optimized to mitigate layout shifts with .

FAIL **Document request latency**

Est savings of 1,460 ms

Your first network request is the most important. by avoiding redirects, ensuring a fast server response, and enabling text compression.

FAIL **Minimize main-thread work**

5.7 s

Consider reducing the time spent parsing, compiling and executing JS. You may find delivering smaller JS payloads helps with this.

FAIL **Reduce unused JavaScript**

Est savings of 379 KiB

Reduce unused JavaScript and defer loading scripts until they are required to decrease bytes consumed by network activity. .

FAIL **Reduce unused CSS**

Est savings of 26 KiB

Reduce unused rules from stylesheets and defer CSS not used for above-the-fold content to decrease bytes consumed by network activity. .

WARN **Image elements without explicit dimensions**

Set an explicit width and height on image elements to reduce layout shifts and improve CLS.

WARN **Avoid enormous network payloads**

Total size was 2,976 KiB

Large network payloads cost users real money and are highly correlated with long load times. .

FAIL **Reduce JavaScript execution time**

4.1 s

Consider reducing the time spent parsing, compiling, and executing JS. You may find delivering smaller JS payloads helps with this. .

AI RECOMMENDATIONS

This page scores 27/100 because render is gated by a slow server and a heavy front end: the document alone is 1,144.5 KiB with a Document-request-latency penalty of ~1,460 ms, pushing FCP to 7.4 s, while an un-preloaded LCP image plus 780 ms of render-blocking CSS/JS drive LCP to 18.2 s and Speed Index to 20.1 s. Interactivity is crippled by 5.7 s of main-thread work and 4.1 s of JS execution (379 KiB unused), producing a 2,120 ms TBT. Note that CrUX field data PASSED (no/low real-user traffic), so these are worst-case lab numbers — but the root causes (TTFB, render-blocking, oversized JS/HTML, missing LCP preload) are real and will hit real users at scale.

CRITICAL **Cut document request latency (TTFB)**

The main document at https://www.gravitasgroup.co.uk/ carries a ~1,460 ms latency penalty and is 1,144.5 KiB. Enable full-page/edge caching (e.g. CDN HTML caching or a server-side page cache) and serve the document with a warm cache so TTFB drops under 200 ms. Confirm the origin returns Cache-Control and a CDN HIT rather than regenerating HTML per request.

Impact: LCP/FCP — est. -1.4 s (removes the ~1,460 ms server wait that blocks every downstream resource)

```
# Response header target for the HTML document
Cache-Control: public, max-age=0, s-maxage=600, stale-while-revalidate=60
# Verify edge cache is serving it:
curl -sI https://www.gravitasgroup.co.uk/ | grep -Ei 'cf-cache-status|x-cache|age|cache-control'
```

CRITICAL **Eliminate render-blocking CSS and JS**

Render-blocking requests cost ~780 ms. Add defer to non-critical <script> tags in the <head>, inline critical above-the-fold CSS, and load the rest with media="print" onload swap or rel=preload. Move third-party/analytics scripts to defer or load after the load event.

Impact: FCP/LCP — est. -0.8 s (unblocks first paint at 7.4 s FCP)

```
<!-- Non-blocking script -->
<script src="/app.js" defer></script>

<!-- Non-blocking stylesheet -->
<link rel="preload" href="/styles.css" as="style" onload="this.rel='stylesheet'">
<noscript><link rel="stylesheet" href="/styles.css"></noscript>
```

CRITICAL Preload and prioritize the LCP image

LCP request discovery and LCP breakdown both fail — the hero image (likely f86e7b43-6d05-4ecb-b33d-55742faa486a, 56 KB) is discovered late. Add a <link rel=preload as=image fetchpriority=high> in the <head> and fetchpriority="high" on the <img>, and remove loading="lazy" from it. Do NOT lazy-load the LCP element.

Impact: LCP — est. -3 to -5 s (moves image discovery earlier in the critical path)

```
<link rel="preload" as="image"
href="/.../f86e7b43-6d05-4ecb-b33d-55742faa486a"
fetchpriority="high">

<img src="/.../f86e7b43-6d05-4ecb-b33d-55742faa486a"
fetchpriority="high" width="..." height="..." alt="...">
```

HIGH Remove unused JavaScript and split bundles

379 KiB unused JS drives 4.1 s execution, 5.7 s main-thread work and 2,120 ms TBT. Code-split with dynamic import(), tree-shake dead code, and defer/lazy-load below-the-fold widgets. Break long tasks into <50 ms chunks. Audit the 4 files flagged with unused CSS/JS (Save 405 KiB) and the single asset over 500 KB.

Impact: TBT/INP — est. -1,000 ms TBT (fewer/shorter long tasks on the main thread)

```
// Lazy-load non-critical modules instead of shipping them upfront
const onIdle = window.requestIdleCallback || (cb => setTimeout(cb, 1));
onIdle(() => import('./below-the-fold-widget.js'));
```

HIGH Shrink the 1,144 KiB HTML document

The document is 1,144.5 KiB — abnormally large and part of the 2,976 KiB total payload. Enable Brotli/gzip compression on the HTML response, strip inlined data/base64 blobs, and remove server-rendered dead markup. A compressed HTML doc should be well under 100 KiB over the wire.

Impact: FCP/LCP — est. -0.5 to -1 s (less bytes to download and parse before render)

```
# Confirm compression is applied to the HTML
curl -sI -H 'Accept-Encoding: br,gzip' https://www.gravitasgroup.co.uk/ \
| grep -i content-encoding
# Expect: content-encoding: br
```

MEDIUM Set efficient cache lifetimes

433 KiB is served with short/no cache lifetimes. Add Cache-Control: public, max-age=31536000, immutable to fingerprinted static assets (JS/CSS/images/fonts) so repeat visits skip re-download.

Impact: LCP/FCP on repeat visits — est. -0.4 s and 433 KiB saved on return loads

```
# Static, hashed assets
location ~* \.(js|css|woff2|png|svg|jpg|webp)$ {
add_header Cache-Control "public, max-age=31536000, immutable";
}
```

MEDIUM Add image dimensions and next-gen formats

logo_light_en.svg has no explicit dimensions (CLS risk); add width/height. Convert f86e7b43-6d05-4ecb-b33d-55742faa486a (56 KB) and transparent.png (7 KB) to WebP/AVIF for ~21 KiB savings. Set width/height on all <img> to reserve space.

Impact: CLS/LCP — prevents layout shift and saves ~21 KiB on the hero path

```
<img src="logo_light_en.svg" width="180" height="40" alt="Gravitas Group">

<picture>
<source srcset="hero.avif" type="image/avif">
<source srcset="hero.webp" type="image/webp">
<img src="f86e7b43-..." width="1200" height="600" alt="...">
</picture>
```

LOW Drop legacy JS and fix font-display

28 KiB of legacy JS (polyfills/transpiled code) ships to modern browsers — target ES2020+ in your build (e.g. Babel/browserslist) to skip it. Add font-display: swap to @font-face to remove the 20 ms font-blocking penalty.

Impact: TBT — est. -50 ms parse/exec plus faster text paint

```
// package.json / .browserslistrc
"browserslist": ["defaults and supports es6-module"]

/* CSS */
@font-face { font-family:'Brand'; font-display: swap; src:url('brand.woff2') format('woff2'); }
```