AI AGENT-READINESS REPORT

danielwellington.com

Ran 46 of 81 checks

OVERALL

83

Grade B

SURFACE SUB-SCORES

CHATGPT / ACP	GOOGLE AI MODE	PERPLEXITY
81	86	80

META	MICROSOFT COPILOT
81	85

Findings · 9 need attention

NEEDS WORK CRITICAL **HTTPS enforced sitewide + HSTS (>= 6-month max-age)**

Enforce HTTPS sitewide and ship a Strict-Transport-Security header with max-age >= 6 months

WHAT WE FOUND

Confirmed the homepage is HTTPS (status 200), probed http://danielwellington.com/ for redirect behaviour, and parsed the Strict-Transport-Security header (value: "max-age=7889238").

How we checked: URL scheme + homepage status check, an http://host/ redirect probe, and a Strict-Transport-Security max-age parse (>= 180-day threshold).

- HSTS max-age is below the 6-month minimum

```
Checked: danielwellington.com
```

FAILED HIGH **Product pages discoverable without JavaScript**

Make product pages discoverable without JavaScript

WHAT WE FOUND

Counted product pages discovered by the non-JavaScript crawl. None were found — JS-only storefront, products missing from sitemap, or the crawl was blocked.

How we checked: Count the product pages a non-JavaScript crawl could discover via the sitemap or initial HTML (no JS execution). The fetcher already attempted discovery; we read ctx.pdpSample.

- No product pages discoverable from a non-JavaScript crawl

```
Checked: danielwellington.com/sitemap.xml, danielwellington.com
```

FAILED HIGH **UCP profile carries all four required top-level keys**

Add every required top-level key to the UCP profile

WHAT WE FOUND

Profile is missing required key(s): signing_keys.

How we checked: Read the profile root (or top-level `ucp` wrapper) and verify the presence of `version`, `services`, `capabilities`, and `signing_keys` keys.

- Required top-level key `signing_keys` is missing

```
Checked: danielwellington.com/.well-known/ucp
```

FAILED MEDIUM **HSTS policy carries the includeSubDomains directive**

Add `includeSubDomains` to your Strict-Transport-Security header

WHAT WE FOUND

Inspected the homepage Strict-Transport-Security header ("max-age=7889238") and the includeSubDomains directive is absent.

How we checked: Parse the homepage `Strict-Transport-Security` header for the `includeSubDomains` directive (§6.1.2).

- HSTS header is missing the includeSubDomains directive

`Checked: danielwellington.com`

FAILED MEDIUM **Organization/OnlineStore JSON-LD with contactPoint on homepage**

Add an Organization (or OnlineStore) JSON-LD block to your homepage with a contactPoint

WHAT WE FOUND

Found a homepage Organization node but its contactPoint is missing both email and telephone.

How we checked: Parse homepage `<script type="application/ld+json">` blocks, flatten `@graph`, and look for an Organization/OnlineStore/Store node with a contactPoint carrying `email` or `telephone`.

- Homepage Organization node has no contactPoint with email or telephone

`Checked: danielwellington.com`

FAILED LOW **About page reachable with substantive copy**

Publish a substantive About page at a standard URL

WHAT WE FOUND

Probed 4 candidate About-page paths and none returned a 2xx body

How we checked: URL probe of platform-specific about-page paths via politeFetch; the first 2xx response whose HTML-stripped body length is >= 200 chars counts as a pass.

- No About page reachable at any standard URL

`Checked: danielwellington.com/pages/about, danielwellington.com/pages/about-us, danielwellington.com/about, danielwellington.com/about-us`

FAILED LOW **HSTS policy carries the preload directive**

Add `preload` to your Strict-Transport-Security header and submit to hstspreload.org

WHAT WE FOUND

Inspected the homepage Strict-Transport-Security header ("max-age=7889238") and the preload directive is absent.

How we checked: Parse the homepage `Strict-Transport-Security` header for the `preload` directive (hstspreload.org vendor extension to).

- HSTS header is missing the preload directive

`Checked: danielwellington.com`

FAILED INFO **Apple Pay markers detected (informational)**

Enable Apple Pay through your payment processor (informational only)

WHAT WE FOUND

Scanned the homepage and 0 sampled PDPs for Apple Pay markers; none matched.

How we checked: Substring match on known Apple Pay SDK/markup signatures (ApplePaySession, apple-pay-button, /apple-developer-merchant-tid-domain-association) across the homepage and every sampled PDP HTML.

- No Apple Pay markers detected on the homepage or PDPs

`Checked: danielwellington.com`

FAILED INFO **Google Pay markers detected (informational)**

Enable Google Pay through your payment processor (informational only)

WHAT WE FOUND

Scanned the homepage and 0 sampled PDPs for Google Pay markers; none matched.

How we checked: Substring match on known Google Pay SDK/markup signatures (pay.google.com/gp/p/js/pay.js, google.payments.api, <google-pay-button) across the homepage and every sampled PDP HTML.

- No Google Pay markers detected on the homepage or PDPs

`Checked: danielwellington.com`

PASSING CRITICAL **Googlebot allowed on product paths**

Allow Googlebot on product paths

WHAT WE FOUND

Checked robots.txt for Googlebot at the representative product path `/products/test`. The active group resolves to Allow.

How we checked: group match on `User-agent: Googlebot` rules at the representative product path `/products/test` (via the parsed robots.txt isAllowed predicate).

`Checked: danielwellington.com/robots.txt, /products/test`

NOT APPLICABLE CRITICAL **Offer price + priceCurrency valid**

Set price as a number and priceCurrency as an ISO 4217 code

WHAT WE FOUND

No PDP carried a Product JSON-LD node, so there is nothing to inspect for Offer price + priceCurrency.

How we checked: Parse Offer `price` (or AggregateOffer `lowPrice`) as a parseable numeric price >= 0; require `priceCurrency` to match /^[A-Z]{3}$/i.

PASSING CRITICAL **OAI-SearchBot allowed**

Allow OAI-SearchBot in robots.txt

WHAT WE FOUND

Checked robots.txt rules for OAI-SearchBot (OpenAI's ChatGPT search/discovery crawler) at path /. The active group resolves to Allow.

How we checked: group match on `User-agent: OAI-SearchBot` rules at path `/` (via the parsed robots.txt isAllowed predicate).

`Checked: danielwellington.com/robots.txt`

PASSING CRITICAL **No global wildcard root disallow**

Remove the wildcard `Disallow: /` from robots.txt

WHAT WE FOUND

Scanned the robots.txt wildcard `User-agent: *` group for a root Disallow (wildcard group present; root Disallow absent).

How we checked: Line-by-line scan of robots.txt; track membership of the `User-agent: *` group (stacked UA lines combine into one group per section 2.2.1) and flag the file when a root `Disallow: /` appears in that group with no offsetting `Allow: /`.

`Checked: danielwellington.com/robots.txt`

PASSING HIGH **Bingbot allowed**

Allow Bingbot in robots.txt

WHAT WE FOUND

Checked robots.txt rules for Bingbot (Microsoft's web crawler — also the source for Copilot Shopping's index) at path /. The active group resolves to Allow.

How we checked: group match on `User-agent: Bingbot` rules at path `/` (via the parsed robots.txt isAllowed predicate).

`Checked: danielwellington.com/robots.txt`

PASSING HIGH **Contact page exposes email or phone**

Add a `mailto:` email link or `tel:` phone link to your contact page

WHAT WE FOUND

Reached the contact page and found a contact method: phone (2026-05-07).

How we checked: URL probe of contact paths; the first 2xx body is scanned for `mailto:` / `tel:` hrefs, plain emails (placeholder hosts excluded), and phone-shaped numbers.

`Checked: danielwellington.com/pages/contact`

NOT APPLICABLE HIGH **MerchantReturnPolicy finite-window has positive merchantReturnDays**

Add a positive `merchantReturnDays` to finite-window return policies

WHAT WE FOUND

No MerchantReturnPolicy node used the MerchantReturnFiniteReturnWindow category, so the `merchantReturnDays` check has nothing to evaluate.

How we checked: For each MerchantReturnPolicy node whose returnPolicyCategory normalizes to MerchantReturnFiniteReturnWindow, require `merchantReturnDays` to be a positive number (or a numeric string > 0).

NOT APPLICABLE HIGH **MerchantReturnPolicy satisfies Option A (country+category) or B (returnLink)**

Make every MerchantReturnPolicy node satisfy Option A or Option B

WHAT WE FOUND

No PDP carried a `hasMerchantReturnPolicy` node, so Option A/B shape cannot be evaluated.

How we checked: For each PDP, walk every `hasMerchantReturnPolicy` node (Product or Offer level) and require either (applicableCountry + returnPolicyCategory) OR a syntactically-valid `merchantReturnLink` URL.

NOT APPLICABLE HIGH **MerchantReturnPolicy node present on Product or Offer**

Emit `hasMerchantReturnPolicy` on Product or Offer JSON-LD

WHAT WE FOUND

No PDP carried a Product JSON-LD node, so there is nothing to inspect for `hasMerchantReturnPolicy`.

How we checked: On each PDP, locate the Product JSON-LD node and check for a `hasMerchantReturnPolicy` object/array at Product level OR Offer level. Pass band >= 85% coverage, partial >= 50%.

NOT APPLICABLE HIGH **Offer `availability` is a Schema.org URL**

Use a canonical Schema.org availability IRI on every Offer

WHAT WE FOUND

No PDP carried a resolvable Offer, so there is nothing to inspect for `availability`.

How we checked: On each Offer, accept `availability` only if it matches one of the canonical Schema.org ItemAvailability IRIs (http or https, trailing slash optional).

NOT APPLICABLE HIGH **Offer JSON-LD carries shippingDetails (OfferShippingDetails)**

Emit shippingDetails (OfferShippingDetails) on Offer JSON-LD

WHAT WE FOUND

No PDP carried a Product JSON-LD node, so there is nothing to inspect for `shippingDetails`.

How we checked: On each PDP, locate the Product JSON-LD node and check for `shippingDetails` (single object or array) at Product or Offer level. Pass band >= 85% coverage.

NOT APPLICABLE HIGH **Sampled PDPs are not gated behind a login wall (401 / 403)**

Open PDPs to anonymous fetches

WHAT WE FOUND

Wanted to evaluate PDP status codes, but the sample is empty.

How we checked: For each sampled PDP, evaluate the captured HTTP status code (preferred) or fall back to body-presence; 401 / 403 fails.

- No product pages were sampled

NOT APPLICABLE HIGH **No sampled PDP returns a noindex directive**

Remove the noindex directive from every PDP

WHAT WE FOUND

Wanted to inspect sampled PDPs for noindex directives, but the sample is empty.

How we checked: For each sampled PDP, inspect the HTML for `<meta name="robots" content="...noindex...">` and the response headers for `X-Robots-Tag:...noindex...`.

- No product pages were sampled

NOT APPLICABLE HIGH **Each PDP carries at most one Product JSON-LD node**

Emit a single Product JSON-LD node per PDP

WHAT WE FOUND

Wanted to count Product JSON-LD nodes per sampled PDP, but the sample is empty.

How we checked: For each sampled PDP, count JSON-LD nodes whose @type is `Product` or whose @type array contains `Product`. Each PDP must expose at most one.

- No product pages were sampled

PASSING HIGH **PerplexityBot allowed**

Allow PerplexityBot in robots.txt

WHAT WE FOUND

Checked robots.txt rules for PerplexityBot (Perplexity's shopping index crawler) at path /. The active group resolves to Allow.

How we checked: group match on `User-agent: PerplexityBot` rules at path `/` (via the parsed robots.txt isAllowed predicate).

`Checked: danielwellington.com/robots.txt`

PASSING HIGH **Privacy policy page reachable**

Publish a privacy policy page and link it from your site nav/footer

WHAT WE FOUND

Found a privacy policy page at https://danielwellington.com/policies/privacy-policy (27163 chars of stripped body text).

How we checked: Discover candidate URLs by scoring homepage nav/footer anchors for privacy/gdpr/cookie keywords, then append platform-conventional paths; probe each with politeFetch and pass on the first 2xx with >=200 stripped-body chars.

`Checked: danielwellington.com/policies/privacy-policy`

NOT APPLICABLE HIGH **Brand attribution on PDPs**

Surface brand attribution on every PDP

WHAT WE FOUND

No PDPs were sampled, so we have nothing to inspect for brand attribution.

How we checked: On each PDP, accept brand attribution from either (a) `extractBrand` on the first Product JSON-LD node OR (b) an HTML brand signal (OG `product:brand`, `brand` meta, `og:brand`, Microdata `itemprop="brand"`).

NOT APPLICABLE HIGH **GTIN coverage on PDPs**

Populate `gtin` on every branded Product node

WHAT WE FOUND

No PDPs were sampled, so we have nothing to inspect for GTINs.

How we checked: Extract gtin / gtin8 / gtin12 / gtin13 / gtin14 from the first Product JSON-LD node on each PDP; validate digit length.

NOT APPLICABLE HIGH **Product `image` populated**

Add a resolvable image URL to every Product node

WHAT WE FOUND

No PDP in the sample carried a Product JSON-LD node, so there is nothing to inspect for `image`.

How we checked: Resolve `image` on each Product node into a list of URL strings (string, array, or ImageObject.url/contentUrl); require at least one non-empty URL.

NOT APPLICABLE HIGH **Product JSON-LD present on PDPs**

Publish a Product JSON-LD block on every PDP

WHAT WE FOUND

No PDPs were sampled, so we have nothing to inspect for Product JSON-LD.

How we checked: Walk each sampled PDP's parsed `jsonLdBlocks`, flatten `@graph` containers, and count the page as passing if any node has `@type` Product / ProductGroup / IndividualProduct / ProductModel.

NOT APPLICABLE HIGH **Product `name` populated**

Populate `name` on every Product JSON-LD node

WHAT WE FOUND

No PDP in the sample carried a Product JSON-LD node, so there is nothing to inspect for `name`.

How we checked: On each PDP with a Product JSON-LD node, require `name` to be a string of length > 0 after trimming.

NOT APPLICABLE HIGH **Product JSON-LD includes `offers`**

Add an `offers` object to every Product node

WHAT WE FOUND

No PDP carried a Product JSON-LD node, so there is nothing to inspect for `offers`.

How we checked: On each Product node, require a resolvable Offer (or first Offer inside an AggregateOffer) via `findOffer`.

PASSING HIGH **Products are machine-discoverable**

Publish a product feed or a crawlable product sitemap

WHAT WE FOUND

Confirmed agents can discover your products via `platform_api` (20 product pages verified).

How we checked: Read the product-discovery cascade result from ctx.discovery. Score by discovery method (feed / platform_api / sitemap_typed -> pass when verifiedProductCount >= MIN_CONFIDENT_PRODUCTS; content_verified -> partial; none or under-threshold -> fail).

`Checked: danielwellington.com`

PASSING HIGH **robots.txt present at root**

Publish a non-empty robots.txt at the site root

WHAT WE FOUND

Reached a non-empty /robots.txt at the site root (5235 bytes, 4 declared Sitemap lines).

How we checked: Check whether the fetcher reached a non-empty /robots.txt at the site root (§2.2.3 access method).

`Checked: danielwellington.com/robots.txt`

PASSING HIGH **Terms of service page reachable**

Publish a terms of service page and link it from your site nav/footer

WHAT WE FOUND

Found a terms-of-service page at https://danielwellington.com/policies/terms-of-service (16887 chars of stripped body text).

How we checked: Discover candidate URLs by scoring homepage nav/footer anchors for terms/tos/legal/conditions keywords, then append platform-conventional paths; probe each with politeFetch and pass on the first 2xx with >=200 stripped-body chars.

`Checked: danielwellington.com/policies/terms-of-service`

NOT APPLICABLE HIGH **UCP profile Cache-Control is shared-cacheable with max-age >= 60s**

Serve `/.well-known/ucp` with `Cache-Control: public, max-age=…`

WHAT WE FOUND

Wanted to inspect the UCP profile's Cache-Control header, but the runner did not surface transport metadata.

How we checked: Parse the `Cache-Control` header on the `/.well-known/ucp` response; require `public`, `max-age >= 60`, and no `no-store`/`no-cache`/`private`.

- Transport metadata not available — runner update pending

`Checked: danielwellington.com/.well-known/ucp`

PASSING HIGH **/.well-known/ucp response Content-Type is application/json**

Serve /.well-known/ucp with `Content-Type: application/json`

WHAT WE FOUND

UCP profile Content-Type is "application/json; charset=utf-8".

How we checked: Check that the `Content-Type` header on /.well-known/ucp starts with `application/json` (optionally with a charset parameter).

`Checked: danielwellington.com/.well-known/ucp`

PASSING HIGH **/.well-known/ucp is publicly fetchable with no auth**

Allow unauthenticated access to /.well-known/ucp

WHAT WE FOUND

Unauthenticated GET /.well-known/ucp returned 200.

How we checked: Confirm an unauthenticated GET to /.well-known/ucp returns a 2xx status.

`Checked: danielwellington.com/.well-known/ucp`

PASSING HIGH **/.well-known/ucp returns 200 directly with no redirects**

Serve /.well-known/ucp directly with a 200 response

WHAT WE FOUND

GET /.well-known/ucp returned 200 directly with no redirect chain.

How we checked: Inspect the final HTTP status of GET /.well-known/ucp and whether any 3xx redirect was followed to reach it.

`Checked: danielwellington.com/.well-known/ucp`

PASSING HIGH **/.well-known/ucp profile is present with a `version` field**

Publish `/.well-known/ucp` with at minimum a `version` field

WHAT WE FOUND

Found /.well-known/ucp with version "2026-04-08".

How we checked: Confirm `ctx.wellKnownUcp` is non-null and carries a non-empty `version` string (the only universally-required UCP profile field).

`Checked: danielwellington.com/.well-known/ucp`

PASSING HIGH **Each service satisfies the transport-conditional field requirements**

Populate the conditional fields required by each service's transport

WHAT WE FOUND

Validated 2 services with recognised transports (2 satisfy their transport's required fields).

How we checked: For each services[] entry with a recognised transport, require the transport-conditional fields: rest/mcp -> endpoint+schema; a2a -> endpoint; embedded -> schema.

`Checked: danielwellington.com/.well-known/ucp`

PASSING HIGH **Each service `transport` is rest, mcp, a2a, or embedded**

Set transport to one of rest, mcp, a2a, or embedded

WHAT WE FOUND

Validated 2 service transports (2 match the UCP enum).

How we checked: For each services[] entry, require `transport` to be one of: rest, mcp, a2a, embedded.

`Checked: danielwellington.com/.well-known/ucp`

PASSING HIGH **UCP profile declares a valid shopping service entry**

Declare a shopping service entry with a recognised transport and an HTTPS endpoint

WHAT WE FOUND

Found 1 valid shopping service entry (transport + HTTPS endpoint).

How we checked: List every services[] entry whose namespace is `shopping` (or contains `shopping`) and require at least one with `transport` {rest,mcp,a2a,embedded} AND a syntactically valid `https://` endpoint.

`Checked: danielwellington.com/.well-known/ucp`

NOT APPLICABLE HIGH **Every signing_keys[] entry is a valid JWK**

Make every signing_keys[] entry a JWK with kty + kty-specific params

WHAT WE FOUND

Profile declares no signing_keys; JWK validation has no entries to evaluate.

How we checked: Walk signing_keys[] and validate each entry per §4.1 (kty required) + §6 (kty-specific required parameters). `kid` is OPTIONAL per §4.5 and not enforced here.

`Checked: danielwellington.com/.well-known/ucp`

NOT APPLICABLE MEDIUM **MerchantReturnPolicy merchantReturnLink URL is reachable**

Repair every merchantReturnLink URL

WHAT WE FOUND

No MerchantReturnPolicy node carried a `merchantReturnLink` URL, so reachability has nothing to evaluate.

How we checked: Collect every unique `merchantReturnLink` URL across all MerchantReturnPolicy nodes; probe each once via politeFetch (failSoft). 2xx counts as reachable.

NOT APPLICABLE MEDIUM **MerchantReturnPolicy applicableCountry uses ISO 3166-1 alpha-2 codes**

Use ISO 3166-1 alpha-2 country codes in applicableCountry

WHAT WE FOUND

No MerchantReturnPolicy node carried `applicableCountry`, so the ISO-code check has nothing to evaluate.

How we checked: On each MerchantReturnPolicy node where `applicableCountry` is set, extract every candidate string and require every one to match /^[A-Z]{2}$/i.

NOT APPLICABLE MEDIUM **MerchantReturnPolicy returnPolicyCategory uses valid Schema.org enum**

Use a valid Schema.org returnPolicyCategory enum value

WHAT WE FOUND

No MerchantReturnPolicy node carried `returnPolicyCategory`, so the enum check has nothing to evaluate.

How we checked: On each MerchantReturnPolicy node where `returnPolicyCategory` is set, accept the bare enum name or the schema.org URL form; reject any other string.

NOT APPLICABLE MEDIUM **OfferShippingDetails shippingDestination is a valid DefinedRegion**

Emit shippingDestination as a DefinedRegion with ISO addressCountry

WHAT WE FOUND

No OfferShippingDetails node carried `shippingDestination`, so the DefinedRegion check has nothing to evaluate.

How we checked: On each OfferShippingDetails node where `shippingDestination` is set, require it to be a DefinedRegion (or array) and every entry to carry `addressCountry` matching /^[A-Z]{2}$/i.

NOT APPLICABLE MEDIUM **OfferShippingDetails shippingRate is a valid MonetaryAmount**

Emit shippingRate as a valid MonetaryAmount

WHAT WE FOUND

No OfferShippingDetails node carried `shippingRate`, so the MonetaryAmount check has nothing to evaluate.

How we checked: On each OfferShippingDetails node where `shippingRate` is set, require an object with numeric value/maxValue (typed or numeric string) and a 3-letter ISO 4217 currency.

NOT APPLICABLE MEDIUM **Product `brand` is a string or Brand/Organization object**

Emit `brand` as either a string or a typed Brand object on every Product

WHAT WE FOUND

No PDP carried a Product JSON-LD node, so there is nothing to inspect for `brand`.

How we checked: On each Product node, accept `brand` if it's a non-empty trimmed string OR an object with a non-empty `name`. Objects with `@type` Brand/Organization but no `name` are rejected.

NOT APPLICABLE MEDIUM **Product `description` present**

Populate `description` on every Product JSON-LD node

WHAT WE FOUND

No PDP carried a Product JSON-LD node, so there is nothing to inspect for `description`.

How we checked: Read `description` on each Product node; strip HTML tags and collapse whitespace; require length > 0.

NOT APPLICABLE MEDIUM **Product `sku` populated**

Populate `sku` on every Product JSON-LD node

WHAT WE FOUND

No PDP carried a Product JSON-LD node, so there is nothing to inspect for `sku`.

How we checked: On each PDP with a Product node, accept `sku` if it is a non-empty trimmed string or a number.

NOT APPLICABLE MEDIUM **Product title not a placeholder**

Replace placeholder and slug-shape titles with real product names

WHAT WE FOUND

No PDP carried a Product JSON-LD node, so there is nothing to inspect for placeholder titles.

How we checked: Read `name` on the first Product JSON-LD node. Fail if empty, matches a known placeholder list (Default Title / Untitled / Product N / sample / test / draft / placeholder), or matches slug shape (lower-case alnum + at least one hyphen).

PASSING MEDIUM **Returns/refund policy page reachable**

Publish a returns policy page and link it from your site nav/footer

WHAT WE FOUND

Found a returns/refund policy page at https://danielwellington.com/policies/refund-policy (6383 chars of stripped body text).

How we checked: Discover candidate URLs by scoring homepage nav/footer anchors for return/refund/exchange keywords, then append platform-conventional paths; probe each with politeFetch and pass on the first 2xx with >=200 stripped-body chars.

```
Checked: danielwellington.com/policies/refund-policy
```

PASSING MEDIUM **Third-party review-platform integration detected**

Install a third-party review platform so agents see syndicated reviews on your storefront

WHAT WE FOUND

Found a third-party review-platform integration (reviews.io, trustpilot) on 1 of 1 scanned source.

How we checked: Substring scan of homepage and sampled PDP HTML for known review-platform asset fingerprints (judge.me, yotpo, stamped.io, reviews.io, okendo, loox, trustpilot, bazaarvoice).

```
Checked: danielwellington.com
```

PASSING MEDIUM **Shipping policy page reachable**

Publish a shipping policy page and link it from your site nav/footer

WHAT WE FOUND

Found a shipping policy page at https://danielwellington.com/policies/shipping-policy (3975 chars of stripped body text).

How we checked: Discover candidate URLs by scoring homepage nav/footer anchors for shipping/delivery/dispatch keywords, then append platform-conventional paths; probe each with politeFetch and pass on the first 2xx with >=200 stripped-body chars.

```
Checked: danielwellington.com/policies/shipping-policy
```

PASSING MEDIUM **Sitemap declared in robots.txt**

Add a `Sitemap:` line to robots.txt

WHAT WE FOUND

Read robots.txt and counted 4 declared Sitemap lines.

How we checked: Read parsed Sitemap: directives from robots.txt (sitemaps.org / implementation note).

```
Checked: danielwellington.com/robots.txt, danielwellington.com/sitemap.xml, danielwelling-
ton.com/cdn/shop/files/faceted-sitemap.xml
```

PASSING MEDIUM **Sitemap <loc> entries are entity-escaped**

Entity-escape `&`, `<`, `>` in every <loc>

WHAT WE FOUND

Sampled 201 <loc> entries across 3 sitemap resource(s); every one is properly entity-escaped.

How we checked: Sample the first 100 <loc> entries per sitemap document and check for raw &, <, or > (sitemaps.org entity escaping rules).

```
Checked: danielwellington.com/sitemap.xml, danielwellington.com/sitemap_agentic_discovery.xml, danielwelling-
ton.com/sitemap_products_1.xml?from=8133598282035&to=15431552696644
```

PASSING MEDIUM **Sitemap resolvable and includes at least one product URL**

Publish a sitemap containing product URLs

WHAT WE FOUND

Resolved an XML sitemap with 1000 <loc> entries; 998 look like product URLs (100%).

How we checked: Parse `<loc>` entries from the resolved sitemap (or sitemap index) and classify each against product-URL patterns (`/products/...`, `/product/...`, `/p/<id>`, etc.).

```
Checked: danielwellington.com/sitemap.xml
```

PASSING MEDIUM **Sitemap entries share the host of the containing sitemap**

Keep every sitemap entry on the sitemap's own host

WHAT WE FOUND

Compared 1477 <loc> entries against their sitemap host across 3 resource(s); every entry shares its sitemap's host.

How we checked: For each resolved sitemap resource, parse the sitemap URL's host and compare it against every parsed <loc> URL's host.

```
Checked: danielwellington.com/sitemap.xml, danielwellington.com/sitemap_agentic_discovery.xml, danielwelling-
ton.com/sitemap_products_1.xml?from=8133598282035&to=15431552696644
```

PASSING MEDIUM **Sitemap root declares the sitemaps.org 0.9 namespace**

Add the sitemaps.org 0.9 xmlns to the root element

WHAT WE FOUND

Inspected 3 sitemap resource(s); every one declares the sitemaps.org 0.9 namespace.

How we checked: Substring-match `xmlns="http://www.sitemaps.org/schemas/sitemap/0.9"` against the raw XML of every resolved sitemap document.

```
Checked: danielwellington.com/sitemap.xml, danielwellington.com/sitemap_agentic_discovery.xml, danielwelling-
ton.com/sitemap_products_1.xml?from=8133598282035&to=15431552696644
```

NOT APPLICABLE MEDIUM **Each capability has version + spec + schema**

Populate version, spec, and schema on every capabilities[] entry

WHAT WE FOUND

Profile declares no capabilities; required-field checks have nothing to evaluate.

How we checked: For each capabilities[] entry, require non-empty string values for `version`, `spec`, and `schema`.

`Checked: danielwellington.com/.well-known/ucp`

PASSING MEDIUM **Each service's `spec` URL origin matches its namespace authority**

Point each service `spec` URL at the canonical UCP authority

WHAT WE FOUND

Validated 2 service spec URLs (2 match the namespace authority).

How we checked: For each service with a `spec` URL, require the URL origin to be a canonical UCP authority OR the host/path to include the namespace token.

`Checked: danielwellington.com/.well-known/ucp`

PASSING MEDIUM **Every service `version` matches YYYY-MM-DD**

Use ISO-date `version` strings on every service

WHAT WE FOUND

Validated 2 service version strings (2 match YYYY-MM-DD).

How we checked: For each services[] entry, require `version` to be a string matching /^\d{4}-\d{2}-\d{2}$/.

`Checked: danielwellington.com/.well-known/ucp`

NOT APPLICABLE LOW **BreadcrumbList present on PDPs**

Add a BreadcrumbList JSON-LD block to every PDP

WHAT WE FOUND

No PDP carried a Product JSON-LD node, so there is no eligible PDP to inspect for BreadcrumbList.

How we checked: Search every JSON-LD block on each PDP for `@type: BreadcrumbList` with a non-empty itemListElement.

PASSING LOW **ChatGPT-User allowed**

Allow ChatGPT-User in robots.txt (advisory)

WHAT WE FOUND

Checked robots.txt rules for ChatGPT-User (OpenAI's user-initiated live fetcher (advisory)) at path /. The active group resolves to Allow.

How we checked: group match on `User-agent: ChatGPT-User` rules at path `/` (via the parsed robots.txt isAllowed predicate).

`Checked: danielwellington.com/robots.txt`

NOT APPLICABLE LOW **Alt text on at least 80% of PDP images**

Add descriptive alt text to product images (WCAG 2.x SC 1.1.1)

WHAT WE FOUND

No PDPs were available in the sample — alt-text coverage could not be evaluated.

How we checked: Per PDP, count <img> tags via regex; a tag 'has alt text' when its `alt` attribute is present AND non-empty after trim. A PDP passes when it carries no <img> at all OR >=80% of its <img> tags have non-empty alt.

NOT APPLICABLE LOW **Product images meet Google’s 50,000-pixel area threshold**

Upload higher-resolution product images (area >= 50,000 pixels)

WHAT WE FOUND

No PDPs were available in the sample — image area could not be evaluated.

How we checked: For every sampled PDP, parse <img> tags and read explicit `width` and `height` attributes; a PDP passes when at least one image has width x height >= 50,000. PDPs without any explicit-dimension <img> are marked indeterminate (this check does not HEAD image URLs).

NOT APPLICABLE LOW **MerchantReturnPolicy enrichment enums use valid Schema.org values**

Use Schema.org enum values for returnFees / returnMethod / refundType

WHAT WE FOUND

No MerchantReturnPolicy node carried returnFees, returnMethod, or refundType, so the enum check has nothing to evaluate.

How we checked: On each MerchantReturnPolicy node, inspect `returnFees`/`returnMethod`/`refundType` if set; require the bare name or schema.org URL form of a value in the corresponding Schema.org enum.

NOT APPLICABLE LOW **Offer `itemCondition` is canonical when present**

Either omit `itemCondition` (defaults to NewCondition) or set it to a canonical IRI

WHAT WE FOUND

No PDP carried a resolvable Offer, so there is nothing to inspect for `itemCondition`.

How we checked: On each Offer: if `itemCondition` is omitted, count as pass (Google defaults to NewCondition). If present, accept only when it matches a canonical Schema.org ItemCondition IRI.

NOT APPLICABLE LOW **OfferShippingDetails deliveryTime is a valid ShippingDeliveryTime**

Emit a ShippingDeliveryTime with handlingTime and/or transitTime populated

WHAT WE FOUND

No OfferShippingDetails node carried `deliveryTime`, so the ShippingDeliveryTime check has nothing to evaluate.

How we checked: On each OfferShippingDetails node where `deliveryTime` is set, require an object with at least one of handlingTime / transitTime populated as a QuantitativeValue.

PASSING LOW **Perplexity-User allowed**

Allow Perplexity-User in robots.txt (advisory)

WHAT WE FOUND

Checked robots.txt rules for Perplexity-User (Perplexity's live user-initiated fetcher (advisory)) at path /. The active group resolves to Allow.

How we checked: group match on `User-agent: Perplexity-User` rules at path `/` (via the parsed robots.txt isAllowed predicate).

`Checked: danielwellington.com/robots.txt`

NOT APPLICABLE LOW **Product `aggregateRating` present**

Add an AggregateRating to Product nodes when you have real reviews

WHAT WE FOUND

No PDP carried a Product JSON-LD node, so there is nothing to inspect for `aggregateRating`.

How we checked: On each Product node, parse `aggregateRating` (or the first element if it's an array) and require ratingValue in [0,5] AND reviewCount or ratingCount >= 1.

NOT APPLICABLE LOW **Product title quality (present, not all-caps)**

Use sentence-case product titles

WHAT WE FOUND

No PDP carried a Product JSON-LD node, so there is nothing to inspect for title quality.

How we checked: Read `name` on the first Product JSON-LD node. Fail if missing/empty after trimming OR if the string contains letters and they're all upper-case.

PASSING LOW **/robots.txt is served as text/plain**

Send Content-Type: text/plain on /robots.txt

WHAT WE FOUND

Inspected /robots.txt Content-Type ("text/plain; charset=utf-8"); recognised as text/plain.

How we checked: Inspect the /robots.txt response `Content-Type` header for a `text/plain` media type per §2.3.

`Checked: danielwellington.com/robots.txt`

PASSING LOW **/robots.txt is under 500 KiB (RFC 9309 §2.5 parser cap)**

Trim /robots.txt below 500 KiB

WHAT WE FOUND

Measured /robots.txt at 5,236 bytes; within the 512,000-byte (500 KiB) cap.

How we checked: Measure the raw byte size of the /robots.txt body and compare against the §2.5 parser cap (>= 500 KiB).

`Checked: danielwellington.com/robots.txt`

PASSING LOW **/robots.txt is served as UTF-8**

Serve /robots.txt as UTF-8

WHAT WE FOUND

Inspected the /robots.txt byte stream (5236 bytes); decodes as UTF-8.

How we checked: Inspect the raw byte stream of /robots.txt for UTF-8 decodability per §2.3.

`Checked: danielwellington.com/robots.txt`

PASSING LOW **Every sitemap <loc> URL is under 2048 characters**

Keep every <loc> URL under 2,048 characters

WHAT WE FOUND

Inspected 1477 <loc> entries across 3 sitemap resource(s); every one is under 2,048 characters.

How we checked: Iterate every parsed <loc> URL across all resolved sitemap resources and check length against the 2,048-character cap.

`Checked: danielwellington.com/sitemap.xml, danielwellington.com/sitemap_agentic_discovery.xml, danielwellington.com/sitemap_products_1.xml?from=8133598282035&to=15431552696644`

PASSING LOW **Sitemap respects 50 MiB / 50,000-URL caps per document**

Split over-cap sitemaps into a sitemap index

WHAT WE FOUND

Checked 3 sitemap resource(s) against the 50 MiB / 50,000-URL caps; every one is within the limits.

How we checked: Check raw byte size (<= 52,428,800 B) and entry count (<= 50,000) for every resolved sitemap resource.

`Checked: danielwellington.com/sitemap.xml, danielwellington.com/sitemap_agentic_discovery.xml, danielwellington.com/sitemap_products_1.xml?from=8133598282035&to=15431552696644`

PASSING LOW **Sitemap is served as UTF-8**

Serve every sitemap document as UTF-8

WHAT WE FOUND

Inspected 3 sitemap resource(s); every one decodes as UTF-8.

How we checked: Inspect every resolved sitemap document's raw byte stream for UTF-8 decodability (sitemaps.org encoding requirement).

`Checked: danielwellington.com/sitemap.xml, danielwellington.com/sitemap_agentic_discovery.xml, danielwellington.com/sitemap_products_1.xml?from=8133598282035&to=15431552696644`

PASSING LOW **UCP MCP-transport entries have valid HTTPS endpoints**

Make every declared MCP transport endpoint an absolute HTTPS URL

WHAT WE FOUND

Validated 1 MCP transport entry; all carry absolute https:// endpoints.

How we checked: Filter services[] to entries where transport=mcp and validate that `endpoint` is an absolute https:// URL.

`Checked: danielwellington.com/.well-known/ucp`

PASSING INFO **llms.txt present (informational)**

Publish an /llms.txt manifest (optional)

WHAT WE FOUND

Reached an /llms.txt at the site root (4352 bytes).

How we checked: Check whether the fetcher reached an /llms.txt at the site root. Informational only — no failure path per llmstxt.org being a voluntary community convention.

`Checked: danielwellington.com/llms.txt`