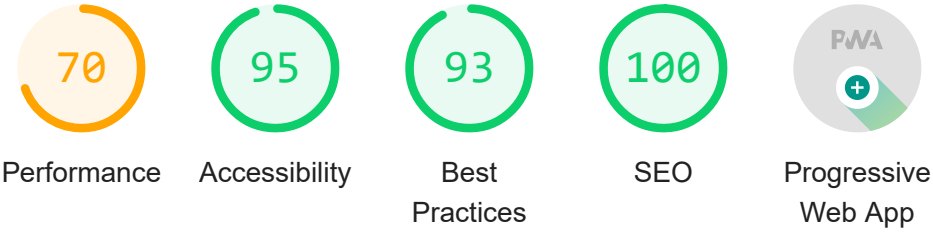
http://localhost:3000/create



▲ 0–49 50–89 90–100

Performance

Metrics

First Contentful Paint	1.3 s	▲ Time to Interactive	10.5 s
Speed Index	1.3 s	Total Blocking Time	210 ms
Largest Contentful Paint	1.4 s	Cumulative Layout Shift	0

Values are estimated and may vary. The performance score is calculated directly from these metrics. See calculator.

View Original Trace

Opportunities — These suggestions can help your page load faster. They don't directly affect the Performance score.

Opportunity	Estimated Savings
Remove unused JavaScript	0.28 s

Remove unused JavaScript to reduce bytes consumed by network activity. Learn more.

If you are not server-side rendering, split your JavaScript bundles with `React.lazy()`. Otherwise, code-split using a third-party library such as loadable-components.

Show 3rd-party resources (0)

URL	Transfer Size	Potential Savings
…js/vendors~main.chunk.js (localhost)	869.5 KiB	360.9 KiB

Diagnostics — More information about the performance of your application. These numbers don't directly affect the Performance score.

▲ Avoid enormous network payloads — Total size was 12,087 KiB

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

Show 3rd-party resources (0)

URL	Transfer Size
/image-beach1.jpg (localhost)	4,030.1 KiB
/image-beach3.jpg (localhost)	3,701.5 KiB
/image-beach2.jpg (localhost)	3,464.4 KiB
…js/vendors~main.chunk.js (localhost)	869.5 KiB
…js/main.chunk.js (localhost)	10.6 KiB
…js/bundle.js (localhost)	7.2 KiB
/script.js (localhost)	1.2 KiB
/create (localhost)	1.2 KiB
/Home.jsx (localhost)	0.9 KiB
/sw_cached_site.js (localhost)	0.0 KiB

▲ Serve static assets with an efficient cache policy — 3 resources found

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

Show 3rd-party resources (0)

URL	Cache TTL	Transfer Size
…js/vendors~main.chunk.js (localhost)	None	870 KiB
…js/main.chunk.js (localhost)	None	11 KiB
…js/bundle.js (localhost)	None	7 KiB

Avoid chaining critical requests — 3 chains found

The Critical Request Chains below show you what resources are loaded with a high priority. Consider reducing the length of chains, reducing the download size of resources, or deferring the download of unnecessary resources to improve page load. Learn more.

Maximum critical path latency: **390 ms**

Initial Navigation

/create (localhost)

…js/bundle.js (localhost) **- 40 ms, 7.23 KiB**

…js/vendors~main.chunk.js (localhost) **- 320 ms, 869.54 KiB**

…js/main.chunk.js (localhost) **- 40 ms, 10.57 KiB**

Keep request counts low and transfer sizes small — 10 requests • 12,087 KiB

To set budgets for the quantity and size of page resources, add a budget.json file. Learn more.

Resource Type	Requests	Transfer Size
Total	10	12,086.6 KiB
Other	5	11,196.9 KiB
Script	4	888.5 KiB
Document	1	1.2 KiB
Stylesheet	0	0.0 KiB
Image	0	0.0 KiB
Media	0	0.0 KiB
Font	0	0.0 KiB
Third-party	0	0.0 KiB

Largest Contentful Paint element — 1 element found

This is the largest contentful element painted within the viewport. Learn More

Element

h1

Avoid long main-thread tasks — 1 long task found

Lists the longest tasks on the main thread, useful for identifying worst contributors to input delay. Learn more

☐ ~~Show 3rd-party resources~~ (0)

URL	Start Time	Duration
…js/vendors~main.chunk.js (localhost)	10,249 ms	259 ms

Passed audits (30)

Eliminate render-blocking resources

Resources are blocking the first paint of your page. Consider delivering critical JS/CSS inline and deferring all non-critical JS/styles. Learn more.

Properly size images

Serve images that are appropriately-sized to save cellular data and improve load time. Learn more.

Defer offscreen images

Consider lazy-loading offscreen and hidden images after all critical resources have finished loading to lower time to interactive. Learn more.

Minify CSS — Potential savings of 138 KiB

Minifying CSS files can reduce network payload sizes. Learn more.

If your build system minifies CSS files automatically, ensure that you are deploying the production build of your application. You can check this with the React Developer Tools extension. Learn more.

☐ ~~Show 3rd-party resources~~ (0)

URL	Transfer Size	Potential Savings
`/*! * Bootstrap v4.6.0 (https://getbootstrap.com/) * Copyright 2011-2021 The Bootstrap Authors * ...`	167.2 KiB	138.4 KiB

Minify JavaScript

Minifying JavaScript files can reduce payload sizes and script parse time. Learn more.

If your build system minifies JS files automatically, ensure that you are deploying the production build of your application. You can check this with the React Developer Tools extension. Learn more.

Remove unused CSS — Potential savings of 166 KiB

Remove dead rules from stylesheets and defer the loading of CSS not used for above-the-fold content to reduce unnecessary bytes consumed by network activity. Learn more.

☐ ~~Show 3rd-party resources~~ (0)

URL	Transfer Size	Potential Savings
`/*! * Bootstrap v4.6.0 (https://getbootstrap.com/) * Copyright 2011-2021 The Bootstrap Authors * ...`	167.2 KiB	166.5 KiB

Efficiently encode images

Optimized images load faster and consume less cellular data. Learn more.

Serve images in next-gen formats

Image formats like JPEG 2000, JPEG XR, and WebP often provide better compression than PNG or JPEG, which means faster downloads and less data consumption. Learn more.

Enable text compression

Text-based resources should be served with compression (gzip, deflate or brotli) to minimize total network bytes. Learn more.

Preconnect to required origins

Consider adding `preconnect` or `dns-prefetch` resource hints to establish early connections to important third-party origins. Learn more.

Initial server response time was short — Root document took 10 ms

Keep the server response time for the main document short because all other requests depend on it. Learn more.

Show 3rd-party resources (0)

URL	Time Spent
/create (localhost)	10 ms

Avoid multiple page redirects

Redirects introduce additional delays before the page can be loaded. Learn more.

If you are using React Router, minimize usage of the `<Redirect>` component for route navigations.

Preload key requests

Consider using `<link rel=preload>` to prioritize fetching resources that are currently requested later in page load. Learn more.

Use HTTP/2

HTTP/2 offers many benefits over HTTP/1.1, including binary headers and multiplexing. Learn more.

Use video formats for animated content

Large GIFs are inefficient for delivering animated content. Consider using MPEG4/WebM videos for animations and PNG/WebP for static images instead of GIF to save network bytes. Learn more

Remove duplicate modules in JavaScript bundles

Remove large, duplicate JavaScript modules from bundles to reduce unnecessary bytes consumed by network activity.

Avoid serving legacy JavaScript to modern browsers

Polyfills and transforms enable legacy browsers to use new JavaScript features. However, many aren't necessary for modern browsers. For your bundled JavaScript, adopt a modern script deployment strategy using module/nomodule feature detection to reduce the amount of code shipped to modern browsers, while retaining support for legacy browsers. Learn More

Preload Largest Contentful Paint image

Preload the image used by the LCP element in order to improve your LCP time. Learn more.

Avoids an excessive DOM size — 33 elements

A large DOM will increase memory usage, cause longer style calculations, and produce costly layout reflows. Learn more.

Consider using a "windowing" library like `react-window` to minimize the number of DOM nodes created if you are rendering many repeated elements on the page. Learn more. Also, minimize unnecessary re-renders using `shouldComponentUpdate`, `PureComponent`, or `React.memo` and skip effects only until certain dependencies have changed if you are using the `Effect` hook to improve runtime performance.

Statistic	Element	Value
Total DOM Elements		33
Maximum DOM Depth	 input.from-control	6
Maximum Child Elements	 nav.navbar.bg-dark.container	7

User Timing marks and measures

Consider instrumenting your app with the User Timing API to measure your app's real-world performance during key user experiences. Learn more.

Use the React DevTools Profiler, which makes use of the Profiler API, to measure the rendering performance of your components. Learn more.

JavaScript execution time — 0.2 s

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

Show 3rd-party resources (0)

URL	Total CPU Time	Script Evaluation	Script Parse
…js/vendors~main.chunk.js (localhost)	147 ms	2 ms	73 ms
…js/main.chunk.js (localhost)	109 ms	108 ms	2 ms
Unattributable	98 ms	3 ms	0 ms
/create (localhost)	50 ms	3 ms	2 ms

Minimizes main-thread work — 0.4 s

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

Category	Time Spent
Other	179 ms
Script Evaluation	116 ms
Script Parsing & Compilation	78 ms
Parse HTML & CSS	15 ms
Garbage Collection	12 ms
Style & Layout	4 ms
Rendering	3 ms

All text remains visible during webfont loads

Leverage the font-display CSS feature to ensure text is user-visible while webfonts are loading. Learn more.

Minimize third-party usage

Third-party code can significantly impact load performance. Limit the number of redundant third-party providers and try to load third-party code after your page has primarily finished loading. Learn more.

Lazy load third-party resources with facades

Some third-party embeds can be lazy loaded. Consider replacing them with a facade until they are required. Learn more.

Avoid large layout shifts

These DOM elements contribute most to the CLS of the page.

Uses passive listeners to improve scrolling performance

Consider marking your touch and wheel event listeners as `passive` to improve your page's scroll performance. Learn more.

Avoids `document.write()`

For users on slow connections, external scripts dynamically injected via `document.write()` can delay page load by tens of seconds. Learn more.

Avoid non-composited animations

Animations which are not composited can be janky and increase CLS. Learn more

Image elements have explicit `width` and `height`

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

95

Accessibility

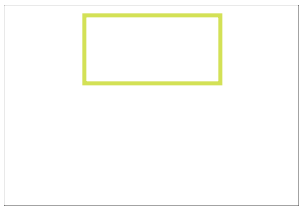
These checks highlight opportunities to improve the accessibility of your web app. Only a subset of accessibility issues can be automatically detected so manual testing is also encouraged.

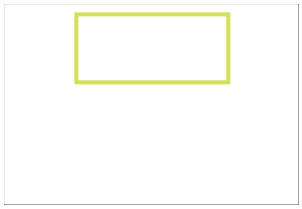
Contrast — These are opportunities to improve the legibility of your content.

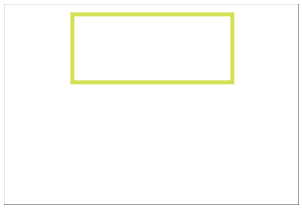
▲ Background and foreground colors do not have a sufficient contrast ratio.

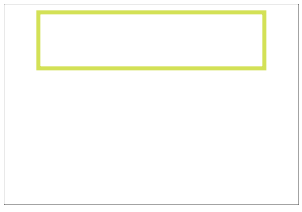
Low-contrast text is difficult or impossible for many users to read. Learn more.

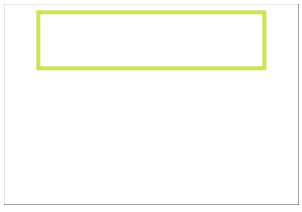
Failing Elements

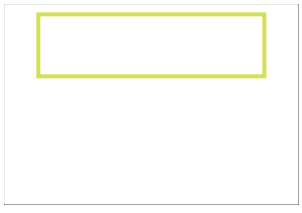

a.Link


a.Link


a.Link


a.Link


a.Link


a.Link

Failing Elements

a.Link

button.btn.btn-lg.btn-info

Additional items to manually check (10) — These items address areas which an automated testing tool cannot cover. Learn more in our guide on conducting an accessibility review.

The page has a logical tab order

Tabbing through the page follows the visual layout. Users cannot focus elements that are offscreen. Learn more.

Interactive controls are keyboard focusable

Custom interactive controls are keyboard focusable and display a focus indicator. Learn more.

Interactive elements indicate their purpose and state

Interactive elements, such as links and buttons, should indicate their state and be distinguishable from non-interactive elements. Learn more.

The user's focus is directed to new content added to the page

If new content, such as a dialog, is added to the page, the user's focus is directed to it. Learn more.

User focus is not accidentally trapped in a region

A user can tab into and out of any control or region without accidentally trapping their focus. Learn more.

Custom controls have associated labels

Custom interactive controls have associated labels, provided by aria-label or aria-labelledby. Learn more.

Custom controls have ARIA roles

Custom interactive controls have appropriate ARIA roles. Learn more.

Visual order on the page follows DOM order

DOM order matches the visual order, improving navigation for assistive technology. Learn more.

Offscreen content is hidden from assistive technology

Offscreen content is hidden with display: none or aria-hidden=true. Learn more.

HTML5 landmark elements are used to improve navigation

Landmark elements (<main>, <nav>, etc.) are used to improve the keyboard navigation of the page for assistive technology. Learn more.

Passed audits (10)

`[aria-hidden="true"]` is not present on the document `<body>`

Assistive technologies, like screen readers, work inconsistently when `aria-hidden="true"` is set on the document `<body>`. Learn more.

Buttons have an accessible name

When a button doesn't have an accessible name, screen readers announce it as "button", making it unusable for users who rely on screen readers. Learn more.

The page contains a heading, skip link, or landmark region

Adding ways to bypass repetitive content lets keyboard users navigate the page more efficiently. Learn more.

Document has a `<title>` element

The title gives screen reader users an overview of the page, and search engine users rely on it heavily to determine if a page is relevant to their search. Learn more.

Heading elements appear in a sequentially-descending order

Properly ordered headings that do not skip levels convey the semantic structure of the page, making it easier to navigate and understand when using assistive technologies. Learn more.

`<html>` element has a `[lang]` attribute

If a page doesn't specify a lang attribute, a screen reader assumes that the page is in the default language that the user chose when setting up the screen reader. If the page isn't actually in the default language, then the screen reader might not announce the page's text correctly. Learn more.

`<html>` element has a valid value for its `[lang]` attribute

Specifying a valid BCP 47 language helps screen readers announce text properly. Learn more.

Form elements have associated labels

Labels ensure that form controls are announced properly by assistive technologies, like screen readers. Learn more.

Links have a discernible name

Link text (and alternate text for images, when used as links) that is discernible, unique, and focusable improves the navigation experience for screen reader users. Learn more.

`[user-scalable="no"]` is not used in the `<meta name="viewport">` element and the `[maximum-scale]` attribute is not less than 5.

Disabling zooming is problematic for users with low vision who rely on screen magnification to properly see the contents of a web page. Learn more.

Not applicable (33)

`[accesskey]` values are unique

Access keys let users quickly focus a part of the page. For proper navigation, each access key must be unique. Learn more.

`[aria-*]` attributes match their roles

Each ARIA `role` supports a specific subset of `aria-*` attributes. Mismatching these invalidates the `aria-*` attributes. Learn more.

`button`, `link`, and `menuitem` elements have accessible names

When an element doesn't have an accessible name, screen readers announce it with a generic name, making it unusable for users who rely on screen readers. Learn more.

`[aria-hidden="true"]` elements do not contain focusable descendents

Focusable descendents within an `[aria-hidden="true"]` element prevent those interactive elements from being available to users of assistive technologies like screen readers. Learn more.

ARIA input fields have accessible names

When an input field doesn't have an accessible name, screen readers announce it with a generic name, making it unusable for users who rely on screen readers. Learn more.

ARIA `meter` elements have accessible names

When an element doesn't have an accessible name, screen readers announce it with a generic name, making it unusable for users who rely on screen readers. Learn more.

ARIA `progressbar` elements have accessible names

When an element doesn't have an accessible name, screen readers announce it with a generic name, making it unusable for users who rely on screen readers. Learn more.

`[role]`s have all required `[aria-*]` attributes

Some ARIA roles have required attributes that describe the state of the element to screen readers. Learn more.

Elements with an ARIA `[role]` that require children to contain a specific `[role]` have all required children.

Some ARIA parent roles must contain specific child roles to perform their intended accessibility functions. Learn more.

`[role]`s are contained by their required parent element

Some ARIA child roles must be contained by specific parent roles to properly perform their intended accessibility functions. Learn more.

`[role]` values are valid

ARIA roles must have valid values in order to perform their intended accessibility functions. Learn more.

ARIA toggle fields have accessible names

When a toggle field doesn't have an accessible name, screen readers announce it with a generic name, making it unusable for users who rely on screen readers. Learn more.

ARIA `tooltip` elements have accessible names

When an element doesn't have an accessible name, screen readers announce it with a generic name, making it unusable for users who rely on screen readers. Learn more.

ARIA `treeitem` elements have accessible names

When an element doesn't have an accessible name, screen readers announce it with a generic name, making it unusable for users who rely on screen readers. Learn more.

`[aria-*]` attributes have valid values

Assistive technologies, like screen readers, can't interpret ARIA attributes with invalid values. Learn more.

`[aria-*]` attributes are valid and not misspelled

Assistive technologies, like screen readers, can't interpret ARIA attributes with invalid names. Learn more.

`<dl>`'s contain only properly-ordered `<dt>` and `<dd>` groups, `<script>`, `<template>` or `<div>` elements.

When definition lists are not properly marked up, screen readers may produce confusing or inaccurate output. Learn more.

Definition list items are wrapped in `<dl>` elements

Definition list items (`<dt>` and `<dd>`) must be wrapped in a parent `<dl>` element to ensure that screen readers can properly announce them. Learn more.

`[id]` attributes on active, focusable elements are unique

All focusable elements must have a unique `id` to ensure that they're visible to assistive technologies. Learn more.

ARIA IDs are unique

The value of an ARIA ID must be unique to prevent other instances from being overlooked by assistive technologies. Learn more.

No form fields have multiple labels

Form fields with multiple labels can be confusingly announced by assistive technologies like screen readers which use either the first, the last, or all of the labels. Learn more.

`<frame>` or `<iframe>` elements have a title

Screen reader users rely on frame titles to describe the contents of frames. Learn more.

Image elements have `[alt]` attributes

Informative elements should aim for short, descriptive alternate text. Decorative elements can be ignored with an empty alt attribute. Learn more.

`<input type="image">` elements have `[alt]` text

When an image is being used as an `<input>` button, providing alternative text can help screen reader users understand the purpose of the button. Learn more.

Lists contain only `<li>` elements and script supporting elements (`<script>` and `<template>`).

Screen readers have a specific way of announcing lists. Ensuring proper list structure aids screen reader output. Learn more.

List items (`<li>`) are contained within `<ul>` or `<ol>` parent elements

Screen readers require list items (`<li>`) to be contained within a parent `<ul>` or `<ol>` to be announced properly. Learn more.

The document does not use `<meta http-equiv="refresh">`

Users do not expect a page to refresh automatically, and doing so will move focus back to the top of the page. This may create a frustrating or confusing experience. Learn more.

`<object>` elements have `[alt]` text

Screen readers cannot translate non-text content. Adding alt text to `<object>` elements helps screen readers convey meaning to users. Learn more.

No element has a `[tabindex]` value greater than 0

A value greater than 0 implies an explicit navigation ordering. Although technically valid, this often creates frustrating experiences for users who rely on assistive technologies. Learn more.

Cells in a `<table>` element that use the `[headers]` attribute refer to table cells within the same table.

Screen readers have features to make navigating tables easier. Ensuring `<td>` cells using the `[headers]` attribute only refer to other cells in the same table may improve the experience for screen reader users. Learn more.

`<th>` elements and elements with `[role="columnheader"/"rowheader"]` have data cells they describe.

Screen readers have features to make navigating tables easier. Ensuring table headers always refer to some set of cells may improve the experience for screen reader users. Learn more.

`[lang]` attributes have a valid value

Specifying a valid BCP 47 language on elements helps ensure that text is pronounced correctly by a screen reader. Learn more.

`<video>` elements contain a `<track>` element with `[kind="captions"]`

When a video provides a caption it is easier for deaf and hearing impaired users to access its information. Learn more.

Best Practices

General

Uses deprecated APIs — 1 warning found

Deprecated APIs will eventually be removed from the browser. Learn more.

Show 3rd-party resources (0)

Deprecation / Warning	Source
SharedArrayBuffer will require cross-origin isolation as of M91, around May 2021. See https://developer.chrome.com/blog/enabling-shared-array-buffer/ for more details.	vendors~main.chunk.js:87136

Passed audits (16)

Uses HTTPS

All sites should be protected with HTTPS, even ones that don't handle sensitive data. This includes avoiding mixed content, where some resources are loaded over HTTP despite the initial request being served over HTTPS. HTTPS prevents intruders from tampering with or passively listening in on the communications between your app and your users, and is a prerequisite for HTTP/2 and many new web platform APIs. Learn more.

Links to cross-origin destinations are safe

Add `rel="noopener"` or `rel="noreferrer"` to any external links to improve performance and prevent security vulnerabilities. Learn more.

Avoids requesting the geolocation permission on page load

Users are mistrustful of or confused by sites that request their location without context. Consider tying the request to a user action instead. Learn more.

Avoids requesting the notification permission on page load

Users are mistrustful of or confused by sites that request to send notifications without context. Consider tying the request to user gestures instead. Learn more.

Avoids front-end JavaScript libraries with known security vulnerabilities

Some third-party scripts may contain known security vulnerabilities that are easily identified and exploited by attackers. Learn more.

Allows users to paste into password fields

Preventing password pasting undermines good security policy. Learn more.

Displays images with correct aspect ratio

Image display dimensions should match natural aspect ratio. Learn more.

Serves images with appropriate resolution

Image natural dimensions should be proportional to the display size and the pixel ratio to maximize image clarity. Learn more.

Page has the HTML doctype

Specifying a doctype prevents the browser from switching to quirks-mode. Learn more.

Properly defines charset

A character encoding declaration is required. It can be done with a `<meta>` tag in the first 1024 bytes of the HTML or in the Content-Type HTTP response header. Learn more.

Avoids `unload` event listeners

The `unload` event does not fire reliably and listening for it can prevent browser optimizations like the Back-Forward Cache. Consider using the `pagehide` or `visibilitychange` events instead. Learn more

Avoids Application Cache

Application Cache is deprecated. Learn more.

Detected JavaScript libraries

All front-end JavaScript libraries detected on the page. Learn more.

Name	Version
React	

No browser errors logged to the console

Errors logged to the console indicate unresolved problems. They can come from network request failures and other browser concerns. Learn more

Page has valid source maps

Source maps translate minified code to the original source code. This helps developers debug in production. In addition, Lighthouse is able to provide further insights. Consider deploying source maps to take advantage of these benefits. Learn more.

☐ ~~Show 3rd-party resources~~ (0)

URL	Map URL
…js/vendors~main.chunk.js (localhost) `Error: Timed out fetching resource.`	…js/vendors~main.chunk.js.map (localhost)
…js/main.chunk.js (localhost) `Error: Timed out fetching resource.`	…js/main.chunk.js.map (localhost)
…js/bundle.js (localhost) `Error: Timed out fetching resource.`	…js/bundle.js.map (localhost)

No issues in the `Issues` panel in Chrome Devtools

Issues logged to the `Issues` panel in Chrome Devtools indicate unresolved problems. They can come from network request failures, insufficient security controls, and other browser concerns. Open up the Issues panel in Chrome DevTools for more details on each issue.

Not applicable (1)

Fonts with `font-display: optional` are preloaded

Preload `optional` fonts so first-time visitors may use them. Learn more

SEO

These checks ensure that your page is optimized for search engine results ranking. There are additional factors Lighthouse does not check that may affect your search ranking. Learn more.

Additional items to manually check (1) — Run these additional validators on your site to check additional SEO best practices.

Structured data is valid

Run the Structured Data Testing Tool and the Structured Data Linter to validate structured data. Learn more.

Passed audits (9)

Has a `<meta name="viewport">` tag with `width` or `initial-scale`

Add a `<meta name="viewport">` tag to optimize your app for mobile screens. Learn more.

Document has a `<title>` element

The title gives screen reader users an overview of the page, and search engine users rely on it heavily to determine if a page is relevant to their search. Learn more.

Document has a meta description

Meta descriptions may be included in search results to concisely summarize page content. Learn more.

Page has successful HTTP status code

Pages with unsuccessful HTTP status codes may not be indexed properly. Learn more.

Links have descriptive text

Descriptive link text helps search engines understand your content. Learn more.

Links are crawlable

Search engines may use `href` attributes on links to crawl websites. Ensure that the `href` attribute of anchor elements links to an appropriate destination, so more pages of the site can be discovered. Learn More

Page isn't blocked from indexing

Search engines are unable to include your pages in search results if they don't have permission to crawl them. Learn more.

Document has a valid `hreflang`

hreflang links tell search engines what version of a page they should list in search results for a given language or region. Learn more.

Document avoids plugins

Search engines can't index plugin content, and many devices restrict plugins or don't support them. Learn more.

Not applicable (5)

robots.txt is valid

If your robots.txt file is malformed, crawlers may not be able to understand how you want your website to be crawled or indexed. Learn more.

Image elements have `[alt]` attributes

Informative elements should aim for short, descriptive alternate text. Decorative elements can be ignored with an empty alt attribute. Learn more.

Document has a valid `rel=canonical`

Canonical links suggest which URL to show in search results. Learn more.

Document uses legible font sizes

Font sizes less than 12px are too small to be legible and require mobile visitors to "pinch to zoom" in order to read. Strive to have >60% of page text ≥12px. Learn more.

Tap targets are sized appropriately

Interactive elements like buttons and links should be large enough (48x48px), and have enough space around them, to be easy enough to tap without overlapping onto other elements. Learn more.

Progressive Web App

These checks validate the aspects of a Progressive Web App. Learn more.

Installable

Web app manifest and service worker meet the installability requirements

Service worker is the technology that enables your app to use many Progressive Web App features, such as offline, add to homescreen, and push notifications. With proper service worker and manifest implementations, browsers can proactively prompt users to add your app to their homescreen, which can lead to higher engagement. Learn more.

PWA Optimized

Registers a service worker that controls page and `start_url`

The service worker is the technology that enables your app to use many Progressive Web App features, such as offline, add to homescreen, and push notifications. Learn more.

Redirects HTTP traffic to HTTPS

If you've already set up HTTPS, make sure that you redirect all HTTP traffic to HTTPS in order to enable secure web features for all your users. Learn more.

Configured for a custom splash screen

A themed splash screen ensures a high-quality experience when users launch your app from their homescreens. Learn more.

Sets a theme color for the address bar.

The browser address bar can be themed to match your site. Learn more.

Content is sized correctly for the viewport

If the width of your app's content doesn't match the width of the viewport, your app might not be optimized for mobile screens. Learn more.

Has a `<meta name="viewport">` tag with `width` or `initial-scale`

Add a `<meta name="viewport">` tag to optimize your app for mobile screens. Learn more.

Provides a valid `apple-touch-icon`

For ideal appearance on iOS when users add a progressive web app to the home screen, define an `apple-touch-icon`. It must point to a non-transparent 192px (or 180px) square PNG. Learn More.

▲ Manifest doesn't have a maskable icon

A maskable icon ensures that the image fills the entire shape without being letterboxed when installing the app on a device. Learn more.

Additional items to manually check (3) — These checks are required by the baseline PWA Checklist but are not automatically checked by Lighthouse. They do not affect your score but it's important that you verify them manually.

Site works cross-browser

To reach the most number of users, sites should work across every major browser. Learn more.

Page transitions don't feel like they block on the network

Transitions should feel snappy as you tap around, even on a slow network. This experience is key to a user's perception of performance. Learn more.

Each page has a URL

Ensure individual pages are deep linkable via URL and that URLs are unique for the purpose of shareability on social media. Learn more.

Runtime Settings

URL	http://localhost:3000/create
Fetch Time	Apr 25, 2021, 7:50 PM GMT+2
Device	Emulated Desktop
Network throttling	40 ms TCP RTT, 10,240 Kbps throughput (Simulated)
CPU throttling	1x slowdown (Simulated)
Channel	devtools
User agent (host)	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/90.0.4430.85 Safari/537.36
User agent (network)	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/90.0.4420.0 Safari/537.36 Chrome-Lighthouse
CPU/Memory Power	973
Axe version	4.1.2

Generated by **Lighthouse** 7.2.0 | File an issue