




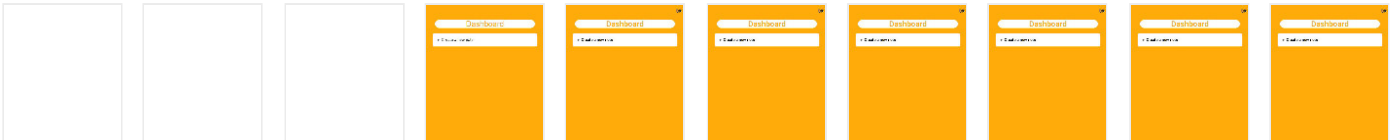
# Performance

Metrics

 First Contentful Paint	1.7 s	Time to Interactive	1.9 s
Speed Index	1.7 s	Total Blocking Time	210 ms
Largest Contentful Paint	1.8 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
Minify JavaScript	0.24 s ^

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.](#)

☐ Show 3rd-party resources (0)

URL	Transfer Size	Potential Savings
...js/0.chunk.js (localhost)	647.2 KiB	246.1 KiB
...js/bundle.js (localhost)	14.1 KiB	6.7 KiB
...js/main.chunk.js (localhost)	21 KiB	5.4 KiB

Remove unused JavaScript	0.16 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/0.chunk.js (localhost)	647.2 KiB	216.9 KiB
...front-end/node_modules/react-dom/cjs/react-dom.development.js	178.2 KiB	76.1 KiB
...front-end/node_modules/lodash/lodash.js	102.5 KiB	29.2 KiB
...front-end/node_modules/react-error-overlay/lib/index.js	73.5 KiB	11.5 KiB
...front-end/node_modules/history/esm/history.js	5.9 KiB	4 KiB
...front-end/node_modules/react-redux/es/components/connectAdvanced.js	4.2 KiB	3.8 KiB

**Diagnostics** — More information about the performance of your application. These numbers don't [directly affect](#) the Performance score.

▲ 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/0.chunk.js (localhost)	None	647 KiB
...js/main.chunk.js (localhost)	None	21 KiB
...js/bundle.js (localhost)	None	14 KiB

Avoid chaining critical requests — 9 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: **1,320 ms**

Initial Navigation

- /dashboard (localhost)
- ...css/bootstrap.min.css (maxcdn.bootstrapcdn.com) - **60 ms, 19.37 KiB**
- /css?family=PT+Serif|Open+Sans:300,400,600,700,800 (fonts.googleapis.com) - **140 ms, 1.3 KiB**
- /css2?family=Roboto:wght@300;400;500&display=swap (fonts.googleapis.com)
- ...v20/KFOICnqEu....woff2 (fonts.gstatic.com) - **40 ms, 11.14 KiB**
- ...js/bootstrap.min.js (maxcdn.bootstrapcdn.com) - **80 ms, 9.86 KiB**
- /0075a9b63d.js (kit.fontawesome.com) - **470 ms, 3.74 KiB**
- ...js/bundle.js (localhost) - **50 ms, 14.14 KiB**
- ...js/0.chunk.js (localhost) - **440 ms, 647.2 KiB**
- ...js/main.chunk.js (localhost) - **320 ms, 20.95 KiB**
- ...webfonts/free-fa-solid-900.woff2 (ka-f.fontawesome.com) - **70 ms, 86.81 KiB**

User Timing marks and measures — 108 user timings ^

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.](#)

Name	Type	Start Time	Duration
⌘ (React Tree Reconciliation: Completed Root)	Measure	972.74 ms	58.35 ms
⌘ Provider [mount]	Measure	977.77 ms	52.5 ms
⌘ App [mount]	Measure	981.7 ms	48.52 ms
⌘ Routes [mount]	Measure	982.48 ms	47.71 ms
⌘ BrowserRouter [mount]	Measure	984.43 ms	45.71 ms
⌘ Router [mount]	Measure	993.21 ms	36.9 ms
⌘ Switch [mount]	Measure	994.28 ms	35.77 ms
⌘ AuthenticatedRoute [mount]	Measure	998.81 ms	31.2 ms
⌘ Dashboard [mount]	Measure	1,001.13 ms	28.74 ms
⌘ LogOutButton [mount]	Measure	1,009.28 ms	8.42 ms
⌘ withRouter(LinkContainer) [mount]	Measure	1,022.8 ms	5.94 ms
⌘ LinkContainer [mount]	Measure	1,023.64 ms	5.05 ms
⌘ Route [mount]	Measure	1,024.85 ms	3.77 ms
⌘ Route [mount]	Measure	1,024.85 ms	5.11 ms
⌘ ListGroupItem [mount]	Measure	1,025.64 ms	2.81 ms
⌘ (Committing Changes)	Measure	1,032.6 ms	11.98 ms
⌘ (Committing Snapshot Effects: 0 Total)	Measure	1,032.9 ms	4.78 ms
⌘ (Committing Host Effects: 6 Total)	Measure	1,037.81 ms	2.66 ms
⌘ (Calling Lifecycle Methods: 6 Total)	Measure	1,040.8 ms	3.63 ms
⌘ Route.componentDidMount	Measure	1,042.06 ms	0.73 ms
⌘ Router.componentDidMount	Measure	1,043.9 ms	0.17 ms
⌘ BrowserRouter.componentDidMount	Measure	1,044.11 ms	0.13 ms
⌘ (React Tree Reconciliation: Completed Root)	Measure	1,215.54 ms	13.94 ms
⌘ Dashboard [update]	Measure	1,216.91 ms	12.42 ms
⌘ LogOutButton [update]	Measure	1,223.17 ms	1.7 ms
⌘ withRouter(LinkContainer) [update]	Measure	1,225.82 ms	3.28 ms
⌘ LinkContainer [update]	Measure	1,226.18 ms	2.87 ms
⌘ Route [update]	Measure	1,227.34 ms	1.67 ms
⌘ ListGroupItem [update]	Measure	1,228.29 ms	0.65 ms

Name	Type	Start Time	Duration
⌘ (Committing Changes)	Measure	1,229.55 ms	2.56 ms
⌘ (Committing Snapshot Effects: 0 Total)	Measure	1,229.6 ms	0.69 ms
⌘ (Committing Host Effects: 3 Total)	Measure	1,230.32 ms	0.96 ms
⌘ (Calling Lifecycle Methods: 3 Total)	Measure	1,231.33 ms	0.73 ms
⌘ Route.componentDidUpdate	Measure	1,231.78 ms	0.16 ms
⌘ (React Tree Reconciliation: Completed Root)	Measure	1,232.43 ms	8 ms
⌘ Dashboard [update]	Measure	1,233.18 ms	6.67 ms
⌘ LogOutButton [update]	Measure	1,233.68 ms	0.25 ms
⌘ withRouter(LinkContainer) [update]	Measure	1,234.08 ms	5.48 ms
⌘ LinkContainer [update]	Measure	1,234.28 ms	5.22 ms
⌘ Route [update]	Measure	1,234.4 ms	5.04 ms
⌘ ListGroupItem [update]	Measure	1,234.62 ms	4.68 ms
⌘ (Committing Changes)	Measure	1,240.53 ms	3.38 ms
⌘ (Committing Snapshot Effects: 0 Total)	Measure	1,241.64 ms	0.9 ms
⌘ (Committing Host Effects: 3 Total)	Measure	1,242.59 ms	0.53 ms
⌘ (Calling Lifecycle Methods: 3 Total)	Measure	1,243.17 ms	0.69 ms
⌘ Route.componentDidUpdate	Measure	1,243.68 ms	0.06 ms
⌘ (React Tree Reconciliation)	Mark	972.79 ms	
⌘ Provider [mount] (#4)	Mark	977.82 ms	
⌘ App [mount] (#8)	Mark	981.77 ms	
⌘ Routes [mount] (#10)	Mark	982.51 ms	
⌘ BrowserRouter [mount] (#12)	Mark	984.48 ms	
⌘ Router [mount] (#14)	Mark	993.25 ms	
⌘ Switch [mount] (#20)	Mark	994.31 ms	
⌘ AuthenticatedRoute [mount] (#24)	Mark	998.85 ms	
⌘ Route [mount] (#26)	Mark	999.28 ms	
⌘ Dashboard [mount] (#32)	Mark	1,001.17 ms	
⌘ LogOutButton [mount] (#36)	Mark	1,009.33 ms	
⌘ withRouter(LinkContainer) [mount] (#48)	Mark	1,022.84 ms	
⌘ LinkContainer [mount] (#53)	Mark	1,023.68 ms	
⌘ Route [mount] (#55)	Mark	1,024.87 ms	

Name	Type	Start Time	Duration
✱ ListGroupItem [mount] (#61)	Mark	1,025.66 ms	
✱ (Committing Changes)	Mark	1,032.65 ms	
✱ (Committing Snapshot Effects)	Mark	1,032.92 ms	
✱ (Committing Host Effects)	Mark	1,037.85 ms	
✱ (Calling Lifecycle Methods)	Mark	1,040.83 ms	
✱ Route.componentDidMount (#55)	Mark	1,042.09 ms	
✱ Router.componentDidMount (#14)	Mark	1,043.93 ms	
✱ BrowserRouter.componentDidMount (#12)	Mark	1,044.13 ms	
✱ (React Tree Reconciliation)	Mark	1,215.57 ms	
✱ Provider [update] (#4)	Mark	1,216.03 ms	
✱ App [update] (#8)	Mark	1,216.28 ms	
✱ Routes [update] (#10)	Mark	1,216.36 ms	
✱ BrowserRouter [update] (#12)	Mark	1,216.43 ms	
✱ Router [update] (#14)	Mark	1,216.51 ms	
✱ Switch [update] (#20)	Mark	1,216.62 ms	
✱ AuthenticatedRoute [update] (#24)	Mark	1,216.73 ms	
✱ Route [update] (#26)	Mark	1,216.81 ms	
✱ Dashboard [update] (#32)	Mark	1,216.92 ms	
✱ LogOutButton [update] (#36)	Mark	1,223.25 ms	
✱ withRouter(LinkContainer) [update] (#48)	Mark	1,225.87 ms	
✱ LinkContainer [update] (#53)	Mark	1,226.2 ms	
✱ Route [update] (#55)	Mark	1,227.38 ms	
✱ ListGroupItem [update] (#61)	Mark	1,228.32 ms	
✱ (Committing Changes)	Mark	1,229.57 ms	
✱ (Committing Snapshot Effects)	Mark	1,229.61 ms	
✱ (Committing Host Effects)	Mark	1,230.34 ms	
✱ (Calling Lifecycle Methods)	Mark	1,231.34 ms	
✱ Route.componentDidUpdate (#55)	Mark	1,231.8 ms	
✱ (React Tree Reconciliation)	Mark	1,232.44 ms	
✱ Provider [update] (#4)	Mark	1,232.57 ms	
✱ App [update] (#8)	Mark	1,232.65 ms	

Name	Type	Start Time	Duration
⌘ Routes [update] (#10)	Mark	1,232.71 ms	
⌘ BrowserRouter [update] (#12)	Mark	1,232.79 ms	
⌘ Router [update] (#14)	Mark	1,232.86 ms	
⌘ Switch [update] (#20)	Mark	1,232.95 ms	
⌘ AuthenticatedRoute [update] (#24)	Mark	1,233.03 ms	
⌘ Route [update] (#26)	Mark	1,233.1 ms	
⌘ Dashboard [update] (#32)	Mark	1,233.2 ms	
⌘ LogOutButton [update] (#36)	Mark	1,233.69 ms	
⌘ withRouter(LinkContainer) [update] (#48)	Mark	1,234.09 ms	
⌘ LinkContainer [update] (#53)	Mark	1,234.29 ms	
⌘ Route [update] (#55)	Mark	1,234.41 ms	
⌘ ListGroupItem [update] (#61)	Mark	1,234.63 ms	
⌘ (Committing Changes)	Mark	1,240.56 ms	
⌘ (Committing Snapshot Effects)	Mark	1,241.69 ms	
⌘ (Committing Host Effects)	Mark	1,242.6 ms	
⌘ (Calling Lifecycle Methods)	Mark	1,243.19 ms	
⌘ Route.componentDidUpdate (#55)	Mark	1,243.7 ms	

Keep request counts low and transfer sizes small — 18 requests • 841 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	18	840.7 KiB
Script	5	695.9 KiB
Font	2	98 KiB
Other	7	24.4 KiB
Stylesheet	3	21.4 KiB
Document	1	1 KiB
Image	0	0 KiB
Media	0	0 KiB
Third-party	10	150.7 KiB

Largest Contentful Paint element — 1 element found



This is the largest contentful element painted within the viewport. [Learn More](#)

Element

h1

Avoid large layout shifts — 1 element found ^

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

Element	CLS Contribution
i.fas.fa-sign-out-alt	0

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/0.chunk.js (localhost)	1,449 ms	496 ms

Passed audits (26) ^

Eliminate render-blocking resources — Potential savings of 80 ms ^

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

☒ Show 3rd-party resources (4)

URL	Transfer Size	Potential Savings
...css/bootstrap.min.css (maxcdn.bootstrapcdn.com)	19.4 KiB	320 ms
/css?family=PT+Serif Open+Sans:300,400,600,700,800 (fonts.googleapis.com)	1.3 KiB	260 ms
/css2?family=Roboto:wght@300;400;500&display=swap (fonts.googleapis.com)	0.8 KiB	260 ms
...js/bootstrap.min.js (maxcdn.bootstrapcdn.com)	9.9 KiB	280 ms

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 ^

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](#).

Remove unused CSS — Potential savings of 31 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 (1)

URL	Transfer Size	Potential Savings
...css/bootstrap.min.css (maxcdn.bootstrapcdn.com)	19.4 KiB	19.1 KiB
/*! * Font Awesome Free 5.15.1 by @fontawesome - https://fontawesome.com * License - https://fonta...	12 KiB	12 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 0 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
/dashboard (localhost)	0 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, multiplexing, and server push. [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 — Potential savings of 6 KiB ^

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](#)

☐ Show 3rd-party resources (0)

URL	Potential Savings
...js/0.chunk.js (localhost)	6 KiB
0.chunk.js:67645	@babel/plugin-transform-classes
0.chunk.js:67645	Object.getOwnPropertyNames
0.chunk.js:67645	Array.from
0.chunk.js:5833	Array.isArray
0.chunk.js:5881	Object.keys
0.chunk.js:5955	Object.entries
0.chunk.js:5975	Object.values

Avoids enormous network payloads — Total size was 841 KiB ^

Large network payloads cost users real money and are highly correlated with long load times. [Learn more.](#)

☒ Show 3rd-party resources (6)

URL	Transfer Size
...js/0.chunk.js (localhost)	647.2 KiB
...webfonts/free-fa-solid-900.woff2 (ka-f.fontawesome.com)	86.8 KiB
...js/main.chunk.js (localhost)	21 KiB
...css/bootstrap.min.css (maxcdn.bootstrapcdn.com)	19.4 KiB
...js/bundle.js (localhost)	14.1 KiB
...css/free.min.css (ka-f.fontawesome.com)	12.4 KiB
...v20/KFOICnqEu....woff2 (fonts.gstatic.com)	11.1 KiB
...js/bootstrap.min.js (maxcdn.bootstrapcdn.com)	9.9 KiB
/logo192.png (localhost)	5.5 KiB
...css/free-v4-shims.min.css (ka-f.fontawesome.com)	4.2 KiB

Avoids an excessive DOM size — 18 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		18
Maximum DOM Depth	<b>	8
Maximum Child Elements	<body>	9

JavaScript execution time — 0.5 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 (1)

URL	Total CPU Time	Script Evaluation	Script Parse
...js/main.chunk.js (localhost)	289 ms	270 ms	15 ms
...js/0.chunk.js (localhost)	280 ms	80 ms	100 ms
Unattributable	182 ms	4 ms	0 ms
/dashboard (localhost)	170 ms	5 ms	2 ms
/0075a9b63d.js (kit.fontawesome.com)	52 ms	48 ms	1 ms

Minimizes main-thread work — 1.0 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
Script Evaluation	408 ms
Other	331 ms
Script Parsing & Compilation	122 ms
Style & Layout	76 ms
Parse HTML & CSS	47 ms
Garbage Collection	12 ms
Rendering	5 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 blocked the main thread for 0 ms ^

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](#).

☐ Show 3rd party resources (0)

Third-Party	Transfer Size	Main-Thread Blocking Time
<a href="#">FontAwesome CDN</a>	108 KiB	0 ms
...webfonts/free-fa-solid-900.woff2 (ka-f.fontawesome.com)	87 KiB	0 ms
...css/free.min.css (ka-f.fontawesome.com)	12 KiB	0 ms
Other resources	9 KiB	0 ms
<a href="#">Bootstrap CDN</a>	29 KiB	0 ms
...css/bootstrap.min.css (maxcdn.bootstrapcdn.com)	19 KiB	0 ms
...js/bootstrap.min.js (maxcdn.bootstrapcdn.com)	10 KiB	0 ms
<a href="#">Google Fonts</a>	13 KiB	0 ms
...v20/KFOICnqEu....woff2 (fonts.gstatic.com)	11 KiB	0 ms

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](#)

## Runtime Settings

<b>URL</b>	http://localhost:3000/dashboard
<b>Fetch Time</b>	Jan 3, 2021, 11:05 PM GMT-3
<b>Device</b>	Emulated Desktop
<b>Network throttling</b>	40 ms TCP RTT, 10,240 Kbps throughput (Simulated)
<b>CPU throttling</b>	1x slowdown (Simulated)
<b>Channel</b>	devtools
<b>User agent (host)</b>	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4280.88 Safari/537.36
<b>User agent (network)</b>	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/84.0.4143.7 Safari/537.36 Chrome-Lighthouse

