

On Page SEO

I am highly delighted to represent my skills on page-load-speed optimization.

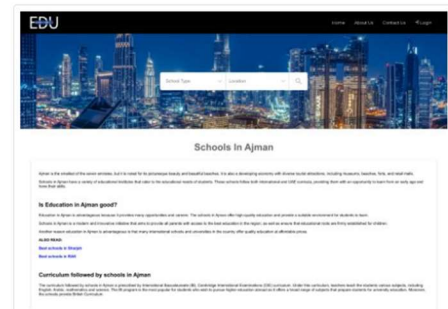
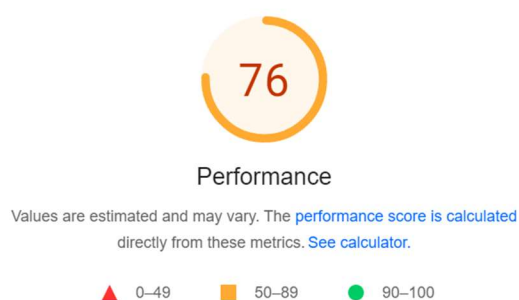
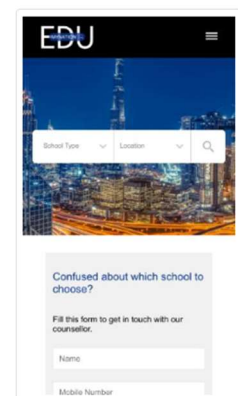
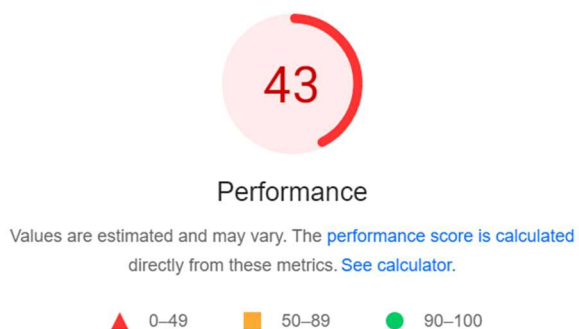
I performed a detailed insight test of the website and found the following flaws and improvements for the same.

I tested the website performance on **Google's page Speed insights** and found the following results :

Link to the Website : <https://edumynation.com/category/schools-in-ajman>

Results/Observations of test performed :

- The performance of the website is **43 and 76** (out of 100) for mobile and desktop screens respectively. For a good website, the score must be **above the mark of 90**.

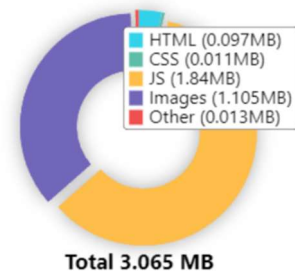


- Hence by observation, I can conclude that website's performance is not up to the mark and must be boosted to cross the mark of 90.

- The approximate loading time is around 7.8 sec and 2.1 sec for mobile and desktop screens respectively, which is enough to bounce off around 50% of the user before loading in case of mobile devices, which is definitely not a good sign.

- The size of page is about 3.065 MB which is reasonably low which is good for Page Load Speed and user experience.
- Out of total size, 1.1 MB is consumed by images which is around 33.33% of the total page.
- The first content (First content paint abbreviated as FCP) took 3.1 sec for mobile device which drastically reduces the page load speed and increase the bounce off rate of the users.
- The page took 7.8 sec for mobile device to interact which means user has to wait around 8 sec for the page to become interactive fully.
- Images are taking around 3-4 sec to load properly, which is comparatively high for usual loading time.

Page Size Breakdown









Suggestions/Measures must be taken to reduce the page speed

- The first and basic step to be taken is **to minify all the JS and CSS files**(removing all the extra line space and comments in the code).
- Images being used in the site **are of JPG/PNG formats** which does not helps in the reduction of page load speed. Therefore, Image **formats like WebP and AVIF** often provide better compression than PNG or JPEG, which means **faster downloads and less data** consumption.
- I noticed that the images are properly sized in the code, hence while loading browser have to explicitly resize the images which wastes the time. Ideally, **page should never serve images that are larger than the version that's rendered on the user's screen.** Anything larger than that just results in wasted bytes and slows down page load time.
- The main strategy for serving appropriately sized images is called "responsive images". With responsive images, you generate multiple versions of each image, and then specify which version to use in your HTML or CSS using media queries, viewport dimensions, and so on.
- **Use and serve image in the `next/image` component to set the appropriate `sizes`**

- Compressing the landing page background image will lead to boost in the loading time.
- Reduce unused JavaScript and defer loading scripts until they are required to decrease bytes consumed by network activity, 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`.
- Use `Webpack Bundle Analyzer` to detect unused JavaScript code.
- 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. Use the `next/script` component to defer loading of non-critical third-party scripts.
- Reduce initial server response time, Keep the server response time for the main document short because all other requests depend on it. If you are server-side rendering any React components, consider using `renderToPipeableStream()` or `renderToStaticNodeStream()` to allow the client to receive and hydrate different parts of the markup instead of all at once.
- Serve static assets with an efficient cache policy. A long cache lifetime can speed up repeat visits to your page.
- 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.
- Avoid an excessive DOM size. A large DOM will increase memory usage, cause longer style calculations, and produce costly layout reflows.
- Optimizing images format and removing unused java-script will boost the website in desktop device.

Lab matrix report for reference for mobile device :

METRICS	
 First Contentful Paint 2.6 s	 Time to Interactive 7.9 s
 Speed Index 4.7 s	 Total Blocking Time 1,080 ms
 Largest Contentful Paint 6.9 s	 Cumulative Layout Shift 0.002

I assure that all the measures and improvements mentioned in the report are truly based on my skills, up to the mark and speed report by google insight .

Reported by:
Mr. Tanuj Kalonia