Newser PWA

This document discusses building a PWA with ReactJS as an answer for the evaluation test for a Senior Front End Developer. The test subject can be found by following the link: <https://github.com/DigitalHubUAE/frontend-developer>.

First of all, the work can be reviewed in many ways, you can try all of them to check different parts of the application:

* Running directly on your browser:

In order to see the resultant application running, and test it on your browser/Mobile, a Docker Image has been added, and it’s runnable using the following command:

Linux: docker run -p 8000:80 -it yaziderman/newser-pwa-5

Windows: winpty docker run -p 8000:80 -it yaziderman/newser-pwa-5

As per what is mentioned in the previous command, after executing it, open the following link on your browser: <http://localhost:8000>, or from any mobile connected to the same network: http://<your PC IP>:8000.

However, the docker repository url is the following: <https://cloud.docker.com/repository/docker/yaziderman/newser-pwa-5>

* Manually setup the application on your local machine:

The script is uploaded on Github, and publicly accessible using the url: <https://github.com/yaziderman/newser.3>

You can clone it to your machine, and then execute the following two commands:

cd <the application directory name>

npm run build

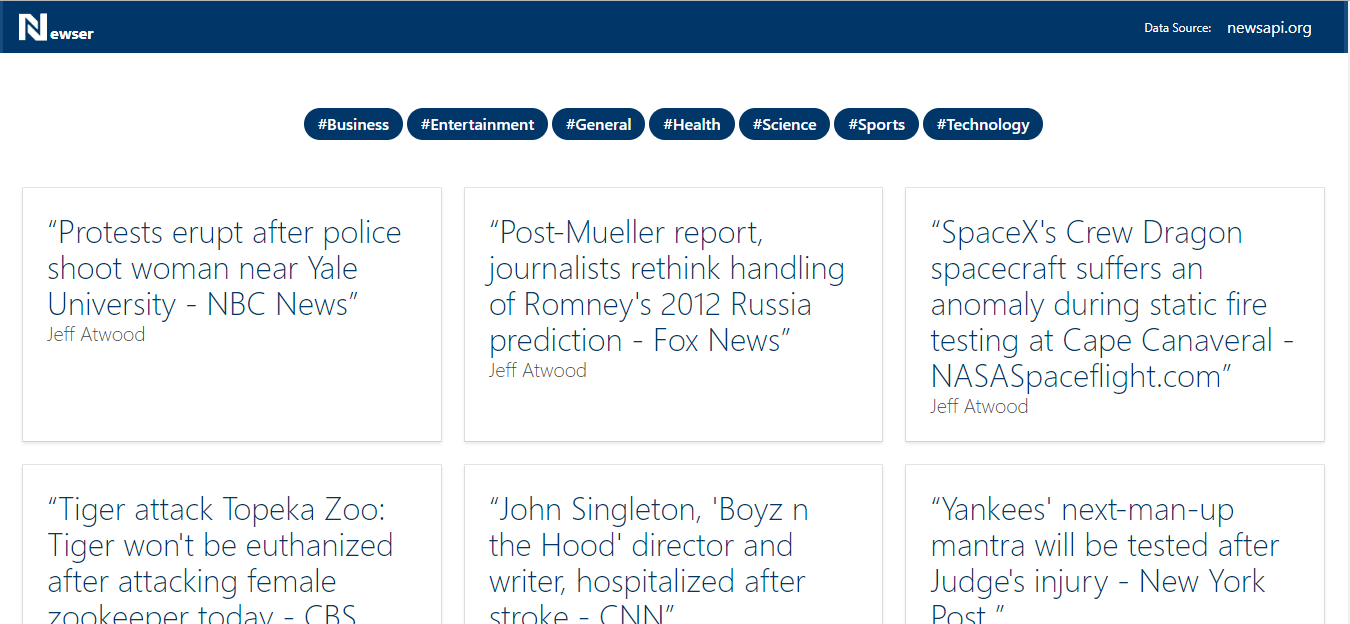
npm install -g serve <If not already installed>

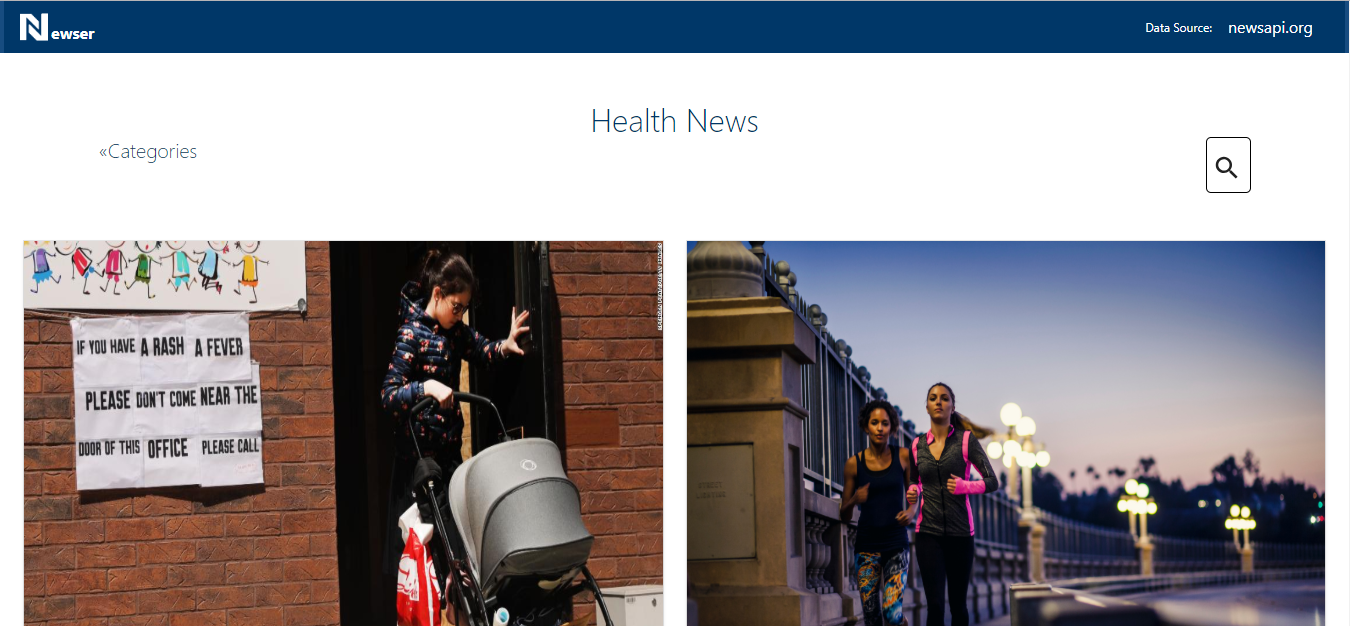
serve -s build

In this case, the application will be accessible using the link: [http://localhost:5000](http://localhost:8000), or from any mobile connected to the same network: http://<your PC IP>:5000

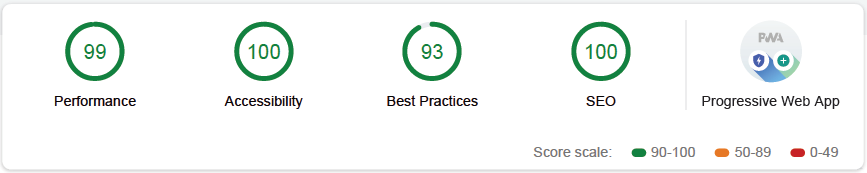
|  |
| --- |
|  |

After running the application in either of the two methods, you should be able to see an interface similar to the two screenshots:





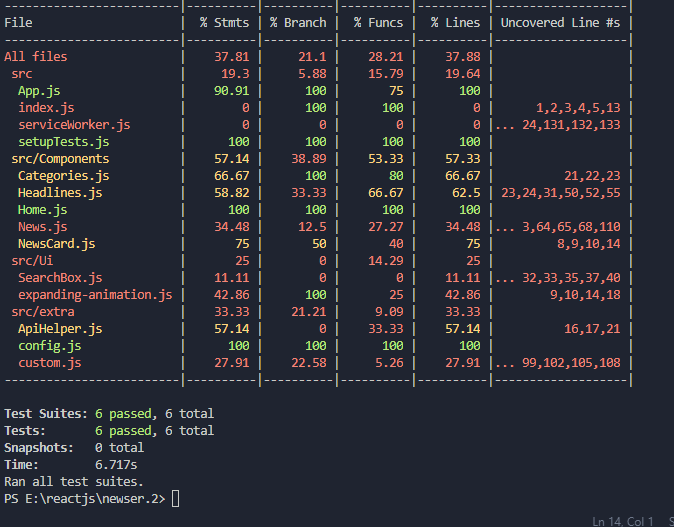
As per the attached report: [Lighthouse Report.pdf](https://github.com/yaziderman/newser.3/blob/master/Documentation/Lighthouse%20Report.pdf), we have got the following ranking by Lighthouse extension:



And here you are the test coverage result:

By executing the command:

npm test -- --coverage



## Specific Comments of the Use Case:

## **First Challenge**

Create a free account here [https://newsapi.org](https://newsapi.org/) , news API has the news for different online newspapers, we want to connect our website with these services.

On the first challenge, we need to

* Create the account in [https://newsapi.org: Done](https://newsapi.org/)
* Create a boilerplate project[Done](https://newsapi.org/)
* Show the list of categories in the homepage[Done](https://newsapi.org/), but hardcoded, as there is no API for the categories of the service provider
* Show the headlines on the homepage[Done](https://newsapi.org/)
* Show the news filtered by categories on a new page[Done](https://newsapi.org/), But the news service does not support filteration by categories, hence I have used the Headlines service again for illustration purposes.

#### **Considerations**

* Need to be a Single Page Application
* Need to be PWA
* Show images
* We are not going to evaluate your design choices regarding colours and fonts, but we are going to take on considering how the overall experience is.

## **Second Challenge**

As modern developers, we understand that PWA is the path to follow. In this point, we are going to apply as many good practices we can but mostly focus on offline content render and static content cache.

At least we should be able to

* Access offline to a previously visited content Done
* Have a proper Manifest in place Done
* Cache all the static content Done

## **Third Challenge**

Instead of waiting to have all the content before rending the page we should apply progressive loading. You can check this [article](https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps/Loading) about this point: Done, in addition to caching the Web Services responses, and applying ReactJS Suspense techniques, the images are being loaded progressively and also splash templates have been added to give the used the impression of loaded page even during loading.

## **Forth Challenge**

Add at least 1 micro-interaction, you can understand a bit more about this topic in this [article](https://uxplanet.org/creating-meaningful-micro-interactions-99cbde1fbee7)

This is the creative moment of the test, you can innovate as much as you want here. The most appropriate Micro-Interaction to be added during short time is a quick search box with interactive anumation. We have also Slider to Top and that’s it.

## **Fifth Challenge**

Test all the things. We are going to focus on functional tests. We can use MochaJs or whatever else tool, but we need to be able to run all the tests headless from the command line. You can use docker for all the dependencies.

I have mainly used JestJS with React to test the main functionality of the main components, the time wasn’t enough to have a bigger coverage.

## **Lately**

Document the project, how to spin up the project, how to run the tests.

The README file should have all the step to set up the project locally. Remember as less command for setting up the project much better.

Yazeed Erman

+971 50 4516768