

|  |  |  |
| --- | --- | --- |
|  |  | |
|  |  |  | | |  | |
|  |  | BookWorm Documentation  Testing, difficulties faced and implementation proof. | | |  | |
|  |  | | | | |  |
|  | **Introduction**  This documentation will describe everything from the project plan, testing, implementation proof, show the functional UI, technical difficulties faced etc.  Author: K.P.I. Shenesh Perera  Date: 25/06/2019  IDM | | | | |  |
|  |  | | |  | | |

**Contents**

[Project Plan 3](#_Toc14983554)

[Technical Difficulties 4](#_Toc14983555)

[Not having previous experience with PWAs and insufficient resources to learn from 4](#_Toc14983556)

[No browser means no safety net 4](#_Toc14983557)

[Mobile application-looking navigation 5](#_Toc14983558)

[Caching and cache invalidation 5](#_Toc14983559)

[Smooth UX 5](#_Toc14983560)

[Proof of implementation 5](#_Toc14983561)

[Screenshots of the application 8](#_Toc14983562)

[Testing 10](#_Toc14983563)

[Network request to check DOMContentLoaded speed and memory usage 10](#_Toc14983564)

[Performance Test 11](#_Toc14983565)

[General testing 11](#_Toc14983566)

# Project Plan

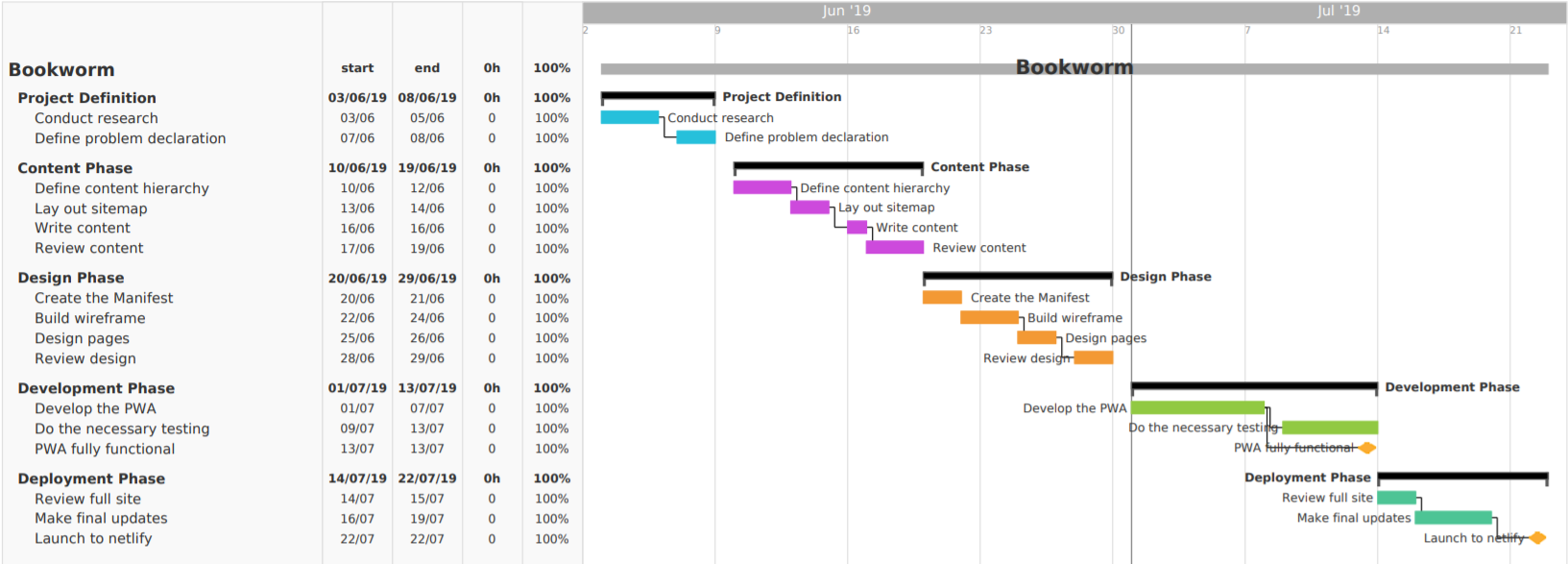


Figure 1.0, Shenesh Perera, 03/06/2019

The ambitious project, BookWorm the digital library for anyone around the world, began on the 03rd of June and ended successfully on the 22nd of July 2019. The project faced a lot of issues as I was the only individual working on the project.

The project plan was designed according to the Software Development Lifecycle architecture and is distinctly divided in to their respective phases. The following is the descriptive project plan of BookWorm:

* Project Definition, this phase was one of the crucial phases as I had to understand the business related problem, set up a basic set of user and system requirements and grasp the objectives of the business related problem’s solution. Extensive primary and secondary researches were done in order to understand the business related problem and to identify the existing problems in the traditional way of reading books in a library then a problem declaration was drawn.
* Content Phase, this was more of a strategy taken in order to ensure that I wouldn’t have to spend a lot of time in the development phase trying to figure out what text will go where and what I will have to come up with. In this stage I birthed the name for the project, BookWorm, declared the sitemap so that search engines can crawl the website and wrote the necessary content required.
* Design Phase, this must be the core of the process of the development, it helped me understand how a PWA should look before I actually developed it with the tech stack that I chose to work with. I did some basic wireframing on paper, wrote some basic HTML and did some basic CSS to have a proper structure to work with. One very important thing I did in this phase is making the manifest, which is very important for a PWA to work.
* Development phase, the most challenging phase in the entire process. Admittedly, I’ve worked with a lot of fullstack applications, written a several production ready backend and frontend services but this was my first time doing caching and working with the Service Workers introduced the Chrome V8 engine as a result of the ECMAScript 2015 (ES6). Service Workers are the most crucial component in the PWA, working with the Workbox Service worker and maintaining my own service worker were quite the challenge. The necessary testing was also done during this stage.
* Deployment phase, by now I had already with me a fully functioning application that satisfied the objectives and fulfilled all the requirements of the business related problem. Now the new problem was to find a suitable service that would help me deploy the PWA, and upon research I came to find Netlify. A great service that serves static serverless architecture based web applications for free with rich features like SSL/TSL support and Continuous Integration(using travis and the like) plus A/B testing.

Upon the completion of this project, BookWorm was released to netlify at this address: [https://bookwormz.netlify.com](https://bookwormz.netlify.com/)

# Technical Difficulties

## Not having previous experience with PWAs and insufficient resources to learn from

This was considerably the biggest technical difficulty I faced during the process of doing this project, I had barely any experience with a PWA. I barely even knew what a PWA is. The topic of progressive web application development is fairly new to this world, as first introduced in 2015 by Alex Russel, the resources that are around the internet to learn to develop a PWA are quite outdated at times and do not cover the full picture.

Even experienced developers don’t seem to have a full grasp on what a PWA is and how to make one, what kinds of challenges that have to be faced when making one. However, the inspiration to making a PWA instead of a general native mobile application is from Dev.to’s PWA that is downloadable on mobile. It’s fast and elegant, due to this inspiration I kept pushing myself in order to create the PWA.

## No browser means no safety net

This isn’t a security issue, this is more of a problem that I had to face regarding the utility functions that are provided by the browser to any website that’s run in the browser. Although this is a good thing on mobile platforms, it is certainly not a good thing for the website that is run on your browser. This is solved easily by adding a navigation bar on the side.

## Mobile application-looking navigation

A PWA must look and feel like a general mobile application, so it must retain the position of content and listings of the sort. Users must be capable of navigation to and from the application easily. As the PWA will be run on mobile as a mobile app, it is very important to think about touch interactions. A typical example of this is the hover pseudoclass in CSS, although this functions properly in a browser, it wouldn’t work as nearly as well on the mobile version of the PWA.

## Caching and cache invalidation

The main purpose of the service worker was to cache static assets, including the html and css files for atleast a week long. The problem with this that I faced is that when the Google Play Books API updated some kind of dynamic content, although the client-side sends HTTP requests to the Google Play Books API through the Fetch API, even if the data is successfully returned, the user interface will not update with the new data as the data that is used is from the cache. So the biggest challenge was to implement proper cache invalidation so that the user interface will update with new content from the Google Play Books API.

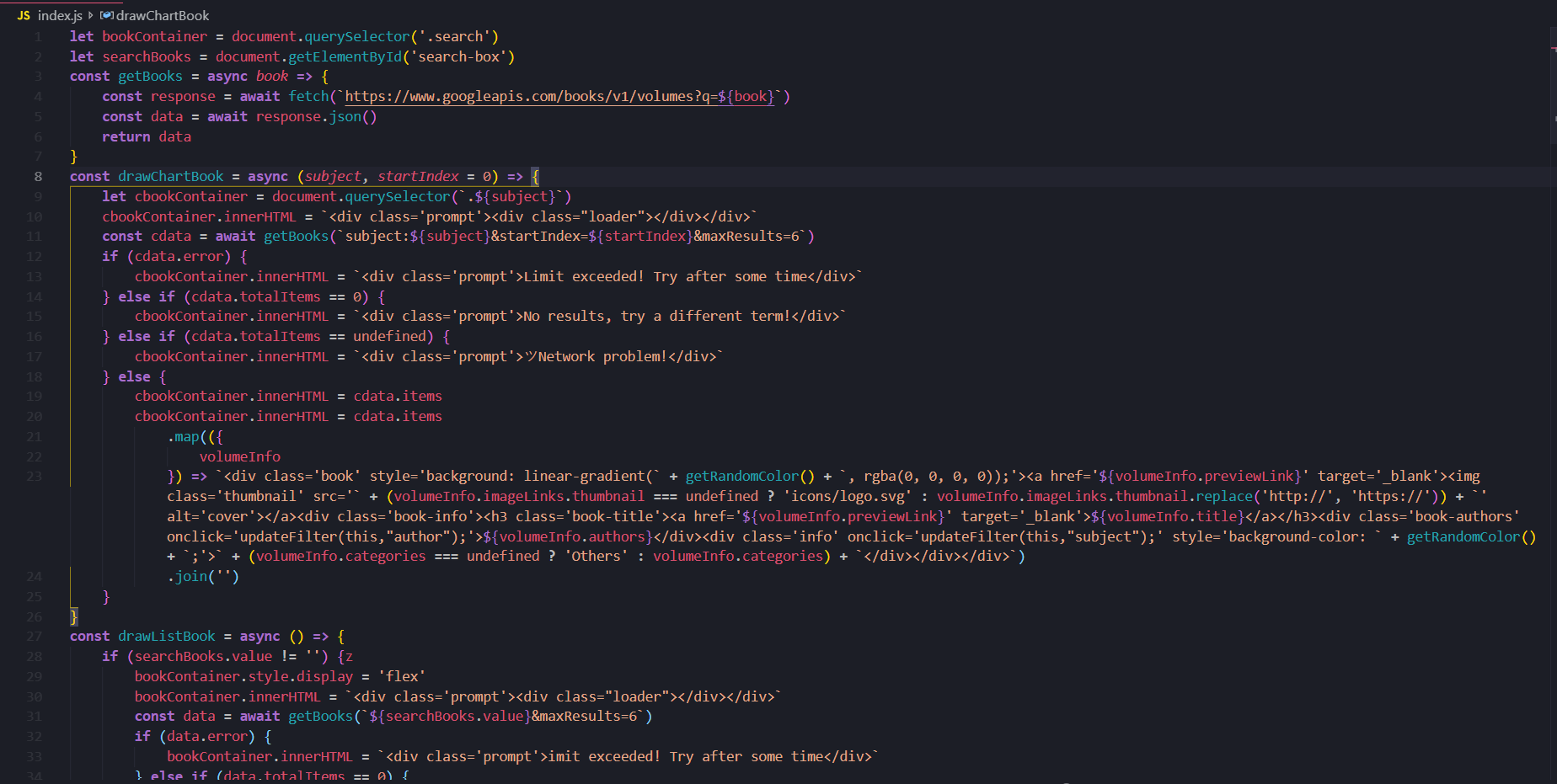
Due to this difficulty, the codebase of the sw.js file was over 1200 lines long, and upon refactoring was bought down to a lower size.

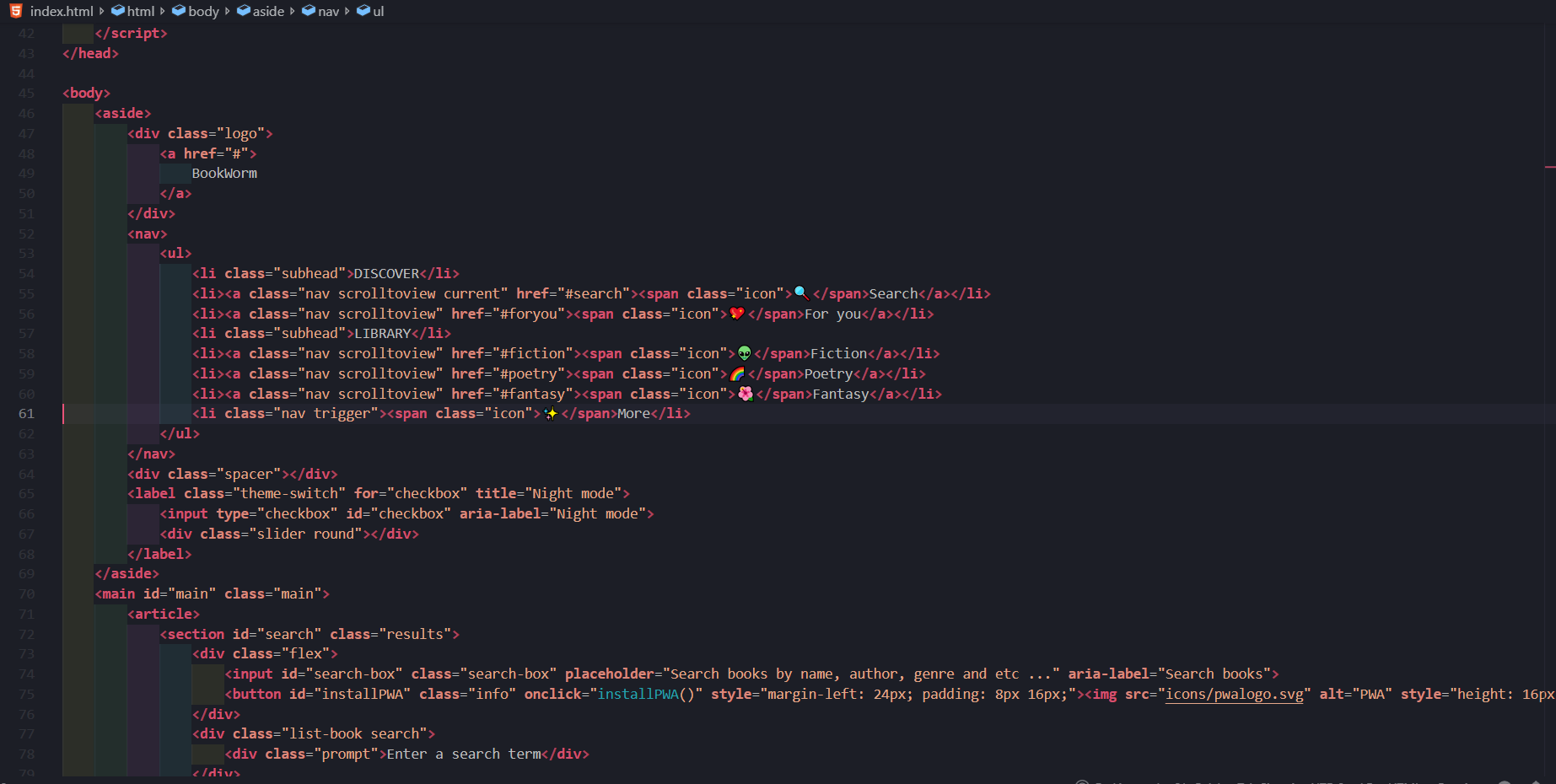
## Smooth UX

Something I was not used to doing, because I’ve been developing web applications for most of my life. Having smooth user experience on mobile platforms must be one of the toughest challenges to face as a developer that has not done mobile development. Managing things like animations and implementing variable smooth scroll speeds depending on the device width were certainly a big difficulty.

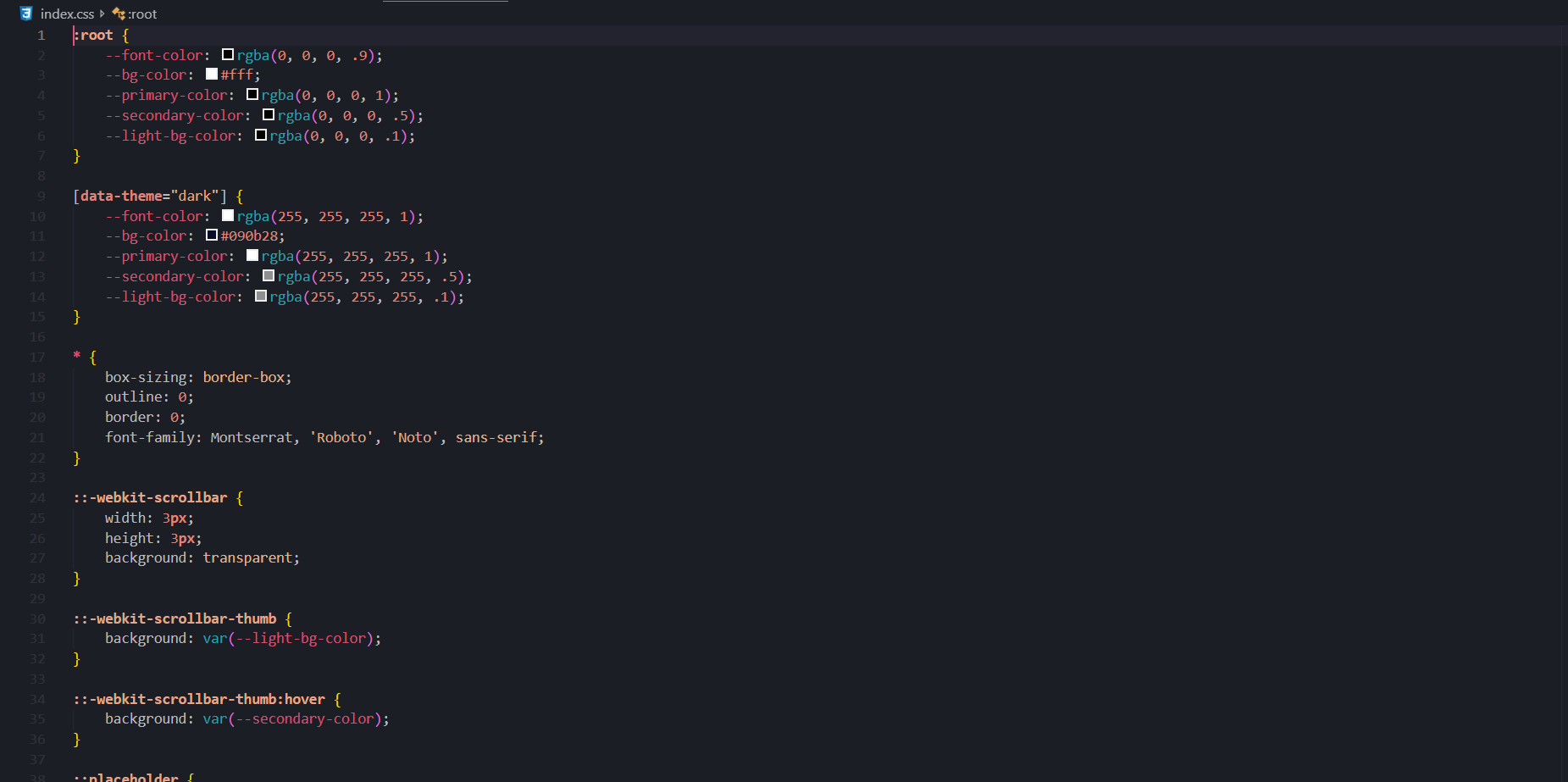
# Proof of implementation

The source code is included with the assignment; the following screenshots are not the entire code as the code is too big to be included in the documentation. Relevant screenshots have been taken.

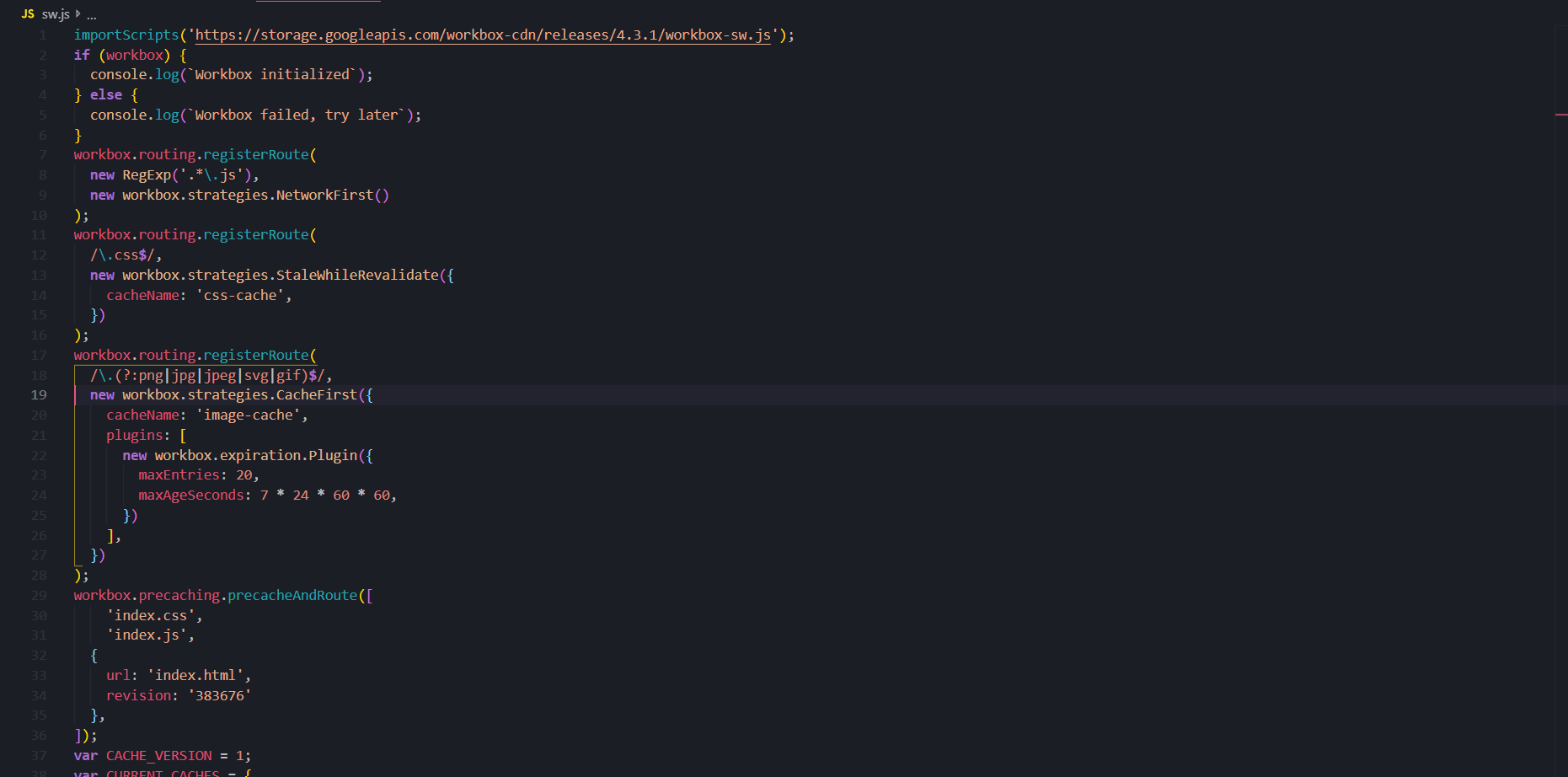
Index.js – the main file that does most of the DOM manipulation and handling HTTP requests

Index.html - the main html file with all the semantic markup

Index.css – the main stylesheet that styles the semantic markup

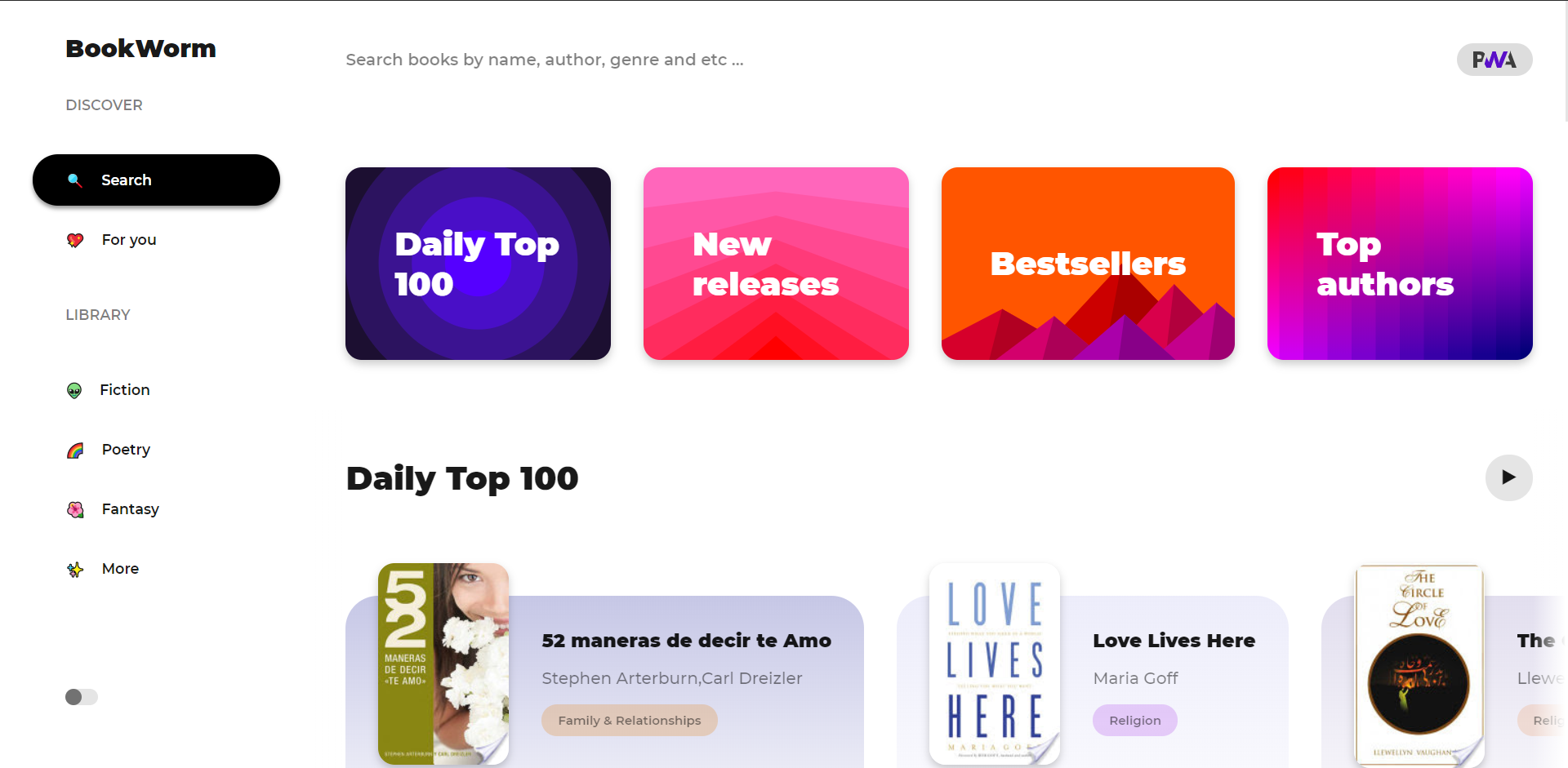


Sw.js – The service worker that handles cache and cache invalidation, and any interactions with the workbox service worker that assists with PWA development

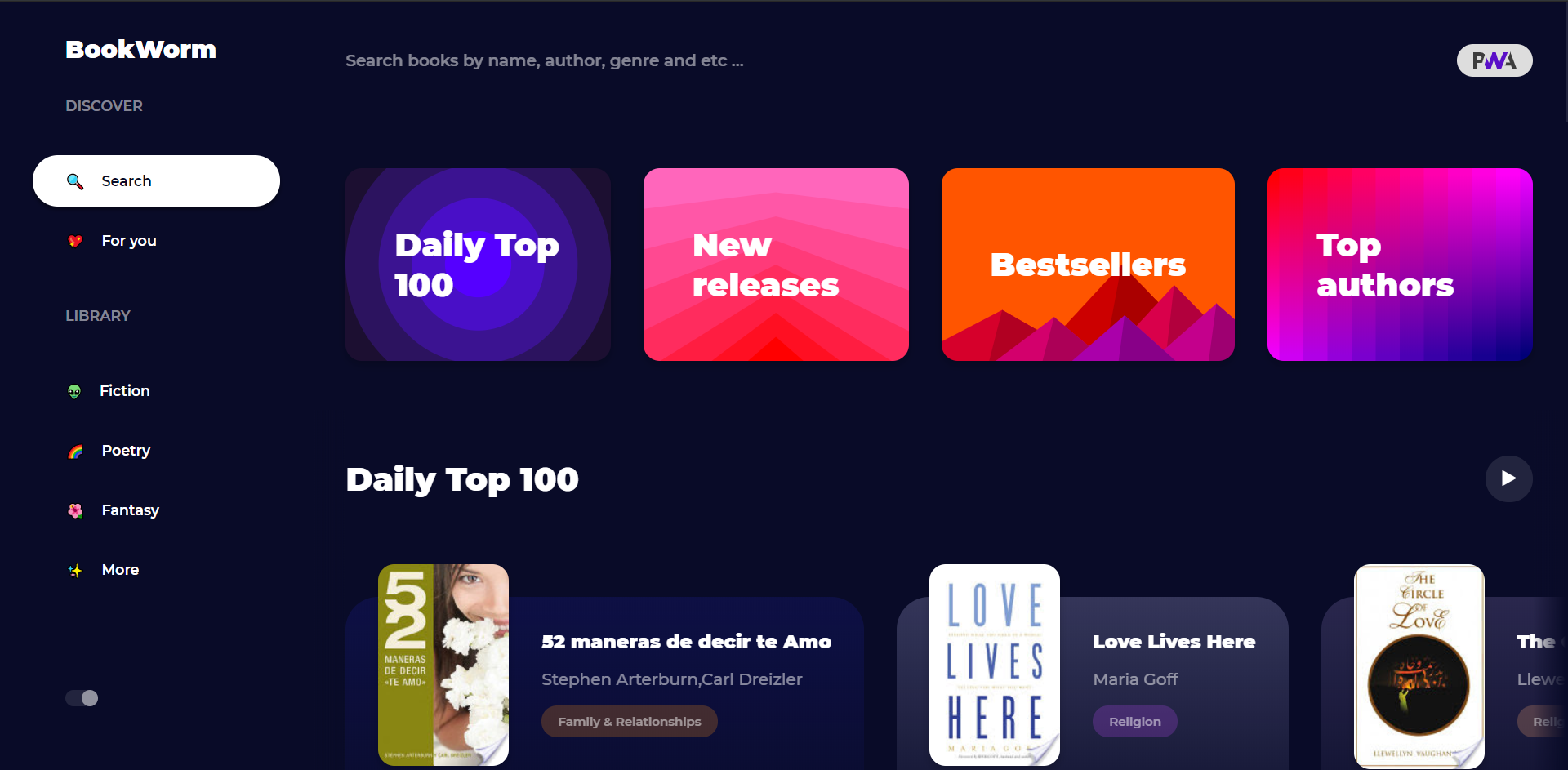


## Screenshots of the application

Application on light mode - for daytime



Application in darkmode – for night time

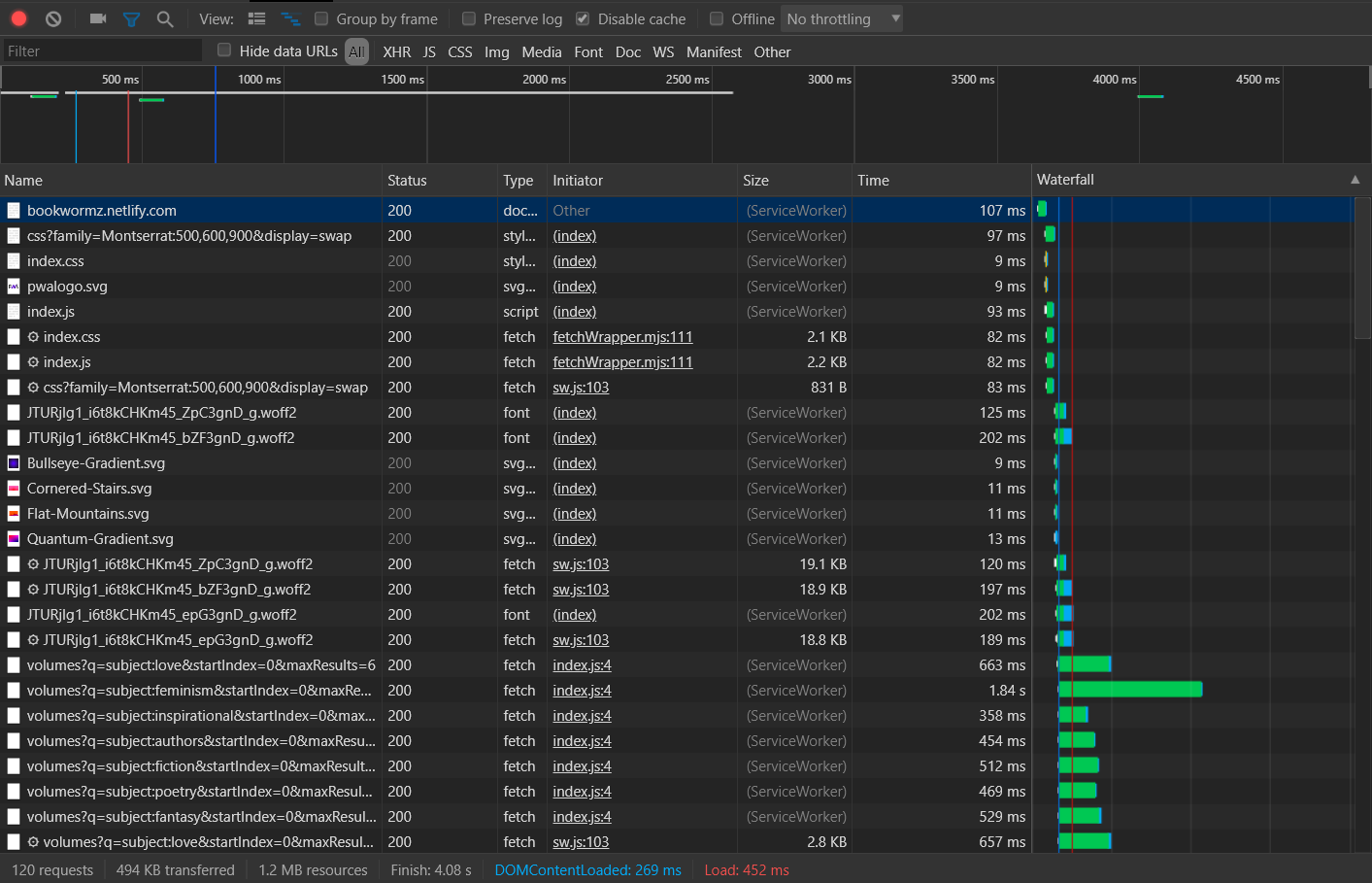


Application in mobile view in darkmode

|  |  |
| --- | --- |
| C:\Users\Ryft\AppData\Local\Microsoft\Windows\INetCache\Content.Word\mob1.png | C:\Users\Ryft\AppData\Local\Microsoft\Windows\INetCache\Content.Word\mob2.png |

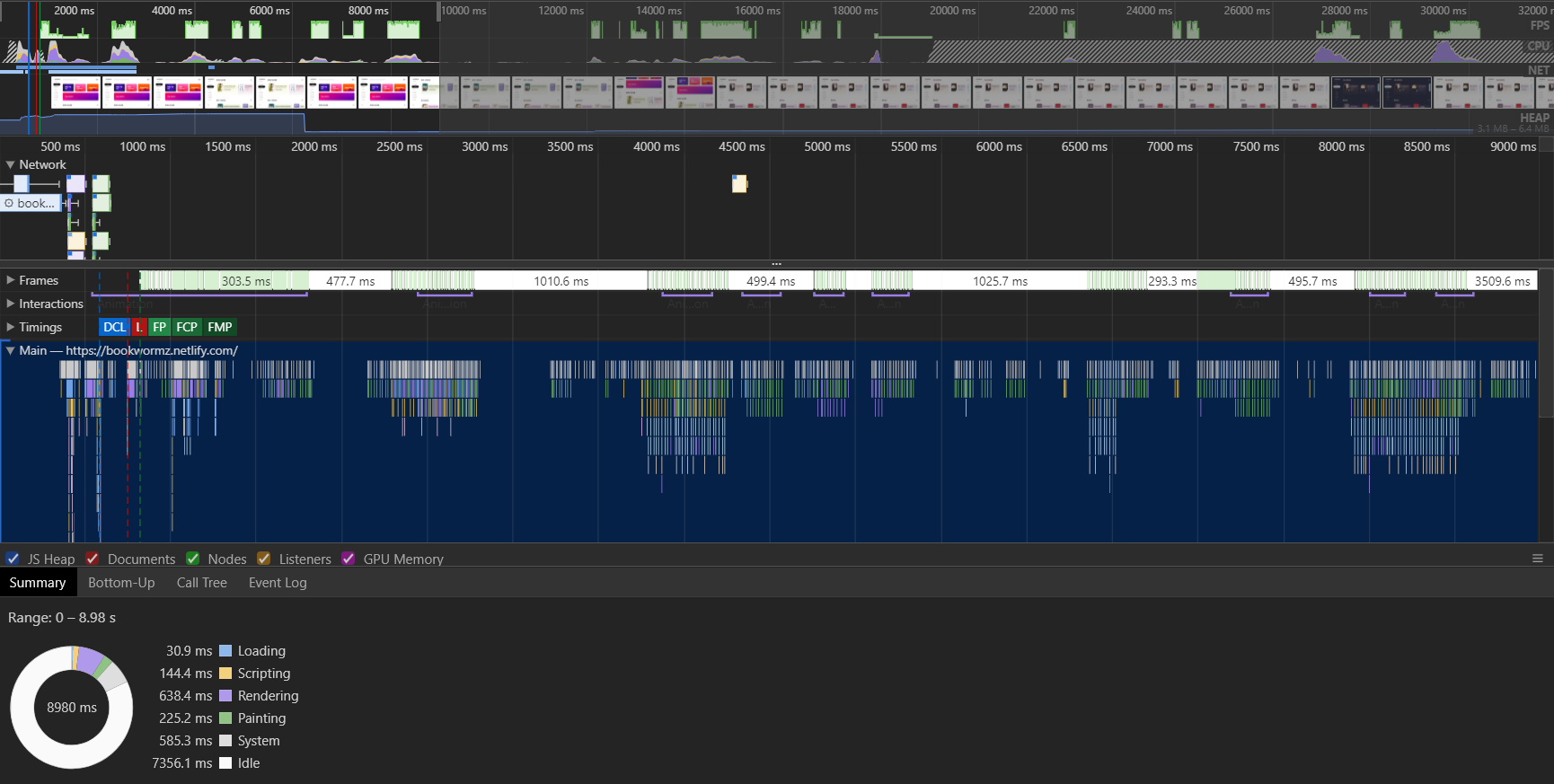
# Testing

## Network request to check DOMContentLoaded speed and memory usage



As you can see the DOM is completely loaded within 269ms, with optimal painting performance and the entire application is loaded within 452ms. It is a general rule of thumb that any good application has to load it’s UI entirely within half of a second and it seems that BookWorm has met that standard. As you see the entire application requires no more than 1.2MB of resources so it’s minimal as it can be. There are no network errors that are shown in the waterfall, so the network request test was successful.

## Performance Test



In the performance test, the main goal was to identify if the painting performance is 60FPS and if there is any lag in between moving from section to section. The smooth scroll works optimally and as you see, the total time taken for entire test was somewhere around 9 seconds, and around 7.5 of those seconds were idle time that was spent with me moving my cursor from place to place for the purpose of the test.

## General testing

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case** | **Expected** | **Actual** | **Evaluation** |
| Smooth scroll functions properly on mobile | Extremely lagging and in fact jumps from one place to another | Scrolling must be smooth | The dom manipulations and the calculations done for smooth scroll are off. |
| Smooth scroll retry | No scrolling at all | Scrolling must be smooth | The fixes to the calculations have broken the dom manipulations |
| Smooth scroll retryx2 | Scrolling is smooth | Scrolling must be smooth | Test success |
| Check if PWA sends a popup to install itself on mobile and add to the home screen | Popup pops up but the app is not installed to the home screen | The popup must come and ask the user for consent to add the PWA to the home screen | There was an error in the installPWA() function that was related to Boolean arithmetic |
| Check if PWA sends a popup to install itself on mobile and add to the home screen retry | Popup pops up and asks the user for consent, the app is on the homescreen | The popup must come and ask the user for consent to add the PWA to the home screen | Test success |
| Check if deployed version of the application runs | PWA is displayed, and all the test cases are successful | When visiting bookwormz.netlify.com, the PWA must be displayed and all the above test cases must be successful | Test success |

# Justifications & possible improvements

## Feedback from r/books and my own comments to their feedback

Most of the feedback from r/books can be summarized with the following 3 statements:

* BookWormz runs great and functions really great
* The darkmode is very helpful and it helps with using the app at night
* BookWormz relies on Google Play Books, which doesn’t have some books most people read, the reading functionality is limited to google play books

The dark mode was introduced mainly to help readers to read at night without any problems and as it seems it worked quite well.

I agree on the comment about the limited functionality but this is because I do not have the sufficient resources to fully deploy an application that has a database nor fund hosting services enough to make this a complete alternative to bigger ebook readers like Amazon Kindle.

## Why a PWA?

In the simplest words, it is declared the future of mobile and desktop development. It relies on web technologies, so it’s blazingly fast. With the evolution of JIT compilers (Just in time compilers) in the V8 engine, Javascript has become faster than ever being one of the most demanded languages in the world. It’s better to use a technology that has lived for a very long time and keeps getting more and more attention.

Tech giants like facebook, twitter, Instagram etc. have their own Progressive Web Apps, if the tech giants themselves have adopted this new approach, why shouldn’t we? Why shouldn’t I use it for BookWorm? That is simply why BookWorm is a PWA.

## Possible Improvements and why/why not to include them in the future

* The application must be made so that there is a user/login function, in otherwords it must have database support. This must be included as this way, more features can be introduced like persistent bookmarking etc.
* The dependence on google play books must entirely be erased. This feature is considerably but doesn’t really have to be included. As this would mean that the existing base of users that rely on the books in google play books API may navigate to using their former services.
* Support for interacting with other ebook reader APIs like Amazon Kindle, Kobo etc. must be done so that users from those ebook readers may use BookWorm instead. This is a considerable feature, it doesn’t have to be included but if it does the traffic towards BookWorm will be higher.
* With sufficient polling from the users, the light mode can be entirely eliminated. The famous chat client for gamers called Discord, eliminated their light theme after determining from the users that the dark mode is used more. This feature may have to be considered after atleast 6 to 7 months have passed by.
* User interface customizability, this feature must be included in the future, this would mean that the users will be capable of having their own background colors, font sizes and font types which is important as some people are used to particular styles of reading.