Venxia Report for Guild Stationers And Web Corp

Caleb Irwin
January 9th, 2024

Preface

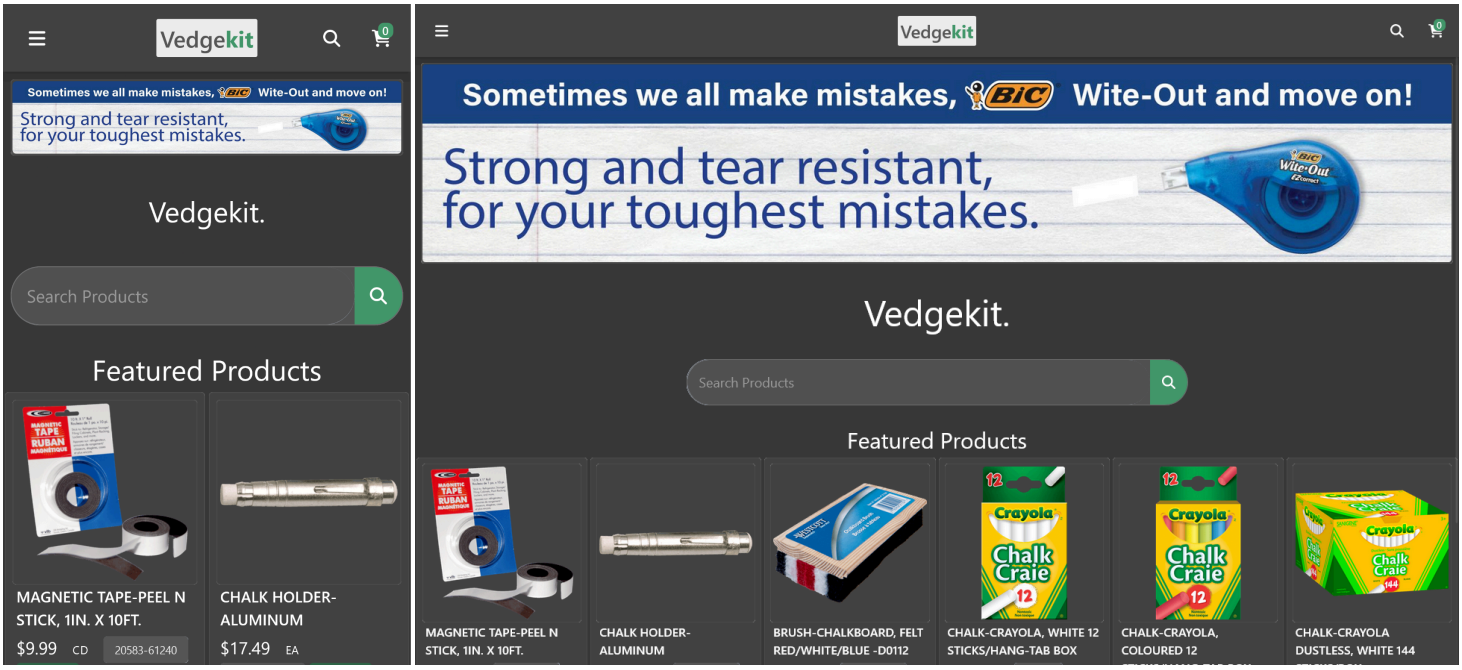
My goal in writing this report is to help you improve Venxia so that it is a suitable product for Office Experts and the entire buying group. I will explain major issues and provide potential solutions. I will also briefly cover some smaller but important issues. To show you that improvement is possible I have created Vedgekit. My next section will explain what Vedgekit is and how it works. Thank you in advance for reading my report.

Caleb Irwin
Office Experts Office Pro
me@calebirwin.ca
info@officeexperts.ca

Table of Contents

Preface **1**
Table of Contents **2**
Vedgekit **3**
What? 3
Why? 3
How? 3
Waterfall 5
Issues And Shortcomings 6
Major Issues **7**
Server: Response Time 7
Architecture: Key Resource Chaining (Waterfall) 8
Content Delivery: Images 11
Content Delivery: Improper Practices 12
Client: Mobile Usability 13
Client: Lack of A Unified and Modern Design System 14
Outdated Technology And Technical Debt 15
Other Issues **16**
Server: Search Results 16
Client: Sparse Content 17
Client: Cart Issues 18
Client: Content Shift From Sticky Navigation Bar 18
Client: Popups 18
Client: Preload Main Image 19
Client: Lazy Load Images 19
Client: Icons Should Not Be PNGs 20
Conclusion **21**

Vedgekit



What?

Vedgekit is effectively a replacement for Venxia's front end. It is a demo and is not feature-complete. I have currently implemented…

- Session management
- Home page (with optimized images)
- Product page (including related products)
- Search page (with infinite scroll, but without filters)
- Cart page
- Navigation bar
- Checkout (proxied)
- Order confirmation page
- Dark and light mode

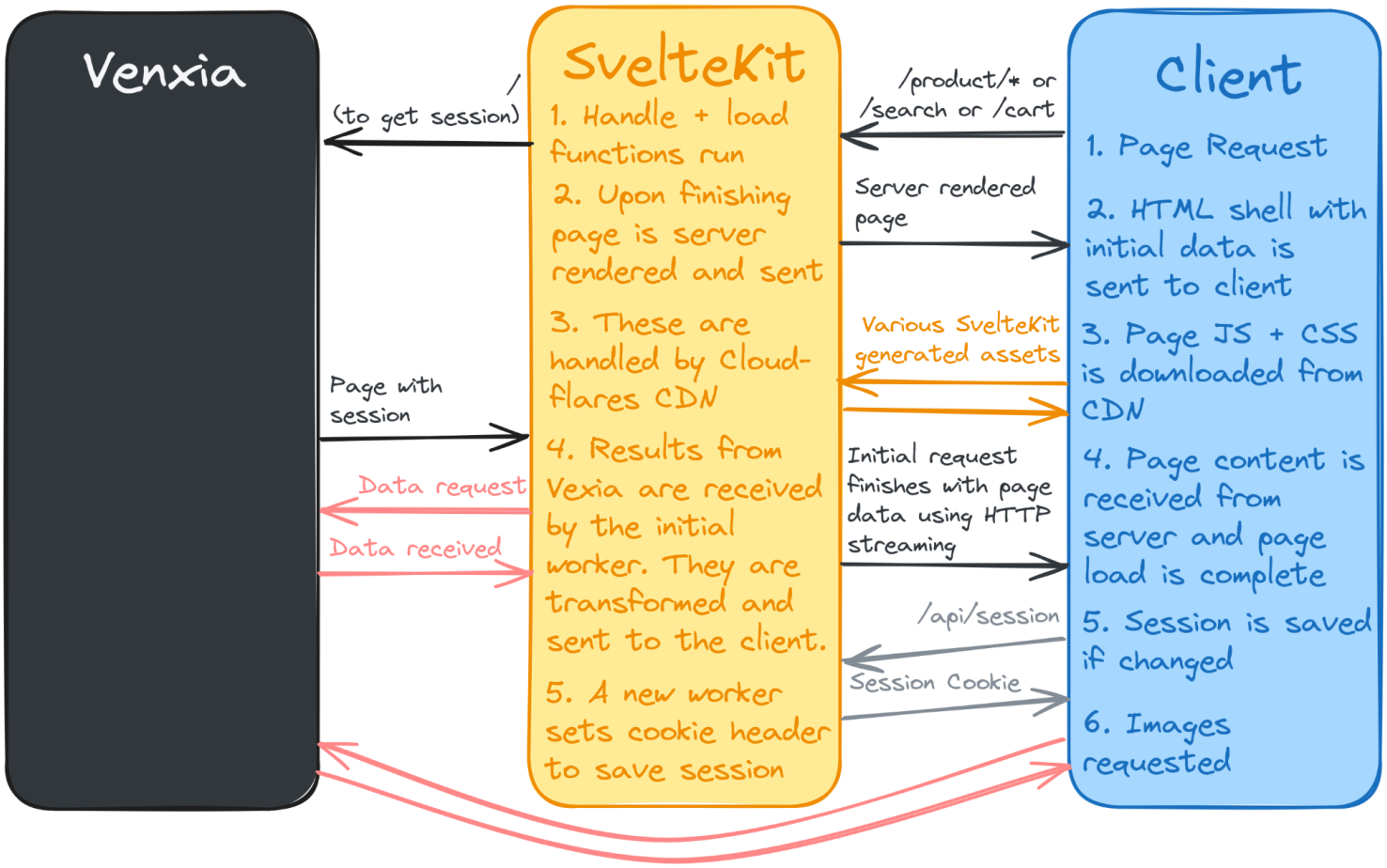
Why?

I made Vedgekit to prove that your website could be made vastly more performant for a very reasonable outlay of effort. I will also use it to display many of my proposed solutions. This will provide a real-world comparison.

How?

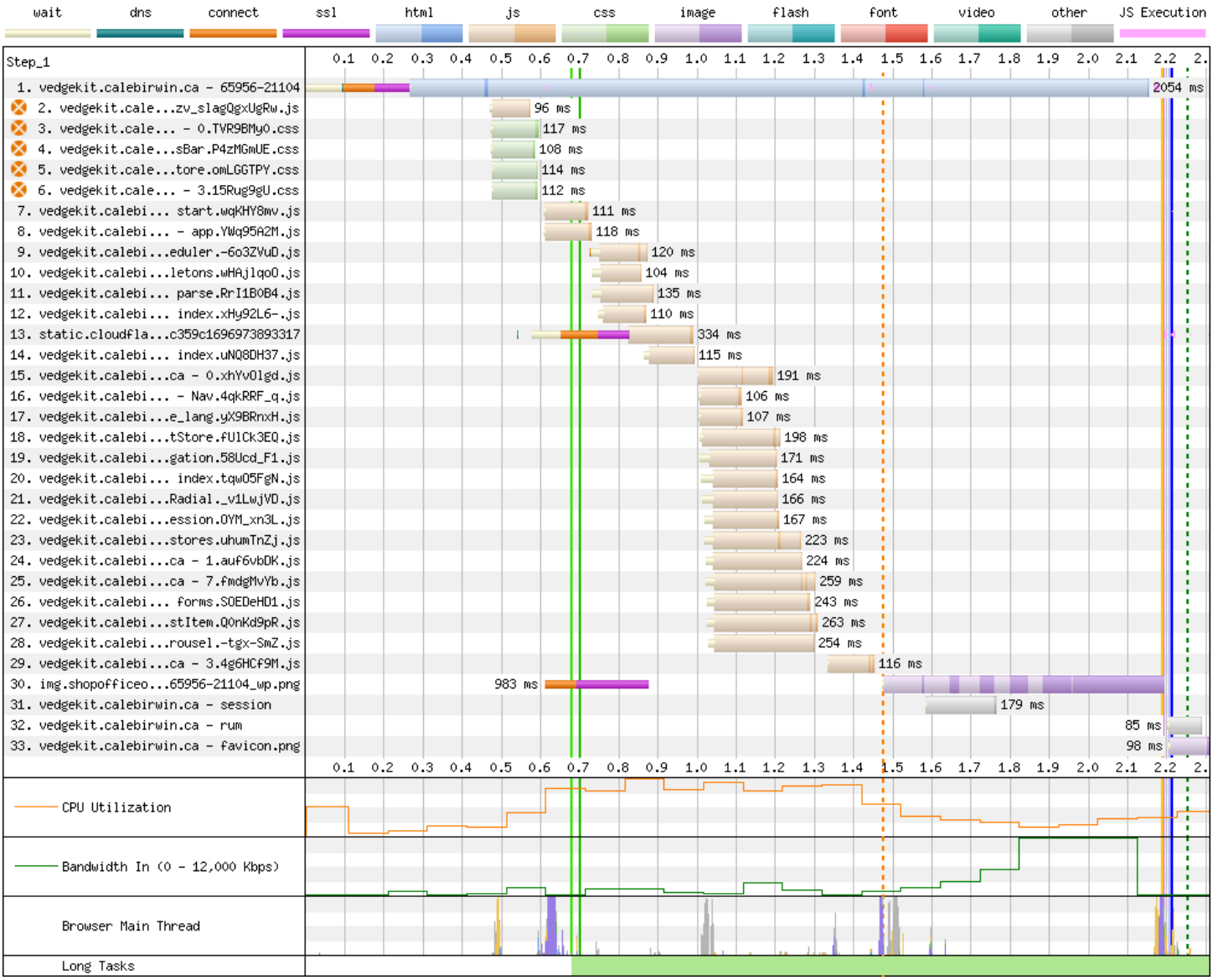
The code is available to read on GitHub. Vedgekit's tech stack is in it's name:

- **Venxia**: Your eCommerce platform
- Cloudflare's **Edge** Workers: A globally distributed serverless runtime. It is incredibly cheap to use, has zero cold starts (on the paid plan), and offers the low latency of a CDN. It runs in 310 data centers in Toronto, Vancouver, Montreal, Ottawa, Winnipeg, Calgary and Saskatoon. This is the best web runtime for this project. However, SvelteKit can run on many different platforms, including Netlify (AWS Lambda), Vercel (AWS Lambda and Cloudflare Workers), or any server running Node.js, Bun, or Deno. It should not be hosted using Lambda because that runtime does not support the Stream's API.
- Svelte**Kit**: A modern full-stack Javascript framework. My demo version of Vedgekit is hosted on Cloudflare Pages. SvelteKit is powered by Svelte which is highly performant, compiled, and feature complete.



This is a simplified chart. All data requests to Venxia are processed by SvelteKit and streamed to the client as soon as each is finished. So first the session is sent, then initial data (ex. product information), and then supplementary data is sent (ex. related products). Learn about the Streams API here.

Waterfall



Page: https://vedgekit.calebirwin.ca/product/65956-21104
Test: First View. 12000/12000 Kbps, 70ms Latency. Toronto, Canada.

First Contentful Paint: 0.7s (solid darker green vertical line)	**Data Paint:** 1.5s (layout shift, dashed orange line)	**Largest Contentful Paint:** 2.2s (dash dark green line)

Issues And Shortcomings

Being an early demo Vedgekit has many issues large and small. These include…

- Does not support user accounts (except through proxied checkout)
- Does not have its own checkout flow
- Search filters, including categories, are not currently implemented
- Does not support French
- Sessions can be corrupted (breaking the user's cart till a new session is issued)
- Overreliance on the Streams API (ideally would be completely server rendered but not due to your APIs slowness)
- SEO tags are not added
- No sitemap
- Added ~800ms of information delay on first load (needed to get user session and cart; avoided if the landing page is the home page)
- Static home page (generated at build time)
- Light/dark mode toggle does not persist preference (this is easily fixable)
- Massive images (due to just using yours; the homepage uses smaller ones generated at build time)
- Various smaller missing features

These issues are solvable but I can not currently justify the time commitment to do so. More systemic issues include…

- Reliance on a reverse-engineered API (there are likely many unaccounted-for edge cases)
- Will break if you change your site much

Major Issues

Server: Response Time

What: This is the time it takes to connect to the server, establish TLS, and receive the first part of your document (<!DOCTYPE html>). This is measured by time to first byte (TTFB).
Issue: 750ms (in a test simulating 70ms latency, Toronto). Real-world: ~800ms (my laptop with DNS cached).

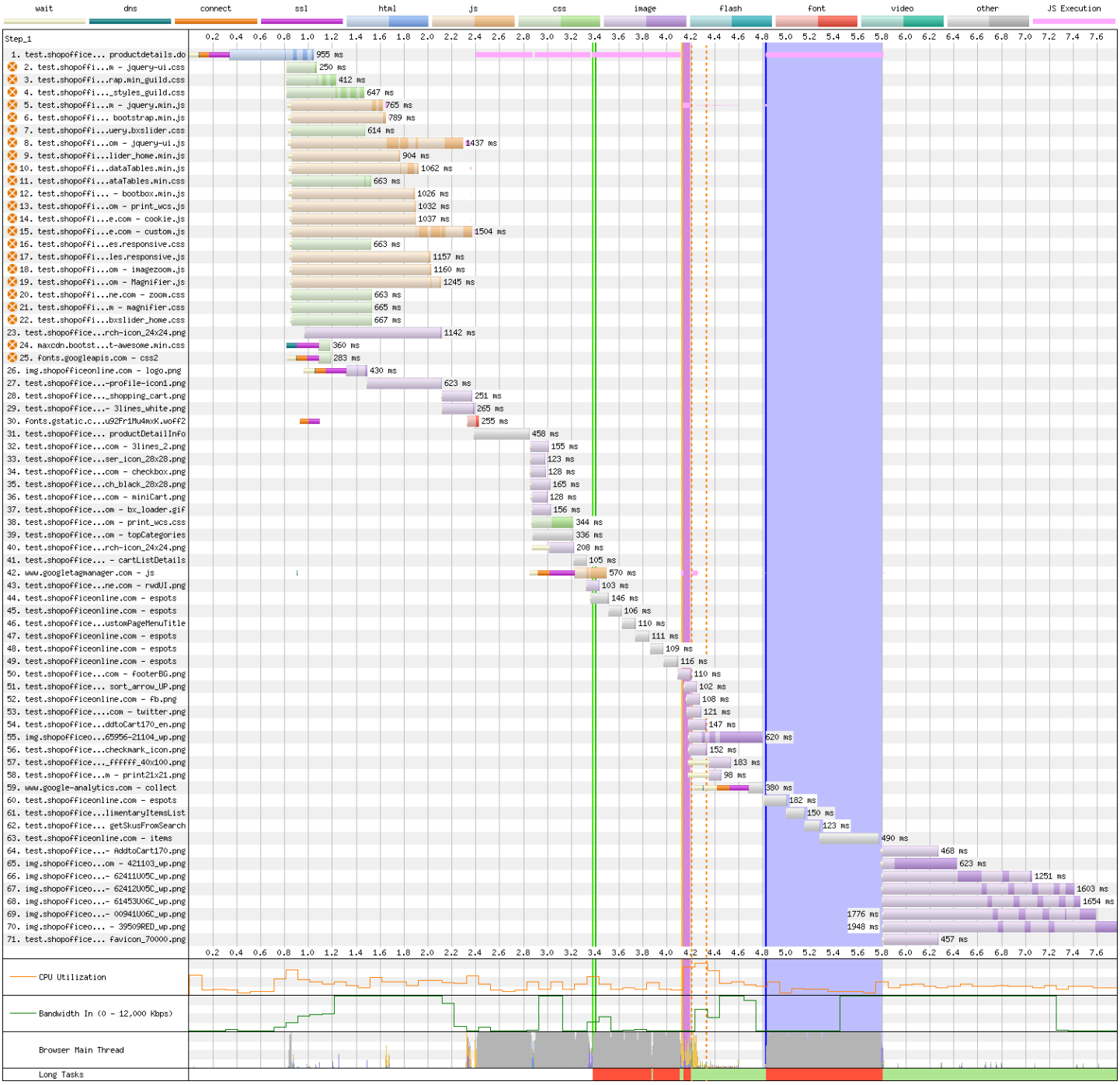
Home Page

Solution: I do not know how to speed up your server. On many API requests, the waiting time is over 500ms. I do not know how this is possible and all efforts should be made to improve your server's performance. The connection and TLS setup time can be reduced if you proxy your site through Cloudflare's free CDN. When your site is visited people will connect to Cloudflare's incredibly fast edge network. The actual traffic will be routed to your servers but the connection and TLS will go through their closer servers. A bonus is that with the right caching headers, you can also cache static files on their network reducing the work your server does and improving your website's speed. While this is not a perfect solution it is free.
Vedgekit: 450ms (in a test simulating 70ms latency, Toronto). Real-world: ~200ms (my laptop with DNS cached). Hosted on Cloudflare.

Home Page

Architecture: Key Resource Chaining (Waterfall)

What: This issue is when display critical resources are loaded sequentially rather than in parallel. This occurs when you have to wait for one resource to load to start to fetch the next one.
Issue:



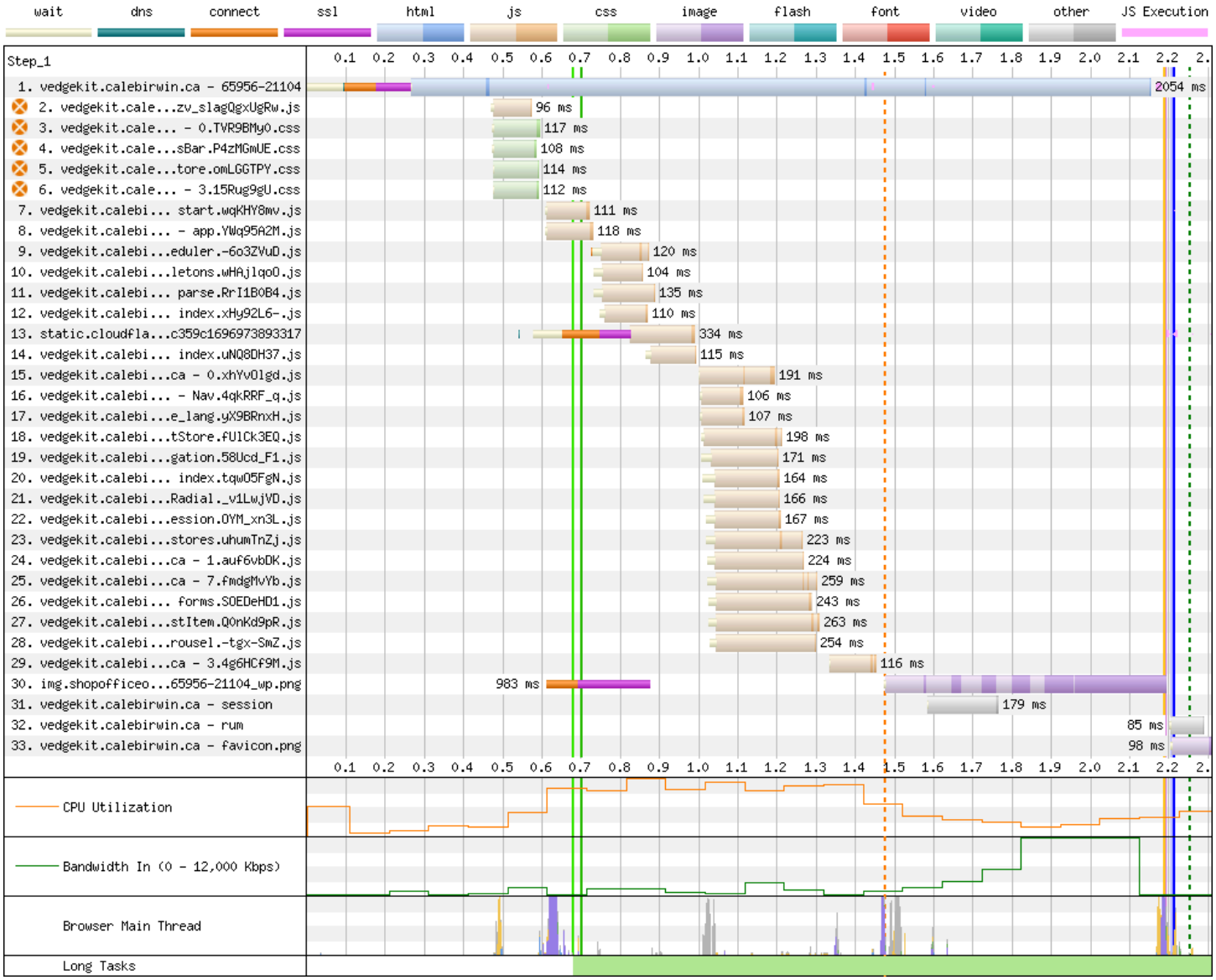
Product Page Waterfall

Your waterfall shows key requests are chained.

1. 0.8s: Slow initial request (HTML)
2. 2.4s: All your massive resources (CSS + JS)
3. 2.9s: Product data request
4. 3.4s: Cart
5. 4.1s: Espots
6. 4.8s: Other images
7. 5.8s: Complementary products
8. 7.7s: Complementary images

Solution: Include all critical data in the initial request. This would eliminate your chained requests. I am not certain what architecture you are using. You are not using [Jamstack]() because you are not using a CDN and because your initial request includes data. You are not doing monolith well due to your reliance on client-side requests.

Vedgekit:



Vedgekit Product Page

1. 0.4s: Initial request starts and the entire HTML page is downloaded
2. 0.6s: Render critical CSS + JS is downloaded (loading skeleton rendered)
3. 1.4s: Hydration JS is downloaded (page becomes fully interactive)
4. 1.5s: Product data is streamed to the client through the initial request (as well as cart data, which is streamed at an earlier not clearly delineated time)
5. 1.8s: Session is received from initial request and is saved (request sets session cookie).
6. 2.1s: Complementary product data is streamed to the client and the initial request ends (images not loaded due to lazy loading and being off-screen on mobile).
7. 2.2s: Product image finishes downloading
8. Done: All future pages are loaded in PWA style and are quicker to load.

The request waterfall occurs on the server, not the client. This speeds up page load due to the server starting the requests instantly, as opposed to having to wait for the client to download the JS bundle. In an ideal world, your API would respond quicker meaning my server would respond quicker. To render the page there is exactly one set of requests that has to be made.

Content Delivery: Images

What: You should serve responsive images in modern formats as outlined in this article.
Issue: You use one oversized image in the worst format. You MUST add thumbnails. 250kb images for search result thumbnails is crazy. For example, loading 48 search results requires 12.8 MB of images.
Solution: There are multiple ways to do this. The cheapest way (if we ignore engineering time) is to generate multiple image sizes on your server. If you proxied your site through Cloudflare you could ensure that the correctly sized images are delivered quickly. This option is free except for engineering time.
A second option is to use Cloudflare Images. The 2 relevant services they offer are Cloudflare Images and Cloudflare Images Resizing. The first offers storage and resizing but requires more setup. The second option resizes your images on the fly. Pricing is $5 per month per 100k images stored and $1 per month per 100k images delivered (plus various plan fees of ~$20/month)..
Vedgekit: I do not optimize your images because there is no free way for me to do so. However, I do optimize the images displayed on the home page. The slide show images are ~90 KB JPEGs each and the thumbnails are ~15 KB PNGs each. This is not even using next-gen formats.

Content Delivery: Improper Practices

What: All content should be served in the most efficient and performant way. Static content should be served with long-lasting caching headers. Static content should be served from a CDN.
Issue:

Minify CSS — Potential savings of 131 KiB

Minifying CSS files can reduce network payload sizes. Learn how to minify CSS. FCP LCP

URL	Transfer Size	Potential Savings
shopofficeonline.com 1st Party	577.2 KiB	130.9 KiB
...css/print_wcs.css (test.shopofficeonline.com)	261.6 KiB	69.1 KiB
...css/wci_styles_guild.css (test.shopofficeonline.com)	155.7 KiB	40.8 KiB
...css/bootstrap.min_guild.css (test.shopofficeonline.com)	107.4 KiB	13.2 KiB
...css/jquery-ui.css (test.shopofficeonline.com)	36.8 KiB	5.6 KiB
...css/jquery.dataTables.min.css (test.shopofficeonline.com)	15.6 KiB	2.2 KiB

Minify JavaScript — Potential savings of 263 KiB

Minifying JavaScript files can reduce payload sizes and script parse time. Learn how to minify JavaScript. FCP LCP

URL	Transfer Size	Potential Savings
shopofficeonline.com 1st Party	829.5 KiB	246.4 KiB
...1.11.4/jquery-ui.js (test.shopofficeonline.com)	476.2 KiB	154.4 KiB
...js/custom.js (test.shopofficeonline.com)	353.4 KiB	92.0 KiB
Unattributable	53.3 KiB	16.5 KiB
inline: $(function () { _	28.4 KiB	10.3 KiB
inline: if('G' != 'G'){ var sCartId = '_	25.0 KiB	6.2 KiB

Serve static assets with an efficient cache policy — 18 resources found

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

URL	Cache TTL	Transfer Size
shopofficeonline.com 1st Party		5,714 KiB
...Attachment/04-OP-Top-Ad-E-EN.png (img.shopofficeonline.com)	None	900 KiB
...850/65956-21104_wp.png (img.shopofficeonline.com)	None	582 KiB
...850/20583-61240_wp.png (img.shopofficeonline.com)	None	540 KiB
...Attachment/01-OP-Top-Ad-E-EN.png (img.shopofficeonline.com)	None	478 KiB
...Attachment/02-OP-Top-Ad-E-EN.png (img.shopofficeonline.com)	None	474 KiB
...Attachment/03-OP-Top-Ad-E-EN.png (img.shopofficeonline.com)	None	466 KiB
...850/51-0812_wp.png (img.shopofficeonline.com)	None	459 KiB
...850/51-0312_wp.png (img.shopofficeonline.com)	None	384 KiB
...850/51-1408_wp.png (img.shopofficeonline.com)	None	361 KiB
...Attachment/05-OP-Top-Ad-E-EN.png (img.shopofficeonline.com)	None	265 KiB
...850/65956-03504_wp.png (img.shopofficeonline.com)	None	243 KiB
...Attachment/01-OP-Bottom.png (img.shopofficeonline.com)	None	203 KiB
...Attachment/02-OP-Bottom.png (img.shopofficeonline.com)	None	126 KiB
...Attachment/OP-F-Brand-1.png (img.shopofficeonline.com)	None	56 KiB
...Attachment/OP-F-Brand-3.png (img.shopofficeonline.com)	None	55 KiB
...Attachment/OP-F-Brand-2.png (img.shopofficeonline.com)	None	48 KiB
...Attachment/OP-F-Brand-4.png (img.shopofficeonline.com)	None	46 KiB
...70000/logo.png (img.shopofficeonline.com)	None	29 KiB

- **Incorrect Cache Headers:** Your JS + CSS including libraries like jQuery are intentionally not cached! The exact header is: "Cache-Control: no-cache, no-store, max-age=0, must-revalidate." Some images are missing any cache policy.
- **Unminified Resouces:** jQuery UI, custom.js, and various CSS files are not minified.
- **PNGs:** This is the largest format you could use.
- **No CDN:** All static assets should be served from a CDN. They offer many benefits.

Solution: Once again use Cloudflare's CDN. Set the correct headers on your server. Cloudflare can be configured to minify your files for you.
Vedgekit: All static files have infinite browser cache and are stored on Cloudflare's CDN. SvelteKit's build step creates unique names for the files and minifies and splits all code for efficient caching.

Client: Mobile Usability

What

1. Page width should always match screen width
2. Buttons should be sized for touch
3. Smaller images should be used on mobile
4. Slow high-latency connections should be anticipated

Issue: These are a by-product of other issues and incorrect styles.

1. You can scroll in the x-axis. The page is too wide!
2. Various buttons are too small
3. All images are too large
4. Long waterfalls are awful on high-latency connections. Furthermore, pages require too much code to be downloaded.

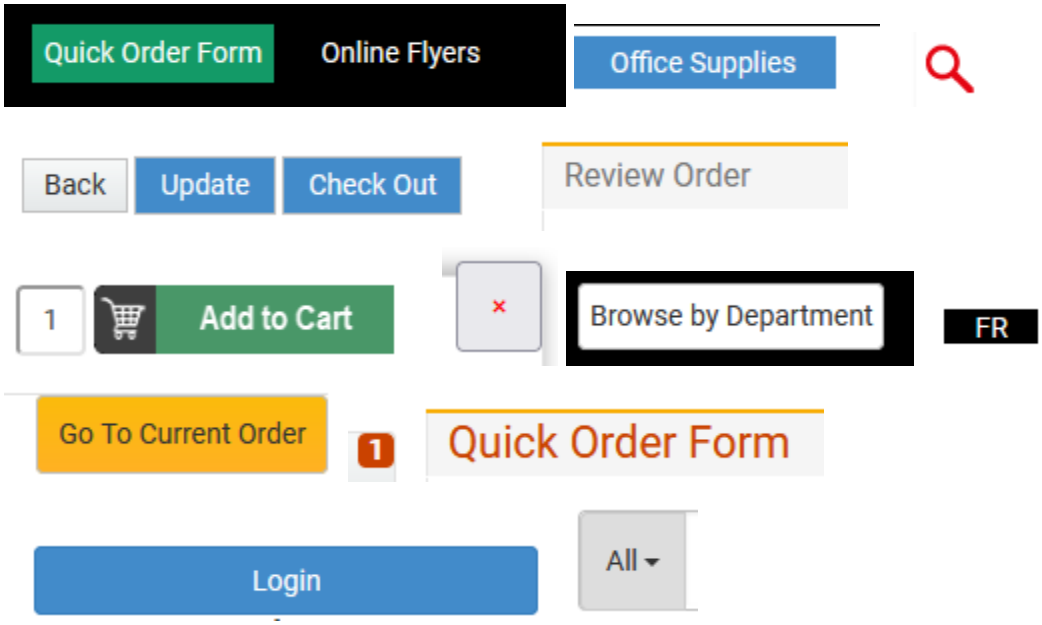
Solution

1. Fix styles
2. Fix styles
3. Fix images
4. Fix architecture (and other performance issues)

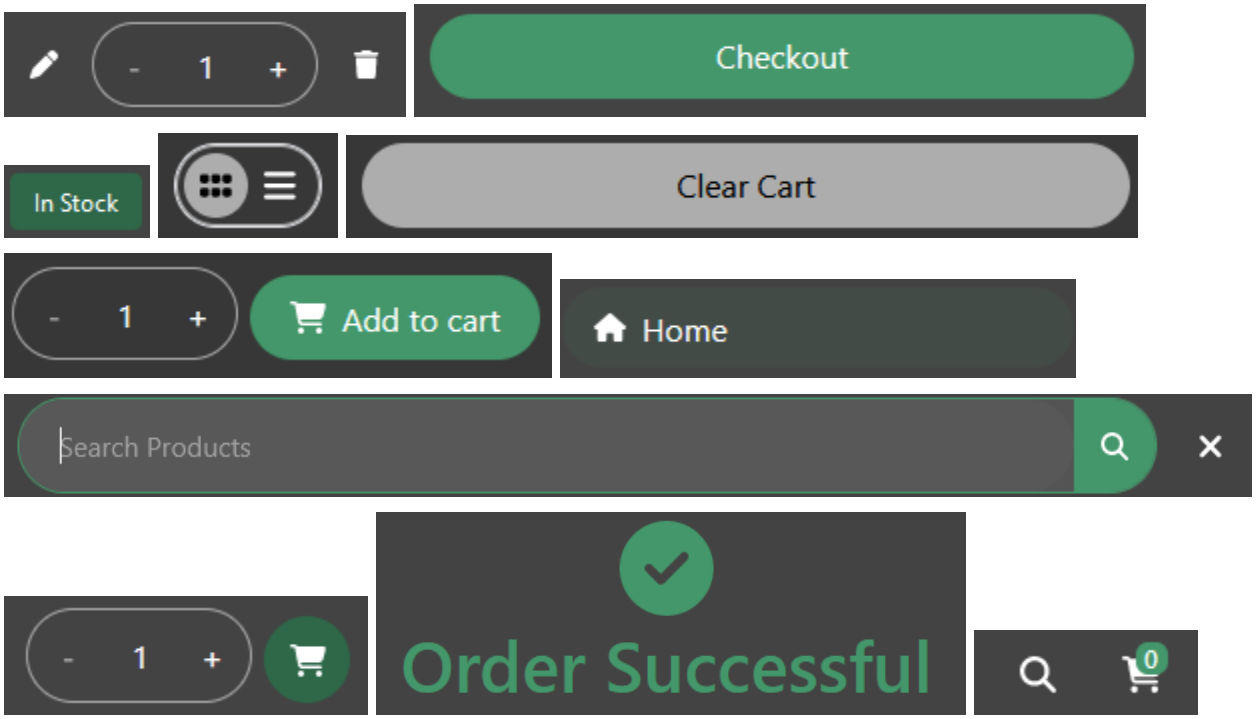
Vedgekit: This was developed with mobile in mind. It works great on mobile. Unfortunately, the large image sizes reduce performance.

Client: Lack of A Unified and Modern Design System

What: You should have a limited amount of button designs and accent colours. Style should generally follow modern sensibilities.
Issue: 4 accent colours and countless button designs.



Solution: Restyle. There are many ways to do this.
Vedgekit: 3 accent colours (guild green, darker guild green, gray). It is built using Skeleton, Tailwind, and Font Awesome. Light mode and dark mode are both supported. All colours and most styles can be easily edited in a single place.



Outdated Technology And Technical Debt

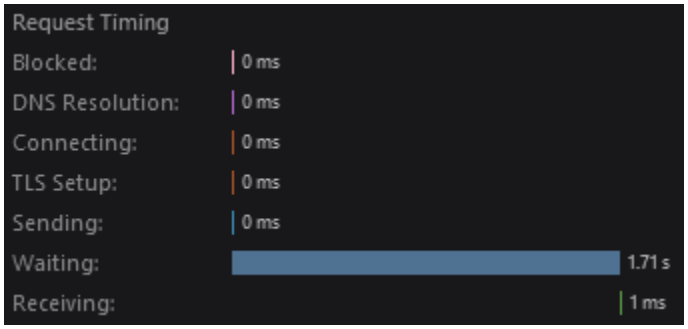
What: “Technical debt… is the implied cost of future reworking required when choosing an easy but limited solution instead of a better approach that could take more time” -Wikipedia.
Issue: Your site is powered by old technology, namely jQuery, jQuery UI, and Bootstrap. According to ChatGPT, “jQuery, jQuery UI, and Bootstrap formed a popular UI stack from the late 2000s to the early 2010s, offering developers versatile tools for web interactivity and design.” Your front end heavily relies on the global namespace. This makes it very hard to refactor. jQuery’s imperative model is generally worse than declarative methods because it makes it hard to keep track of application state. Your architecture shows significant technical debt. Having so many sequential requests makes no sense performance-wise. As I have been reverse engineering your APIs, I have found many, many empty fields, strange fixes, and questionable choices. Your JS has seemingly dead code and never used local variables. Simply put, your code is a tangled mess of global functions that rely on large outdated libraries.
Solution: Solving all your site's many problems will take much effort. If you choose the right solutions you can greatly reduce your technical debt. A *potentially* simpler solution is a complete rewrite of your front end. If you were to rewrite your front end you would want to edit some of your APIs so that fewer round trips are required (ex. /productDetails returns all relevant product information including related products). While I love SvelteKit it is not the most popular framework. A good bet would be to use Next.js which is powered by React. Nuxt, a Vue framework, is also supposed to be very good and feature-complete.
Vedgekit: Vedgekit is built on very modern technology. I am sure some technical debt has already accumulated, but it impacts performance significantly less than yours does. Replicating Vedgekit would take a senior engineer with domain knowledge less than a week of work. Getting this new site to be fully usable would most certainly take longer. This is because it would take time to add all features and ensure reliability. Vedgekit has proved that a rewrite utilizing your existing APIs is not that hard.

Other Issues

Server: Search Results

The most relevant items are not always at the top of the results. I do not know the best way for you to resolve this issue.
Ex: https://vedgekit.calebirwin.ca/search?query=red+letter+paper. The only correct result is the 15th result.

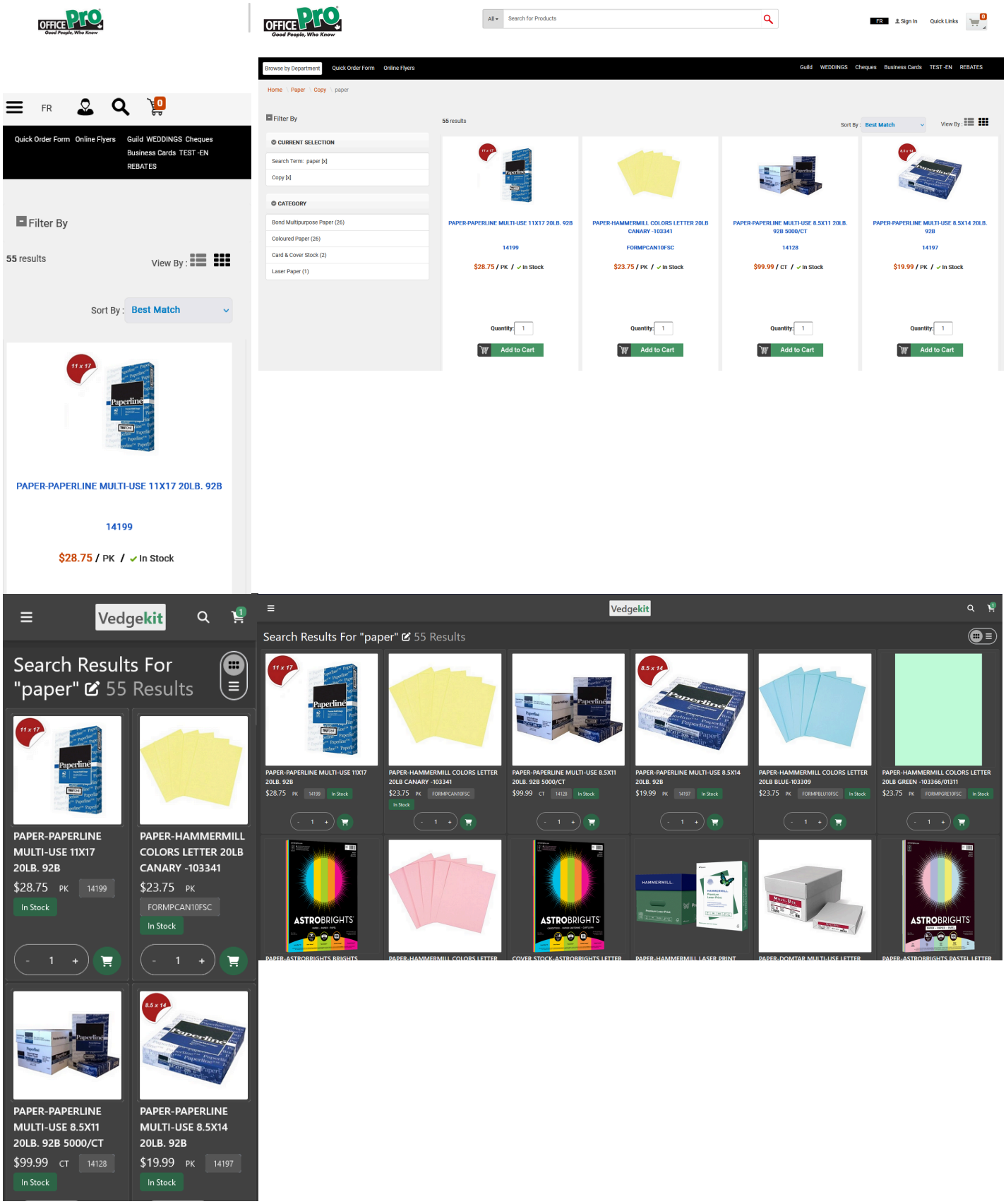
Search results are very sluggish (API takes ~1700ms per page of 12 items). This is unusually slow and I am sure you will find some major problems with the serverside code that generates results.



https://test.shopofficeonline.com/search?searchTerm=paper&... (on my laptop)

Client: Sparse Content

Solution: Change styles to increase content density.



Client: Cart Issues

- Pressing the update button on the cart page kicks the user to the homepage in WebKit (all iOS and iPad OS browsers and Safari on Mac). Firefox asks if you are sure you want to resubmit the form. It only works properly on Chromium browsers.
- Also, in Safari the cart button is still disabled after adding a product to the cart on the homepage. A reload is required to enable it.

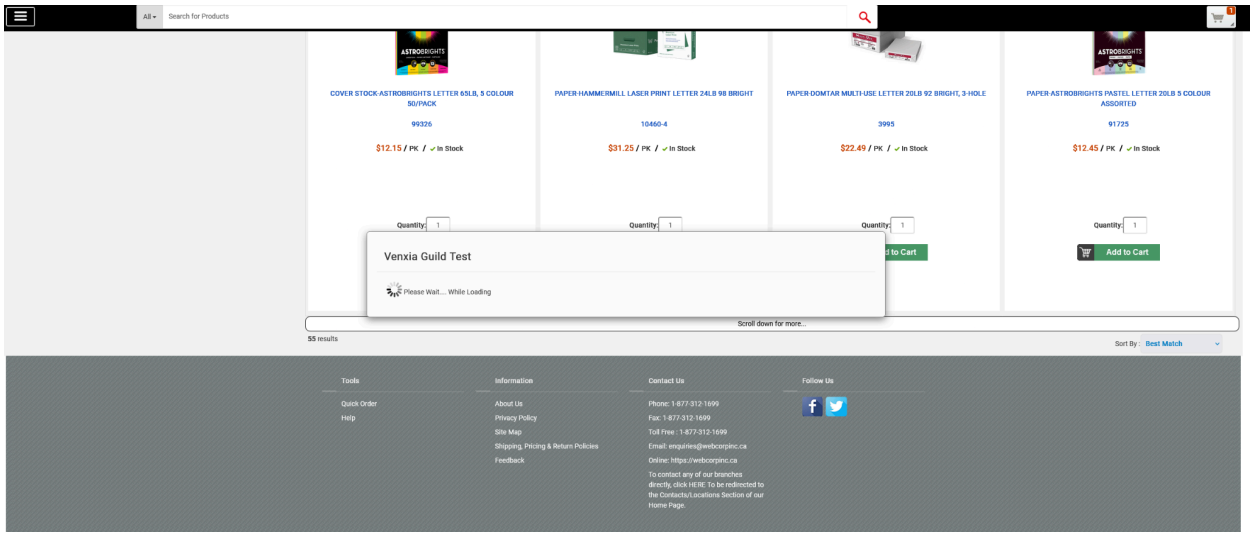
These both require bug fixes. I do not know exact solutions but these are not difficult problems.

Client: Content Shift From Sticky Navigation Bar

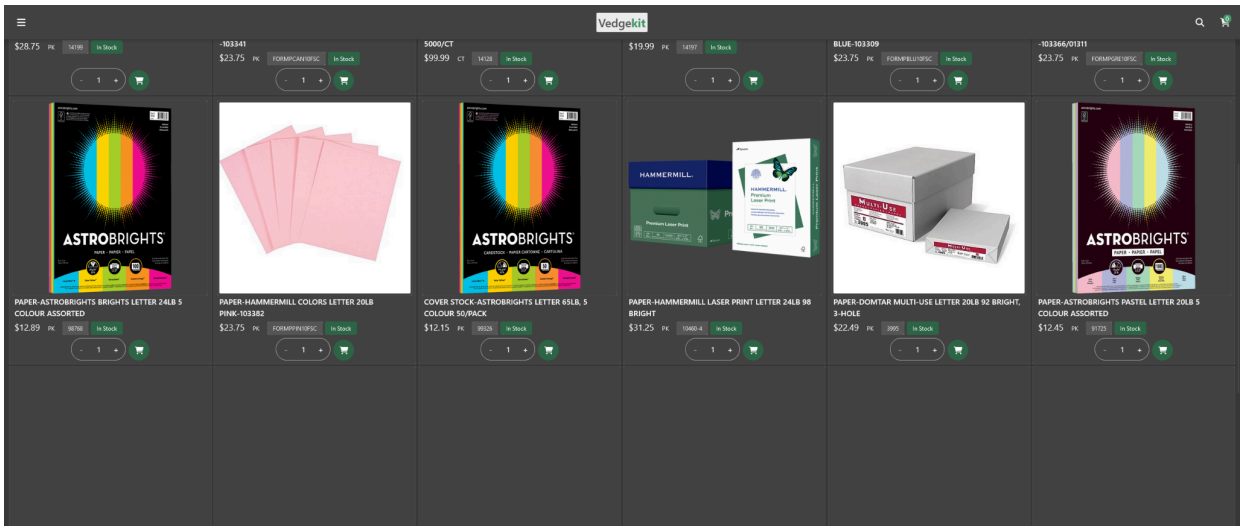
When you scroll from the top of the page the sticky navigation bar pops into existence. This causes content to shift down on the x-axis. This can be fixed by adjusting the styles. You appear to be using JavaScript to trigger its appearance. This effect can be achieved in CSS which is likely better (no scroll event listeners).

Client: Popups

Your use of popups is generally excessive. Having a full-screen notification for more results on infinite scroll results is a poor user experience. If you must notify the user of something a toast is better.



This popup prevents all interaction. It is abrupt.



Vedgekit has empty boxes to represent loading items. They slowly fade in and out as the next 12 items are loading.

Client: Preload Main Image

Main images should be preloaded. On product pages, this would be the primary product image. On the home page, this would be the first image of the promotional carousel. See this article. Related article on preloading.

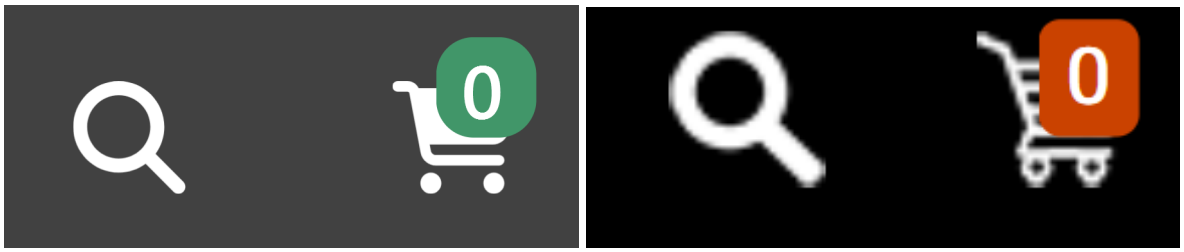
```
<head>
…
<link rel="preload" as="image" href="important.png">
<!-- or -->
<link rel="preload" as="image" href="wolf.jpg" imagesrcset="wolf_400px.jpg
400w, wolf_800px.jpg 800w, wolf_1600px.jpg 1600w" imagesizes="50vw">
…
</head>
```

Client: Lazy Load Images

As a general rule, non-key images that can be off-screen initially should be lazy-loaded. This is especially true if your images are oversized.

```
<img src="image.png" loading="lazy" alt="…" width="200" height="200">
```

Client: Icons Should Not Be PNGs



Your icons have a larger file size but are infinitely less crisp. SVGs are great. I use Font Awesome. It is free.

Conclusion

It is my firm belief that the current state of Venxia is unacceptable. If the major issues of performance, appearance, and usability are addressed, I believe that Office Experts would be able to use Web Corp's product. I understand that addressing these issues would be a lot of work. A secondary solution would be for Web Corp to provide Office Experts with some assurance that the Venxia API will stay the same and a better way to handle sessions. This would allow us to, with further development, use Vedgekit as our primary website. My sincere hope is that something mutually positive will come from this report. Please feel free to reach out to me for clarification or to arrange future discussions. Thank you for your careful consideration.

Caleb Irwin
Office Experts Office Pro
me@calebirwin.ca
info@officeexperts.ca