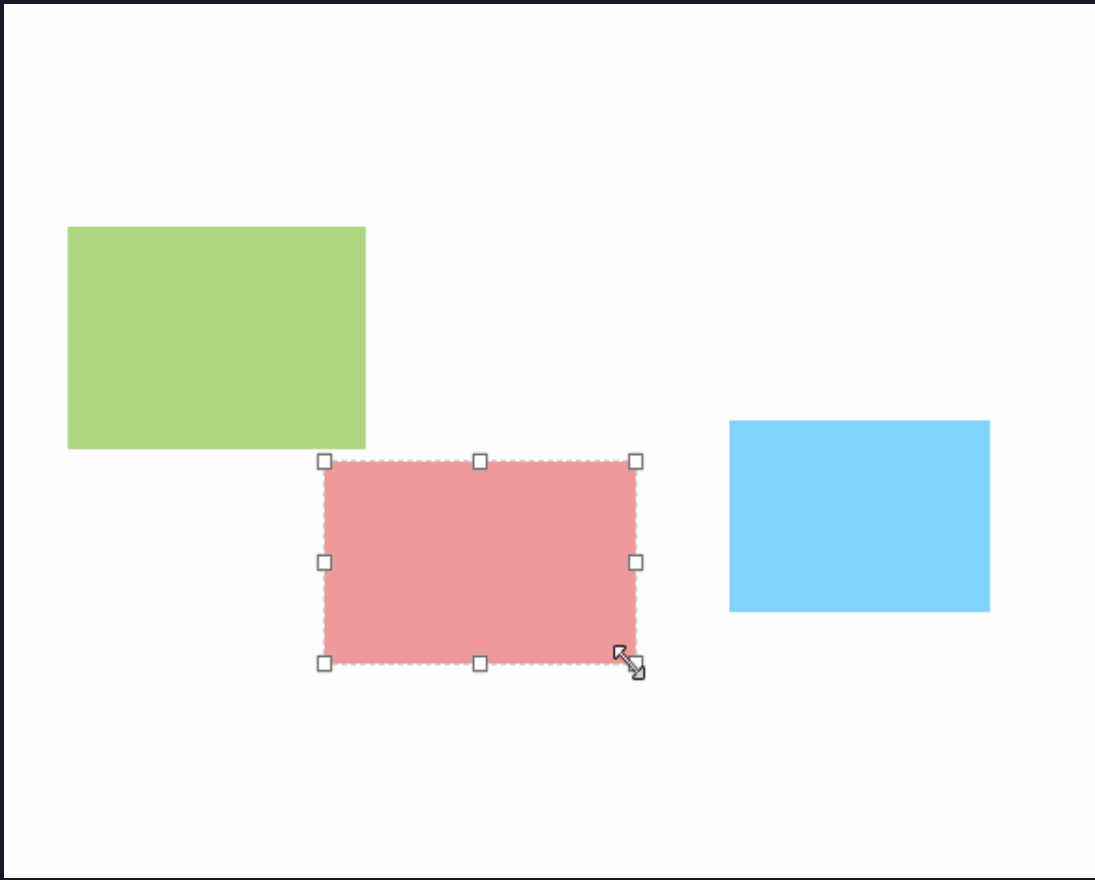
CONTENT MANAGEMENT SYSTEM IMPROVEMENTS

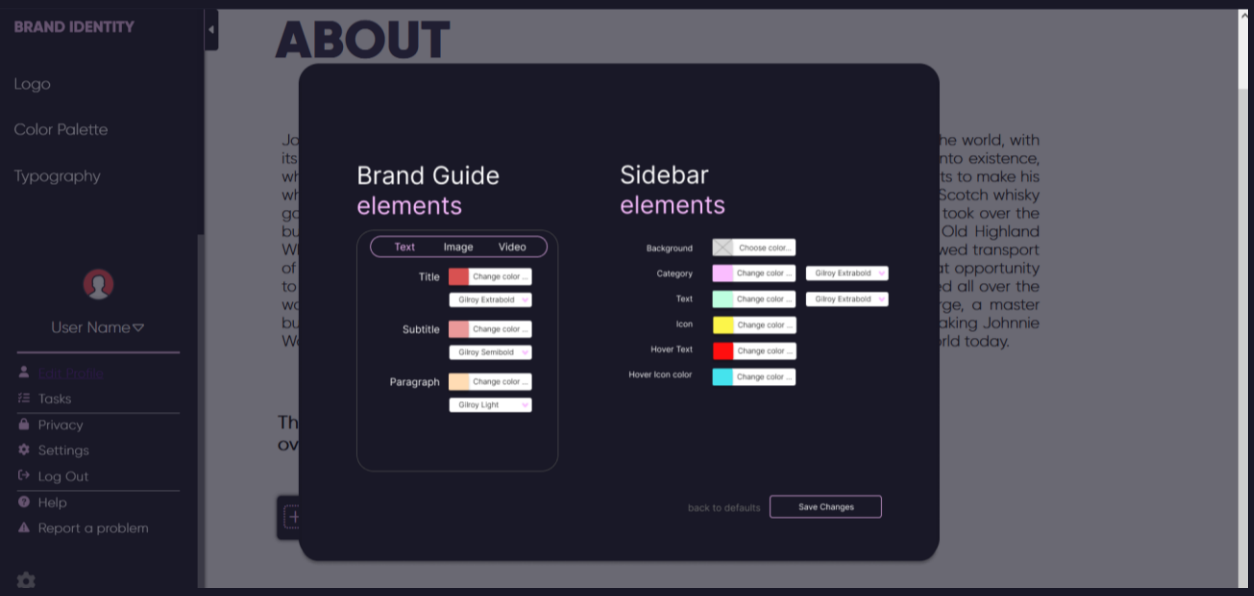
Techniques

Layout renderer with dragging and resizing
Choose between fixed, free, and other asset containers

https://vuejsexamples.com/vue-component-for-draggable-and-resizable-elements/

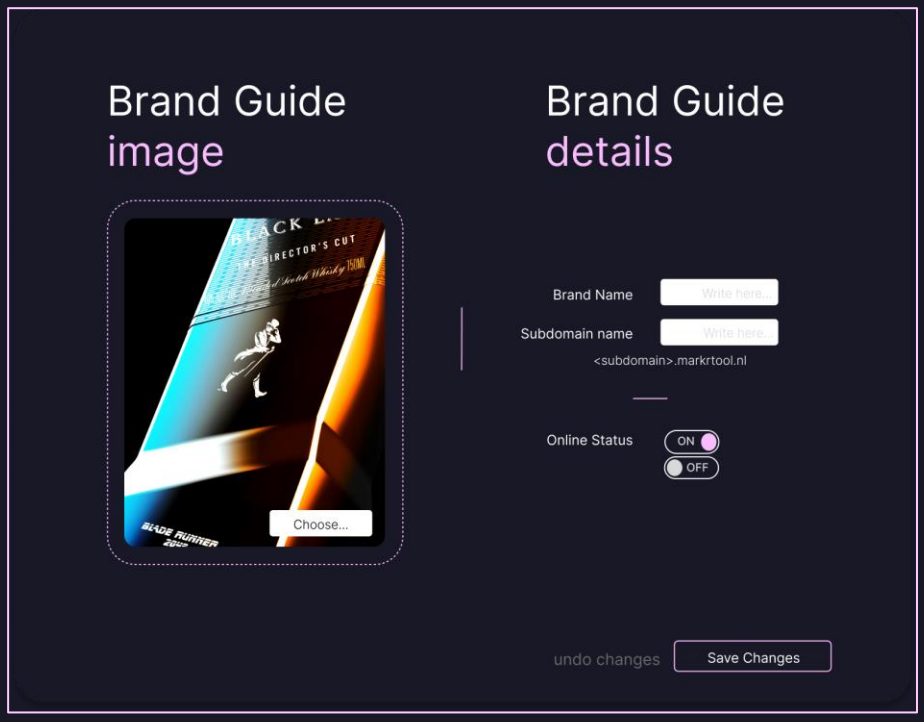


Look and feel customisation



TRANSCENDING CONTENT MANAGEMENT

Markr tool as a platform. *The future through the lens of UX*



In the context of Brand Guide Systems, We created a tool to write and author changes and have those changes propagate to the published BGS. This, in many ways, is still about content management.

We have also figured out a way to deploy and host the published BGS conveniently on its own subdomain of our tool, this of course helps **client perception** of your work on a BGS, as a person with a marketing, design, and or branding role within a company.

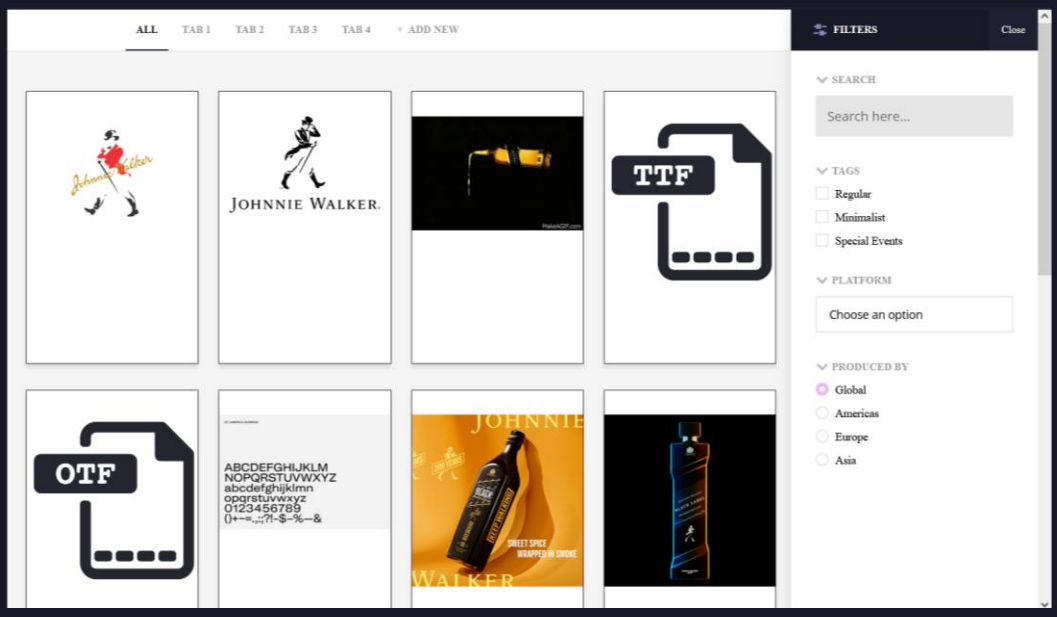
But there is more that can be done.

We believe that Markr can transcend content management and become THE branding platform to act as an anchor point for all your agencies branding needs. Below are some ways we could make this a reality.

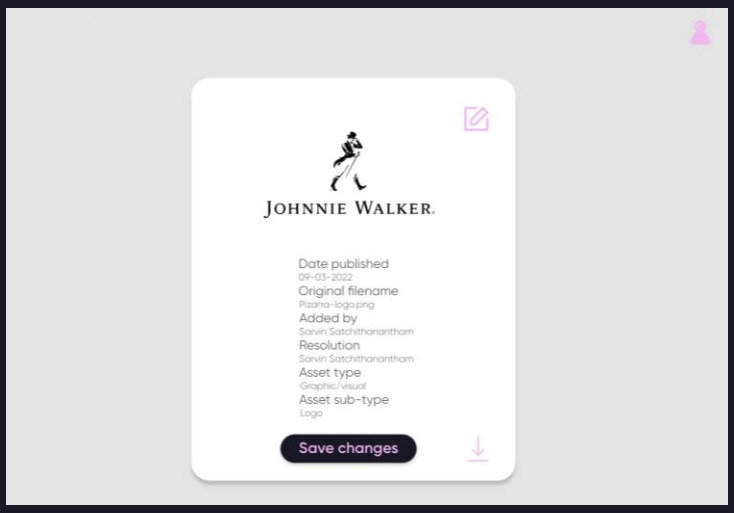
Techniques

Digital asset management

A home for all branding related company assets



The first step would be to create a route on the backend to get all assets from all pages, but with information like which page it came from, and where else on the web this asset is found.



Real-time collaboration

The first step would be to make use of firebase observable events, the idea is that your web application subscribes to data changes in our API and then our API would, for the lifetime of our session connection, subscribe to changes in the database. This is quite simple to implement, but its not nearly as flexible as the way Google Docs or Figma does it.

They utilize this concept called Operational Transformations. In brief terms, OTs help us keep track of conflicts while multiple people are working on the same file, similar to what git does.

https://medium.com/coinmonks/operational-transformations-as-an-algorithm-for-automatic-conflict-resolution-3bf8920ea447

https://www.figma.com/blog/how-figmas-multiplayer-technology-works/

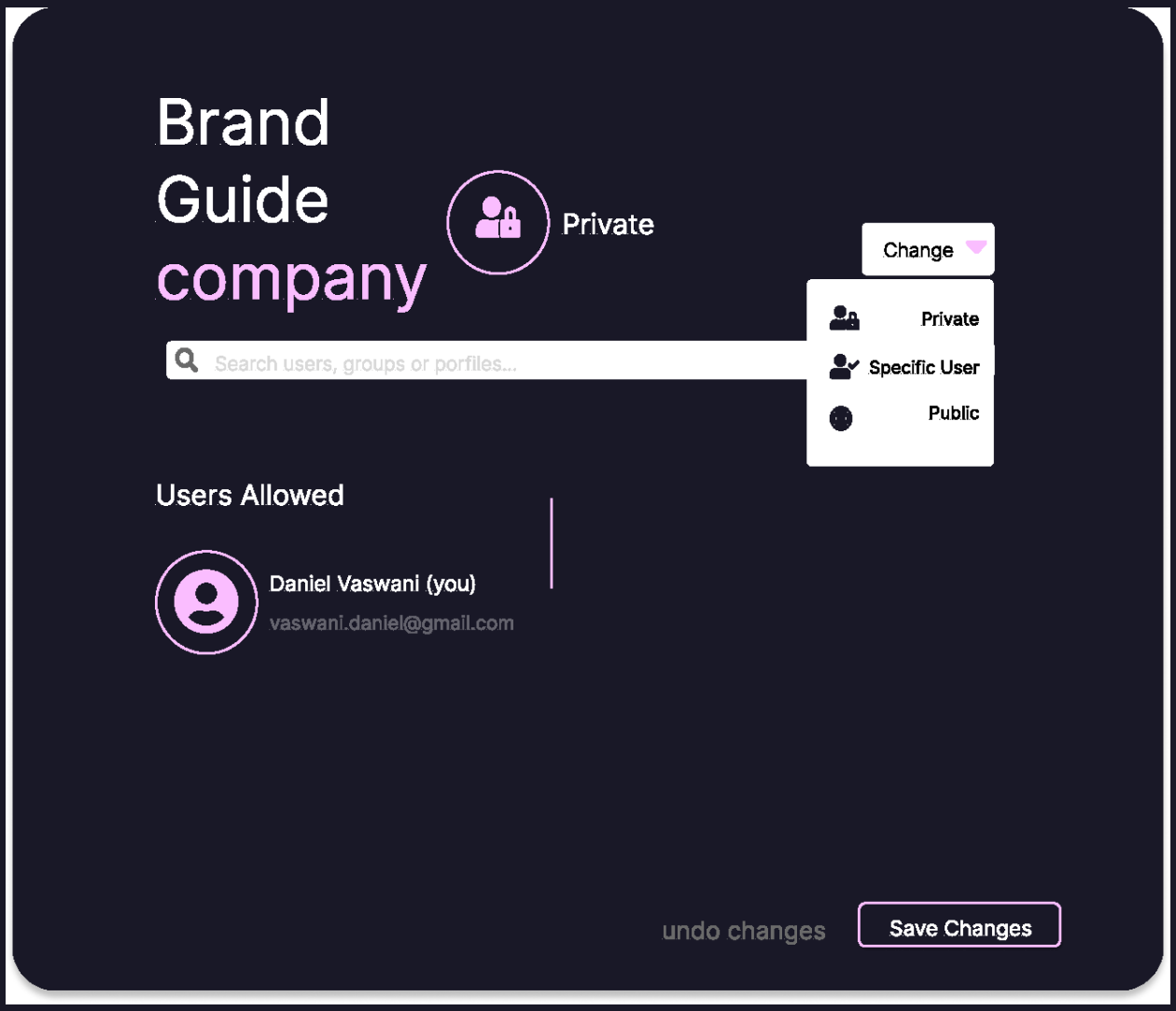


Image manipulation

Data driven design workflow
https://helpx.adobe.com/photoshop/using/creating-data-driven-graphics.html

https://helpx.adobe.com/photoshop/using/scripting.html

Using data driven design, we can make images dynamic, the only thing designers would need to adapt to is that they would mockup any information shown in a graphic, in a csv. E.g. titles, colors, and even imported image locations

Photoshop Document layers give us a good look at the creation process of an image, from a data standpoint.

Above is an example of designers using a CSV file to make the contents of their PSD Document dynamic, to clarify, changing one value on the CSV would propagate into the finished PSD, and then from there you could create a simple script to export it into various formats with options.

Converting from CSV to JSON and vice versa

Once converted, the process to store and retrieve the data in a NOSQL Document oriented database such as MongoDB or Cloud Firestore, is rather simple.

What's Next?

Implementing a dependency system on Markr tool, allowing assets to be dependent on others. For example, you could set text with different colors and image assets to be dependent on the brand guide colors you chose.

To get text to regenerate based on a changed color would be simple, but to get an image containing that color in its background would require extra care.

Photoshop does not run server-side, as it is a paid and proprietary image authoring SaaS meant to run on a desktop. We need photoshop to apply a changed dataset.

Photoshop API

https://developer.adobe.com/photoshop/api/ is a new API Adobe themselves launched, it is in public beta at the moment, but advertised on their website are the features we are looking for, and more. This of course, will mean that there is an added cost to worry about. There could also be an **unpredictable amount of time** before it is mature enough to use commercially.

Another positive, is that it might solve the 5MB limit on images on Markr. Due to the nature of the Photoshop API, it would be possible to use their own cloud storage to store images, and then from there we could consume the API directly in the frontend of our tool.

Their demo illustrates the best, the features/power we are looking for, specifically what's important is the Text Layer, Smart Object and the Image Cutout API.

With Photoshop API we can simplify our backend by taking advantage of JAMStack (JavaScript, APIs, Markup), in general, this is the trend of most new web apps created since the last year or so.

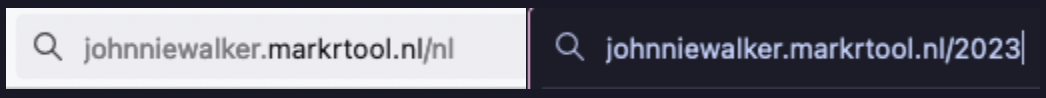
https://developer.adobe.com/photoshop/api/demo/

A homegrown solution

A solution that we could explore is to create a Markr Desktop Agent. A way to prompt the user to regenerate images on their local machine. Web browser tools such as Postman and Figma use these as a way to offer native OS features in the browser. This would require the user to install the user agent onto their computer. In this agent, we would prompt the user to open photoshop and run a script to apply new changes to the PSD, then collect it again, via the agent.

There is a lot of speculation on this route on the side of what Photoshop and the user's OS allows, and what the user is comfortable with, so our personal advice is to wait for the Photoshop API to be completed, for the simplest workflow and tech stack.

Brand Guide System versioning



By far the simplest to implement, and it has quite a lot of utility from a user standpoint.

All it would require is another layer of nesting data (in our NOSQL Document approach).

Instead of going through Users > BGSs > Pages it would need to look through to Users > BGSs > **Versions** > Pages

RECOMMENDED STACK

Vue + ExpressJS Admin Server + Firebase(Auth, Cloud Firestore, Cloud Storage)

We can recommend this stack simply for how lightweight it makes the tool and the displayed Brand Guide System application, no imports of firebase on either app directly.

We have 3 repositories for our code (This is where the vercel bot deploys directly from main branches)

1. markr-tool https://github.com/danielvaswani/markr-tool
2. markr-displayed-bgs https://github.com/danielvaswani/markr-displayed-bgs
3. Markr-admin-server https://github.com/danielvaswani/markr-admin-server/

   Has a README.md with routes you can use

Steps to get the subdomain configuration

Hosting through Vercel allows us to have

1. Buy domain
2. Invite vercel to the necessary repositories
3. Set DNS to vercel (unlimited wildcard * subdomains)
4. Point A record domain to **the tool** hosted website
5. Set * wildcard subdomains to route to displayed brand guide application
6. Link to <subdomain>.markrtool.nl from tool
7. Set markr-displayed-bgs to return a 404 if BGS with relevant subdomain not found

```
const API = import.meta.env.VITE_API;

fetch(`${API}/api/brandguides/${bgsName}?subdomain=true`)
  //parsing the json
  .then((response) => {
    // console.log(response);
    return response.json();
  })
  .then((data) => {
    // console.log(data);
    createApp(data.pages);
  })
  .catch((error) => {
    window.location = `https://${bgsName}.markrtool.nl/404`;
  });
```

host.split(".")[0] gives you the subdomain text