Advisory report

Shopify Hydrogen template / Masita B2C store prototype

Date	:	**12-06-2024**
Version	:	**1.1**
Status	:	**Definitive**
Author	:	**Luc Swinkels**

Version history

Version	Date	Author(s)	Amendments	Status
0.1	10-06-2024	Luc Swinkels	First draft	Draft
1.0	11-06-2024	Luc Swinkels	First definitive version	Definitive
1.1	12-06-2024	Luc Swinkels	Added recommendations and intro	Definitive

Table of contents

Introduction

This document serves as an advice to Moonly Software. In this document,
advice on how to continue the project after this internship, as well as an overview of the progress that has been made, will be discussed.

The aim of this document is to make the handover of documents and other deliverables as smooth as possible.

Context

Context

Masita, a leading Dutch sportswear brand, had a Magento webshop developed a few years ago. However, it turned out that there were a lot of different requirements and desires that were built as extensions to Magento by various developers, including Moonly. Additionally, Masita worked with other parties for tasks such as order picking and shipping, the ability to customise items, and selling products on other channels (such as bol.com, etc.). To integrate these software systems with the Magento webshop, a number of connections were written.

Due to insufficient consideration for future-proofing, scalability, and the Magento framework as a basis when creating these integrations, they did not function properly, leading to more and more cracks in the system. Whenever an external integration was updated, the entire order process behind the store would come to a halt. These technical problems alongside a lack of priority from the owner due to upcoming ownership changes, meant the webshop kept declining.

Ultimately, Masita chose to cease operations, and the webshop was taken offline due to escalating problems with the website. Currently, these integrations no longer work, and they have not been updated for over a year.

Last year, Masita was acquired by the current owner, and they are in the process of completely redesigning their webshops. Subsequently, Moonly was asked to start developing a B2B store so that they could begin selling items to businesses.

Now that Moonly is working on completing this B2B store, we want to think ahead and commission a study on the development of a brand new B2C store.

The current problem analysis is as follows: Masita is currently unable to sell products to B2C customers because all integrations are disabled on the current website (masita.com, which is often inaccessible), and it is not possible to order products. Additionally, this website, having not been updated for over a year, often experiences downtime and has low speed. Since the Magento store is currently unusable, and Masita does want to sell products again, we need to explore how a new B2C store can be realised.

Assignment

Moonly and Masita have decided to collaborate on a new B2B and B2C store so that Masita can resume selling their products. Since it was important for Masita to start selling to businesses first, a B2B store is currently in development. In the future, a B2C store will also need to be developed.

For the B2C store, Masita has provided several requirements. There must be the ability to sell printed shirts, customers must be able to manage orders, and it should be possible to sell products through external stores. The B2C store should have similar styling to the already in development B2B store, as it is based on Masita's branding. The lay-out, however, can be changed freely to cater towards individual customers instead of businesses.

Since Moonly wants to avoid the same pitfalls and problems encountered with the Magento store when developing this new store, we want to first conduct research into the best solution for developing this new store. This research should not only consider Masita's various requirements but also focus on how the B2C shop can encourage customers to start ordering from Masita again. While the B2B shop primarily focuses on functionality for retailers, the B2C shop needs to be optimised for UX and conversion to help Masita regain its position in the market.

Additionally, the results of this research are intended to be translated into a proof of concept/prototype of a B2C store so that Moonly can develop a new B2C shop with a validated foundation.

To validate the designs and/or prototypes, user tests should be set-up that can be done in-house, or through personal connections and social media channels as the target audience is not very niche. In this case, the target audience is anyone who wants to buy clothes for working out. While there is access to user databases with previous Masita customers, this project does not yet include global testing for the prototype.

The research will be conducted with Moonly as the client and has been initiated by Moonly in preparation for developing the B2C shop. Masita is not the client in this case (though they may be a stakeholder).

Research

Research questions

Main research question

- How can a new, user-friendly B2C webshop be built and validated by research?

Sub questions

- Which problems did the old magento-based B2C-webshop face and how can I avoid them?
- Which features are needed for the B2C-webshop and with which priority should they be implemented?
- Which platforms can and should be used to develop the new webshop?
- How can Masita's technical requirements be integrated?
- How can I make sure the new webshop is user-friendly and optimised for conversion?

Results and conclusions

- Which problems did the old magento-based B2C-webshop face and how can I avoid them?
    - By interviewing developers who worked on the old Magento webshop, I found out that the technology is quite limited when it comes to building custom features and there is not a lot of freedom in development. I also found out that there were organisational issues that caused more technical problems such as big feature requests in a small time frame causing hacky, temporary solutions.
- Which features are needed for the B2C-webshop and with which priority should they be implemented?
    - I made a list of requirements based on the project's needs and ordered them using the MoSCoW method. I discussed these requirements with my company mentor to see if I was missing anything.
- Which platforms can and should be used to develop the new webshop?
    - I looked at lots of different e-commerce platforms and their pros and cons. I ranked them on things like integrated tools, freedom of development, development community (and documentation), and ease of development. Ultimately deciding to go for a headless solution with Shopify and their Hydrogen framework due to it being a very popular platform that has all the requirements I need, and because a big part of the project was being able to reuse code for future projects that required a similar solution.
- How can Masita's technical requirements be integrated?
    - By researching ways to build custom integrations in specific e-commerce platforms, I looked at which platform had the most freedom of development. I figured out that Masita's requirements were fairly standard which meant developing these features such as a custom item builder would be doable as long as I had the freedom to build it within the technology/platform I chose.
- How can I make sure the new webshop is user-friendly and optimised for conversion?
    - By gathering constant feedback from peers and users on my designs, as well as testing a design prototype, I validated my designs and iterated on them with improvements.

Technology readiness level (TRL)

Current level (available features)

The current prototype has a TRL of 6. It is a system that can be used for basic demonstration of features, however there are still extra features that need to be added for it to become a fully functional live webshop.

The current available feature list consists of:

- Overview of collections (homepage)
  - Dynamically loaded based on Shopify collection data
- Overview of a specific collection's products (collection page)
  - Dynamically loaded based on Shopify collection data
- Overview of a specific product (product page)
  - Dynamically loaded based on Shopify product data, including extra product information based on metafields
  - Variant images if there is more than one including lightbox with gallery based on the product's images
  - Labels and other checks in place for sold out / low on stock products
- Adding a product to your cart
  - Includes adding multiple products / removing products
- Customising a product (if the product is customisable, defined in Shopify metafield)
- Overview of your cart
  - Shows product information including customisations
  - Includes both overview in an aside (product page) and a separate overview page (cart overview)
- Add (valid) discounts to your cart
  - Includes removing discounts
- Checkout
  - Only with dummy payment data for now, no active payment methods configured
- Search
  - Includes predictive search suggestions
  - Includes both search in an aside (anywhere) and a separate results page (search results)
- Navigation
  - Menu loaded dynamically based on Shopify menu data including subitems
- Design/responsiveness
  - Current prototype is based on the previously made designs and is fully responsive

Actions to reach the next level (missing features)

- Filtering through products (collection page)
  - UI already in place, query logic needs to be added
- Overview of collection products bug
  - Currently shows the first variant of the product, which means that if that variant is out of stock, it will look like the product is out of stock when a different variant could be in stock
- Product reviews (product page)
  - Not a lot of free options available that work well with Hydrogen
- Product variant selectors
  - Works, but since the built-in component (VariantSelector) from Shopify is very limited in configuration, I wasn't able to pre-emptively disable a certain size or color if the combination of that size/color in combination with the already selected size/color is out of stock, meaning that it could look like something is in stock

    - before you click on it, when in reality it is out of stock (you would still see the availability after clicking on the size/color)
- Homepage collections shown based on metafield instead of array with strings of collection names
    - Metafield already available in Shopify backend, query also available but has to be done with custom admin API client instead of storefront client
- Account creation / login
    - Not available for Shopify development stores - upgrade to a premium version first
    - Also includes purchase history, account settings, and other account-related tasks
- Multilingual support
    - Currently only available in English
- Sales channel integrations (Channable)
    - Only requires Channable plugin configuration in Shopify backend
- Fully dynamic content based on metafields in Shopify backend
    - For example: homepage sections (which collections should be shown)
    - Currently only available for extra product information
- Relevant/related products
    - For example: “people also bought”, “products similar to this”, “you might also like” sections on product page / cart overview
- Display errors on cart overview if product stock is lower than cart quantity
    - Currently only blocks the user from adding another item if it is out of stock (for example: product has a stock of 5, user has 5 in their cart, 6th will not be added, but also doesn’t show an error message)
- Hosting (online demo) / CI/CD
    - Currently only available locally through a dev server
    - Code is available on GitHub (private Moonly repository) but is not configured to automatically run CI/CD integrations yet

Recommendations

There are a couple of recommendations based on the project's final status.

Research

In terms of research, no extra research needs to be done to verify certain parts of the design or development, however it could be beneficial to research hosting solutions to see if Oxygen is the best fit to host Hydrogen applications, or if a different solution is preferred. While Oxygen is recommended by Shopify, I have not done any research on this to validate if that recommendation is truly the best solution.

Design

The design is done and validated by reviews and testing. No changes have to be made to the current designs unless there are plans to expand, like adding more designs for account settings.

Designs should maintain a similar style and use components as designed in Figma as much as possible.

Development

As for the development of the prototype / store, there are plenty of missing features that should still be developed as mentioned above. When developing these features, the same code structure should be maintained to ensure a consistent codebase.

An overview of how to develop the missing features can be found in the next section.

Advice

To make the store a fully functional online store, the following steps should be taken:

1. Developing missing features

Here are some ways that the missing features can be developed:

- Filtering through products (collection page)
    - Can be developed by querying products and sending filters as parameters, filter values and UI are already available
    - A new Hydrogen template is being developed by Shopify, it could be valuable to explore their filter system as seen here: GitHub repo
- Overview of collection products bug
    - The query should instead be based on the first available variant, or refactored to include all variants of the product (and then show them on hover for example)
- Product variant selectors
    - I would either leave it as it currently is, or develop a custom variant selection component that allows pre-emptive querying of available variant combinations
- Product reviews (product page)
    - I would use Okenda for this as they seem to be one of the leading review plugins that support the Hydrogen framework
- Homepage collections shown based on metafield instead of array with strings of collection names
    - Metafield already available in Shopify backend and query is developed, the only remaining step is to perform the query on a custom admin API client (see: GitHub repo)
- Account creation / login
    - Can be added by upgrading to a non-development store
- Multilingual support
    - Can be added by finding a translation plugin, I would put this on a low priority as English is already available and this requires a good amount of work
- Sales channel integrations (Channable)
    - Can be added by configuring the Channable Shopify plugin
- Fully dynamic content based on metafields in Shopify backend
    - Can be added by creating relevant metafields in Shopify and then querying them where needed
- Relevant/related products
    - Can be developed by creating a custom query that searches for products in a similar category or with similar specifications (or both) to the active product by sending them as parameters
- Display errors on cart overview if product stock is lower than cart quantity
    - Shopify likely already returns this somewhere when this happens, so once that is found you can render the response in the front-end. If not, run a custom function when adding/removing items from the cart

2. Gathering data / setting up Shopify backend

The current store holds roughly ~50 products and ~10 collections. If all products are to be instantly available, import the product data that is required or manually create them. A checklist of data that should be filled in for a minimal user experience is:

- Navigation (menu's)
- Collections

- Products
    - Product variants
    - Product metafields (extra information)

Additional data could be:

- Other metafields such as homepage collections
- Discount codes

3. Hosting / CI/CD

Explore the Oxygen platform first (Shopify's dedicated Hydrogen hosting system). If that is not optimal, look for a custom solution like hosting on a custom server.

Conclusion

While there are a good amount of missing features, most of those features are extra's and not must-haves. The current prototype is functional at its core and allows users to perform basic actions that you would require from an online store.

Due to most of the components already being built and a lot of data queries being in place, it should be possible to develop most of the missing features without too many issues.