3D product configurator

A guide to mass customization

Ruud Peters

Creative Media and Game Technology

Saxion University of Applied Science

October 27, 2020

Abstract

A configurator on the web is key to mass customization. Unfortunately, it is difficult to develop one due to its inherent complexity. This is especially true when the product requires realistic visualization. Until recently, only dedicated software companies could make these kinds of applications. Even then, the concept used to be far too expensive to be viable for small to medium sized companies. But, with recent developments in open source software, it has become increasingly viable to develop a custom configurator in an affordable manner. This report aims to provide an efficient and effective manner to build a custom configurator on the web using 3D technologies for product visualization. It achieves this through answering four supporting questions that represent the fundamental challenges that were encountered during the development of an online configurator. The answers are found through a literature review and case study. The literature review serves as a theoretical foundation, while the case study describes the development process of the paper's client in detail.

Using the literature review and experience from the development process, it has been determined that the company's and product's fit for mass customization are the most important factors to a configurator's success. Products that do not require mass customization to be viable, but do benefit from personalization, are a particularly good fit. Companies that have a low sales volume and high amount of revenue per sale are also suitable; it allows some of the configurators work to be done manually by employees, of which the cost does not substantially raise the product's price.

In the case study, Lean Startup Methodology (LSM) has successfully been applied to increase customer value and decrease wasted resources. LSM achieves this through the build-measure-learn loop; an iterative approach to product development. It uses a Minimal Viable Product (MVP) to test a product concept through early customer validation, ensuring that a concept is viable before investing in its further development.

The requirements for a configurator are determined by distinguishing between functional requirements (i.e. what actions the customers should be able to perform), and non-functional requirements (i.e. which requirements ensure the quality of a configurator on the web). The functional requirements must be based on the needs of the project owner and target audience, while the non-functional requirements should be based on the current best practises of software development on the web.

A configurator's technology stack should be based on the functional and non-functional requirements. The report provides an example of a technology stack that can serve multiple purposes in terms of product configuration. The technology stack generally consists of Google Cloud Platform (GCP) for hosting, React and Material design for the user interface (UI), Node and Webpack for building the web application, Gitlab and Docker for continuous deployment, FreeCAD and Blender for CAD model conversion, and finally ThreeJS for rendering 3D models in the web browser.

Preface

This report is a graduation thesis that serves to learn more about 3D product configuration on the web. It has been written for a Dutch company called Kachelbouwer, which directly translates to “stove builder”. Run by Ewald and Bernadette Ten Hagen, they seek to build highly efficient, wood burning stoves that can heat homes in a carbon neutral manner. Two years ago, Ewald came to me with the idea of customers building their own stove from modular parts. He imagined a digital playground, in which people could experiment with different designs that could be easily translated to reality. What started as an experiment, has now turned into a business concept. This is where our goal to build a product configurator began.

First and foremost, I would like to thank Ewald and Bernadette Ten Hagen for their support over the years. It has been a great experience to work with them. I would also like to thank Henri Bruel for being a great graduation coach. His feedback, together with that of Lukas Malec, was very helpful for writing this report.

Glossary

2D, 3D	Two dimensional, three dimensional.
Configurator	A software application for designing products exactly matching customers' individual needs (cyLEDGE Media, 2008).
Combustion chamber	The part of a wood burning stove in which wood is burned.
C#	Programming language used by Unity3D.
CAD	Computer Aided Design.
Exhaust ducts	The part of a wood burning stove that connects the combustion chamber to the chimney. It acts as a heat sink to improve efficiency and radiate heat over long periods of time.
E-commerce	Electronic commerce.
Facade	The part that encapsulates the exterior of a stove for the sake of appearance.
GAE	Google App Engine.
GCP	Google Cloud Platform.
glTF	Graphics Library Transmission Format.
HTML	Hypertext Markup Language.
IAP	Identity Aware Proxy.
JavaScript	Programming language on the web.
LSM	Lean Startup Methodology
Mantle	The part of a stove that encapsulates the exhaust ducts.
MMP	Minimal Marketable Product.
MVP	Minimal Viable Product.
Pivot	A change of strategy without a change of vision (Ries, 2012).
SEO	Search Engine Optimization.
Stove	Wood burning stove, used for generating heat.
USP	Unique Selling Point.
UV	2D coordinates of a texture on a 3D mesh.

UX	User Experience.
WebGL	Web Graphics Library.
Web apps	Web Applications.

Table of contents

1. Introduction

As computer graphics technologies on the web (WebGL) matures, new opportunities arise for innovative solutions. Using WebGL, it is possible to create applications on websites (web-apps) that visualize realistic looking three dimensional (3D) environments. An interesting commercial use case of this technology is a webshop in which customers can customize a product. Such an app is called a configurator, which is an upcoming trend in e-commerce. Utilizing WebGL for a configurator on the web opens up new possibilities in terms of interactivity, customization and visualization. Compared to traditional two dimensional (2D) configurators, WebGL provides more opportunities for businesses. It not only enhances user experience; it enables the creation of new services that were previously deemed infeasible.

Client Kachelbouwer, the company for which this report has been written, wants to develop such a service. The client wants customers to be able to design their own wood burning stove from modular components on a website in an easy manner; a feat that requires WebGL. Developing a configurator with WebGL is a significant challenge due to the inherent complexity of the technology stack. It requires a substantial investment, making the development a risky venture. This report addresses this problem by explaining the developmental process of the use case of the client.

1.1 Goal

This report aims to serve as a guide for creating a product configurator that allows for mass customization in 3D. By substantiating the design choices of a real use case, the reader is taken through a configurator's design and development process. The given methods act as a foundation for developing a web-based solution from the ground up in an efficient and effective manner. The methods are, at the time of writing, a compound of the current best practices. Moreover, they are suitable for many purposes, including:

- Lean design and development of software.
- Creating web apps using an extensible and scalable technology stack.
- The application of WebGL for commercial purposes.

1.2 Objective of the client

Client Kachelbouwer is a small company in the east of the Netherlands that focuses on custom built wood-burning stoves. Due to each customer's situation and wishes being different, every house requires a different heating solution to achieve optimal efficiency. Deciding the best solution requires expertise, and is therefore a costly process.

To provide a more scalable solution, the client wants a web configurator in which customers can easily design their own stove from modules. The client desires a web-app to guide customers through the design and order process to ensure a user friendly experience and optimal end result.

As the design of a modular stove gets quite complex, a configurator utilizing 2D images wouldn't suffice. Hence, the client wishes for a more sophisticated solution for customers to easily visualize and customize the product. Consequently, the client came up with the idea of utilizing a 3D configurator on the web. However, the client has no experience with developing software. Due to the inherent risk of developing a new product, the client wishes to spend minimal funds until the concept is deemed valid.

1.3 Problem statement

The first and foremost problem is that the client lacks customer validation of the product concept. This applies to both the product and its configurator. Therefore, every development is a risky investment. If a feature turns out to be undesirable, it is not only costly; it also delays the product launch. The problem is compounded by the fact that the client has a huge wishlist that describes the ideal solution. There is no plan on how to gradually grow to the ideal solution, making it difficult to establish criteria and prioritize requirements to test the validity of the concept.

Another challenge is to choose the most suitable technology stack for the solution. As the stack is not easily changed mid development, it is important that it is future proof in terms of features and scalability. Its implementation will greatly influence the user experience (UX) and development costs. Therefore, it is critical to developing a successful solution in the long run.

1.4 Main and supporting questions

Based on the problems stated above, the main question of this report is;

- *How to efficiently and effectively develop a product configurator on the web that allows for mass customization in 3D?*

Which can be divided in the following supporting questions;

1. *What are the preconditions for a company to successfully utilize a product configurator for mass customization?*
2. *How to approach the development of a product configurator so that it is efficient and effective?*
3. *How to determine the requirements of the project owner and target audience for the product configurator?*
4. *Which technologies and platforms should a product configurator on the web utilize to meet the requirements and support mass customization in 3D?*

2. Literature review

2.1 Mass customization through configuration

A product configurator is a tool for customising a product to meet the needs of a particular customer (cyLEDGE Media, 2008). Usually, this tool takes the shape of a software solution

in which the customer can co-design a product (Franke & Piller, 2003). Configurators come in many varieties; ranging from simple ones that can customize the colors of a product to complex ones in which a complete product can be built from modular components (Combeenation, 2019). Note that a configurator's complexity greatly influences its benefits and downsides. One considerable benefit is its key role to mass customization. An advanced configurator allows a product to be visualized, customized, assessed, represented and priced in real-time (Wankel, 2007). In the case of business-to-consumer setting, this can provide the following advantages (Cyledge, n.d.);

1. Differentiation through individuality.
2. Reduced capital commitment and less overproduction.
3. Better knowledge of customers' needs.
4. Higher customer loyalty.
5. Shopping as experience.

A precondition to customer satisfaction from co-design is that the process is felicitous and successful. The customer has to be capable of performing the task, making user friendly design a must. Other key factors to a successful configurator are its technological capabilities, its integration in the sales environment, its ability to allow for learning by doing, its ability to provide experience and process satisfaction, and its integration into the brand concept (Wankel, 2007).

Although a configurator is the way to mass customization, it is by no means an easy venture. It is tricky to implement in both the shopping experience and company's workflow (Product Designer, 2018). A company has to adapt its business model for a configurator's implementation to become successful (Heuvelmans, 2010). An important precondition is that the product is well suited for mass customization.

A configurator's price should also be considered. There are affordable, pre-made solutions available on the market. These are usually sold on a per-license basis through a recurring fee. The benefit of these pre-made configurators is that there is little to no upfront development cost. The quality of the configurator is also ensured, as the companies which sell them offer demonstrations of their capabilities. They may be a viable choice for simple use cases, such as products that merely require their appearance to be customizable. However, when the required customizability necessitates complex functionality, the one-size-fits-all approach of pre-made solutions does not suffice without custom modification. At that point, the question becomes whether it is wise to build upon proprietary software (causing vendor lock-in), or build a custom solution (in-house or through outsourcing) using open-source software.

Developing a custom configurator offers many advantages in the long term. Because the software is tailor made, it is better suited to the needs of a company and its target audience (Marsner, n.d.). Because the company truly owns the solution, the recurring license fees will be little to none (IdeaRoom, n.d.). However, developing custom software requires a significant initial investment, and comes with a risk of cost overrun (Flyvbjerg & Budzier, 2011). It also takes a long time to develop.

2.2 Lean methodology

When the validity and composition of a product concept are unknown, like in the case of the report's client, it is recommended to follow Lean methodology; an approach to maximize customer value while minimizing waste (Lean Enterprise Institute, n.d.). This is achieved by iteratively improving the value proposition of a product concept using the Build-Measure-Learn feedback loop (Belyh, 2019). The loop revolves around the Minimal Viable Product (MVP); a minimum feature set (Blank, 2010) which is used as an experiment to explore a hypothesis about what customers really want. The MVP allows a team to collect the maximum amount of validated learning about customers with the least effort (Ries, 2009). Due to this broad definition, an MVP can range considerably in complexity depending on the amount of previous iterations, available knowledge and resources. But fundamentally, an MVP is about finding a product-market fit with minimal waste through early customer validation. To accomplish this, an MVP has to convey its greater concept by delivering core features to visionary customers. These customers can be referred to as innovators (Rogers, 2003) or earlyvangelists (Blank, 2010), and constitute a tiny but important fraction of the target market. Their feedback determines if the development of a concept is worth continuing, and if so, what changes and additions have to be made. So, not only should an MVP test if the proposed solution is suitable; it should test if the presumed problem is serious enough to justify a solution in the first place. When testing an MVP, some crucial questions are:

- How much do customers desire the proposed product or service?
- How willing are customers to pay for the proposed solution?
- What unique selling points of the proposed product are the most and least interesting?
- What changes to the product's vision would improve the value proposition?

To answer crucial questions with minimal investment through real customer feedback is the essence of an MVP, and the key to maximizing customer value while minimizing waste. The challenge of developing an MVP is determining the minimum feature set. It requires carefully balancing the "minimal" with the "viable" part of each MVP iteration. Too little change results in inefficient feedback, while too much change may result in wasted resources. Designing an MVP involves analyzing the product's vision and extracting the features that best capture the value proposition. The features have to provide enough value for innovators to desire the (envisioned) solution, and should be implemented with the least effort possible. This way, a company can validate a concept without having to develop the whole product.

In the case of software, such as a configurator, partially Implementing features and utilizing placeholder content can accelerate the development process. As long as the MVP can be tested and conveys its future utility, innovators can provide feedback. In the case of a negative response, a pivot (also known as a course correction) can be made before having spent significant resources. When the response is positive, the provided feedback can be used to iteratively improve the MVP. After multiple iterations have been validated and the solution has been deemed adequate, the goal becomes to transform the MVP into a Minimal Marketable Product (MMP).

An important distinction has to be made between a MVP and MMP. These two terms are commonly used interchangeably (Ambler, 2017), as both of them are minimal feature sets. The main difference lies in their focus and target audience;

- The MVP focuses on validating the problem and establishing the (ideal) product-market fit by learning from innovators.
- The MMP focuses on reducing the time-to-market by meeting only the essential criteria for market launch and targeting early adopters (Pichler, 2013).

In simple terms; the MVP focuses on learning as fast as possible, while the MMP focuses on earning as soon as possible. The MMP can be seen as “the step after” an adequate solution has been found using an MVP. However, it often takes more than one MVP to achieve this. So, Transitioning to an MMP should be a careful consideration. It should not be the goal of every MVP. This is one of the biggest misconceptions of Lean startup methodology; it is not about being cheap and gaining revenue quickly. These two effects are a result of finding the right product concept by focussing on maximizing value and minimizing waste through customer validation.

3. Methodology

This report answers the main question by answering the supporting ones in their respective order. The supporting questions formulate the fundamental challenges that were encountered during the development of a configurator, from its inception to its implementation. The supporting questions are answered through the combination of a literature review and case study. The literature review serves as a foundation for the case study, by defining a theoretical basis for making grounded design decisions. The literature review has been conducted through qualitative research. The case study puts the reviewed theory in practise using the case of the report’s client. A prototype is developed and tested during the case study. The development is split in two parts; a conceptualization and implementation. In the conceptualization, the prototype’s design process is described. The design is determined using Lean methodology, and based on the functional requirements of both the report’s client and target audience. The implementation section describes the technology choices, based on the previously determined non-functional requirements. Finally, testing is achieved through real customer validation, ensuring the result’s validity.

The first supporting question, “*What are the preconditions for a company to successfully utilize a product configurator for mass customization?”,* is answered using the literature review. By explaining the strengths and weaknesses of configurators, its preconditions for success are determined. The given information was acquired through secondary research. The sources consist of websites that explain the definitions, a research paper on mass customization, and a book on the benefits of co-design. The sources are reliable, as their conclusions correspond about the benefits of mass customization and co-design. However, as stated by (Franke & Piller, 2003), there is hardly any empirical evidence on the benefits of configurators. Therefore, the conclusions deducted from the sources may not be completely valid.

The second supporting question, "*How to approach the development of a product configurator so that it is efficient and effective?"*, is answered using both the literature review and case study. First, the literature review explains the fundamentals of Lean software development. Then, the concluded best practises are used to conceptualize the prototype of a configurator in the case study. The utilized sources for the literature study are found through secondary research. The sources mainly consist of the work from Ries (20xx) and Blank (20xx) due to their prominent connection with Lean product development. As Lean startup methodology is empirically proven to increase chance of success (Schwery, 2018), and the sources originate from its founders, it can be said that the sources are both reliable and valid.

Supporting questions three, "*How to determine the requirements of the project owner and target audience for the product configurator?*", is answered using the case study. By splitting the requirements in functional and non-functional ones, it becomes possible to distinguish between the needs of the client and the desired specifications for online configurators in general. The functional requirements are deduced from primary research on a previous development iteration, which was tested by real customers. The concluded functional requirements are therefore valid, but slightly unreliable due to the limited amount of individuals that tested the application. The non-functional requirements are deduced from secondary research on the general criteria of web applications. Its sources consist of multiple websites that explain the criteria and their importance. Due to these criteria being widely researched and coherent, the non-functional requirements can be deemed valid and reliable.

The fourth and last supporting question, "*Which technologies and platforms should a product configurator on the web utilize to meet the requirements and support mass customization in 3D?*", is answered in chapter "Implementation". It provides a suitable technology stack for developing the prototype of an online configurator, conforming to the requirements that have been established through supporting question three. The stack has been used to create a prototype for the report's client. The prototype is tested using customer validation, of which the details can be found in chapter "Results". The supporting question is answered using a combination of primary and secondary research on suitable technologies for a 3D configurator. The utilized sources consist of experience gained from a previous development iteration, documentation from different software, libraries and even fora. Therefore, the sources are valid and reliable.

4. Conceptualization

4.1 Minimal Viable Product

In the case of the client, the goal of the MVP is to first gauge the desire for a stove configurator and then discover its desirable features. To achieve this, the features that best represent the core value proposition have to be implemented in the MVP. In other words; the features have to convey how the configurator would look, so that visionary customers can validate the concept and provide feedback. As stated in the *"Objectives of the client",* the ideal solution for the client would be a configurator that enables customers to easily design a

stove themselves. The designing process should allow for changes in shape and materials, so that the resulting stove fits the needs of the customer. A configurable stove consists of four parts; a combustion chamber, a long exhaust duct, a mantle and an outer facade.

1. The type of combustion chamber determines the amount of heat that can be created. It should be well suited to the customer's living situation.

2. The duct is made from dense stone that acts as a heatsink for the hot exhaust gasses. The heat is stored in the duct's building material, like a natural battery. It then slowly radiates the heat to the outside, providing a heat source over a long period of time. The duct consists of modular pieces that can make straight sections and corners. The duct's modularity allows it to be built into any kind of shape. It is suitable for different setups such as inner and outer corners of a room. The combustion chamber and chimney length determine the ideal length.

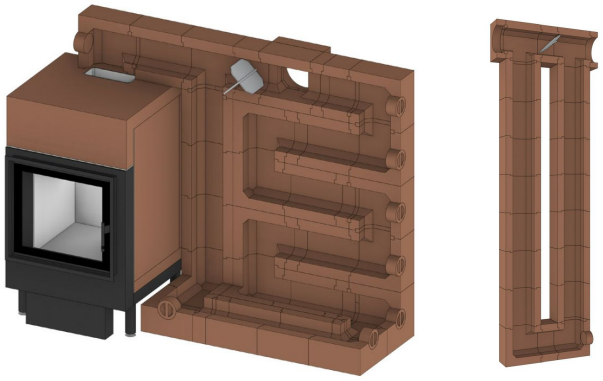

Figure 1. Stove with a cross section of exhaust ducts.

3. The mantle is a "concrete box" around the exhaust ducts. At the start of the MVP's development, there was no design for it yet. However, it was known that the mantle was going to be modular to ensure that the ducts always fit inside it. The mantle is going to be made from square cement blocks.

4. The facade is basically a layer of square panels that cover the mantle. The panels can be made from different materials, and are purely decorative. Customers should be able to choose from many materials in the configurator.

Ideally, the customer should be able to customize all four parts of the stove. However, there are limitations to the flexibility of the stove's design. The configuration has to follow certain rules (e.g ideal length of the duct) to be viable. Moreover, the configured stove should fit the house of the client. To approximate the ideal configuration, certain calculations have to be made using information on the customer's house. Examples of these are the level of insulation, room size and chimney height. Not only does this make configuration process difficult for customers, it also makes the development of a minimal viable configurator quite challenging. Hence, the first version of the MVP should answer some essential questions

before a marketable solution can be made. Using the questions from the Lean methodology section as an example, the following questions can be formed;

- How much do customers desire to configure a stove?
- What features are essential for customers to design a stove?
- How would the ideal configuration process of a stove look like?

The last two questions are essential for developing features that customers find useful. The answers tell what the functional requirements for the configurator should be. Note that questions about pricing are left out. This is because the client is already on the stove market, and the price of the new configurable stove will be lower than the ones currently sold (due to the easy assembly of modular components). However, this relies on the premise that the perceived value of the modular stove is the same or better. This should be tested in the next version of the MVP, when designing the facade.

The MVP's conceptualization, implementation, and results are further explained in the appendices. This is because the MVP's development can be considered an independent project. It was developed by Stanislav Mutafchiev during his graduation, and co-designed by me, the author. The MVP's conclusions were essential to the development of the MMP, which this paper will explain in the following sections.

4.2 Minimal Marketable Product

In short, The MVP concluded two things; the vision of the product configurator is legitimate, but the approach was wrong. The client needs a change of strategy without a change of vision (Ries, 2012), better known as a pivot. The changes should be based on the lessons learned from the MVP. In the case of the client, the MVP showed that the technology stack and minimum feature set were both unviable. Both of these have to be redetermined. It is important to start with the new minimum feature set, as "you've got to start with the customer experience and work back towards the technology" (Jobs, 1997).

The most valuable lesson learned was that customers desire a user friendly configurator that allows for the customization of a stove's shape and materials. The difficulty is that the design should be both appealing and functional. These two aspects influence each other, making the design process a complicated task for the user. During the development of the MVP, the client assumed that the functionality should be leading within the design process; first the combustion chamber and ducts, then the mantle and facade. This is how the client would manually design a stove.

The main challenge is that the amount of functionality required to configure a complete stove conflicts with the customer's wish for ease of use. More functionality enables more flexibility, but also increases complexity. It is difficult to make the design process both flexible and user friendly, while simultaneously guaranteeing a functional end result. It would require a lot of work and user testing to accomplish, which is undesirable at this stage due to the limited product validation of the MVP. Still, the client idealizes a configurator in which customers can design their own stove.

The whole problem can be summarized to the following question: “What simple solution allows customers to easily design the appearance of a stove, while simultaneously ensuring the stove’s functionality?”. In the case of the client, this is the critical question to a successful pivot. After careful consideration and many discussions, the client found two wrong assumptions;

- Functionality has to be guaranteed, but it does not have to be leading in the design process. It is possible to design the appearance of a stove first and then fit in the ducts. Not every exterior design will be functional, but it is often correctable with small adjustments.

- Although it would be ideal for customers to design the complete stoves themselves, it is infeasible to accomplish in one go. The client’s vision is appropriate, but the approach has to be divided in steps.

Dropping these two assumptions opens up a new possibility; a concierge test (Kromer, n.d.). This is a strategy whereby customers are provided with a manual service (bmilab, n.d.). It allows a feature to be tested without the need for it to be completely ready, optimized, or automated. A concierge test may also provide helpful insight into the future implementation. The strategy is particularly well suited for the case of the client; customers can design the appearance of the stove, and let the inner workings be handled for them by the client. The strategy would also significantly reduce the amount of development required. The price of the client’s product even allows for the manual labour cost to be included, until the service is automated.

The first development step is to allow customers to configure the facade. The stove’s shape can be manually designed by the client, based on the requirements of a specific customer. This is not ideal, as it would require a lot of back and forth communication. It would only be viable for visionary customers, who are willing to invest significant effort into the product. Fortunately, it will result in feedback that can be used for further developments. It also provides initial stove designs and realisations. Pictures of these realised designs can be showcased inside the configurator, next to their virtual model. This way, future customers can see what to expect from the design process. If a particular model is liked by another customer, the facade can easily be adjusted. One model can thus serve multiple customers. This step is already enough for a viable MMP. However, the focus should still be on testing and iterative improvement. For example; the configurator should provide a diverse amount of models and materials to test what customers really want, and adjust for that in the future. The unique selling points should be listed and explained, to test early marketing material.

The second step is to allow customers to configure the general shape of the stove. This would not always result in viable models, but it is great for customers to experiment with. The configurator should be flexible and user friendly. But, the client should be able to help a customer if needed. When the feedback from the first iteration step has been implemented, early adopters can be approached through initial marketing. It is a great moment to experiment with different design possibilities and marketing strategies. The focus of this step

should be on finding scalable practises that allow for growth. For example, finding and implementing the features that make the manual service effortless.

The third development step would be the ideal vision of the client; a configurator that enables customers to design a complete stove based on the requirements of their home. Although this stage aims to serve the general majority of customers, not every one of them will be capable of using the configurator. Therefore, the client should still offer the manual service from step one and two. The focus of this stage should be to offer a complete experience to the average customer in the target market. The customer should be able to find and understand the product, as well as configure their ideal version. Product marketing, education on biomass, and guidance in the configurator are important examples for the customer acquisition process.

The strategy that is explained in these three steps can be used to determine the minimum feature set. The features can then be divided in requirements, which can be used to determine the ideal technology stack.

4.3 Establishment of requirements

The technology stack should be chosen based on the requirements of the minimum feature set. There are two types of requirements. First, there are functional requirements; these tell what a system should be able to do. Second, there are non-functional requirements; these describe in what manner the system should do it. In this section, an overview of both types of requirements is given. The functional requirements are based on the first development step of the Minimal Marketable Product, that has been determined in the previous section. The non-functional requirements are based on the current best practises for web applications in general, which have been acquired from experience, previous colleagues and a multitude of online sources like fora. The requirements are minimal and viable, like the minimum feature set they are inferred from.

4.3.1 Functional requirements

Overview of unique selling points

As requested by one of the users of the MVP, an overview of the USPs would be helpful. In the case of the client, the value proposition is not immediately obvious. Providing a simple and clean overview with short explanations is critical to acquire new customers who are unfamiliar with the concept. In a later development step, these USPs can be further explained on dedicated web pages. The USP overview has to be on the top of the landing page so it can immediately be seen by new customers. Preferably, the USPs have illustrations to make the content more iconic and appealing.

Overview of product models

Because customers had troubles with imagining the possibilities of the configurator, an overview of product models has to be provided. The overview serves an interactive showcase, and preferably displays both the realised and virtual version of the product. The overview has to allow models to be added by the client. Each model needs to have a name

and a price attached to it. When a customer clicks on one of the models, the customer has to be directed to its details.

Overview of product details
Each model gets its own details page that visualizes the 3D model, and preferably a picture of the realisation. The overview needs a table that contains its technical information, such as its weight, heat output and price. The visualized 3D model on the overview has to be interactive, allowing customers to rotate and zoom. Easy switching between facade presets would be desirable, to show the possibilities of the configurator.

Product customization
If a customer is interested, he or she has to be able to configure the facade manually. Changing the facade should also update the current price. A multitude of materials has to be available, and has to be based on customer interviews during the MMP phase. Initially, traditional materials such as different kinds of stone can be used.

Order creation
After a customer configures the facade of a stove, he or she has to be able to create an order. Initially, this can be a simple email service that brings the customer and client in contact. Later, the order should list all the parts and their associated prices.

Contact information
It is essential for a company's credibility to have a contact page (Barret, 2018). It should at least contain a physical address, email address and phone number. Preferably, the page contains a map of where the company is located. When users are on their phone, it would be ideal if clicking on the phone number or email address directs them to the dial pad or mail client, respectively.

4.3.2 Non-functional requirements

Usability
Customers may use the website on different devices. Thus, a responsive user interface is required (LePage & Andrew, 2019). Preferably, the UI is optimized for phone, tablet and PC separately. For example, the website menu has to be optimized for touch on phone and tablet, and mouse on PC.

The web pages should load fast on all devices (Cloudflare, n.d.). Clicking on a link and opening a new page should not require a reload of the 3D model data, for example. Long reload times bring the experience to a halt, and cause users to bounce to other websites. The included code on the web pages should be optimized for size. Leaving unnecessary code on a web page leads to a slower initial load time, which should be avoided as much as possible.

The website should also support different browsers and devices. A great example are iPhones; websites behave quite differently on Safari compared to Chrome, resulting in bad user interfaces and broken buttons if not taken into account (SiteGround, n.d.). As a large section of the browser market consists of safari (16,82% as of October 2020) (Statcounter,

n.d.), it is absolutely necessary to support it. Ignoring this can result in broken pages that significantly increase bounce rate (Massey, 2015).

marketability
Search engine optimization (SEO) is essential to marketing success of a website. SEO basically determines how high a website is ranked on web searches, such as on Google (WebFX, n.d.). Web pages that have a high SEO score will attract more potential customers, increasing the product's exposure. It is therefore essential for increasing sales. SEO is also free, which saves advertising costs in the long run. However, It takes a significant amount of time before a search engine updates the SEO score of web pages (Google, n.d.). Website age also contributes to a higher SEO score, which is why the SEO should be prioritized early in the development of a website.

Website analytics are key to the build-measure-learn feedback loop, especially at the MMP stage. Using analytics, the client can learn about customer behaviour and demographics (Thakur, 2017). Analytics are essential to reduce bounce rate and increase customer retention. Analytics can also be extended by AB testing, whereby different versions of a web page are served and tested. Tools such as Google Analytics are free, and worth integrating in a website.

Security
Online security is just as important as any of the other requirements. Cyberattacks are costly to clean-up, damage reputation and discourage visitors from coming back (Tammany, 2018) . Preventing them prevents unnecessary costs and even improves the SEO of a website. When choosing a platform or technology, it is important to take their security measures into consideration. A useful example is HTTPS; a protocol that secures the communication between a website and a user. Not every hosting service sets up HTTPS automatically and for free, which should be taken into account.

An important security aspect is data and identity protection. Although the client does not need user accounts immediately, it is beneficial to consider this aspect for the future. Protecting customer and product data is a necessity for every company. Customer data leaking would be disastrous for a company's reputation. Sensitive 3D models should not fall in wrong hands either, as it could spark product imitation.

Stability
Last but not least, a website needs to be stable. It cannot go offline accidently, as it would result in lost customers. A broken feature that goes unnoticed over a long period of time is even worse, as it can cause customers to never come back to the "broken" website.

The website's deployment process should be reproducible; easy to update without the chance of something breaking unexpectedly. The chance of a mistake being made during a complicated, manual deployment process increases when a web app becomes more complex or when the development team grows in size. Fixing these kinds of mistakes is undesirable, which is why the process should ideally be automated. It is also advisable to

have a test domain set up, to check the web app's integrity before uploading it to the main domain.

A website's scalability is also important. When the amount of visitors grows, the hosting service has to scale to prevent long loading times. Preferably, this can happen dynamically depending on server load, as running unnecessary servers is costly. Using a good hosting service can also prevent DDOS cyber attacks, which try to overwhelm a website's servers.

5. Implementation

An online configurator, like a physical system, consists of several components. These components form the technology stack, and serve as the foundation on which a configurator is built. This chapter explains the composition of the technology stack, and substantiates its design choices.

To prevent reinventing the wheel in software development, it is advisable to use pre-existing software applications and libraries. There is a large offer of free, open-source software available on the internet. Without getting too far into the details, open-source software can prevent a lot of work and license costs, under the condition that changes to the original source code are given back to the open-source community. The technology stack provided by this report is mostly open-source, except for the back-end. This is because the back-end is set up in the cloud, meaning that a company does need server hardware.

The technology stack has been chosen on the basis that it at least supports the functional requirements. In other words, it supports all the desired functionalities. The non-functional requirements were also taken into consideration, ensuring its effectiveness in terms of usability, marketability, security and stability. The chosen technology stack is therefore not only viable, but also suitable for many web applications. Its flexibility lies in its modularity; all of its components can be changed, and new modules can easily be added. It is therefore a strong base for any kind of (3D) configurator.

For the purpose of simplicity, the provided technology stack is split into five parts; a front-end, a back-end, building tools, devops tools and auxiliary tools.

a list of back-end and front-end technologies. The front-end meaning any technology which the user directly comes in contact with (i.e. the presentation layer), and the back-end meaning all the technologies that handle the data in the background. In practise, some of these technologies are somewhere in between these two terms. The list is therefore disputable, and only serves as a guideline.

5.1 Front-end

5.1.1 3D rendering

ThreeJS is a JavaScript library that serves as a WebGL wrapper or API. It is used to easily make 3D applications in the web browser. It is therefore very suitable for the visualization of

products in a configurator. Its glTF loader allows for loading lightweight 3D models. It also comes with a smooth orbit camera, making it possible to view a product from any angle.

visualizing a complete product in ThreeJS is therefore easily achieved. The library itself is also lightweight. Its size is currently around 700 kilobytes, which can be further reduced using WebPack.

5.1.2 User interface

ReactJS is a JavaScript library for building user interfaces. By rewriting the content of a single webpage instead of loading a completely new page, ReactJS allows for the creation of a single-page application (SPA). This makes it a lot easier to create websites with dynamic content, such as configurators. An SPA is not only fast, but also responsive, making the user experience close to a native app. Using an SPA framework like ReactJS ensures that a configurator is fast and mobile friendly. It is definitely possible to switch ReactJS for another SPA framework like VueJS or AngularJS.

Combining ReactJS with Google's Material Design system, otherwise known as Material UI, ensures a professional looking UI and UX. As Material UI offers many prefabricated components, creating responsive web applications ends up taking little effort.

5.2 Back-end

5.2.1 Hosting

Google App Engine (GAE) is part of Google Cloud Platform (GCP), and serves as a platform to host web applications using the data centers of Google. It is completely serverless, meaning that the cloud takes care of running the server. GAE manages the allocation of resources based on demand, thereby scaling up and down automatically. This allows the hosting costs to be based on the application's demand, meaning that users of the platform only pay for what is needed.

GAE supports both single-page applications,multi-page applications, and even combinations thereof. It can also host multiple versions at the same time under different (sub)domains using Google Domains, allowing for a development version to be tested before publishing. GAE has many more useful features, and is very well integrated with other cloud functionalities. A great example is the Identity Aware Proxy (IAP) of GCP. This service can put certain pages or complete (sub)domains behind a password, by requiring users to login to their google account. Thus, only a limited number of people can be given access.

5.2.2 Database

The reason why GCP was chosen over other cloud platforms, such as Amazon Web Services or Microsoft Azure, was because of a specific service of GCP called Firebase. Firebase is basically GCP "lite". It offers many of the same functionalities as GCP, but is more user friendly regarding web application development. It offers a JavaScript package (see section 5.3.2) that serves as an API for many GCP functionalities. This makes creating an MVP a lot simpler, as only one service is needed to implement a lot of different features.

Firebase Cloud Storage can, for example, save files such as 3D model (glTF) files, while Firebase Firestore serves as a traditional NoSQL database. It offers many other functionalities such as analytics, testing, messaging and authentication. The latter is especially helpful, as creating an authentication system is very difficult due to its security requirements. Like GCP, Firebase only bills for what is used. A major downside is that it is relatively expensive compared to GCP. But, for the creation of an MVP, it is an ideal solution. It can later be replaced by a cheaper alternative when it is financially viable.

5.3 Building tools

5.3.1 Runtime and package management

Although JavaScript does work in the web browser by itself, build tools are required to effectively develop web applications such as configurators. NodeJS, together with the Node Package Manager (NPM), are a suitable option. NPM is used to install packages of build tools and libraries. It can even install libraries for the front-end, such as ReactJS, ThreeJS, Material-UI, Google Firebase and many more. NodeJS, together with NPM, make the developing environment very simple to set up and use.

5.3.2 Bundling and minification

Webpack is a core building tool for creating optimized web applications in an effective manner. It serves multiple purposes, and even allows plugins to be installed. Webpack and its additional plugins can be installed using NPM. First and foremost, Webpack is a module bundler; it can make optimized bundles of JavaScript, HTML and CSS. Bundling decreases the total size of the web application, and increases its load speed. It achieves this by removing unused code (in a process called tree shaking), and merges multiple packages. Webpack can also convert newer versions of JavaScript (that are not yet supported by all browsers) to older, more widely supported versions. It achieves this using another build tool called Babel, through a process called transpilation. Thus, Webpack and Babel ensure that web applications work on older devices and browser versions. Finally, Webpack can also host a development with live reloading. When a change is made during development, it can immediately be seen and tested locally in the web browser. This speeds up the development of web applications significantly.

5.4 Devops tools

5.4.1 Version control

Git is a version-control system that tracks the changes in source code development . Git is helpful when multiple developers are working on the same code, as it assists with merging changes. It achieves this by having multiple branches that act as different stages of code integration. For example, the main branch should always have a stable version of application, while a develop branch can have a more recent but experimental version. These different branches can automatically be deployed to the cloud using a Continuous Deployment pipeline (CD, see 5.4.3). Thus, different versions of a configurator can continuously be tested in an online environment. A Git repository, containing all the different

versions and branches, can be hosted for free on GitLab. GitLab also offers free project management tooling and a continuous integration and deployment pipeline.

5.4.2 Project management

Gitlab offers an issue tracker and project board for free. The issue tracker can be used to describe desired features and bugs through referencing specific Git commits and branches. The project board provides an overview of all the issues and merge requests. The issues can be given labels, so that the overview categorizes them automatically in columns. The merge requests act as review points. It is therefore possible to create a semi-automated Kanban board. Lastly, it is possible to mention other developers, which can trigger an email notification. All of these tools are essential to project management and quality assurance.

5.4.3 Continuous deployment

Through Gitlab Pipeline, it is possible to automate the deployment process. It achieves this by automatically building the application in the cloud when a new version of the software is saved via Git (i.e. when a commit is pushed). After the build is finished, it is automatically published to the back-end (GAE in the case of the client). GitLab Pipeline achieves this through an application called docker; virtualization software that can create “mini” operating systems, which are known as containers. These containers can run alongside the main operating system, and copied over to other computers or servers, together with all the installed software inside them. Thus, Docker ensures that a build process will work on any computer by making the environment completely independent and fixed. Although setting up Docker requires knowledge and effort, it is a worthwhile investment. When implemented, it saves a lot of work by automating a tedious process and ensuring its reliability. It is especially effective in bigger teams, in which it is undesirable to give everyone access to a server.

5.5 Auxiliary tools

5.5.1 CAD model conversion

An important aspect of a product configurator are the 3D models of the product. Usually, products are designed using Computer Aided Design (CAD) programs. The resulting CAD models of the product contain 3D information, but are incompatible with hardware accelerated rendering APIs such as WebGL. This is because CAD geometry is made of non-uniform rational b-splines (NURBS), while WebGL requires mesh geometry. To use the products made in CAD for configurators on the web, their geometry has to be converted. The conversion process is critical to the quality of the 3D visualization. Unfortunately, it is difficult to carry out the conversion process effectively, as the model's geometrical complexity has to be carefully balanced between visual quality and performance. After the geometrical conversion, materials have to be assigned to the model as well through a process called UV mapping. Finally, the model should be exported to the glTF file format. Previous research (Koster & Peters, 2020) showed that there is only one program that can effectively convert from CAD to glTF directly, namely a software suite called PiXYZ. Unfortunately, it can be considered quite expensive. There is an open source solution available; a combination of FreeCAD and Blender. although it is not as effective as PiXYZ, it can certainly get the job

done. FreeCAD can be used to convert the CAD geometry to a mesh, while Blender can be used to assign materials and export to glTF. Although the process is quite labour intensive, it might be a viable alternative.

5.5.2 Browser support

When developing a web application that aims to support a wide array of devices, it is quite common to encounter browsers that do not support certain JavaScript functionalities. This is due to the fact that the specifications of the web are constantly updated, with some browser developers choosing to forgo the support of every feature. A classic example is Apple devices not supporting WebGL 2.0 (*Hackernews*, 2019). These limitations have to be carefully considered before choosing to implement features, or it might lead to unexpected bugs on some devices. It is therefore recommended to regularly test across browsers and devices. Companies such as Browserstack allow this to be seamlessly done online. Sometimes, it is possible to implement these features in browsers using a Polyfill; a technique to retrofit JavaScript functionalities in browsers that do not natively support them.

6. Results

Although the development process of the client's configurator is still ongoing, it was already possible to test a prototype of the MMP with two real customers. This was achieved through an interview, in which the client discussed the customized product using a 3D visualization of its model in the configurator. Thus, the results reflect early customer validation on the configurator that has been developed during the case study. The results are split in three sections that validate the prototype's design, technology stack and development approach. These sections provide the answers to the supporting questions of this report.

6.1 Design validation

The customer validation on the configurator's design confirmed the assumptions that were found in the literature review (section 2.1). The customers said that a configurator is a positive contribution to the complete product concept due to its ability to personalize the stove. Although the customers were earlyvangelists, it can be said that personalization creates customer loyalty and differentiation due to individuality, as the customers are more than willing to invest time into co-designing a unique product. The customers also stated that they are enjoying the co-designing process, which shows that configurators can have a positive influence on the shopping experience. Although it remains to be seen if mass customization leads to reduced capital commitment and less overproduction, the price of the modular product will be lower than the standard models in case of the client. This is due to the modular product being easier and faster to build. Everything considered, it can be said that a configurator fits well with the product concept of the client. This might be the case because of the following reasons;

- The product is viable without mass customization. Thus, a configurator can be a bonus.
- The product exterior is important to customers. Thus, a product configurator can add personalization value.

- The product price allows for manual labour to be included. Thus, a product configurator does not need to be completely automated (and can therefore be developed more easily).

The results provide answers to supporting questions one and three, as it contains information on the preconditions for a successful configurator and the functional requirements.

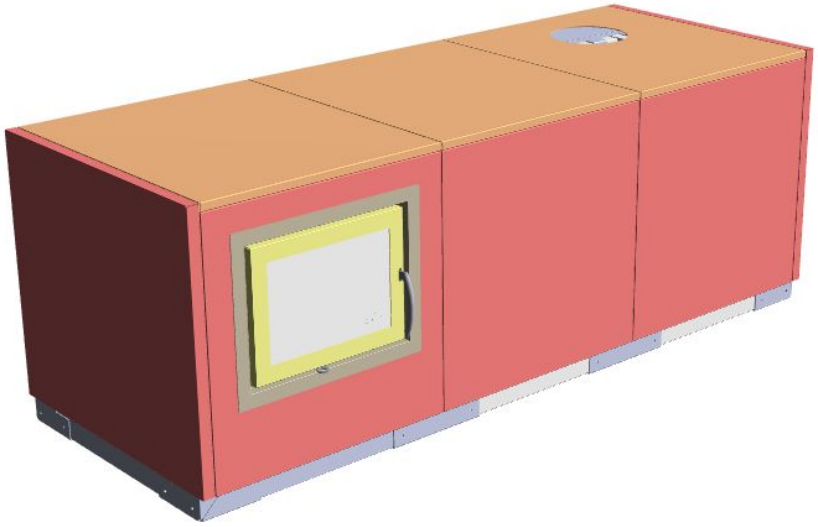

Figure 2. 3D visualization of the product's model that was shown to the customers.

Strengths ● Users enjoy co-designing a product, and are willing to invest effort. ● The overview of USPs was clear. ● The 3D visualization was helpful for understanding the expected end result. ● Users liked the idea of testing materials in the configurator.	**Weaknesses** ● Users noticed that the configurator is still not on the level of an MMP. ● Users found the 3D visualization clear, but not pretty. ● Users missed a call to action at the bottom of the landing page, directing to the overview of models. ● Website did not work on an user's Apple computer. ● Users missed content on the landing page, such as pictures. ● Users did not like the name of the product.
Opportunities ● Users would like to see more models. ● Users suggested multiple materials, such as slate stone and weathered steel (Cor-Ten). ● After the first model of a user has been built, pictures of its realisation can be added to the website.	**Threats** ● Users had a lot of questions on the process of co-designing. ● Although the users were generally positive, they still had a lot of criticism on the product itself. ● These specific earlyvangelists are designers, who may be more interested

	in co-design; it is still the question if other customers value the configurator. ● More early adopters are required before the concept can be deemed viable.

Table 1. SWOT analysis of the customer feedback on the prototype of the MMP.

6.2 Technology validation

The technology stack turned out to be suitable for the task of creating a configurator on the web. This is due to it being capable of supporting all the functional requirements. It also works well considering the non-functional requirements. The Node Package Manager allows technologies to easily be added and removed from the project, resulting in the stack being highly flexible. Webpack ensured that the front-end was as lightweight as possible through its modularization, caching and tree shaking abilities. React, combined with Google's Material Design enables the creation of amazing single page web apps that are not only very performant, but also good looking. ThreeJS is a powerful library that allows developers to create a lightweight and performant 3D experience. React and ThreeJS work really well together, but requires a significant understanding of both frameworks. With FreeCAD and Blender, it is possible to convert CAD to lightweight GLTF files that can be loaded by ThreeJS. The continuous delivery pipeline of Gitlab, combined with Docker, allowed for the whole deployment to GCP to be automated. GCP, together with Firebase, are useful platforms for hosting web applications. They are affordable and offer many features in the cloud. The only downside of this technology stack is that it is quite complicated. Because there are many different technologies interacting with each other, it becomes easy to lose track of what is happening. The stack requires a developer that knows the intricacies of full stack development. It might be preferable to have a separate front-end and back-end developer for such configurations. The results provide answers to supporting questions three and four, as they confirm the non-functional criteria and technology needs of a product configurator.

Strengths ● Very flexible technology stack. ● Mobile friendly. ● Lightweight and fast. ● High degree of 3D support. ● Security handled by GCP. ● Automated deployment. ● Automatically scales. ● Very cheap to set up and host. ● Responsive design.	**Weaknesses** ● Very complex due to modularity of technology stack. ● Difficult to learn due to high complexity. ● No 3D editor, making configurator development more difficult.
Opportunities ● SEO can be micromanaged. ● Easy to add new stoves.	**Threats** ● Slow to develop new features due to complexity.

● Google Identities for user accounts. ● Google Analytics is very powerful and free.	● Technology stack is very dependent on Google, causing vendor-lock.

Table 2. SWOT analysis of MMP technology stack.

6.3 Approach validation

Developing an advanced product configurator is inherently difficult and risky. Applying Lean startup methodology is a viable strategy to mitigate these two negative aspects. During the first development cycle, the product concept and technology stack were unsuitable for reaching a product-market fit. However, using the build-measure-learn feedback loop, it was possible to effectively pivot to a concept that is far more promising. Using Lean resulted in a decrease in waste and an increase in customer value. The difficult part was applying Lean startup methodology correctly. Concepts such as the MVP are often misunderstood. Even with a fundamental understanding, it is not always obvious how to correctly apply the theory in practise. It does not guarantee success either, as there are many more factors in creating a product configurator efficiently and effectively. However, with Lean startup methodology, it is possible to approach the ideal product configurator iteratively through the build-measure-learn loop, which is far better than the all-or-nothing approach. Lean Startup Methodology is therefore a valid answer to sub question two.

7. Conclusion

7.1 supporting questions

What are the preconditions for a company to successfully utilize a product configurator for mass customization? The results from the design validation (section 6.1) showed that mass customization should first and foremost fit with the business model and product concept of a company. It is therefore not always a viable strategy. Choosing to build a configurator is quite an undertaking, requiring a significant investment before it pays off. This can be mitigated using Lean Startup Methodology, as can be read in the approach validation (section 6.3).

How to approach the development of a product configurator so that it is efficient and effective? By correctly applying the build-measure-learn loop using an MVP, as can be read in the approach validation (section 6.3). Lean startup methodology reduces waste and increases customer value, therefore improving efficiency and effectiveness. However, the methodology is certainly not easy to understand due to its misconceptions. It is even more difficult to apply, as it is not always clear how to translate the theory into practise. But, it is certainly true that early customer validation is a key to successfully finding a product-market fit.

How to determine the requirements of the project owner and target audience for the product configurator? The functional requirements depend on the problem that the product configurator is trying to solve, as can be read in the design validation (section 6.1). The

non-functional requirements depend on the current best practises in the field of web development, as can be read in the technology validation (section 6.2). There are many standards and recommendations on the web. Each configurator might need a slightly different approach. But for the client of this report, the following four non-functional requirements were deemed most important; usability, marketability, security and stability.

Which technologies and platforms should a product configurator on the web utilize to meet the requirements and support mass customization in 3D? The best way to choose technologies and platforms is to research different solutions and pick the one(s) that fit best based on the requirements of the specific product configurator. This report provides a technology stack that is suitable for many use cases on the web in the technology validation section (6.2), which can be used as a starting point. In short, it utilizes Google Cloud Platform for hosting, React and Material design for UI, Node and Webpack for building, Gitlab and Docker for continuous deployment, FreeCAD and Blender for CAD model conversion, and finally ThreeJS for rendering 3D models in the web browser.

7.2 main question

How to efficiently and effectively develop a product configurator on the web that allows for mass customization in 3D? The first step is to consider if mass customization fits with the company's business model and product, due to it being an essential precondition. Then, Lean methodology must be used to efficiently and effectively find a product-market fit through the build-measure-learn loop using an MVP. Early customer validation prevents wasting time and resources on a solution that nobody wants, and is therefore key to finding a successful product-market fit. The development iterations should be short enough to ensure that a pivot can be made if necessary. The functional requirements should be based on the requirements of the project owner and target audience, while the non-functional requirements should be based on the current best practises of web development. Together, the requirements can be used to determine the most suitable technology stack. This stack should consist of front-end technologies, a hosting, database, and storage platform, build and devops tooling, and finally a CAD conversion workflow. This report provides a multi-purpose stack that can be used as a starting point.

8. Discussion

Based on the results, it can be said that the report provides a practical method for building a configurator on the web. It shows that, through Lean methodology, it is possible to actively approach a successful configurator. However, the method does by no means guarantee success. As explained through the first supporting question, there are preconditions. Hopefully, this report can be used as a roadmap and result in better design decisions.

The explained application of Lean methodology may prove useful for any kind of creator who faces a difficult problem. Lean can be used to effectively develop a successful product of any kind, as long as the methodology is applied correctly. The methods explained in this report

can be used for many purposes. For example, the proposed technology stack is viable for more than just configurators; it can also be used for other kinds of web applications, due to its flexible technology stack.

What the research does not tell is what other technologies are interesting. Any piece of the provided technology stack can be swapped, resulting in many possible configurations. The technologies are constantly evolving, meaning that the provided stack in this report may be outdated in the future. The research also did not go into pre-made configurator solutions, both on and off the web. Although I can not believe that such solutions will offer the same flexibility as the provided technology stack, it might be possible that they are viable for other kinds of products.

9. Recommendations

It might be interesting to compare Google Cloud Platform to Amazon Web Services. They both offer hosting services, but each with their own unique features. The reason why GCP was chosen was because it offers Google Identity Platform; an authentication system that takes care of user logins on a website (Google, n.d.), either through their gmail account or email sign up. It could be that Amazon offers something similar, better or different that may be useful.

PlayCanvas is a WebGL game engine written in JavaScript. It might be useful for online product configurators. In the case of the client, ThreeJS was chosen because it is lightweight and flexible. However, PlayCanvas might be far more powerful in 3D development due to its editor. For WebGL games, it is probably better than Unity3D. So, it is definitely worth looking into.

Lastly, it might be educational to look into pre-made configurator solutions. These solutions are becoming more popular and widespread, due to their ease of use. Unfortunately, almost all of them are proprietary. To test them, it is necessary to contact the company.

Bibliography

Ambler, S. (2017, December 27). *Defining MVP, MMF, MMP, and MMR*. Disciplined Agile. https://www.projectmanagement.com/blog-post/61937/Defining-MVP--MMF--MMP--and-MMR

Barret, L. (2018, July 20). *Factors that influence your website's credibility*. Search Engine Watch.

https://www.searchenginewatch.com/2018/07/20/factors-that-influence-your-websites-credibility/

Belyh, A. (2019, September 20). *How The Build-Measure-Learn Cycle Really Works*. https://www.cleverism.com/how-build-measure-learn-cycle-really-works/

Blank, S. (2010, March 4). *Perfection By Subtraction – The Minimum Feature Set*. https://steveblank.com/2010/03/04/perfection-by-subtraction-the-minimum-feature-set/

bmilab. (n.d.). *Concierge Test*. https://bmilab.com/testing/cards/concierge-test

Cloudflare. (n.d.). *Why Does Site Speed Matter? | Improve Webpage Speed*. https://www.cloudflare.com/learning/performance/why-site-speed-matters/

Combeenation. (2019, August 14). *What Are the Different Types of Product Configurators?* Combeenation Configurators. https://www.combeenation.com/en/what-are-the-different-types-of-product-configurators

Cyledge. (n.d.). *Configurator Database*. https://archive.is/PBn0l

cyLEDGE Media. (2008). *Configurator*. Configurator Database. https://www.configurator-database.com/definitions/configurator

Flyvbjerg, B., & Budzier, A. (2011). *Why Your IT Project May Be Riskier Than You Think*. https://hbr.org/2011/09/why-your-it-project-may-be-riskier-than-you-think

Franke, N., & Piller, F. T. (2003, January). Key research issues in user interaction with user toolkits in a mass customisation system. *International Journal of Technology Management*, *26*, 29. 10.1504/IJTM.2003.003424

Google. (n.d.). *Google Identity Platform*. https://developers.google.com/identity

Google. (n.d.). *Search Engine Optimization (SEO) Starter Guide*. https://support.google.com/webmasters/answer/7451184?hl=en

Hackernews. (2019). https://news.ycombinator.com/item?id=19308027

Heuvelmans, E. (2010). Implementation of a mass customization strategy. 33. http://arno.uvt.nl/show.cgi?fid=121051

IdeaRoom. (n.d.). *Build v. Buy: Deciding between Custom and Licensed Software*. https://www.idearoominc.com/blog/build-v-buy-deciding-custom-licensed-software

Jobs, S. (1997). You've got to start with the customer experience. https://blogs.oracle.com/today/youve-got-to-start-with-the-customer-experience

Koster, L., & Peters, R. (2020). CAD to VR. *A guide to the conversion process*. https://leonkoster.dev/Documents/CADXRPaper.pdf

Kromer, T. (n.d.). *Concierge vs. Wizard of Oz Prototyping*. Kromatic. https://kromatic.com/blog/concierge-vs-wizard-of-oz-test/

Lean Enterprise Institute. (n.d.). *What is Lean?* https://www.lean.org/whatslean/

LePage, P., & Andrew, R. (2019, Febuary 12). *How to create sites which respond to the needs and capabilities of the device they are viewed on*. https://web.dev/responsive-web-design-basics/

Marsner. (n.d.). *Why Custom Software Development Is The Best Approach For Businesses*. https://marsner.com/blog/why-custom-software-development-is-the-best-approach-for-businesses/

Massey, B. (2015, August 19). *What are some strategies for reducing bounce rate?* Landing Page Optimization. https://conversionsciences.com/what-are-some-strategies-for-reducing-bounce-rate/

Pichler, R. (2013, October 9). *The minimal viable product and minimal marketable product*. https://www.romanpichler.com/blog/minimum-viable-product-and-minimal-marketable-product/

Product Designer. (2018, October 2). *3 disadvantages of product customization and how to overcome them*.

https://www.productsdesigner.com/blog/product-customization-disadvantages-challenges/

Ries, E. (2009, July 22). *Minimal Viable Product*. https://www.slideshare.net/startuplessonslearned/minimum-viable-product/5-Minimum_Viable_Productbr_Visionary_customers

Ries, E. (2009, August 3). *Minimum Viable Product: a guide*. Startup Lessons Learned. http://www.startuplessonslearned.com/2009/08/minimum-viable-product-guide.html

Ries, E. (2012, July 6). Twitter. https://twitter.com/ericries/status/221318901018017792?lang=en

Rogers, E. M. (2003). *Diffusion of Innovations* (5th ed.). Simon and Schuster. https://books.google.nl/books?id=9U1K5LjUOwEC&printsec=frontcover#v=onepage&q&f=false

Schwery, M. P. (2018). Lean Startup Orientation. *Empirical Evidence on Venture Success*. https://essay.utwente.nl/74732/1/Schwery_BA_BMS.pdf

SiteGround. (n.d.). *Why does my website look different on different browsers?* https://www.siteground.com/kb/why_does_my_website_look_different_on_different_browsers/

Statcounter. (n.d.). *Browser Market Share Worldwide*. Retrieved 10 1, 2020, from https://gs.statcounter.com/browser-market-share

Tammany, J. (2018, October 11). *What is website security?* SiteLock. https://www.sitelock.com/blog/what-is-website-security/

Thakur, D. (2017, July 20). *10 Good Reasons Why You Should Use Google Analytics*. https://medium.com/@dineshsem/10-good-reasons-why-you-should-use-google-analytics-699f10194834

Wankel, C. (2007). *21st Century Management: A Reference Handbook* (Vol. 1). SAGE Publications, Inc.

https://books.google.nl/books?id=heTVxjr3HtQC&pg=PA425&lpg=PA425&dq=represented+visualized+assessed+priced&source=bl&ots=FArvTYyciT&sig=ACfU3U3tU028aMoCCvAx8yz2haf-aW16IA&hl=en&sa=X&ved=2ahUKEwiykN710LvrAhUL2qQKHcdNCBQQ6AEwDnoECAEQAQ#v=onepage&q=represente

WebFX. (n.d.). *Is SEO Important for Every Business?* https://www.webfx.com/internet-marketing/is-SEO-important-for-every-business.html

Appendices

Minimal Viable Product

The goal of the MVP is to gauge the desire for a stove configurator and discover its desirable features. This is achieved by creating a minimalistic version of a stove configurator, so that visionary customers can provide early validation and feedback. Their feedback is essential to answer critical questions, such as;

- How much do customers desire to configure a stove?
- What features are essential for customers to design a stove?
- What does the ideal configuration process of a stove look like?

Without validated feedback, it is impossible to answer these questions with certainty. A great benefit of validated learning through customers is that it provides answers to unasked questions. A product's development team has a different frame of reference, which does not always portray the product as customers would see it. This is why customer interviews

provide opportunities to think outside of the box. They serve as a recalibration point for the product's vision to ensure a proper market fit.

The challenge of making an MVP is choosing the minimum feature set that best represents the full concept. The process of designing a stove consists of many operations, each of which being a considerable feature. Fortunately, in the case of an MVP, it is unnecessary to implement the complete process. Only an overview of the process has to be provided, as visionary customers can "fill in the gaps" (Ries, 2009). After interviewing the client, it was determined that the configurator's design process that can be simplified to five steps;

1. Choosing a combustion chamber
   There are many kinds of combustion chamber models. The final design heavily depends on which one is picked. Usually, customers already have a rough idea of what they want. Choosing a combustion chamber seemed like a logical first step in the design process. In the MVP, only one combustion chamber was added for simplicity. This way it was possible to show visionary customers what the process would look like, and that more models were planned.

2. Enter house info
   As the design should heavily depend on the house of the customer, it was logical that the configurator provides advice based on the customer's living situation. This requires certain information about the customer's house before the design process starts. However, the client argued that entering (complex) information as the first step might scare away potential customers. The required information (like level of insulation, chimney height, room size, etc) are rarely known by the customer when asked for. Hence, the step was put second. It was also made optional for customers who want to experiment with the configurator before investing significant effort.

3. Design ducts
   The long ducts made from dense stone are what make the stove radiate its warmth for long periods of time. They are a crucial part of the stove, and their layout determines the stove's overall shape. As the ducts connect the combustion chamber to the chimney, their layout should be leading in the design. This is why it was a logical third step. However, it is also the most complex step in the process. Only the most basic functionalities were implemented, as the MVP would have gotten too complex otherwise. In this step, users could preview, create, rotate, delete and connect parts. It was possible to undo and redo these operations as well. Finally, The whole stove setup could be moved relative to the wall, and the ducts could be connected to the chimney.

4. Design facade
   Unfortunately, there was no design of the facade yet during the development of the MVP. This is why the facade design step was not implemented in the MVP. However, it was known what the facade would roughly look like. Hence, it was planned to be the fourth step. The facade consists of a "box" made from concrete tiles that encapsulates the stone ducts, on which panels could be placed from different

materials. The idea was to automatically generate the box around the ducts. The box would be relative to the combustion chamber to ensure a proper fit. Then, customers would be able to select the panels and change their materials from a sidebar.

5. Order
   The order step was not implemented either, but added as a placeholder. The feature is not implemented because it is not a problematic part of the solution (as it has been done countless times before). However, it would have been important to automatically generate a bill of materials from the design process. The bill should show all the parts and their prices, to provide an overview of the costs to the customer. The order process should also send an automatic email to the client with all the necessary information. So although it is not a challenge design wise, its importance should not be underestimated.

During all of the mentioned steps, it is possible for the client to go back to the previous step. It was also possible to save and load a stove design to and from the cloud, to test how well the technology stack would work.

The MVP's market fit is tested through an interview with three clients who have used the MVP. Their feedback is essential to define the functional and non-functional requirements for the MMP. Negative feedback points of the MVP shall be used for testing the MMP, to ensure that the right changes have been implemented.

Implementation

Unity

The MVP was made by Stanislav Mutafchiev in Unity3D, while the cloud save and load system was made by me, the author. We presumed that Unity would allow for quicker prototyping due to it providing a complete solution. The biggest reason why Unity3D was chosen is because of its WebGL build capability, which allows easy development of web applications. The same code that would normally make a standalone PC application could now be put on the web. The code would be transformed into WebAssembly; the latest standard in high performing application code on the web. Using the Unity Editor, it was easy to visualise what was going on when building the configurator. This was especially true while making the duct system, as it has to connect properly. Because Unity has a built-in physics engine, it was possible to check if a new part overlapped with already placed parts. It was also straightforward to load models, textures and scripts. One major benefit of unity compared to traditional web development is its user interface builder. It allows for easy creation of buttons, text boxes and images. There is also the Unity store, which provides many premade features that could save a lot of development time.

Github

The WebGL build from Unity was put on GitHub pages, which is basically a free hosting service. It allows for static pages to be uploaded, such as a WebGL build from Unity3D. Using Git, it was easy to push a new build and test the functionalities on the web. GitHub also offers many project management tools, like project boards that consist of Github issues; small user stories that allow for checkboxes, code snippets and text formatting. These could

be linked together, or referred to from Git commits. Having the whole Unity project hosted, managed and developed from one place is a great experience for developers. It made the development process quick and easy.

Results

Customer validation

Interviews have been given to three clients after they used the MVP. Observations of their use of the MVP have also been conducted.

1. The first user is an old customer of the client. He thought that the concept of a modular stove was promising. However, he argued that it will only work when the final result is pleasing to the eye. This had to be true for the product in real life, and its visualization in the configurator. He stated that the MVP version of the configurator looked rather barebones and boring, which made him doubt the looks of the product in real life. He would like to see the virtual version next to a picture of the released version. The controls of the configurator were unclear from the beginning, as he did not know what to do in the duct design step. After a bit of experimenting with placing duct blocks, he asked what shapes are possible. Again, he would like to see more pictures of more real life examples. The user stated that playing around with the duct blocks was fun, but he could not envision making a whole stove in a configurator.

2. The second user has a design background. She mentioned that the aspect of modularity is really nice, and that she has seen a good configurator in Ikea for modular couches. She was especially curious about the unique selling points of the modular stove, and the materials that could be used. She really liked the idea of a simple configurator, and does not want to tinker with the internal parts (like the ducts). Her other feedback was on the stove design itself, and not on the configurator. The feedback on the product was quite positive.

3. The third and last user was a software developer. He had more comments on the non functional requirements of the configurator. He thought that a loading time of ten seconds is too much, and should be way lower to avoid users bouncing to another website. He also mentioned that the concept is great for people who use tablets, such as ipads. He stressed that it is also important to ensure compatibility with IOS devices and safari. He was curious on how well Unity WebGL would work with the google search engine, and dynamic links. He pointed out that search engine optimization and page load speed are things that should be considered before sticking with a particular technology stack.

The feedback showed that using a configurator to build a modular stove is promising. However, it is not viable in its current form. No one was particularly fond of designing the ducts, as its process was unclear. Overall, the results were mixed.

Strengths	**Weaknesses**

● Configurator is generally appreciated by users. ● Product itself received positive feedback. ● Product concept fits well with a configurator. ● Online configurator is technically feasible.	● Customers do not desire to configure the internal mechanics of a stove. ● Users find it difficult to imagine possibilities. ● Product does not look realistic in the configurator. ● Current technology stack is unsuitable. ● Customers do not understand the unique selling points of the product and configurator without substantial explanation.
Opportunities ● Customers are interested in different materials. ● Customers are interested in possible shapes. ● Support for mobile devices would be appreciated. ● Overview of price.	**Threats** ● Switching technology stack takes a lot of time. ● It is still not completely certain if the concept is viable on the market. ● Unique selling points should be properly explained in order for the product to be interesting.

Table 3. SWOT analysis of user feedback on MVP.

Technology stack

Although the MVP was mostly meant for customer validation, it also provided insight into the non-functional requirements of a technology stack. In this case, the technology stack only consisted of Unity3D and Github.

Github is great for repositories and project management. Github Issues allow for agile development, and is incredibly powerful. The connection between the Git repository and project management functionalities makes it great for small to medium sized teams. There are more sophisticated project management tools out there, like Jira. But four our use case, it was perfect. Github pages is also a nice and simple hosting service. It allows for quick prototyping and manual testing on the web. However, it only hosts one version of an app. This makes Github pages unsuitable for bigger projects, where it is desirable to have a main version for real users and a test version for developers. It also does not offer password protection of the web page as of 2020. The Github environment is rapidly changing, so it is advisable to check on the current status before choosing Github. Their CI and CD pipeline, called Github Actions, is also worth looking into.

The results with Unity are mixed. It is definitely a useful framework for quick prototyping, and its WebGL build capability is surprisingly functional. However, it certainly is not meant for advanced web applications such as configurators. Unity's focus on being a game engine makes it inflexible in the web environment. This is due to it being a closed environment; the code written for the unity engine (C#) does not cover the browser specific functions found in JavaScript. It is possible to call JavaScript functions from C#, but it requires an interface, which adds a lot of complexity. This makes it very difficult to connect to WebAPIs, such as

the ones from databases. Another huge consideration is its lack of search engine optimization support. All the text that is shown in the Unity WebGL build is done without HTML, making it difficult for search engines like Google to know the content (quality) of the page. This might result in a low ranking on Google searches, which is undesirable for businesses. Last but not least, the C# code written for the Unity3D engine cannot be transferred to JavaScript when switching to another framework. Due to the high switching cost, one must definitely know all the limitations of Unity before choosing it as a framework for the web.

Strengths ● Easy development in Unity Editor. ● Complete WebGL engine on the web. ● Easy to create UI. ● Unity store offers many features. ● Might be more performant in some cases due to WebAssembly.	**Weaknesses** ● Poorly interfaced with JavaScript in the browser. ● No asynchronous functionality due to transpilation. ● Slow initial load time. ● Does not support lightweight GLTF format for 3D models. ● Difficult to load 3D models at runtime. ● SEO unfriendly. ● Nasty bugs due to transpilation. ● Slow building time. ● Not free, unlike other web frameworks.
Opportunities ● Unity WebGL is very promising, and might be viable in the future. ● Great for small standalone applications on the web.	**Threats** ● Code cannot be transferred to JavaScript when changing stack. ● WebAssembly might not work on older devices. ● Undesirable for traditional web apps.

Table 4. SWOT analysis of Unity3D for the use of an online configurator.

Conclusion

Everything considered, a pivot is definitely required. This does not mean that the MVP was a failure; a lot of lessons were learned without significant investment. The experience captures the essence of an MVP, and shows its essential role in validating the problem and finding the requirements of a solution. The MVP will not be developed any further, as the Unity3D framework is unsuitable. The next MVP will be built from the ground up. This is not unusual in the build-measure-learn feedback loop. Sometimes, it is necessary to make multiple MVPs. Only once a suitable solution has been found, the MVP can be further developed into an MMP.

In the case of the client, this MVP can be considered successful. However, mistakes have been made. For example, earlier research on the Unity3D WebGL framework would have resulted in a different framework being chosen. If the framework of this MVP was better suited, its development could have continued and resulted in less waste. Another mistake

were some of the MVP’s features; the cloud saving and loading were unnecessary for customers to provide early validation of the concept. Fortunately, both these mistakes taught us a lot about the technical requirements of a configurator’s framework.