Linnæus University
Sweden

Bachelor Degree Project

Designing a Platform for Creating 3D Product Configurators

- A Design Science Study

Authors: Alvin Bjelkendal, Jonathan Olsson
Supervisor: Daniel Toll
Semester: VT 2023
Subject: Computer Science

Abstract

In the rapidly evolving world of e-commerce, businesses are increasingly adopting the idea of providing customers with the ability to visualise and customise products in 3D through a 3D product configurator. However, the creation of 3D configurators is a complex and time-consuming process, requiring technical expertise and extensive communication between businesses and solution providers. This thesis aims to address these challenges by proposing a design for a user-focused approach to streamline the process of creating 3D product configurators. The study undertakes a thorough examination of existing platforms' 3D configurator creation processes to inform the design of an improved solution.
To assess the effectiveness of our proposed design, we identified some key requirements and proceeded to develop a prototype. Testing was conducted, involving the attempt to recreate benchmark configurators with different levels of complexity. Through this process, we assessed the quality and efficiency of our platform in comparison to solutions developed partially or from scratch. Our results demonstrated that a user-focused platform for creating 3D product configurators, can offer extensive time and resource-saving. However, further research and development are necessary to fully explore the scalability, integration possibilities, industry-specific effectiveness, and customer impact of such platforms.

Keywords: *3D Product Configurator, Platform, Design Science*

Acknowledgments

We would like to extend our sincere gratitude towards Daniel Toll, our university lecturer and supervisor, for his invaluable guidance and support throughout the development of this thesis. His expertise, insights, and constructive feedback have played a crucial role in shaping and refining our research.

We would also like to express our appreciation to Beanloop for providing us with the inspiration and idea for this thesis. Their contribution and collaboration have been instrumental in driving our research forward.

Lastly, we would like to acknowledge our families for their unwavering support and understanding throughout the process. Their encouragement and belief in our abilities have been a constant source of motivation.

Without the support and contributions of these individuals and organisations, this thesis would not have been possible. We are deeply grateful for their involvement and assistance in making this research a reality.

Table of Contents

List of Figures

List of Tables

1 Introduction

In today's world, where e-commerce sales continue to surge [1], businesses are increasingly seeking new and innovative ways to provide customers with a personalised and engaging product experience. Recent studies have shown that the use of 3D visualisation in e-commerce, can significantly impact sales and customer satisfaction [13]. These findings underscore the importance of providing customers with the ability to visualise products in 3D before purchasing. One way of providing this is through a software application that allows customers to interact with a product in a virtual environment, this type of application is often referred to as a 3D product configurator. However, the creation of a 3D product configurator can be a complex and time-consuming process that requires extensive technical expertise [10]. The primary objective of this 15 HEC Bachelor Thesis in Computer Science is to streamline the process of creating a 3D configurator. Our goal is to design a solution that accelerates and simplifies the 3D configurator development, making it more accessible and efficient. By developing a prototype, we intend to evaluate the effectiveness of our design.

1.1 Background

In recent years, the popularity of offering 3D-configurators e-commerce has significantly increased [11]. 3D-configurators are powerful tools that allow customers to visualise and customise products in 3D, providing an immersive and engaging online shopping experience [5]. 3D-configurators can be found in a wide range of industries, including furniture, fashion, and automotive, among others. According to recent estimates, there are approximately around 1500 different 3D configurators available online today, spanning across 17 different industries [11].

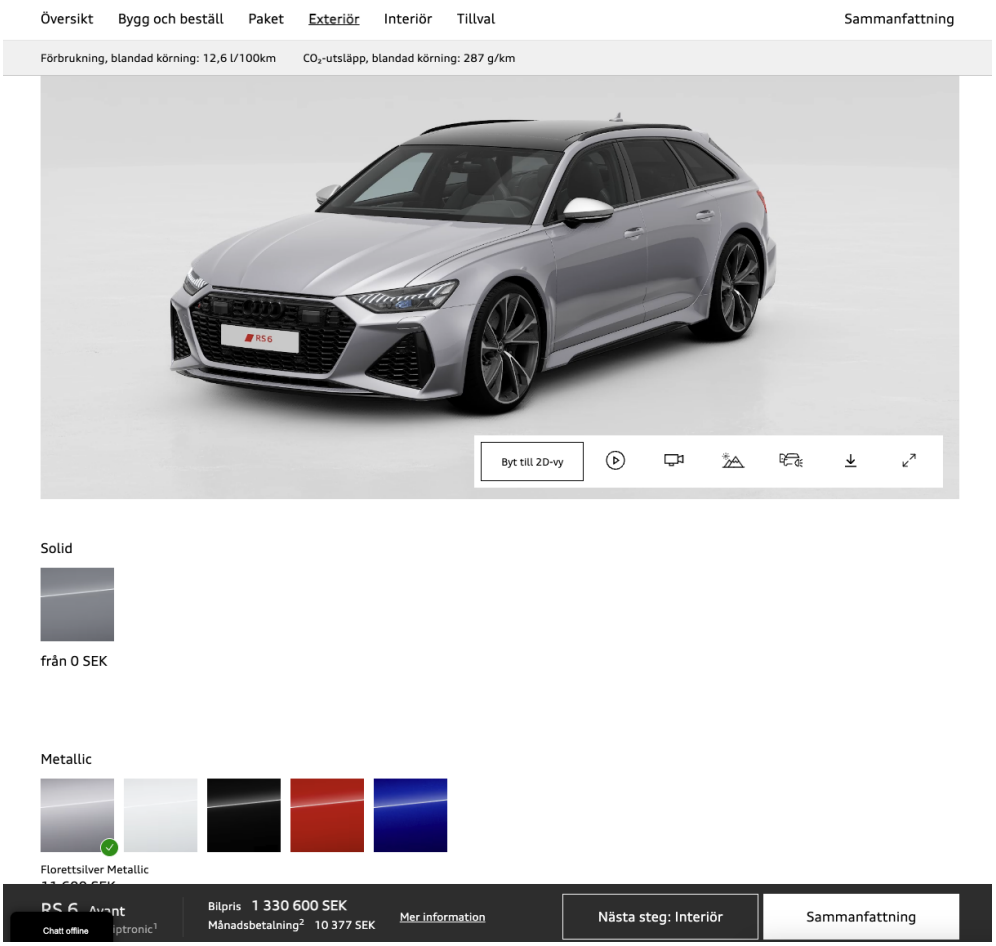

*Figure 1.1 Example of a 3D product configurator (A*udi®)

Figure 1.1 showcases an example of a 3D product configurator. In the figure, the selected colour is visibly applied to the car, highlighting the dynamic real-time interaction achievable through a 3D configurator.

One business sector that has made significant strides in providing customers with the ability to visualise and customise their products in a 3D, is the car industry. Most car brands today have some type of 3D product configurator integrated into their online shopping experience, where the customer can customise and visualise a car in 3D before purchasing it [11]. A good example is Audi's 3D configurator [6] which provides a seamless and engaging user experience, allowing customers to design their dream car in intricate detail. From selecting the model and colour to customising the interior and adding optional features, the 3D configurator offers a comprehensive view of the car before the customer decides to purchase it [6].

The fashion industry is another business sector that's increasingly adopting the concept of integrating 3D visualisation into their online e-commerce

experience [11], particularly with the introduction of 3D virtual try-on services. These services allow customers to envision clothing items on themselves in a three-dimensional virtual environment.
A recent study [13] based on actual customer data, focused on the women's casual L brand and found that 3D virtual try-on had a significant positive effect on online sales. The study analysed actual sales data and conducted qualitative interviews, and showed that virtual try-on increased the average sales per customer by 14,000 won (approximately 13 USD) and decreased the return rate by 27%. These findings indicate that giving customers the ability to visualise products in 3D before purchasing, helps to improve customer satisfaction, boost sales, and decrease returns [13].

Another notable example of a 3D configurator is Fjällräven [7], a Swedish brand that specialises in outdoor equipment and clothing. They have integrated a 3D configurator for their popular “Kånken” backpack product line. With the configurator, customers can fully customise the backpack’s colour scheme, creating a unique and personalised product. Fjällräven’s 3D configurator offers a user-friendly interface that allows customers to visualise their customised backpacks accurately. It provides customers with a range of colour options to choose from, giving them complete control over the final product’s appearance. The configurator also offers an immersive view of the backpack from different angles, providing customers with an accurate representation of the final product [7].

Audi and Fjällräven are two perfect examples of 3D product configurators, however today creating a 3D product configurator can be a complex and time-consuming process that requires extensive technical expertise [10]. If there existed a platform where anyone could create this type of product, the accessibility of 3D product configurators could be significantly increased.

Website creation platforms like Wordpress[8] and Shopify[9] have been popular due to their ability to deliver users the ability to build customised websites and online stores with coding or technical expertise. WordPress offers a large range of plugins and templates, enabling users to create professional websites. On the other hand, Shopify serves as an all-in-one solution for e-commerce websites, allowing businesses to establish their online stores without requiring coding knowledge.
These creation platforms let users create professional-looking websites and fully functional online stores without the need for any programming knowledge. In a similar vein a platform for creating 3D product configurators could let users without any coding experience create a customised and professional looking 3D configurator for their own 3D models.

Beanloop AB is our partner company, who presented us with the idea of creating a platform for creating 3D configurators. They have a great interest in this type of product as they have multiple customers who have expressed a desire to include a 3D product configurator in their online shopping experience, and Beanloop believes that this type of tool could help them save a lot of time and resources while still meeting their customers needs. Beanloop has provided us with a document outlining their vision for the product. That document will be used as the basis when developing the requirements for our prototype, and we will work closely with Beanloop to ensure that the requirements align with their needs and expectations.

1.2 Related Work

In recent years, there have been several efforts to streamline the process of creating 3D configurators for e-commerce platforms. One such platform is the 3D web configurator which utilises a self-learning algorithm to optimise 3D object features based on user preferences [12]. This solution has been beneficial in simplifying the creation of 3D configurators, yet it still requires a certain level of technical skill and knowledge.

As the interest of integrating 3D product configurators in e-commerce experiences rises, the demand for a user-friendly platform that allows for flexible and customizable configurator creation will continue to grow. In response to this need, platforms like Dopple [20], VividWorks [21], and Animech [22] have emerged as strong contenders, helping businesses to create engaging and personalised 3D product configurators.
These platforms offer a range of features and capabilities to facilitate the development of high-quality configurators, catering to the specific needs and products of businesses.

Dopple provides businesses with an immersive and interactive 3D configurator creation process. It begins with a collaborative workshop to understand the desired visual experience, gathering essential product information and files. Dopple's team of 3D artists can also create 3D models if needed. The configurator is then launched, and Dopple collects user data to provide valuable insights to the customer.

VividWorks offers businesses the ability to create engaging, high-quality 3D configurators tailored to their needs. The process involves booking a demo or initiating contact to discuss unique requirements. VividWorks works with existing 3D models or creates them from scratch. They provide a personalised demo for customer feedback, refining the configurator according to the customer's vision. The solution is seamlessly integrated into the business's infrastructure with comprehensive documentation and support.

Animech assists businesses in creating immersive 3D product configurators, as exemplified by their work on Fjällräven's Kånken Me configurator. The process includes an initial consultation to understand goals, followed by an analysis of the product and existing materials. Animech presents a price quote, and upon agreement, plans the project in collaboration with the business. They create a 3D model of the product, configure the behaviour of the configurator, and connect product data and 3D files. Thorough testing is conducted before integration and release.

These processes will be explained further in chapter 3 Theoretical Background.

1.3 Problem Formulation

While the platforms mentioned in 1.2 Related Work offer valuable solutions, and help businesses create engaging high-quality 3D product configurators tailored for their specific needs and products. These solutions are very time consuming as they involve numerous steps and extensive communication between the solution provider and the customer. These steps will be more discussed in chapter 3, Theoretical Background.
If there existed a design for a solution that did not require any programming skills and hence could give the customer control over creating the 3D product configurator themselves, it could potentially speed up this process drastically.

The goal of this thesis is to propose a design for a solution that:

- Makes the creation of 3D product configurators easy and fast, without the need for coding.

- Place the creation of 3D product configurators in the hands of the customers, thereby eliminating the back-and-forth communication between a solution provider and the customer, for the majority of the creation process..

1.4 Solution

To address the challenges outlined in the problem formulation, our thesis focuses on creating a design for a solution that empowers businesses to create their own 3D product configurators effortlessly, eliminating the need for programming knowledge. By enabling businesses to take control of the configurator creation process, our design aims to significantly reduce the time and communication required between businesses and solution

providers. To validate the effectiveness of our design, we will develop a prototype that embodies some of the key features and functionalities of the envisioned solution.

1.5 Research Questions

The research questions for this thesis are the following:

RQ1	Can a no-code solution for creating 3D product configurators generate configurators of equivalent quality compared to those that are programmed from scratch, hence saving time and resources?
RQ2	How can a platform for creating 3D configurators be designed to accommodate the specific requirements for different types of 3D product configurators, such as accessories or cars?

Table 1.1 Research Questions

1.6 Outline

This bachelor thesis is organised as follows. In chapter 2 we discuss the methodology used in our research, such as how we apply design science and work on the basis of an iterative process. In chapter 3, we go deeper into the development techniques we used for developing our prototype. In chapter 4 we will discuss the overall architecture and a flowchart for our prototype to then show our implementation of the prototype in chapter 5. The implementation will then be tested and the results of the testing will be presented in chapter 6. In chapter 7, we will discuss the result of the tests. Lastly in chapter 8, we present the conclusions of the result and discussion and. Here we will also present suggestions for future directions.

2 Methodology

In this chapter, we will explore the methodology employed in this bachelor thesis. Throughout this chapter, we will delve into various components of the methodology utilised, including design science as research strategy, the iterative working process, how we collect and analyse data and ethical considerations.

2.1 Design Science

Design science is a problem-solving methodology that involves the creation of innovative solutions for complex problems through the design, implementation and evaluation of artefacts. It is an approach that is particularly useful in the field of information systems and IT [3]. The design science research (DSR) process typically involves six stages: problem identification and motivation, definition of objectives for a solution, design and development of the artefact, demonstration of artefact effectiveness, evaluation of the artefact's impact on the problem domain, and communication of results. DSR has been widely used in the development of software applications, decision support systems, knowledge management systems, and other types of artefacts that address practical problems in various domains within the information systems and IT field [4].

The use of design science methodology and iterative evaluation will be an important aspect of this project. In our case, we have the intention to solve the problem explained in the Problem Formulation section with two artefacts: Design and Prototype. The Design artefact is a detailed plan or a blueprint that outlines the structure and functionality that could solve the problem. On the other hand, the Prototype artefact is a representation or working model of the design. It aims to simulate the intended functionality to validate and test the feasibility of the design. The prototype may not have all the final features but serves as a functional demonstration of the core concept. While developing the prototype we will follow an iterative approach to design and evaluate it. This involves creating a series of prototypes, testing them, and then making changes based on the feedback received. The iterative process will continue until we have developed a final version of the platform that meets the requirements and objectives of the project. To clarify, the design in the iterative process is not the Design artefact, it is a design of each prototype that involves parts of the core concept.

2.2 Iterative Process

This bachelor thesis will heavily rely on an iterative process as a fundamental aspect of the methodology. This process will involve the

creation of a series of prototypes that will be tested with mainly Beanloop, allowing for the refinement and improvement of the design. The iterative approach will enable the development of a working process that follows a specific flow, including requirements collection, design, implementation, testing, evaluation, and feedback from the working partner, Beanloop. Beanloop is an appropriate partner company to provide feedback for a couple of reasons. Firstly, they have experience and expertise in the relevant field. Their knowledge can contribute to refining and improving the design and our iterative process of developing the prototypes. Secondly, Beanloop is actively involved in the iterative process. Their engagement allows for continuous improvement and feedback throughout each stage of the workflow, from requirements collection to design, implementation, testing and evaluation. Furthermore, Beanloop's role as a working partner means they have a vested interest in the success of the project which ensures their feedback is relevant and important.

To initiate the workflow, we will collect the design requirements from a vision developed by Beanloop. These requirements will then be prioritised based on their importance and feasibility for this thesis, meaning that we will filter out parts of the design that we together with Beanloop find most important to implement. Using these requirements as a guide, we will design and develop an initial prototype for each iteration, which will then be implemented, tested, evaluated, and reviewed by Beanloop.

The feedback we will receive from Beanloop during each iteration will be used to refine the design and improve the application. This step will be critical in motivating the choices we make in the design and in ensuring that the final product meets the project's objectives. Once the design is refined, a new prototype will be developed, and the testing, evaluation, and feedback process will be repeated until a final version of the application is achieved that meets all requirements and objectives.

By following this workflow, we intend to ensure that the design is continuously improved and refined to meet the needs of the target audience, which is both Beanloop and other companies that have an interest in a 3D configurator. The iterative process will allow for early identification and resolution of issues or problems with the design, preventing them from becoming more difficult or time-consuming to address. Overall, the iterative process will play a critical role in the methodology of this thesis.

The figure below demonstrates the structure of an iteration and how the different steps are connected.

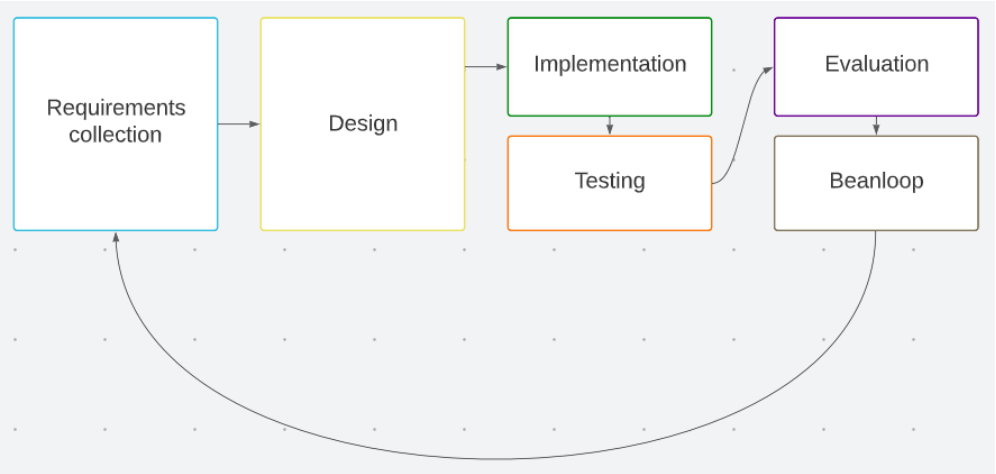

Figure 2.1 Iteration Design

2.2.1 Iteration Report

After completing each iteration, we will create a comprehensive report that will outline every aspect of our work and progress during that iteration. This report will serve as a detailed record of the methodology, process, and outcomes of each iteration.

The report will include a description of the design requirements that were collected at the beginning of the iteration, as well as the prioritisation process used to determine which requirements were feasible for implementation. The report will also document the design decisions that were made, as well as the design elements that were included in the prototype.

The implementation process will also be outlined in the report, including any coding, testing, and debugging that was done to create the prototype. Any issues or challenges encountered during this phase will be documented, along with the steps taken to address them.

Once the prototype is complete, we will conduct a thorough evaluation and testing process to ensure that it meets the desired objectives and requirements. The results of this evaluation will also be included in the report, along with any feedback or suggestions received from Beanloop.

Overall, the report will hopefully provide a comprehensive overview of each iteration, allowing us to track progress over time and identify areas for improvement. It will also serve as a valuable reference for future work.

2.2.2 Data Collection

In our iterative development process, we will actively engage with Beanloop through live demos to gather feedback on our prototype. This feedback will play a crucial role in refining and improving the platform.

Before each iteration, we will collaborate closely with Beanloop to understand their vision and break it down into manageable requirements. Through live demos, we will showcase the prototype to Beanloop and actively involve them in the interactive exploration of its functionalities and features. This hands-on approach will allow Beanloop to provide immediate feedback on the platform's usability, effectiveness, and alignment with their needs and expectations.

During the live demos, we will present the prototype to Beanloop, providing them with an overview of the platform's functionalities, design, and features. Through this demonstration, Beanloop will have the opportunity to explore the prototype and gain a comprehensive understanding of its capabilities.

Following each live demo session, we will thoroughly analyse the feedback received from Beanloop. This analysis will help us pinpoint specific areas that require improvement for future iterations. The insights gained from the live demos will inform the refinement of the platform, ensuring that it evolves to meet the needs and expectations of our users effectively.

By leveraging the power of live demos and the direct involvement of Beanloop, we aim to gather real-time feedback and insights to drive continuous improvement of the platform throughout the iterative development process.

2.2.3 Data Analysis

In our iterative development process, we will gather feedback from live demos with Beanloop, which will serve as the basis for our data analysis. This analysis will provide valuable insights into both the requirements and usability of the platform, guiding our decision-making and improvement efforts. The feedback will be carefully reviewed to ensure that the platform’s requirements are complete, relevant and aligned with Beanloop’s vision.

To conduct a comprehensive analysis of each iteration, we will create an iteration report, as mentioned in 2.2.1 Iteration Report, that incorporates the feedback gathered from the live-demo sessions with Beanloop. This report will serve as a valuable resource for our data analysis process.

2.3 Final Evaluation

Once the final prototype of the platform has been developed through the iterative process, a final evaluation will be conducted to assess its effectiveness. This will be evaluated in chapter 6, Testing and Results. To answer RQ2, we will involve recreating three different types of 3D product configurators: basic, intermediate, and advanced in our testing. These categorizations are based on the number of features each configurator offers, with basic having the smallest set of features and advanced having the most comprehensive set.

The goal of this final evaluation is to ensure that our platform is capable of creating 3D configurators that match the complexity and functionality of existing solutions in the market. By mimicking existing configurators, we aim to assess the platform's ability to handle different levels of configurator complexity while identifying any areas for improvement or potential shortcomings.

By conducting this final evaluation, we aim to validate the effectiveness of our platform in terms of its ability to recreate real-world 3D product configurators and the creation time that it requires, also this evaluation will help us identify areas where further improvements can be made. This will help us answer RQ1 as we will assess the effectiveness of our prototype to create 3D product configurators of compared solutions developed partially or from scratch.

2.4 Ethical Considerations

As with any research project, ethical considerations are an essential component of developing a low code solution for 3D product configurators. The following considerations will be taken into account throughout the research and development process:

Firstly, if we were to implement our design fully, privacy and data protection would be a top priority to ensure that any personal data collected from customers, such as their name, address, and payment details, is handled securely and responsibly. However, as we aim to only implement a prototype with the core functionality for a platform like this, we will not collect any user data.

Secondly, accessibility and inclusion will be at the forefront of the solution's design to ensure that it is accessible to all users, regardless of their technical expertise or ability. The interface will be designed to be intuitive and user-friendly.

Finally, the product configurator will aim to provide an accurate and realistic representation of the final product. The platform should not be used to mislead or deceive customers into purchasing a product that does not meet their expectations.

By considering these ethical considerations, the development of a low code solution for 3D product configurators will be conducted in an ethical and responsible manner, ensuring that the solution is accessible, transparent, and fair to all users.

2.5 Validity and Reliability

To establish the validity and reliability of our design approach in terms of its effectiveness in creating 3D configurators, we will conduct a comprehensive testing process. This process will involve attempts to recreate existing benchmark 3D configurators using our prototype. Additionally, continuously getting feedback from our partner company, Beanloop, who are experts in web development. Will also strengthen the validity of our design approach's effectiveness, as they will ensure that what our prototype is creating is inline with what a 3D configurator is.

3 Theoretical Background

This chapter delves deeper into the processes involved in creating 3D configurators using existing platforms. Additionally, we explore the key techniques that we have carefully chosen to incorporate into our design and prototype.

3.1 3D Configurator Creation Processes

As described in 1.2 Related Work, the process of creating 3D configurators using existing platforms involves several steps. In this section we will dive deeper into how those processes work.

3.1.1 Dopple

Dopple's platform [20] offers a range of features and capabilities that enable businesses to create immersive and interactive 3D configurators. The process of creating a 3D configurator with Dopple involves three main steps, each tailored to ensure a successful implementation.

1. The first step is called "Collaboration," where Dopple engages in a collaborative workshop with the business to understand the desired visual experience for their customers. During this phase, Dopple works closely with the business to gather essential product information and files, ensuring a deep understanding of the customization options and requirements.

2. In the second step, "Asset Creation", Dopple's team of 3D artists comes into play. If the customer does not already possess 3D models of their products, Dopple's team can create 3D models for the customer.

3. The final step, "Launch & Learn," marks the deployment of the 3D configurator and the start of user interactions. Dopple assists the customer in going live with the configurator, integrating it into their website or application. As users engage with the configurator, Dopple collects valuable user data and provides crucial insights to the customer company.

By following this three-step process, Dopple enables businesses to create compelling and effective 3D configurators [20].

3.1.2 VividWorks

VividWorks [21] is a similar platform that also offers businesses the ability to create engaging high-quality 3D configurators tailored to their specific needs. The process of creating one is also similar, and involves these four key steps [21]:

1. The first step begins with businesses booking a demo or initiating contact to discuss their unique business requirements. This initial engagement allows VividWorks to understand the specific needs and goals of the business, enabling them to provide tailored solutions that meet their expectations.

2. In the second step, the customer provides VividWorks with their 3D models. If the customer does not have existing 3D models, VividWorks offers the expertise to create these 3D models from scratch.

3. In the third step, VividWorks creates a personalised demo for the customer to review and provide feedback to. This demo showcases the configurator in action, highlighting its features and capabilities. And the customer has the opportunity to provide valuable input to further refine the configurator to meet their vision.

4. Finally, in the fourth step, VividWorks integrates the solution into the business's existing infrastructure. They provide all the necessary documentation and human support to ensure a smooth integration process.

3.2.3 Animech

Animech [22], the company behind Fjällräven's Kånken Me configurator [7] is another company that helps businesses create immersive 3D product configurators. The process of creating a 3D configurator with Animech includes the following 8 steps [22]:

1. It all begins with an initial consultation, where Animech invites businesses to share their goals and aspirations. During this free consultation, businesses have the opportunity to describe what they aim to achieve through the creation of a 3D product configurator.

2. Next, in the analysis phase, Animech delves deeper into the business's product and current materials. This evaluation helps them gain a comprehensive understanding of the product's characteristics

and existing resources.

3. Once the analysis is complete, Animech presents a price quote to the businesses, allowing them to make decisions at their own pace. This step ensures transparency and provides businesses with the necessary information to evaluate the investment required for their 3D configurator project.

4. Upon agreement, the project takes off, and Animech begins planning the next steps in collaboration with the business. This involves gathering essential files, product information, and other relevant materials needed for the configurator's development.

5. The subsequent step focuses on the visual aspect, where Animech creates a 3D model of the product.

6. To bring the configurator to life, Animech connects all the data in their studio. This involves configuring the behaviour of the configurator as well as connecting product data and 3D files.

7. Before the configurator is released, Animech lets the customer thoroughly test the 3D configurator to ensure its performance and functionality meets the customers requirements.

8. Finally, Animech helps the customer integrate the 3D configurator in their environment and the 3D configurator is released.

3.2 Techniques

In this section, we present the key techniques that we have selected to incorporate into our design and prototype. The majority of these technique choices are completely based on Beanloop's preferences and requirements. However, there is one exception: the decision to use Material UI as a component library.
We opted for Material UI primarily to expedite the development process, specifically the process of designing the user interface. As the focus of our project lies more in the functionality and features of the platform rather than the user interface design itself, using a tool that could accelerate the user interface design process is advantageous. It is important to note that while we chose Material UI, other component libraries that are compatible with React like Bootstrap [24] or Ant Design [25] could have been utilised instead. However, considering our prior familiarity with Material UI and the limited timeframe of the project, it seemed like the most logical choice.

3.2.1 React

React [15] is an open-source JavaScript library developed by Facebook for building user interfaces. It utilises a component-based architecture that allows developers to create reusable UI building blocks and manage application state effectively. React uses a virtual DOM to efficiently update the actual DOM, resulting in faster and more responsive user interfaces. It is flexible and can be used in various contexts, ranging from simple single-page applications to complex applications with multiple pages and dynamic data. React also has a large and active community of developers, offering a wide range of tools and libraries that enhance its functionality. Overall, React is a popular choice for building modern web applications [15].

3.2.2 Three.js

Three.js [17] is an open-source library that was initially created by Ricardo Cabello, and has since become the world's most popular JavaScript framework for displaying 3D content on the web. Three.js provides a comprehensive set of tools and features for creating 3D graphics, including support for textures, materials, lighting, shadows, and more. It also includes a powerful 3D rendering engine, which is responsible for rendering the 3D graphics on the screen. One of the key strengths of Three.js is its ease of use. It provides a simple and intuitive API that makes it easy for developers to create and manipulate 3D objects in the browser. At the same time, it also offers a high degree of flexibility, allowing developers to customise and extend its functionality as needed [17]. Three.js is also highly performant, with optimizations for both desktop and mobile devices. It uses WebGL, a browser API for rendering 3D graphics, to achieve fast and smooth performance in the browser. Overall, Three.js is a powerful and flexible library that provides developers with the tools they need to create engaging and interactive 3D experiences in the browser. Its ease of use, performance, and active community make it an ideal choice for building modern web applications with 3D graphics [17].

3.2.3 React Three Fiber

React Three Fiber [18] is a powerful library that combines React and Three.js, enabling developers like us to create impressive 3D experiences within web applications using React components and Three.js. The library's primary aim is to simplify the process of creating and managing 3D graphics and animations within web applications, leveraging React's declarative approach. With React Three Fiber, developers can create and manipulate 3D objects using familiar React components, without the need to learn a new API. Moreover, React Three Fiber also handles the low-level details of

setting up and managing a Three.js scene [18]. These are the main reasons as to why we've chosen to work with React Three Fiber. The simplified process allows us to focus on the logic of our application rather than the complexities of the underlying 3D rendering engine.

3.2.4 Material UI

Material UI [19] is a user interface library for React that offers a range of pre-built components to streamline the development process. It is based on Google's Material Design guidelines and provides a modern and sleek look to applications. Material UI's components are highly customizable, which allows developers to create a unique and personalised UI for their project. Additionally, Material UI's documentation is well-maintained and offers extensive examples and tutorials, which makes it easy for developers to get started and learn how to use the library effectively [19].

4 System Design

In this section we are going to discuss the Design artefact of the thesis. This section will dive deeper into the design and how we came up with it.

To begin with, we examined three different existing platforms that are currently available in the market: Dopple[20], VividWorks[21] and Animech[22]. By studying these platforms in detail, we gained insights into their process of how to create 3D configurators and what they offer to the customer. This analysis allowed us to identify strengths and weaknesses, as well as potential areas for improvement that we could leverage in our own platform.

In addition to analysing existing platforms, we also examined numerous 3D configurators across different industries from Cyledge's Configurator Database [11]. Through this extensive examination, we were able to identify the wide array of features and capabilities that these configurators offer.

By combining the findings from our analysis of existing platforms and 3D configurators, we were able to develop a comprehensive understanding of the requirements and design considerations for our own platform. This informed our decision-making process when designing the architecture, user interface, and feature set of our platform, ensuring that it aligns with industry standards and user expectations.

To clarify, the platform creates a 3D configurator as shown in the figure below:

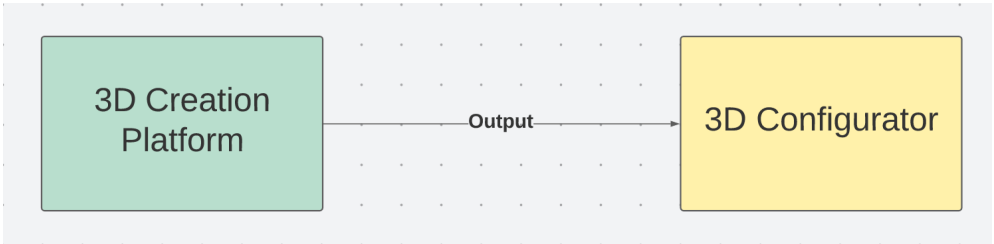

Figure 4.1 Platform Outputs 3D Configurator

Our platform's design is intentionally tailored to facilitate the integration of additional features through the utilisation of extra coding. While our prototype will only present a limited set of examples, it is crucial to underscore that our platform's design enables the inclusion of various features observed in comparable configurators, as evidenced by our test cases in Chapter 6, Testing and Results. A more comprehensive compilation of identified features from these existing 3D configurators is depicted in

Figure 6.7. The modular architecture and well-defined data structures of our platform allow for the seamless addition and customization of functionalities without compromising the overall system integrity. By adhering to industry standards and leveraging popular frameworks and libraries, such as React and Three.js, we can easily incorporate commonly observed features found in existing configurators. This approach simplifies the implementation of new functionalities and enhances the flexibility and adaptability of our platform.

4.1 Process Comparison to Existing Solutions

In this section, we compare our proposed design for 3D configurator creation to existing solutions available in the market.

When looking at the three different 3D configurator creation processes explained in 3.1 3D Configurator Creation Processes, we can observe a general representation of how those steps are aligned, which is very different from our design. Both these processes are envisioned in the figures below.

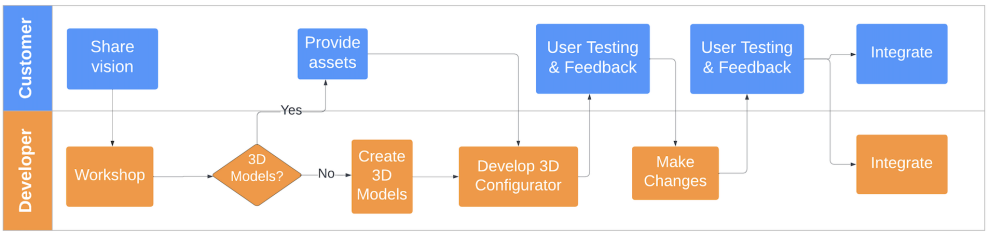

Figure 4.2 General Representation of 3D Configurator Creation Processes

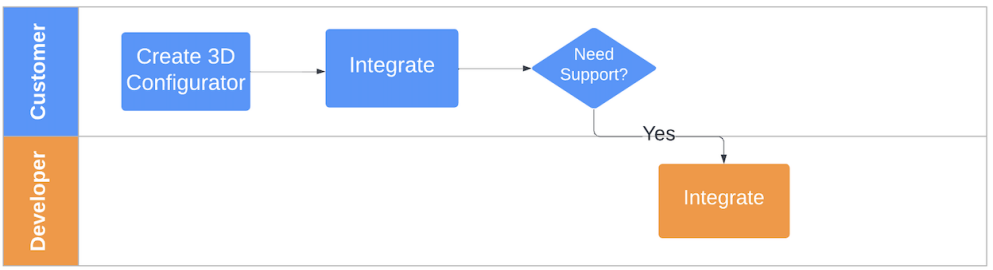

Figure 4.3 Our Design's 3D configurator Creation Process

4.1.1 Simplified Design Process.

Our design introduces a simplified design process compared to existing solutions. Unlike traditional approaches that involve extensive customer-developer interactions, our design empowers the customer to

create the 3D configurator themselves. This eliminates the need for back-and-forth consultations, user testing and feedback.

- Streamlined customer-developer interactions

  One of the significant differences in our solution is the elimination of workshops or user testing sessions between the customer and the developer. Instead, our design allows the customer to actively participate in the entire creation process, making continuous adjustments and modifications to the 3D configurator, and as a result the traditional user testing phase becomes obsolete. It is important to highlight that while our design facilitates customer participation in 3D configurator creation, it does not encompass the creation of 3D models, as that falls outside the scope of our design.

- Real-time iterative creation

  Our design facilitates real-time iterative creation, enabling customers to see immediate results as they customise the 3D configurator. This dynamic feedback loop empowers customers to make informed decisions and explore various options without the need for time-consuming iterations with developers. By providing instant visual feedback, our design accelerates the design process.

- Increased efficiency and cost savings

  Compared to existing solutions that involve prolonged development cycles and multiple iterations, our streamlined design process offers increased efficiency and cost savings for both the solution provider and the customer. By empowering customers to create their 3D configurators, we eliminate the need for extensive developer involvement and associated costs. Additionally, the real-time iterative creation reduces time-to-market, allowing businesses to launch their 3D configurators faster.

4.2 Overall Architecture

The proposed platform for creating 3D configurators is a web-based application that aims to provide users with an intuitive and user-friendly interface to create custom 3D configurators for their models. The design of the system consists of three primary components, each with its own responsibilities.

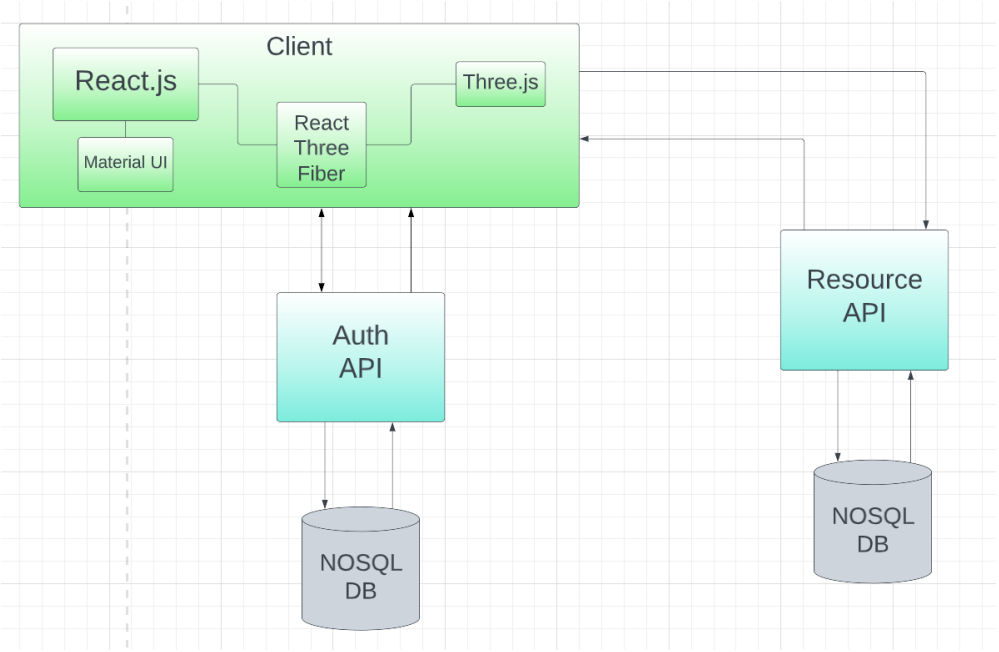

Figure 4.4 Overall Architecture

4.2.1 The Client-Side Application

The client-side application has the main responsibility for providing the functionality to create 3D configurators. This component allows users to upload their 3D models and select which features they want to make configurable in the 3D configurator. The client-side application is responsible for rendering the 3D models and providing users with an interactive configurator that they can use to modify and visualise their design. The techniques of the client-side shown in the Figure 4.4 above is more explained in section 3.2.

4.2.1.1 User Interface Design

To ensure that our Design artefact incorporates a user-friendly and intuitive user interface (UI), significant attention and effort were dedicated to its design. Our design focuses on providing a user-friendly interface that simplifies the 3D configurator creation process. We prioritise intuitive controls, clear instructions, and visual aids to guide customers through each step. By making the interface accessible and easy to navigate, we reduce the learning curve and enable customers of varying technical backgrounds to create compelling 3D configurators with ease. To design the UI, we chose Material UI [19], which provides a range of pre-built UI components that can be easily customised to suit the needs of a project.

The UI design for the whole platform was guided by the principle of simplicity, with the aim of making it as easy to understand as possible for all users, but especially for the configurator as that is the main feature that potential future customers would interact with. Our choice of Material UI as the framework for our design of the UI is rooted in our prior experience with its capabilities and the availability of a wide range of pre-built UI components that can be easily customised. While Material UI was our chosen framework, it's important to note that there are several other excellent alternatives that could also effectively support our design, including React Bootstrap [24] and Ant Design [25].

4.2.1.2 Creating a 3D Configurator

This section provides an example of the process for creating a 3D configurator using the proposed platform, from the client-side perspective. It is important to note that this example is intended to provide a brief overview of the process and is not meant to be an exhaustive representation or guide of all possible configurations. The actual process could vary depending on the complexity of the 3D model, the features selected and other factors.

1. Authentication and Project Selection

   First, the users need to authenticate themselves with their credentials and select a new project. Once the user has created a new project, they can proceed to upload their 3D model that they want to use.

2. Uploading 3D Model

   The user uploads the 3D model they want to create a configurator for. The design supports the most common 3D file formats for the web.

3. Feature Selection

   After the 3D model has been uploaded, the user selects which features they want to include in the configuration.

4. Feature Configuration

   Once the user has selected all the features they want to include, they can begin configuring them. For example, the user can choose colours for the selected materials, add environments with a specified name and create the options that should be included in the configurator.

5. Summary and Saving

   After the configuration is complete, the user gets a summary of the configuration and can create the 3D configurator. Additionally, the user can save the 3D configurator as a JSON file, which can be later accessed and modified.

To illustrate the process of creating a 3D configurator, the following figure is presented.

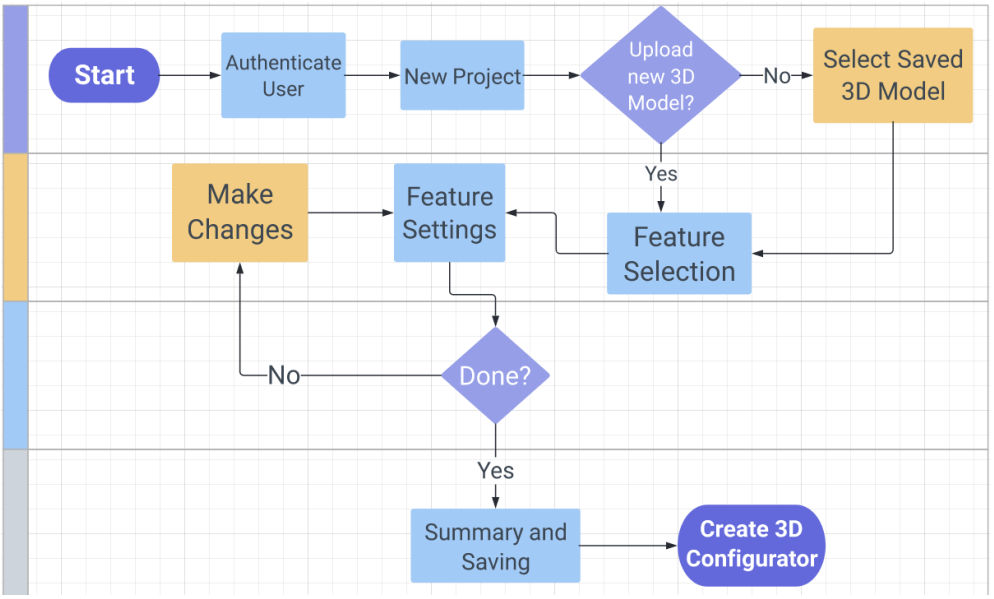

Figure 4.5 Client-Side 3D Configurator Creation Process

4.2.2 The Resource API

The resource API is responsible for saving user-generated content such as 3D-models, projects, 3D configurators in JSON format etc. The resource API will enable users to save their work and be able to return to it later, as well as provide a way for users to share their projects with others. This component is important for the architecture as it ensures that user-generated content is persistent and can be accessed later.

4.2.2.1 Persistent Storage of User-Generated Content

In our design, we have chosen a document database to store user-generated content. Specifically, we have chosen MongoDB [28], a popular NoSQL database that utilises JSON-like documents. This decision was based on a combination of factors, including the vision presented by Beanloop, our own

past experiences, and the widespread adoption of MongoDB within the developer community.

The use of MongoDB offers several advantages for our platform. Its flexible document-based structure allows us to store and manage various types of content generated by users efficiently. This includes 3D models, projects, configurators, and other relevant data. The JSON-like format of MongoDB documents aligns well with the nature of our platform, facilitating seamless integration and retrieval of user-generated content.

It is important to note that while MongoDB was the chosen database solution, other options, such as SQL databases like MySQL, could have also been considered for our design.

4.2.3 The Authentication API

The authentication API is responsible for providing a secure way for users to log in and access their saved projects and configurations. It will handle different types of roles that a user can have, for example administrator, developer or a regular user. Each of these roles will have specific permissions and restrictions. An administrator will have access to all saved projects within the company, as well as the ability to create and manage user accounts. A developer has access to functionalities related to the platform to be able to develop the source code. A regular user has the ability to create and customise configurators.

4.2.3.1 Secure User Authentication and Authorization

To ensure secure authentication and define different roles within our platform, we have designed two distinct authentication methods based on user types: regular users, administrators, and developers.

For regular users, we have chosen the OAuth protocol [26], which enables resource owners to authorise third-party access to their server resources without sharing their credentials. By utilising OAuth, our platform avoids the need to handle users' sensitive credentials directly, enhancing security.

In the case of developers and administrators, we have implemented our own authentication solution using JSON Web Token (JWT) [27]. JWT provides a compact and self-contained approach for securely transmitting information as a JSON object between parties. With JWT, we can handle user roles effectively and maintain secure communication.

To manage user information, we store user data in a document database, ensuring efficient retrieval and storage of user records.

These design choices were informed by our experience, industry best practices, and the guidance provided by Beanloop. We aimed to leverage established techniques and technologies while aligning with Beanloop's existing technology stack. By incorporating these authentication mechanisms, we can provide a secure and role-based access system within our platform.

Overall, the proposed architecture for the 3D configurator platform is a well-considered and robust system that was designed based on research and requirements collected from our partner company, Beanloop. Details of how the application architecture is being put into practice will be discussed in the next chapter, Implementation. In that section, we will present a prototype that has been developed iteratively in response to design decisions and requirements. By demonstrating the most important aspects of the design, we aim to showcase the capabilities of the application and provide insights into the development process.

5 Implementation

This chapter focuses on the development of our prototype, highlighting the iterative process described in Section 2.2. We will delve into the various iterations we went through, outlining the progression and improvements made along the way. Furthermore, we address the technical challenges encountered during development and present the corresponding solutions implemented to overcome them.

5.1 Prototype Development

For our prototype, we worked using an iterative approach and we deliberately made the decision to prioritise the client-side of the application as we considered it to be the most crucial component of the design for the application to work. The techniques of the client-side shown in the Figure 5.1 below is more explained in section 3.2.

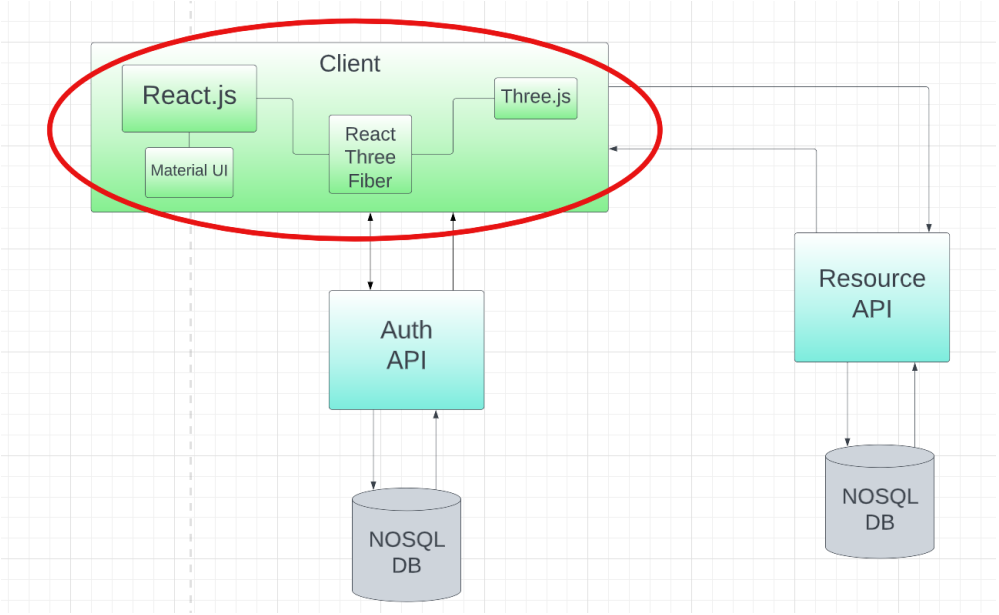

Figure 5.1 Prototype Architecture

To start the development process, we conducted a thorough analysis of the design that we created together with Beanloop. We identified some of the key requirements in the design for the client-side part of the application to work. Once we had a clear understanding of which requirements to start with, we began coding the prototype.

The client-side architecture is demonstrated in Figure 5.1, which showcases the techniques employed such as React, React Three Fiber, Three.js and Material UI. These technologies were chosen based on the vision and design

we formulated with Beanloop. Since Beanloop will eventually take over this project, it was necessary to incorporate technologies that are in line with their existing work process. Therefore, we were required to use the same techniques as Beanloop's preferred stack.

5.1.1 Iteration 1

The first iteration of the prototype was very basic, and only focused on the "configurator part" of the application. Meaning the part where the end-user gets to configure a 3D model.

In this iteration, a 3D model of a car was loaded and all parts of the model were identified and displayed in a list, the user then had the ability to change the colour of each different part of the 3D model.
Although this iteration was basic, it gave us insights into what was required when building a 3D-configurator. These insights are very valuable to us as we need to know what is required to create a 3D-configurator if we're to build a platform that can do it automatically.

Beanloop were pleased about the functionality to change colours and that the configurator were able to modify all the materials of the model. With this feedback we could ascertain that our first iteration was inline with their vision of the prototype, however they expressed concerns about whether we fully understood the bigger picture of the application at this point.They reminded us about their vision about creating a 3D configurator from a platform and select which materials that should be configurable.

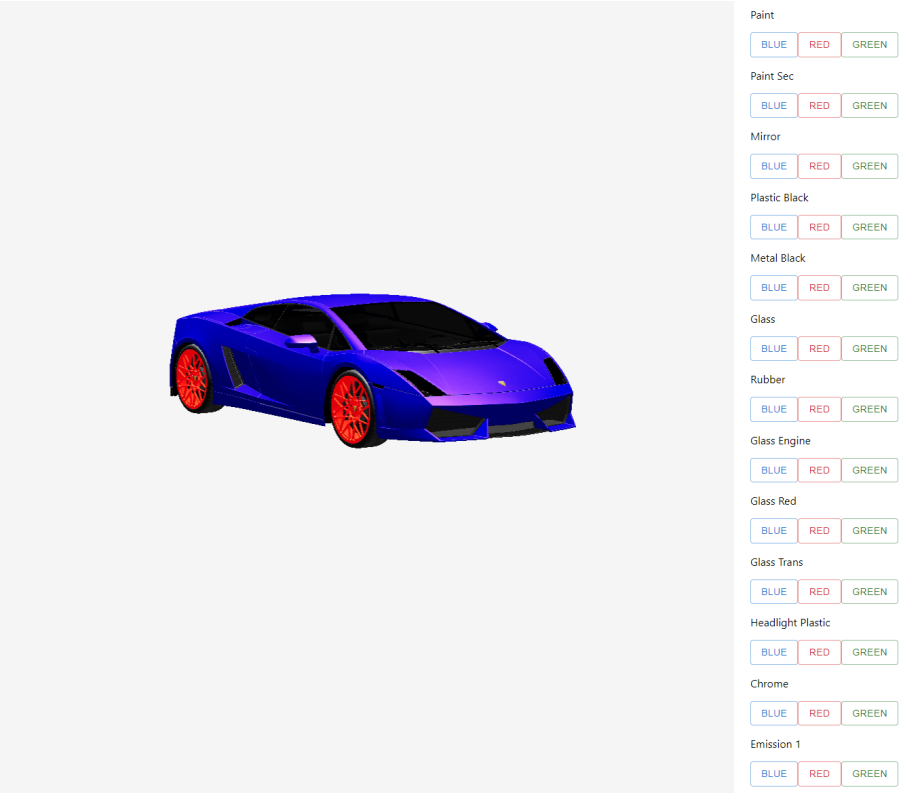

Figure 5.2 Iteration 1

5.1.2 Iteration 2

After collecting feedback from Beanloop and consulting with our supervisor, the main objective for Iteration 2 was to develop the initial version of the "platform" that would enable the creation of a basic configurator. In this iteration the user was first presented with the platform where they could select which parts of the 3D model should be configurable and then choose which colours should be assignable to those parts.

This iteration showed the beginning of the separated structure, with the platform serving as the foundation and the configurator being the final output. Beanloop expressed their positivity and appreciation for our understanding of their vision. They also emphasised the importance of being able to create a configurator with various steps in the future.

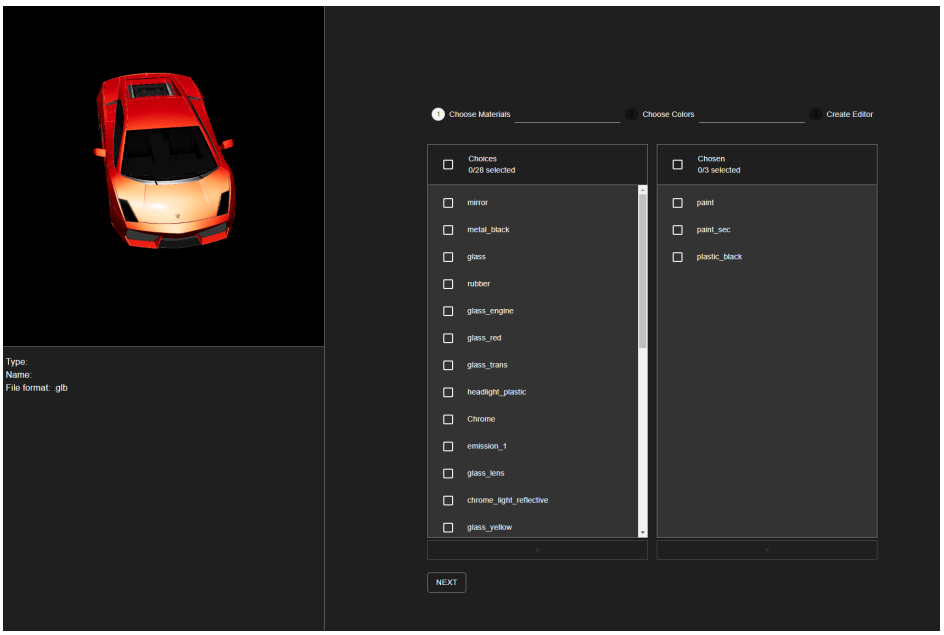

Figure 5.3 Iteration 2 Platform

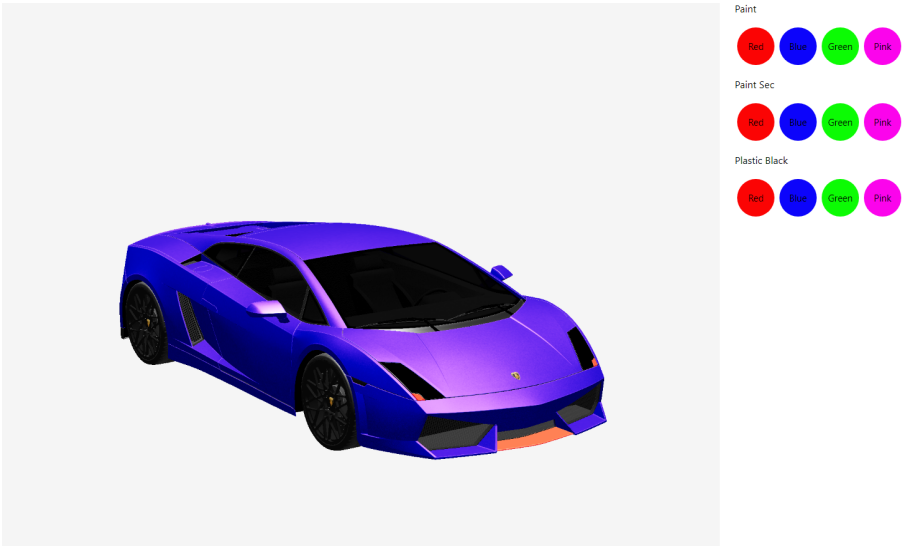

Figure 5.4 Iteration 2 Configurator

5.1.3 Iteration 3

The primary objective for iteration 3 was to enhance the functionality of our platform so it could create more advanced 3D configurators. To achieve this, we needed to expand the selection of features available on our platform. The first step we took was to implement the ability to upload 3D files in .glb format. This allowed users to upload any 3D model and create a configurator for it, which aligned with both ours and Beanloop's vision. Additionally, we

added an "Environment" feature that enabled users to upload 3D environments in .hdri format and then switch between them in the 3D configurator. Meaning you could put a car on a sunny street or place a chair in a living room. This feature is often seen in car 3D configurators, such as Audi's. While the changes in this iteration may have been small in terms of coding, they were significant in improving the overall functionality and user experience of the platform. Feedback from Beanloop was positive, and they acknowledged that the platform was starting to take shape as they envisioned. They were pleased with the new feature for upload of 3D files, which opens up the opportunity to develop a configurator for all 3D models in .glb format. While the ability to include an “Environment” was not initially part of Beanloop’s vision and the feedback was initially mixed, after further explanation of the potential benefits in certain scenarios like putting a car on a sunny street or a chair in a living room, they were positive to the idea.

5.1.4 Iteration 4

In the fourth and final iteration, we continued to refine and enhance the platform to better align with the vision design. We put a strong emphasis on implementing Beanloop's feedback from iteration 2, which highlighted the importance of being able to create a 3D configurator with steps. To accomplish this, we added a basic grouping feature, allowing users to group materials from the 3D model into categories. For instance, for a car, you could group all materials related to the tires in one category and all materials related to the interior in another. This provided a way to simulate "steps" in the configurator, in a basic manner.

In addition to the grouping feature, we also implemented a major new capability that enables the creation of selectable options that are displayed in the final 3D configurator. These options can be customised for different products and scenarios, such as "Rear passenger airbags" for a car or "Extra wheels" for a gaming chair.
To further enhance the realism of our platform, we created a "Final Configuration" view that mimics an order page. This view displays all the selections made by the user in the 3D configurator, allowing them to review their choices before finalising their configuration.

Overall, the final iteration was a success and we were able to deliver a functional platform that met some additional parts of the requirements and vision from Beanloop. They provided feedback regarding the importance of having the ability to configure prices. They also suggested that extra options or special colours should increase the price in order to enhance the realism of the 3D configurator.

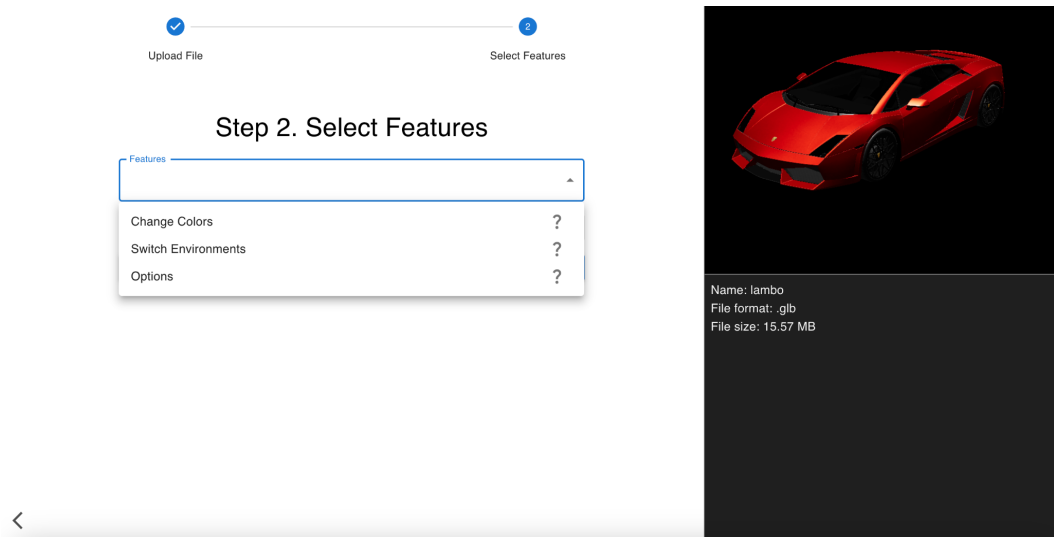

Figure 5.5 Iteration 4 Platform

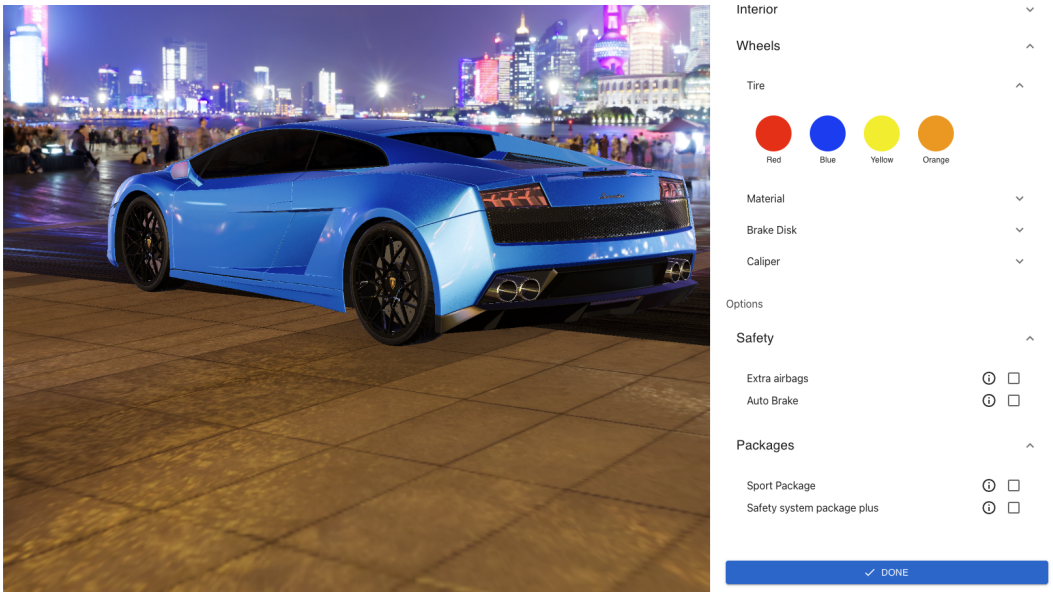

Figure 5.6 Iteration 4 Configurator

5.2 Technical Challenges and Solutions

In the development process, we encountered several technical challenges that we needed to overcome to achieve our goals. If you're looking to tackle a similar project, we recommend considering the following:

- Get familiar with any new technologies that will be used in the project. For example, learning Three.js and React Three Fiber was crucial to our success, but it required some time and effort.
- Prepare for potential setbacks with 3D models. In our case, Beanloop was unable to provide 3D models, which forced us to search for new models online. To avoid delays, consider preparing backup plans and alternative resources for any potential roadblocks. We recommend having good 3D models before you begin and with good models we mean that the model has several materials that are configurable.
- Write code that is generalisable, reusable and easy to maintain. With a complex project like this, it is important to prioritise code that can be adapted and updated for future developments. We were forced to start over from the beginning half way through due to the complexity and difficulty in maintaining our code, which resulted in a time-consuming process.

During our prototype development, we discovered a technical challenge related to the use of colours in the 3D configurator. Specifically, we found that Three.js, one of the key technologies used in our platform, does not handle the RGB format in the same way as it is typically used. In regular RGB, the colour red for example is represented as (255, 0, 0), whereas in Three.js it is represented as (1, 0, 0). This disparity resulted in the need for us to convert all colours to Three.js colours and divide the chosen colours RGB by 255 in order to make it work in the configurator. This was particularly relevant given that the "Choose Colour" feature was a key component of our prototype, and thus ensuring its functionality was crucial to the overall success of the prototype. To clarify how we manage to achieve this in the codebase a block of code is presented:

```
const convertColor = () => {
    const { r, g, b } = color
    return new THREE.Color(r / 255, g / 255, b / 255)
  }
```

Figure 5.7 Technical Challenge Example Part 1

We were able to resolve the issue obtaining the accurate colours while modifying the 3D model. However, to display the colour options available to the user in our User Interface (UI), we had to convert the colour back to its original format, so that the correct colour could be displayed.

```
<Box
  className="color-button"
  style={{backgroundColor:
  `rgb(${props.color.color.r * 255},
   ${props.color.color.g * 255},
   ${props.color.color.b * 255})`,
  }}
  onClick={handleClick}
/>
```

Figure 5.8 Technical Challenge Example Part 2

6 Testing and Results

This section includes the details of our test plan and the results of our evaluation of the usability and functionality of our 3D configurator creation platform prototype. Our objective was to recreate three different existing 3D configurators with varying levels of complexity using our platform, which served as benchmarks to assess the capabilities and limitations of our prototype. Throughout this section, we provide a description of our test cases and their outcomes, emphasising the strengths and limitations of our prototype. The testing will be performed as described in section 2.3 Final Evaluation.

6.1 Tests

We conducted a series of tests to evaluate the capabilities and limitations of our 3D configurator creation platform prototype. Our objective was to recreate three different existing 3D configurators with varying levels of complexity using our platform, which served as benchmarks for our tests. The three configurators used as benchmarks were:

1. Fjällräven's Kånken Me backpack 3D configurator [7] - representing the basic complexity level.
   https://www.fjallraven.com/se/sv-se/vaskor-utrustning/kanken/kanken-ryggsackar/kanken-me

2. Xbox Design Lab's controller 3D configurator [23] - representing the intermediate complexity level.
   https://xboxdesignlab.xbox.com/sv-se/configure/xbox-wireless-controller

3. Audi 's car 3D configurator [6] - representing the advanced complexity level.
   https://www.audi.se/se/web/sv/models.html

The 3D models used in the test were free models found on Sketchfab [29].

6.1.1 Test Case 1 - Basic Configurator

The first 3D configurator that we will try to recreate is Fjällräven's configurator for their popular "Kånken" backpack, which is created by Animech [22], it has a small set of features.

- **View controls:** The user can zoom in/out and rotate the 3D model to view it from different angles.

- **Change colour:** A list is displayed with all parts of the 3D model. When the user clicks on a material, a set of colours are displayed that

the user can switch between.

- **Change colour to group:** It is possible to change the colour of "all outer fabric parts" and "all straps and handles" at the same time.
- **Different colours to different materials:**Some colours are only available selectable for some of the parts of the model, for example the rainbow colour is only available for the straps and handles.
- **Shuffle:** A Shuffle feature that randomises the colour for all parts of the 3D model.
- **Undo Redo:** Undo your latest change or Redo your latest Undo.

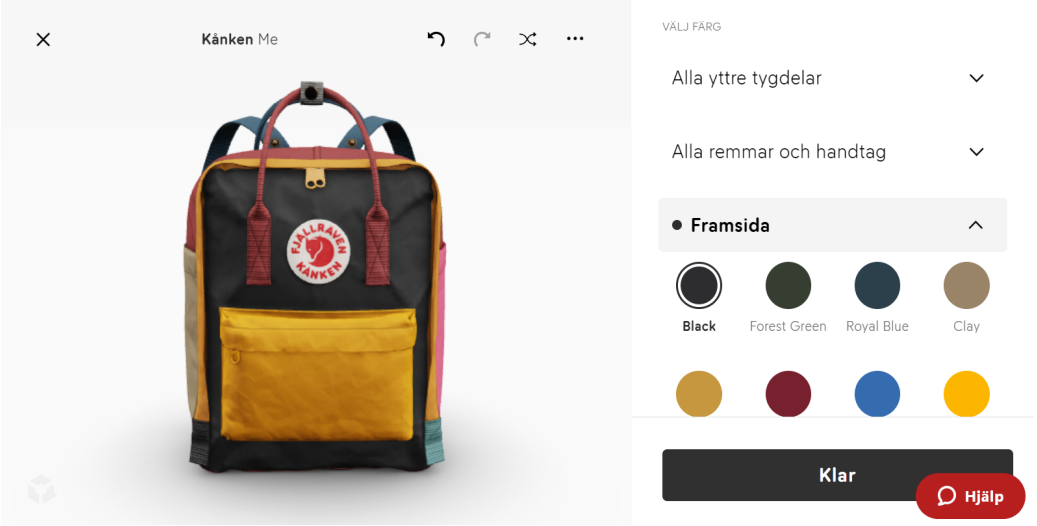

Figure 6.1 Fjällräven's Kånken Me® 3D Configurator

6.1.2 Test Case 2 - Intermediate Configurator

The second 3D configurator that we will try to recreate is Xbox's configurator of the controller for their console.

- **View controls:** The user can zoom in/out and rotate the 3D model to view it from different angles.
- **Change Colour:** The configurator includes a list of different parts of the 3D model that are editable and have the possibility to edit the colours of the different parts.
- **Different colours to different materials:** Some of the materials on the 3D model have specific colours that the rest of the parts do not

have.

- **Add options:** The ability to select options that add extra features to the 3D model. The figure below shows an example of adding "Rubberised grip on the back" as an option.

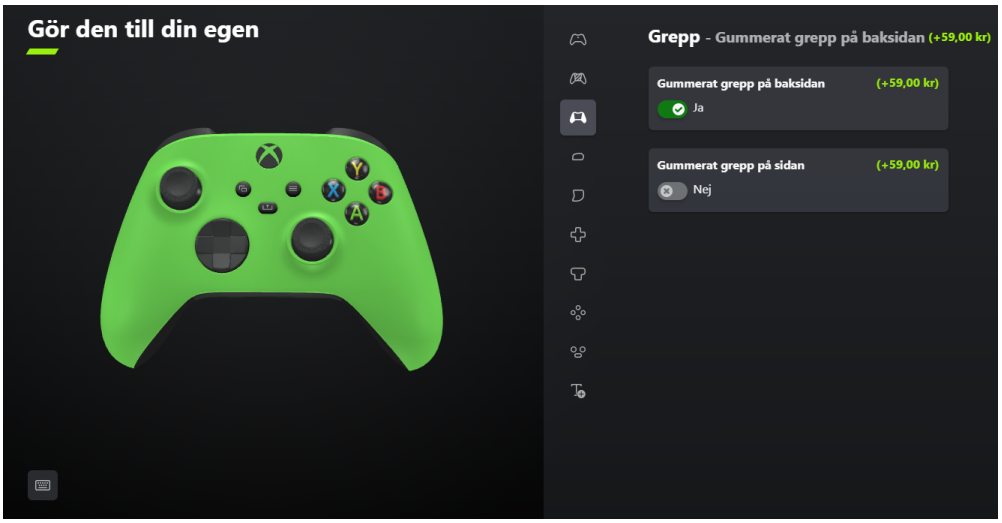

Figure 6.2 Xbox Design Lab®'s 3D Configurator - Different Options

6.1.3 Test Case 3 - Advanced Configurator

The third and final configurator we will try to recreate is Audi's 3D configurator.

- **View controls:** The user can zoom in/out and rotate the 3D model to view it from different angles.

- **Change colour:** The user can change the colour of different things like the exterior paint and interior seats.

- **Change parts:** The user can change parts of the 3D model, for example the rims, the front bumper or the steering wheel.

- **Environments:** The user can switch between different environments that makes it look like the car is standing in a real environment.

- **2D and 3D:** The user can view the car in 2D images and in 3D.

- **Steps:** The configuration process is divided into different steps like “packages”, “exterior”, “interior”.

- **Animations:** The user can trigger animations of the 3D model, like flashing headlights or opening the car doors.

- **Multiple 3D models for the same product (Interior/Exterior):** The Audi configurator offers multiple 3D models for the same product, enabling users to view and configure the car in both interior and exterior perspectives. These 3D models are always separated, allowing for a more detailed and accurate representation of the car's design and features.

- **Options:** The user gets to choose between different options and add-ons, some of the options also apply changes to the look of the 3D model. Some options are not compatible with each other and will automatically remove those

- **Presets:** The user gets to pick between different versions of a car like “Proline”, “Proline advanced”, and “S-Line”. These presets have unique starting appearances and options, but can then be configured further.

- **Popular choices:** A list with popular choices.

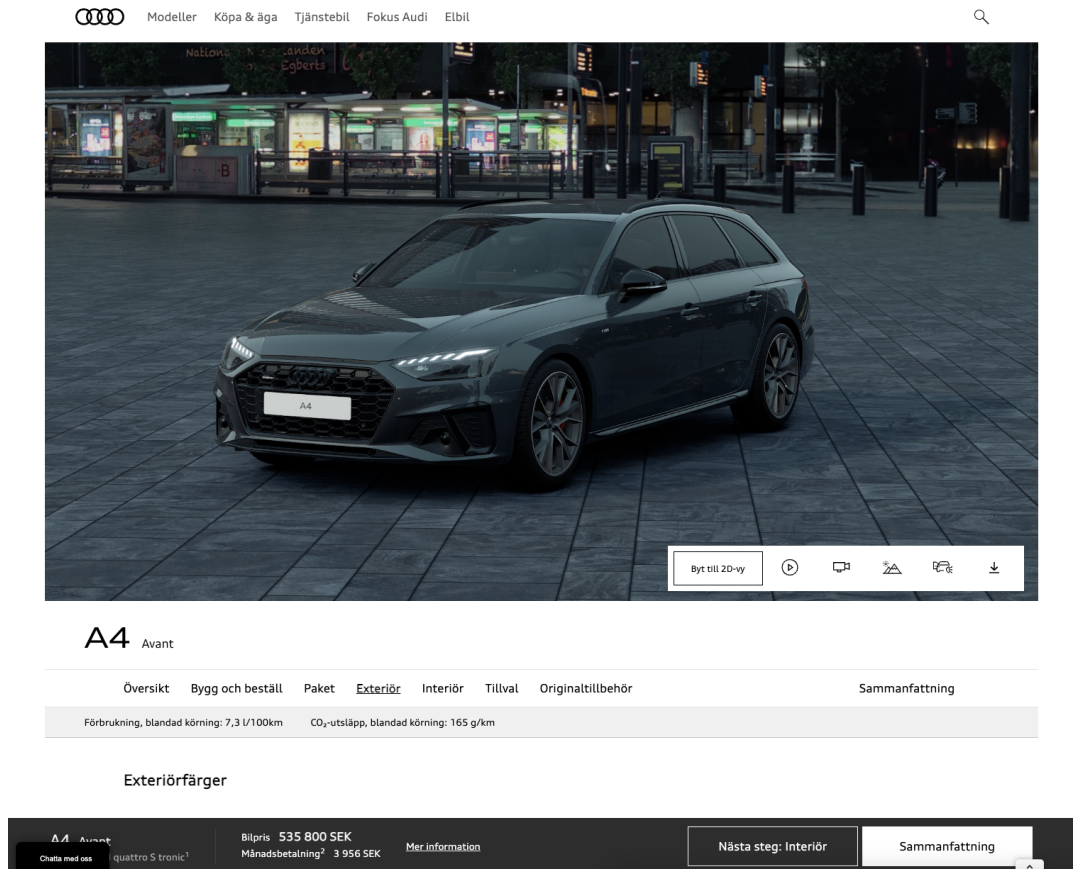

Figure 6.3 Audi®'s 3D Configurator

6.2 Test Results

This section will present the results obtained from our tests and provide insights into their implementation.

6.2.1 Results Test Case 1

As seen in figure 6.7, using our prototype we were able to successfully recreate 6 of the 8 features we identified in Fjällräven's Kånken Me 3D configurator. By uploading a 3D model of a backpack and utilising our "Change colour" feature, we were able to specify which colours could be assigned to each material. This allowed us to create a similar configurator to Fjällräven, where the user can change the colour of each part of the 3D model, and one extra colour is available for a specific material, just like how the rainbow colour is only available for the straps and handles in Fjällräven's configurator. However, there were still some features that we were not able to recreate, such as changing the colour of a whole group of materials simultaneously or the Shuffle feature.

It took roughly 3 minutes to create this 3D product configurator using our prototype.

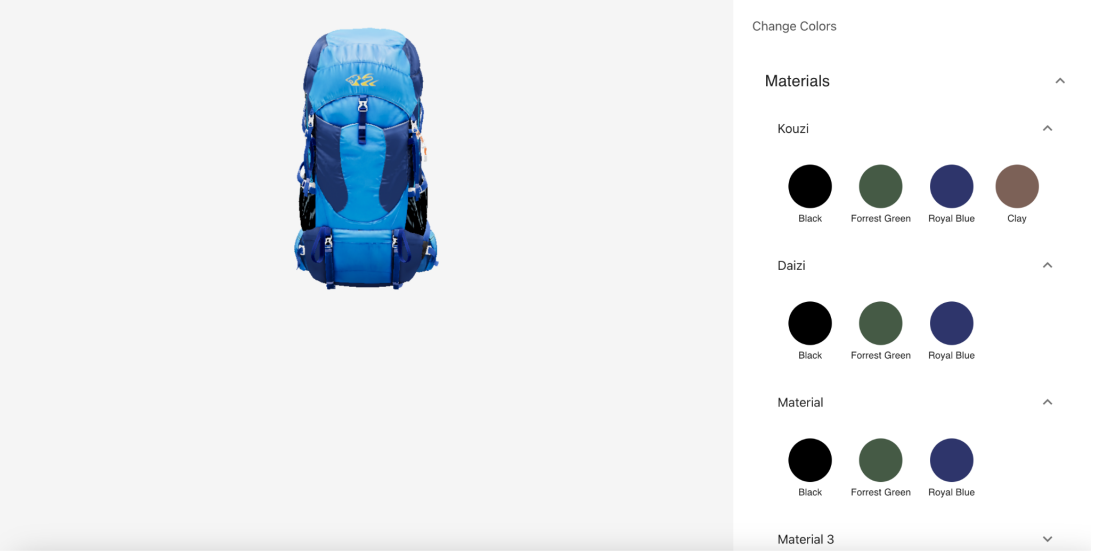

Figure 6.4 Test Case 1 - Screenshot from our prototype

6.2.2 Results Test Case 2

As seen in figure 6.7, using our prototype we were able to successfully recreate 7 of the 10 features that we identified in Microsofts Xbox Design Lab 3D configurator for their Xbox controllers. By uploading a 3D model of an Xbox controller and selecting the "Change colour" feature, we were able to assign different colours to specific parts of the controller, just like in the original configurator.

Using our prototype's feature "Select Options" we could also recreate the extra options part, where the user is asked if they want to have a rubber grip on their controller. In Xbox's configurator this also affects the appearance of the 3D model, this is not the case in ours however there are many cases

when extra options like this doesn't affect the 3D model and is therefore still a valuable feature.

It took roughly 5 minutes to create this 3D product configurator using our prototype.

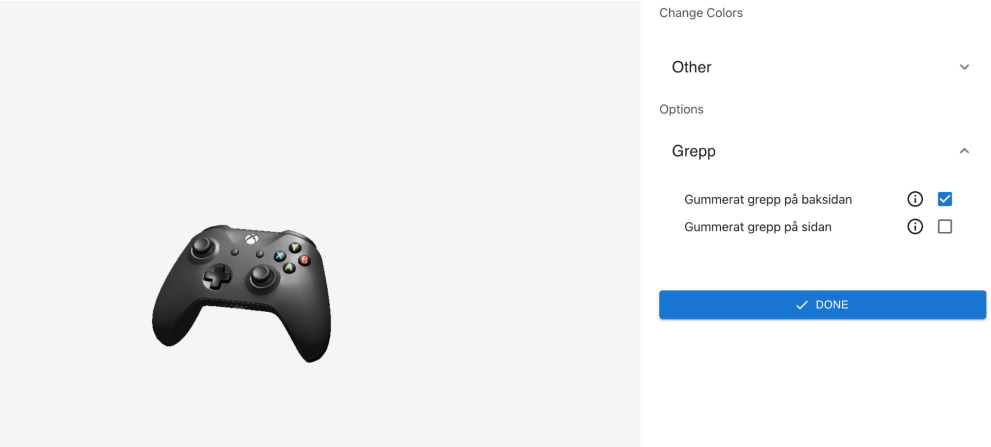

Figure 6.5 Test Case 2 - Screenshot from our prototype

6.2.3 Results Test Case 3

As seen in figure 6.7, using our prototype we were only able to recreate 8 of the 16 features that we identified in Audi's 3D configurator. Although it was a challenging task to fully recreate Audi's 3D configurator through our platform prototype, we were able to create a basic version of it by utilising all the features that we implemented. One of the main features we were able to mimic was the use of steps like "interior" and "exterior" in Audi's configurator. We achieved this by dividing materials into different groups in our "Change colours" feature, even though this does not generate clear steps like in Audi's configurator where the user has to click "next" to proceed to the next step. However, our approach is still a very similar functionality that could be easily modified to work in that way.

One feature that our prototype supports that is a big feature in Audi's configurator is switching between 3D environments where the car is placed. Using our "Switch Environments" feature that we implemented, we could upload three different 3D environments in .hdri format and switch between

them as seen in the figure below. We could also recreate the options functionality that is frequently used in Audi's configurator. These functionalities together makes a pretty good car 3D configurator but is still far behind Audi's version, as it's filled with many different complex features.

It took roughly 8 minutes to create this 3D product configurator using our prototype.

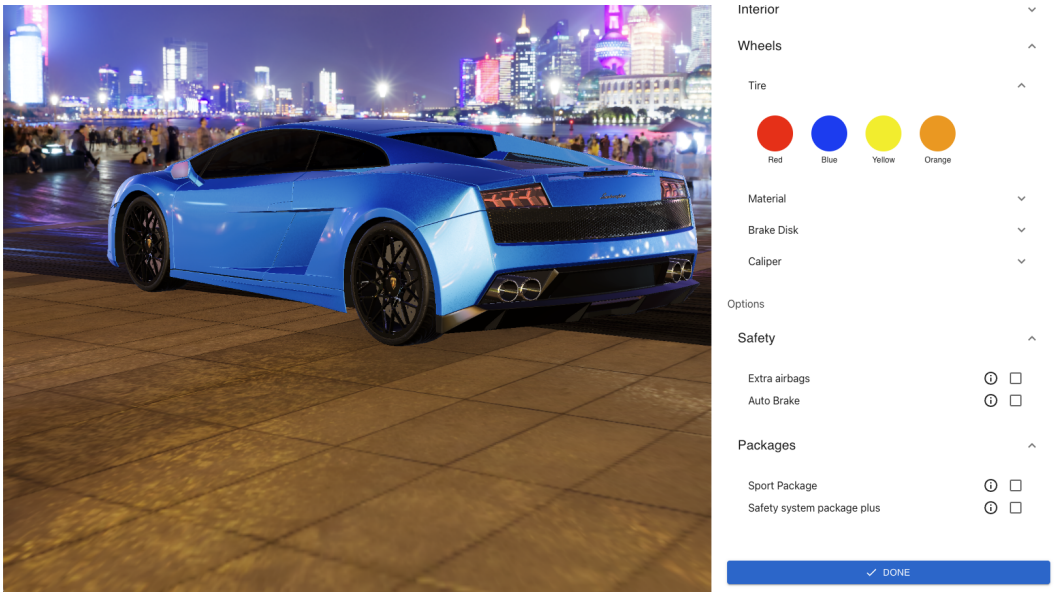

Figure 6.6 Test Case 3 - Screenshot from our prototype

In the figure below, we have summarised all the features that we could identify in the different 3D configurators. As can be seen from the table, we were able to successfully recreate most of the features found in the basic and intermediate configurators, but we faced challenges in recreating the advanced configurator as it had many complex features that were not supported by our prototype. However, our vision for the system design includes all of these advanced features, and we believe that our design would be able to recreate them successfully.

Features	Audi	Xbox	Fjällräven	Our Prototype	Our Design
View Controls					
2D					
3D					
Change Color					
Change Color to group					
Individual Colors					
Change Parts of 3D Model					
Options					
Environments					
Groups					
Steps					
Animations					
Multiple 3D models					
Presets					
Popular Choices					
Randomize					
Summary					

Figure 6.7 Summary of 3D Configurator Features

7 Discussion

In this section we discuss our findings of our study in relation to our research questions, we also reflect on the validity and reliability of our study.

- **RQ1:** Can a no-code solution for creating 3D product configurators generate configurators of equivalent quality compared to those that are programmed from scratch, hence saving time and resources?

- **RQ2:** How can a platform for creating 3D configurators be designed to accommodate the specific requirements for different types of 3D product configurators, such as accessories or cars?

Our study aimed to investigate whether it is possible for a platform dedicated to creating 3D product configurators to produce configurators of equivalent quality compared to solutions that are developed from scratch, while also potentially saving time and resources. When testing our prototype we managed to almost fully recreate Fjällräven's Kånken Me 3D configurator, and we could successfully recreate some additional features from the two other test case configurators. To further explore the concept of "equivalent quality" in our research question, our focus has been on the extent to which we can recreate configurators with similar features and available options compared to custom-built solutions. We have placed emphasis on evaluating the platform's ability to provide a comprehensive range of functionalities that align with or exceed the standards established by bespoke solutions. This assessment involves examining the range of customization options it offers.

One key advantage of using a platform for creating 3D configurators is the potential for time and resource savings. Developing a 3D configurator from scratch or using one of the existing platforms on the market, mentioned in related work, can be a time-consuming and expensive process. Using a platform for creating them instead without a long process, can significantly reduce the time and resources required to develop a 3D configurator.

Our testing shows that using our prototype of the platform for creating a 3D configurator reduces the time and resources significantly as we created all three configurators in under 10 minutes each. However, it is important to note that our familiarity with the platform may have influenced our efficiency. For a person unfamiliar with the platform, it may take longer to create the configurators. Nonetheless, the time and resource savings offered by a platform make it a promising option for businesses looking to develop 3D configurators.

In comparison to the related work discussed earlier, our proposed solution stands out by focusing on accessibility and self-sufficiency for businesses in creating 3D product configurators. While existing platforms offer valuable features and capabilities, like having the ability to help the customer to create 3D models for their products, they all require a certain level of technical skill and knowledge in the 3D configurator creation process. Our design aims to bridge this gap by providing a user-friendly platform that empowers businesses without programming expertise to effortlessly create their own configurators.

There are several opportunities to improve the validity and reliability of different aspects in our study. For example, to improve the validity of our test results and their truthfulness. It would have been better to randomly select three 3D configurators to conduct the testing with. As that would have removed the risk that we may have unconsciously selected 3D configurators that we knew that our platform would be fitted to recreate. Furthermore, considering that we extensively examined Audi's and Fjällräven's 3D configurators in the introduction part of our thesis, it might have been beneficial to exclude them from the testing phase. This would have reduced the potential bias that could have influenced us to tailor our platform specifically to replicate those two 3D configurators. Instead replacing Audi's and Fjällrävens's configurators during the testing phase with configurators from other brands within the same industries would have resulted in more objective and accurate test results.
To improve the validity in regards to our platform's effectiveness. A larger sample size of existing 3D configurators in the testing process could have been used. Also by including participants who are unfamiliar with the prototype in the testing process, would have increased the reliability of our test results.

The implications of our results extend to both the society and the target audience. By providing businesses with a user-friendly platform to create 3D product configurators, our solution has the potential to empower a broader range of businesses. This can lead to increased innovation, enhanced customer experiences, and potentially boost the adoption of 3D configurators in various industries. Additionally, the time and resource-saving aspects of our design can positively impact businesses by reducing their reliance on external providers and allowing them to have greater control over the configurator creation process.

Our study also aimed to investigate how a platform for creating 3D configurators can be designed to accommodate the specific requirements for different types of 3D product configurators. Our findings suggest that to design a platform that accommodates specific requirements for different types of products, it is important to consider the following factors:

- **Product complexity:** The complexity of a product can significantly impact the design of a configurator. Complex products, such as cars, may require a more advanced platform with additional features and customization options to address their specific requirements. On the other hand, simpler products may require a more straightforward and user-friendly platform.
- **Industry requirements:** Different industries may have unique requirements for their product configurators. For example, the automotive industry may require more advanced configurators than the fashion industry. Therefore, it is important to consider the specific requirements of different industries when designing a platform for creating 3D configurators.
- **Flexibility:** A platform for creating 3D configurators should be flexible enough to accommodate different types of products and industries. This can be achieved by designing the platform with modularity and scalability in mind, allowing for customization and expansion as needed.

In conclusion, our study has shown that a platform for creating 3D configurators can produce high-quality configurators and save time and resources, making it a promising option for businesses looking to develop product configurators. However, it is important to consider the limitations of a platform when it comes to accommodating specific requirements for complex products. Our findings suggest that a platform's design should take into consideration factors such as product complexity, industry requirements, and flexibility to create a platform that accommodates different types of products and industries. The platform should be designed with modularity and scalability in mind, allowing for customization and expansion as needed. Overall, our study highlights the potential benefits of implementing a platform for creating 3D configurators. It emphasises the importance of designing a platform that streamlines the configurator creation process and actively involves the customer. By reducing the back-and-forth communication and enabling businesses to take a more active role in the platform, we have uncovered opportunities for enhanced efficiency and customer engagement.

8 Conclusion and Future Work

In this study, we investigated the capabilities and limitations of a platform for creating 3D product configurators and addressed the following research questions.

- **RQ1:** Is it possible for a platform dedicated to creating 3D product configurators to produce configurators of equivalent quality compared to solutions developed partially or from scratch, while also potentially saving time and resources?

- **RQ2:** How can a platform for creating 3D configurators be designed to accommodate the specific requirements for different types of 3D product configurators, such as accessories or cars?

Our findings indicate that a platform for creating 3D configurators can indeed produce configurators of equivalent quality as solutions developed partially or from scratch, particularly for less complex 3D configurators. Our findings also indicated that a platform offers the advantage of significant time and resource savings.

Regarding our second research question, our findings indicate that to accommodate specific requirements for different types of configurators, the design of the platform should consider factors such as product complexity, industry requirements, and flexibility.

In conclusion, one notable advantage of using a platform for creating 3D configurators is the potential for time and resource savings. By enabling customers to create their own configurators, the platform streamlines the configurator creation process, saving time and resources for both the solution provider and customers. However, it is essential to recognize the limitations of a platform when it comes to addressing the specific requirements of complex products.

8.1 Future Directions

While our study demonstrates the potential of using a platform for creating 3D configurators, there are several areas that warrant further exploration.

Firstly, future research should focus on assessing the scalability of this type of platform. Our study was conducted on a small scale, and it remains unclear how well the platform would perform when dealing with a large number of users. It would be valuable to investigate whether the platform

can handle high volumes of traffic and how it impacts the speed and performance of the configurator.

Secondly, further research could delve into the integration options for incorporating the platform into existing e-commerce sites. Although this study did not investigate integration possibilities, it is important to explore different approaches and identify the best option for seamless integration.

Moreover, the effectiveness of platforms for creating 3D configurators in different industries should be explored. While our study primarily focused on the fashion and automotive industries, other sectors like real estate, furniture, and construction may have unique configurator requirements. Future research could investigate how well the platform can accommodate these specific needs and provide a detailed assessment of its effectiveness in diverse industries.

Lastly, future research could investigate the impact of using 3D configurators on customer engagement and satisfaction. While there is evidence suggesting that 3D configurators enhance customer engagement, limited research exists on their effectiveness across various industries. Further investigation can shed light on the impact of 3D configurations on customer satisfaction and their influence on purchasing decisions.

In conclusion, our study provides valuable insights into the potential of using a platform for creating 3D configurators. However, there are still several areas that require further exploration to fully understand the effectiveness and scalability of these platforms.

References

[1] Statistics Sweden, "ICT usage in households and by individuals 2021-E-commerce", 2021. Available online at: https://www.scb.se/publikation/42852 Accessed: [24-March-2023]

[2] U.S. Department of Commerce, "Quarterly Retail E-Commerce Sales", *U.S. Census Bureau News, Washington, D.C. 2023* https://www.census.gov/retail/ecommerce.html Accessed: [24-March-2023]

[3] P. Johannesson and E. Perjons, "An Introduction to Design Science" *Springer, Heidelberg, 2021.*

[4] K. Peffers, T. Tuunanen and M. Rothenberger "A Design Science Research Methodology for Information Systems Research", *Journal of Management Information Systems, Volume 24, Issue 3, Number 3*, M. E. Shape, Inc 2007. Available online at: https://www.tandfonline.com/doi/abs/10.2753/MIS0742-1222240302. Accessed: [23-March-2023]

[5] S. Hewawalpita and I. Perera, "Effect of 3D product presentation on consumer preference in e-commerce", *Moratuwa Engineering Research Conference (MERCon)*, Moratuwa, Sri Lanka, 2017. Available online at: https://ieeexplore.ieee.org/abstract/document/7980532 Accessed: [07-March-2023]

[6] Audi. (n.d.). Audi website
Available online at: https://www.audi.com/en.html
Accessed: [6-March-2023]

[7] Kånken Me. (n.d.). Fjällräven website.
Available on at:
https://www.fjallraven.com/se/sv-se/vaskor-utrustning/kanken/kanken-ryggsackar/kanken-me
Accessed: [6-March-2023]

[8] WordPress. (n.d.). WordPress website.
Available online at: https://wordpress.com/
Accessed: [6-March-2023]

[9] Shopify. (n.d.). Shopify website.
Available online at: https://www.shopify.com/website/builder

[10] P. Blazek and K. Pilsl. "Learnings from setting up product configurator

projects". *Annals of the Faculty of Engineering Hunedoara*, 2017, 15.1: 25.
Available online at:
https://annals.fih.upt.ro/pdf-full/2017/ANNALS-2017-1-02.pdf
Accessed: [24-March-2023]

[11] cyLEDGE Media, "Configurator Database", Vienna, 2023.
Available online at: https://www.configurator-database.com/
Accessed: [24-March-2023]

[12] A. Massaro, V. Vitti, V. Mustich and A. Galiano. "Intelligent real-time 3D configuration platform for customising e-commerce products", *Int. J. Comp. Grsph. Animat.(IJCGA)*, Bari 2019.
Available online at: https://aircconline.com/ijcga/V9N4/9419ijcga02.pdf
Accessed: [24-March-2023]

[13] H. Hwangbo, E. H. Kim, S. -H. Lee and Y. J. Jang. "Effects of 3D Virtual "Try-On" on Online Sales and Customers' Purchasing Experiences," in IEEE Access, vol. 8, pp. 189479-189489, 2020, doi: 10.1109/ACCESS.2020.3023040.
Available online at: https://ieeexplore.ieee.org/document/9189849
Accessed: [24-March-2023]

[14] R. Rolland, E. Yvain, O. Christmann, E. Loup-Escande and S. Richir. 2012. "E-commerce and web 3D for involving the customer in the design process: the case of a gates 3D configurator". In Proceedings of the 2012 Virtual Reality International Conference (VRIC '12). Association for Computing Machinery, New York, NY, USA, Article 25, 1–8.
Available online at: https://doi.org/10.1145/2331714.2331743
Accessed: [25-March-2023]

[15] React. (n.d.). React website.
Available online at: https://react.dev/
Accessed: [5-April-2023]

[16] Three.js. (n.d.). Three.js website.
Available online at: https://threejs.org/
Accessed: [5-April-2023]

[17] Lewy Blue. "Discover three.js".
Available online at: https://discoverthreejs.com/
Accessed: [5-April-2023]

[18] Pmdrs.docs. "React Three Fiber".
Available online at:

https://docs.pmnd.rs/react-three-fiber/getting-started/introduction
Accessed: [5-April-2023]

[19] Material UI. (n.d.). Material UI website.
Available online at: https://mui.com/material-ui/getting-started/overview/
Accessed: [5-April-2023]

[20] Dopple. (n.d.). Dopple website.
Available online at: https://www.dopple.io/3d-configurator
Accessed: [9-May-2023]

[21] Vividworks. (n.d.). Vividworks website.
https://www.vividworks.com/solutions/3d-online-configurator
Accessed: [9-May-2023]

[22] Animech. (n.d.). Animech website.
Available online at: https://www.animech.com/sv/ani-config
Accessed: [9-May-2023]

[23] Xbox Design Lab. (n.d.). Xbox Design Lab 3D Configurator.
Available online at:
https://xboxdesignlab.xbox.com/sv-se/configure/xbox-wireless-controller
Accessed: [10-May-2023]

[24] Bootstrap. (n.d.). Bootstrap website.
Available online at: https://getbootstrap.com/
Accessed: [11-May-2023]

[25] Ant Design. (n.d.). Ant Design website.
Available online at: https://ant.design/
Accessed: [11-May-2023]

[26] OAuth. (n.d.) OAuth website.
Available online at: https://oauth.net/2/
Accessed: [17-May-2023]

[27] JWT. (n.d.) JWT website.
Available online at: https://jwt.io/
Accessed: [17-May-2023]

[28] MongoDB. (n.d.) MongoDB website.
Available online at: https://www.mongodb.com/
Accessed: [17-May-2023]

[29] Sketchfab. (n.d.) Sketchfab website.
Available online at: https://sketchfab.com/feed
Accessed: [17-May-2023]

Appendices

Appendix A: Prototype Code Examples

A.1 Rendering 3D Model

```
import { Environment, OrbitControls, useGLTF } from
'@react-three/drei'
import { Canvas } from '@react-three/fiber'
import React, { useRef } from 'react'
import * as THREE from 'three'

export const Model = (props) => {
 const myMesh = useRef()

 const MyAnimatedBox = () => {
   return (
     <mesh ref={myMesh}>
       <Object model={props.model}
setMaterials={props.setMaterials} rotation={[0, Math.PI / 1.5,
0]} scale={5} />
     </mesh>
   );
 }

 return (
   <Canvas gl={{ logarithmicDepthBuffer: true, antialias: false
}} dpr={[10, 1.5]} camera={{ position: [0, 0, 8], fov: 25 }}>
     {props.environment === undefined ? (
       <>
         <directionalLight intensity={3} />
         <ambientLight intensity={1} />
       </>
     ) : (
       <>
         <Environment ground={{
           height: 10, // Height of the camera that was used to
create the env map (Default: 15)
           radius: 60, // Radius of the world. (Default 60)
           scale: 900, // Scale of the backside projected
sphere that holds the env texture (Default: 1000)
         }} files={props.environment} />
       </>
     )}
     <MyAnimatedBox />
     <OrbitControls minDistance={30} maxDistance={400}
enablePan={false} minPolarAngle={Math.PI / 3.2}
maxPolarAngle={Math.PI / 2.2} />
   </Canvas>
 )
```

```
}

function Object(props) {
 const { scene } = useGLTF(props.model?.url);
 const group = useRef();

 const root = new THREE.Object3D();
 root.add(scene);
 group.current = root;

 const materials = {
   materials: []
 }

 let parts = []

 group.current.traverse(obj => {
   if (obj.material) {
     materials.materials.push(obj.material)
   }
 });
 parts = new Set(materials.materials)
 const test = Array.from(parts)
 props.setMaterials(test)

 return (
   <group ref={group} {...props}>
     <primitive object={scene} />
   </group>
 );
}
```

A.2 Selecting Features

```
import React, { useState } from 'react';
import {
 TextField, MenuItem, Button, Grid, Typography, Dialog,
 DialogTitle,
 DialogContent,
 DialogActions,
 List,
 ListItem,
 ListItemText,
 Tooltip,
 Snackbar,
 Alert,
 IconButton,
 ListItemSecondaryAction
} from '@mui/material';
```

```
import './/CreateStepper.css'
import { QuestionMark } from '@mui/icons-material';

const CreateStepper = (props) => {
 const [component, setComponent] = useState({})
 const [openModal, setOpenModal] = useState(false)
 const [showAlert, setShowAlert] = useState(false)
 const [alertText, setAlertText] = useState({ text: '', type:
'' })

 const handleComponentChange = (event) => {
   const value = event.target.value;
   if (value === 'Select Colors') {
     setComponent({ component: 'Select Colors', needMaterial:
true });
   } else {
     setComponent({ component: value });
   }
 };

 const handleAddSteps = (event) => {
   event.preventDefault()
   if (component.needMaterial) {
     props.setSteps((prevSteps) => [...prevSteps, { component:
'Materials' }])
   }
   props.setSteps((prevSteps) => [...prevSteps, { component:
component.component }])
   setAlertText({ text: 'Successfully added the feature!',
type: 'success' })
   setShowAlert(true)
   setTimeout(() => {
     setShowAlert(false);
   }, 3000)
   setComponent({})
 }

 const handleDone = () => {
   setOpenModal(true)
 };

 const handleModalClose = () => {
   setOpenModal(false)
 };

 const handleModalConfirm = () => {
   setOpenModal(false)
   props.setStepsDone(true)
   // handle what happens after the user confirms they're done
 }
```

```
 return (
   <>
     <Grid container spacing={2} justifyContent="center"
alignItems='center'>
       <Grid item xs={8}>
         <TextField
           select
           label="Features"
           value={component.component || ''}
           onChange={handleComponentChange}
           variant="outlined"
           fullWidth
         >

           <MenuItem value='Select Colors'>
             <ListItemText primary="Change Colors" />
             <ListItemSecondaryAction>
               <Tooltip title={<p style={{ fontSize: 14,
margin: 0 }}>In "Select Colors" you can specify the colors that
the user should be assign to the different components of the 3D
model.</p>}>
                 <IconButton>
                   <QuestionMark />
                 </IconButton>
               </Tooltip>
             </ListItemSecondaryAction>
           </MenuItem>
           <MenuItem value='Environments'>
             <ListItemText primary="Switch Environments" />
             <ListItemSecondaryAction>
               <Tooltip title={<p style={{ fontSize: 14,
margin: 0 }}>In "Switch Environments" you can specify the
background enviroments that the user should be able to
use.</p>}>
                 <IconButton>
                   <QuestionMark />
                 </IconButton>
               </Tooltip>
             </ListItemSecondaryAction>
           </MenuItem>
           <MenuItem value='Options'>
             <ListItemText primary="Options" />
             <ListItemSecondaryAction>
               <Tooltip title={<p style={{ fontSize: 14,
margin: 0 }}>In "Options" you specify options that the user can
select from.</p>}>
                 <IconButton>
                   <QuestionMark />
                 </IconButton>
```

```
              </Tooltip>
            </ListItemSecondaryAction>
          </MenuItem>
        </TextField>
        {showAlert &&
          <Snackbar sx={{ marginBottom: 10 }} open={showAlert}
onClose={() => setShowAlert(false)} autoHideDuration={3000}>
            <Alert
severity={alertText.type}>{alertText.text}</Alert>
          </Snackbar>
        }
      </Grid>
      <Grid item xs={8}>
        <Button variant="contained" color="primary"
onClick={handleAddSteps} fullWidth
disabled={!component.component}>
          Add Feature
        </Button>
      </Grid>
      <Grid item xs={8}>
        <Button variant="contained" color="primary"
onClick={handleDone} fullWidth className='doneButton'>
          Continue
        </Button>
      </Grid>
      <Dialog open={openModal} onClose={handleModalClose}>
        <DialogTitle>Are you done adding
features?</DialogTitle>
        <DialogContent>
          <Typography>
            If you're done adding features, click "Confirm".
Otherwise, click
            "Cancel" to continue adding features.
          </Typography>
          <List>
            {props.steps.map((step, index) => (
              <>
                {step.component === 'Materials' ? (
                  <>
                  </>
                ) : (
                  <ListItem key={index} divider>
                    <ListItemText primary={step.component} />
                  </ListItem>
                )}
              </>
            ))}
          </List>
        </DialogContent>
        <DialogActions>
```

```
            <Button onClick={handleModalClose}
color="secondary">
              Cancel
            </Button>
            <Button onClick={handleModalConfirm}
color="secondary" autoFocus>
              Confirm
            </Button>
          </DialogActions>
        </Dialog>
      </Grid>
    </>
  )
}

export default CreateStepper
```

A.3 Selecting Configurable Materials

```
import { useState } from 'react'
import Grid from '@mui/material/Grid'
import List from '@mui/material/List'
import ListItem from '@mui/material/ListItem'
import ListItemText from '@mui/material/ListItemText'
import ListItemIcon from '@mui/material/ListItemIcon'
import Checkbox from '@mui/material/Checkbox'
import Button from '@mui/material/Button'
import { Alert, Box, Dialog, DialogContent, Divider, Snackbar,
TextField, Typography } from '@mui/material'

function not(a, b) {
 return a.filter((value) => b.indexOf(value) === -1)
}

function intersection(a, b) {
 return a.filter((value) => b.indexOf(value) !== -1)
}

function union(a, b) {
 return [...a, ...not(b, a)]
}

export const MaterialsPicker = (props) => {
 const [checked, setChecked] = useState([])
 const [showAlert, setShowAlert] = useState(false)
 const [alertMessage, setAlertMessage] = useState('')
 const [showGroup, setShowGroup] = useState(false)
 const [groupName, setGroupName] = useState('');
```

```
 const handleToggle = (value) => () => {
   const currentIndex = checked.indexOf(value)
   const newChecked = [...checked]

   if (currentIndex === -1) {
     newChecked.push(value)
   } else {
     newChecked.splice(currentIndex, 1)
   }
   setChecked(newChecked)
 }

 const handleGroupNameChange = (event) => {
   setGroupName(event.target.value);
 };

 const numberOfChecked = (items) => intersection(checked,
items).length


 const handleToggleAll = (items) => () => {
   if (numberOfChecked(items) === items.length) {
     setChecked(not(checked, items))
   } else {
     setChecked(union(checked, items))
   }
 }

 const customList = (title, items) => (
   <>
     <Box display={'flex'} justifyContent={'center'}>
       <Typography variant="h4">{title}</Typography>
     </Box>
     <Box marginTop={5} display='flex'>
       <Checkbox
         onClick={handleToggleAll(items)}
         checked={numberOfChecked(items) === items.length &&
items.length !== 0}
         indeterminate={
           numberOfChecked(items) !== items.length &&
numberOfChecked(items) !== 0
         }
         disabled={items.length === 0}
         inputProps={{
           'aria-label': 'all items selected',
         }}
       />
       <p>Select all</p>
     </Box>
```

```
    <List
      sx={{
        width: '800px',
        height: '50vh',
        overflow: 'auto',

      }}
      dense
      component="div"
      role="list"
    >
      {items.map((value) => {
        const labelId =
`transfer-list-all-item-${value}-label`

        return (
          <>
            <ListItem
              key={value}
              role="listitem"
              onClick={handleToggle(value)}
            >
              <ListItemIcon >
                <Checkbox
                  sx={{ color: '#f5f5f5' }}
                  checked={checked.indexOf(value) !== -1}
                  tabIndex={-1}
                  disableRipple
                  inputProps={{
                    'aria-labelledby': labelId,
                  }}
                />
              </ListItemIcon>
              <ListItemText id={labelId} primary={value.name}
/>
            </ListItem>
            <Divider />
          </>

        );
      })}
    </List>
  </>
);

const handleNext = () => {
  props.setChosenMaterials(prevState =>
    [...prevState,
    {
      name: 'Other',
```

```
        values: checked,
        group: true
      }])
    setShowAlert(true)
    const checkedValues = checked.map((value) =>
value.name).join(", ")
    setAlertMessage(`Added: ${checkedValues}`)
    setChecked([])
  }

  const handleGroup = () => {
    props.setChosenMaterials(prevState =>
      [...prevState,
      {
        name: groupName,
        values: checked,
        group: true
      }])
    setShowAlert(true)
    const checkedValues = checked.map((value) =>
value.name).join(", ")
    setAlertMessage(`Added: ${checkedValues} to ${groupName}`)
    setGroupName('')
    setShowGroup(false)
    setChecked([])
  }

  return (
    <Grid container spacing={2} justifyContent="center"
alignItems="center">
      <Box item>{customList('Select configurable materials',
props.materials)}
        <Button
          sx={{ my: 0.5, width: '100%', borderRadius: 0 }}
          variant="outlined"
          size=""
          onClick={handleNext}
          disabled={checked.length === 0}
          aria-label="Add"
        >
          Add Selections
        </Button>
        <Button
          sx={{ my: 0.5, width: '100%', borderRadius: 0 }}
          variant="outlined"
          size=""
          onClick={() => setShowGroup(true)}
          disabled={checked.length === 0}
          aria-label="Add"
        >
```

```
          Add selections to a group
        </Button>
      </Box>
      {showGroup ? (
        <Dialog open={showGroup} onClose={() =>
setShowGroup(false)}>
          <DialogContent>
            <Typography>What do you want to name the
group?</Typography>
            <TextField
              sx={{ maxWidth: 400 }}
              variant="outlined"
              label="Group name"
              margin="dense"
              value={groupName}
              onChange={handleGroupNameChange}
            />
            <Button
              sx={{ my: 0.5, width: '100%', borderRadius: 0 }}
              variant="outlined"
              size=""
              onClick={handleGroup}
              disabled={checked.length === 0}
              aria-label="Add"
            >
              Add
            </Button>
          </DialogContent>
        </Dialog>
      ) : (
        <></>
      )}
      {showAlert ? (
        <>
          <Snackbar sx={{ marginBottom: 10 }} open={showAlert}
onClose={() => setShowAlert(false)} autoHideDuration={3000}>
            <Alert severity={'success'}>{alertMessage}</Alert>
          </Snackbar>
        </>
      ) : (
        <></>
      )}
    </Grid>
  );
}
```