Development of web application for visualization and calculation for stacking products on pallets

Degree project in Computer Engineering

JOEL PERSSON

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2024
www.chalmers.se

Degree project 2024

Development of web application for visualization and calculation for stacking products on pallets

JOEL PERSSON

UNIVERSITY OF GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2024

Development of web application for visualization
and calculation for stacking products on pallets
JOEL PERSSON

© JOEL PERSSON, 2024.

Supervisors:
Robin Adams, Department of Computer Science and Engineering
Jonas Persson, Evomatic AB

Examiner:
Jonas Almström Duregård, Department of Computer Science and Engineering

Degree project 2024
Department of Computer Science and Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Isometric laptop rendering pallet with products coming out of screen created by the author of the project report.

Written in LaTeX
Department of Computer Science and Engineering
Gothenburg, Sweden 2024

Abstract

This report presents the process applied and the results achieved in the development of a web application which allows users to build, calculate information of interest and visualize stacked products on pallets. The developed web application was to allow a high degree of customizability to be applicable in real life scenarios where each project commitment varies greatly in specific customer requirements. It was also of great importance that the user experience should contribute to ease of use and that the user interface should feel appealing.

Preface

This degree project has been carried out by a student at Chalmers University of Technology under the Department of Computer Engineering. The degree project is the final part of the degree. The work on the project and the writing of the report has been carried out during the spring of 2024.

A big thanks to the company for the granted opportunity to carry out the project.
A big thanks to supervisor Robin Adams for valuable advice and support.

Joel Persson, Gothenburg, June 2024

Contents

1
Introduction

The following chapter shortly introduces the company where the project has been carried out and explains why the project is of interest to the company. The purpose of the project and problem statements for the project are also explained. Set boundaries for the project will also be explained.

1.1 Background

The project has been carried out at a company called Evomatic AB. The company is a complete automation integrator which specializes in the design, programming, installation and operational service of robot cells for production flows in a wide variety of fields.

Integration of automatization within production flows and in the handling of products generally means that some, often repetitive and generic, tasks are replaced by robots. This means that tasks that were previously performed by a human worker are performed by a robot instead. What the precise task to be automated is varies a lot since it spans all kinds of different industries, each with their own needs of automation.

An essential part of the integration of automation in production flows and in handling of products which is frequently automated is palletizing of products on pallets. This is done with a palletizer, a machine which automatically stacks products on top of each other on a pallet. Palletizing of products is usually performed at the end of a production line when the products produced or handled by the line are already placed in a distribution medium such as carton boxes. Carton boxes are often used and are of highest relevance for this project compared to other distribution mediums such as plastic containers. The palletizing usage considered in the project is therefore stacking of carton boxes onto pallets.

Carton boxes containing the goods handled by the palletizer are very often stacked on a pallet. Pallets are a kind of flat structure intended to help support stacked products structurally while being easily movable by for examples forklifts and pallet jacks. They make it easy to handle and move batches of products making for efficient storage and transport of large numbers of products. Pallets are often made of wood, but a variety of materials occur. There exists standardization for the dimensions of pallets. Which standard is used depends largely on one's geographical location in

the world. The EUR-pallet is a standard primarily used in Europe, it is the most widely-used exchange pallet in the world [1].

Development of a web application that can be used to aid at an early stage of a project at the company to visualize and calculate information about the palletizing of products on a pallet according to a customer's specific requirements has the potential to yield great gains in both time and money if integrated into project commitments.

This project builds upon a project previously completed at the company by the same author. That project aimed to provide a proof of concept for sending data from a simple web application on a server hosted on a Programmable Logic Controller (PLC) to the control unit of a robot. That web application also included foundations for building and visualizing pallets. That project also consisted of analyzing what was possible and present a proof of concept of sending data. The report of that project is publicly available [15]. This project is a continued development on the web application portion included in the previously project to create a new stand-alone web application to be used solely for visualization and calculations of pallets. Some of the foundation core functionality from the old web application is therefore reusable for the new web application but needs to be expanded extensively on.

1.2 Purpose

The project's purpose is to investigate different solutions and develop a web application that is capable of building and visualizing stacked products on pallets. The web application should provide a high degree of customizability in terms of pallet settings, product settings and pallet building in order to be applicable even when almost every palletizing scenario is unique.

1.3 Objectives

Develop a web application that acts as an aid applicable at the early stages of different project commitments at the company by visualizing stacked products on pallets. Investigate whether or not integrating the developed web application in the current workflow of project commitments at the company proves beneficial. More concise and specific the objectives for the project can be listed as follows:

- Develop user interface for homepage.
- Develop user interface for entering project name.
- Develop user interface for entering pallet type.
- Develop user interface for entering product dimensions.
- Develop user interface for creating layer patterns.

- Develop user interface for choosing/ordering layers to be included on pallet.
- Developed user interface includes visualization/3D rendering of the pallet being built consisting of selected layers.
- Develop/integrate export and import functionality so that created pallets/projects can be exchanged between e.g. coworkers web application instances.
- Graphics/symbols/buttons are well thought out and are in vector format to scale well for different sizes on the web application.
- Investigate and find solution so the application can be distributed/accessed with relative ease on computers other than the project worker's own computer.
- Perform a concept validation and investigate whether or not integrating the developed web application in the current workflow of project commitments at the company proves beneficial.

1.4 Limitations

- The web application will not have support for storing data associated with a project in a database. Data will only be saved during the time that an instance is used by a user.
- The web application will only support one project at a time for the user, that is, one pallet that is built, edited and visualized at a time.
- The web application will only support one set of product dimensions per project. That means that building a pallet with products of different dimensions will not be supported.
- The web application will only support products to be cuboids.

2
Methods

The following chapter introduces the software, languages and libraries used in the project and describes the methods that have been applied. The project was divided into two primary phases. The first was the largest period which consisted of all the multiple different development phases to create a new web application from the previous project. This phase lasted twelve weeks. Secondly and lastly the result of the project is implemented at the company such that the concept of the developed application can be validated. This phase lasted one week.

2.1 Software, languages and libraries

The following section describes the software, languages and libraries that have been used during the project.

2.1.1 GitHub

GitHub is a web based storage solution for software projects. Some of the tools provided are version handling control and sharing of code [13]. In the project it has been used as remote storage for the source files of the project in a so called repository, making it easier to develop and access the code from different places. It has also made it easier to keep track of updates and changes to the source files. Planned development phases could also be kept enclosed in so called branches until they were not interfering with the preexisting code and until they were finished and then combined, so called merged. Since the repository could easily be shared, this enabled the supervisor of the project to keep track of changes, see progress and help when needed.

2.1.2 Visual Studio Code

Visual Studio Code is a open-source code editor from Microsoft [2]. It was used for all code editing in the project. It is widely used and offers a variety of extensions that can aid in the development process of projects. An extension called Live Server [3] was used during development of the project. This extension was used to run a server for the web application being developed. The server ran live meaning that changes made to the source files of the web application were refreshed and displayed directly. This gave instant feedback of applied changes during programming, excellent for development.

2.1.3 Languages

HyperText Markup Language (HTML) is a markup language which is the standard for documents that are going to be displayed on the web. A HTML file describes the content and structure of the elements that are going to be included on a web page. [4]. A number of HTML documents have been used in the project, each describing portions of the project and combined they make up the foundation of the web application that was developed.

JavaScript (JS) is a programming language which is the most used client-side scripting language of the web [5]. Being a scripting language, it helps with manipulation and customization of an already existing system. In the project, this means making the web application interactive and responsive by for example running scripts when a user clicks on a button. JS has been used in almost all parts of the web application being developed in the project for additional behaviour, some examples includes reacting to user inputs, handling dragging and dropping and rendering three dimensional environments.

Cascading Style Sheets (CSS) is a style language that is used to specify how objects in markup language documents should be presented [6]. In the project it has been used for all settings and configurations on how everything should look that is present in the application interface, this includes for example font-styles, background colors and sizes of objects.

2.1.4 Libraries

Three.js is a cross-browser library to JavaScript used to program and display animated 3D computer graphics in the web browser with the help of WebGL [7]. In the project it has been used to dynamically visualize the pallet with custom layers that is being built by the user.

jQuery is a lightweight, fast and feature-rich library to JavaScript [8]. It is designed to do more with less programming by simplifying traversal and manipulation of objects. Its primary purpose is to make JavaScript easier to use on websites. In the project it is primarily used by for simplifying the code for queries for manipulation and event handling of objects.

jQuery UI is a JavaScript library which is an extension of jQuery with focus on the user interface [9]. In the project it is primarily used for implementing its integrated drag and drop functionality. The drag and drop functionality improves ease of use of the web application for the user.

2.1.5 Blender

Blender is a free and open-source 3D computer graphics piece of software [10]. In the project it has been used to create 3D models that are rendered during the visualization of the pallet being built by the user.

2.1.6 Inkscape

Inkscape is a free and open-source vector graphics editor piece of software [11]. In the project it has been used to make all 2D computer graphics in Scalable Vector Graphics (SVG) format. This was desirable since SVG is scalable without losing quality when zoomed or resized [14].

2.1.7 Apache HTTP Server

The Apache HTTP Server Project is a free open-source cross-platform web server software [12]. An Apache HTTP Server instance was run on a Microsoft Windows machine in the project to easily deploy the developed web application for concept validation at the company. This allowed clients on the same Local Area Network (LAN) as the server to connect to it and test the developed web application.

2.2 Development of the web application

The source code for the web application has been constructed using the source code editor Visual Studio Code. A combination of the languages HTML, CSS and JS was used to build the web application. As explained in 2.1.2, an extension called Live Server was used during the development to act as a host for the source code being programmed. This extension provides the functionality of automatically updating the displayed web application upon saving the source code. Inkscape was used to create 2D vector graphics icons and images that appear in each user interface so it was used throughout the entire development.

The development of the web application has been executed in planned phases. The implementation and construction/rework of functionality has therefore been incrementally added. The main objective for each incremental development of the web application is listed in order of implementation below:

- User interface for homepage
- User interface for entering project name
- User interface for entering pallet type
- User interface for entering product dimensions
- User interface for creating layer patterns
- User interface for choosing/ordering layers to be included on pallet with visualization by 3D rendering
- Develop/integrate export and import functionality so that created pallets/projects can be exchanged between e.g. coworkers web application instances
- Investigate and find solution so the application can be distributed/accessed with relative ease on computers other than the project worker's own computer

2.2.1 Homepage

Firstly a user interface for a homepage was developed that a user lands on when first navigating to the web address of the web application. Starting with the homepage, a general layout and overall theme for the web application was decided on by testing e.g. different color combinations. This was straightforward and quick to test thanks to the live updates by the extension. One more thing that was considered this early in development was to find an appropriate storage solution for the data that would be received and have to be stored from the user later in the web application. This was done by investigating solutions applied by similar projects. The data that needed to be stored was mostly to be user entered and consisted e.g. project name, pallet/product dimensions and layer patterns to name a few. All data types and data structures used in the project are explained in 3.1.

2.2.2 Data gathering

Next, user interfaces that all fell within the realm of project data gathering from the user at the start of a project was developed. These user interfaces was for entering and/or selecting data to be used further in the application. The first of these interfaces was for entering project name including an optional description. The second one was for selecting a pallet type. The third one of these interfaces was for entering dimensions for the product to be used on the layers. This included optional toggling of whether or not to include labels and also option to include custom label image if wanted. The functionality of all of these interfaces where all very similar in that they all stored varying amount of supplied user project data.

2.2.3 Layer patterns

Following the development of user interfaces for data gathering, an interface with functionality to build and save patterns on the selected pallet type by using the selected product was developed. Drag and drop functionality for the pattern building was achieved using jQuery UI. The base for the implementation with similar functionality is found in the source code of the previous project. This implementation was a significant extension of the previous implementation in the sense that it was graphically enhanced, functionally expanded and quite a few bug fixes was done including quality of life updates.

2.2.4 Pallet building and visualization

When the user interface for building and saving patterns was finished the next step was to develop an interface for building a complete pallet where the user could select amongst recently saved layers with created patterns and adding spacers between layers if wanted. A spacer refers to a thin sheet of cardboard that is often put in between stacked layers to obtain structural stability on a pallet. The visualization

of the pallet being built was to be in 3D and was rendered in the browser using the Three.js library to JS. The base for an implementation with 3D rendering using Three.js with similar desired functionality is found in the source code of the previous project. During the development of this user interface additional 3D-models than the ones available from the previous project were created using Blender. The new implementation was a significant extension of the previous implementation in the sense that it was functionally expanded, quite a few bug fixes was done and other general improvements.

2.2.5 Exporting and importing

All the main objective user interfaces was now developed for the web application. The next step consisted of finding and implementing a method for exporting and importing created projects. A created pallet in a project could be of potentially endless combinations of pallet dimensions, products dimensions, labels or not, patterns and order of layers on pallet. Exporting and importing functionality would allow created pallet to easily be downloaded, shared and loaded between e.g. colleagues and customers. This would ensure that pallets, especially complex ones, wouldn't have to be rebuilt from scratch each time. Importing and exporting was done in JS to write and read the data to and from a file.

2.2.6 Accessing the web application

The final development part was to find and implement a method for how the web application was going to be conveniently distributed and/or accessed by the users at the company. This was done by investigating solutions that would be suitable to make the web application accessible. Based on recommendations from the company the method that the project went with was to run an Apache HTTP Server which hosted the developed web application. This made so that the developed web application could be accessed from users on the same LAN as the server at the company.

2.3 Concept validation

The concept validation was was carried out in the company's office. It was performed by having a Windows laptop running an Apache HTTP Server hosting the developed web application. This made the developed web application accessible to all users that was on the same LAN as the host by navigating to a certain address. The point of the concept validation at the company was to demonstrate and realize actual usage of the web application in practice. Since the web application was now accessible to the employees at the company, it could be fully tested for its intended purpose in a more real life scenario. The intended purposes for the developed web application was that employees should be able to navigate to its address with a web browser and furthermore build and 3D visualize pallets of their making. Employees should also be able to export and/or import projects to/from a file that can be exchanged with e.g. a colleague.

Two employees performed the concept validation and the procedure for each employee was to build a complete pallet with customization of their choice. This simulated a real life scenario where a customer might request a custom pallet with everything from non-standard pallet dimensions to unique product patterns. The 3D visualization could then, in an actual scenario, be included as a proposed preview of how the palletizing could be done to strengthen the impression. However, in the concept validation the pallets built by the employees where switched between each other and loaded into each others instances of the web application to simulate the scenario of sharing a created pallet with e.g. a coworker. After the concept validation it was discussed and evaluated between the two participants based on user experience how well the intended purposes had worked.

3
Result

The following chapter describes and shows the result that has been achieved by the project.

3.1 Development of the web application

As explained in 2.2, parts of the web application was developed and implemented incrementally. The result of each incrementally implemented part are presented in order of addition below. Also as mentioned in 2.2 a combination of the languages HTML, CSS and JS was used to build the web application. All of the developed user interfaces consists of varying proportions of each of these languages. Example snippets of each of the languages is shown and explained below to get a better understanding of the different syntax and how they contribute to each other in the development of the web application. Selected portions of the source code for the developed web application that is considered noteworthy and interesting will also showcased under its corresponding user interface section.

HTML was the markup language used in the project. It essentially describes the structure and all the in going elements that should be included in a document i.e. a user interface to be displayed on the web as part of the web application. Simple boilerplate markdown for HTML is shown below as an example for the syntax. In this example a title is set for the page and a header text is added in the body. The body element holds all the content that are going to be displayed on the page.

```
<!-- Example HTML boilerplate -->
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <title>My Web Application</title>
  </head>
  <body>
    <h1>Hello World!</h1>
  </body>
</html>
```

CSS was the style language used in the project. It describes how objects defined in the markup language documents should be presented e.g. color, size or font in the

case of text. Simple CSS language specifying design choices for an example element with made-up id *myElement* is shown below as an example of the styling syntax.

```
/* Example CSS for element with a background image */
#myElement{
  position: absolute;
  width: 100px;
  height: 100px;
  background: url(./path/to/file.svg) no-repeat;
  background-size: contain;
}
```

JS was the scripting language used in the project. It was for example used for all elements and parts that should be responsive and do something on user interaction and input. A simple script function is shown below were responsiveness is added to an example button with made-up id *myBtn* that now loads another page when clicked on by a user.

```
/* Example JS do something when user clicks button */
$("#myBtn").unbind('click').click(function(){
    /* Jump to specified page */
    window.location.replace('myOtherPage.html');
});
```

3.1.1 Homepage, design and data storage

Firstly a homepage that the user first lands on when navigating to the web application was developed. A general layout and overall theme was selected. Important aspects were a simple layout and ease of navigation by a sidebar menu to the left. The solution for data storage for the web application was found to be the *localStorage* property. This is a temporary storage in the browser of the user that the web application utilizes during the session. An overview of the most significant project data and its corresponding data type that is stored in *localStorage* when the application is used is explained in 3.1.6. A discussion about the decision to use the *localStorage* property is provided in 4.1.

The web application part developed in the previous project had no homepage. This was because that web application was limited to conceptual testing of only core functionality whereas the developed application should be more complete, meaning that it should be ready for usage in actual real life scenarios and thus behave and look as a user might expect from similar applications.

3.1.2 Project data gathering

The developed page for entering project name and an optional project description is shown in Figure 3.1. This page is reached by either selecting the button for starting a new project on the homepage or by navigating the sidebar menu on the left selecting the first tab. To make the web application intuitive, notice the red text

under the empty project name input field subtly prompting the user to enter one. A small red triangle is also visible next to current project data tab in the secondary sidebar menu whenever an error or requirement of action is present.

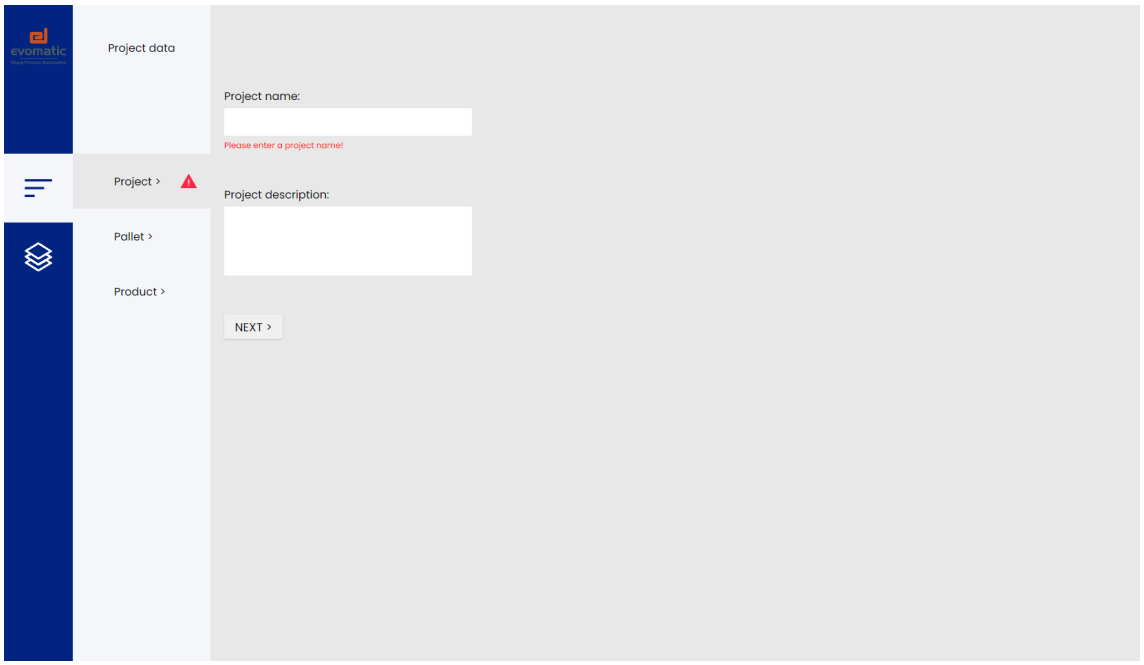

Figure 3.1. The page for the user of the application to enter a project name and an optional description of the project.

The developed page for selecting the pallet for the project is shown in Figure 3.2. The pallet type is selected in a drop down menu. There exist a couple of predefined options to select from reflecting some of the most commonly used pallet types e.g. Euro-pallet which is the standard European pallet. Compared to the application in the previous project, the new application has added support for custom pallet dimensions. When the custom option is selected, new input fields for each of the pallets measures are displayed with a short transition. As is the case with all required user input fields in the application, similar indications as before prompting user action. The image in the right part of the page depicts the currently selected pallet type. All predefined pallet types have their respective design but since the custom type is unknown it is simply shown as a rectangle.

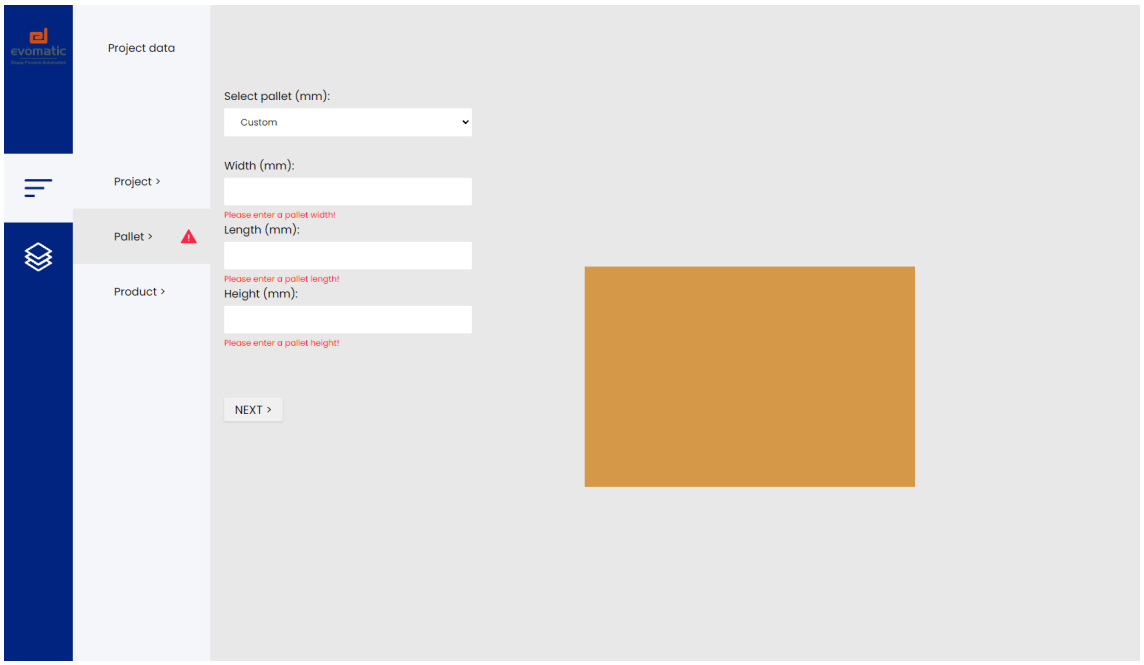

Figure 3.2. The page for the user of the application to select a pallet type or enter dimensions for a custom pallet.

The developed page for entering measurements for the product to be used in the project is shown in Figure 3.3. The required user input fields are indicated here similarly as previous pages prompting the user to action. Only the input fields for dimensions are required as these are necessary hereafter in the application. Whereas weight is simply not tracked if none is entered. The first toggle button below the input fields are for whether or not to include labels on the products in the project. The second toggle button is for, as depicted in the image to the right of the toggle button, whether labels should be on widthwise or lengthwise side of the product. This toggle button is only displayed if the user first selects to include labels in the project in the first place. Lastly there's an optional possibility to upload a custom image to use instead of the default labels used in the application. By default if labels are included and if products are placed in such a way that they are visible, they are simply shown as a white rectangle on the side of the corresponding side of the product in the 3D rendering. If a custom image is uploaded, it will appear instead. This option is also only displayed if the user first selects to include labels in the project in the first place. The custom label functionality is further shown in 3.1.5.

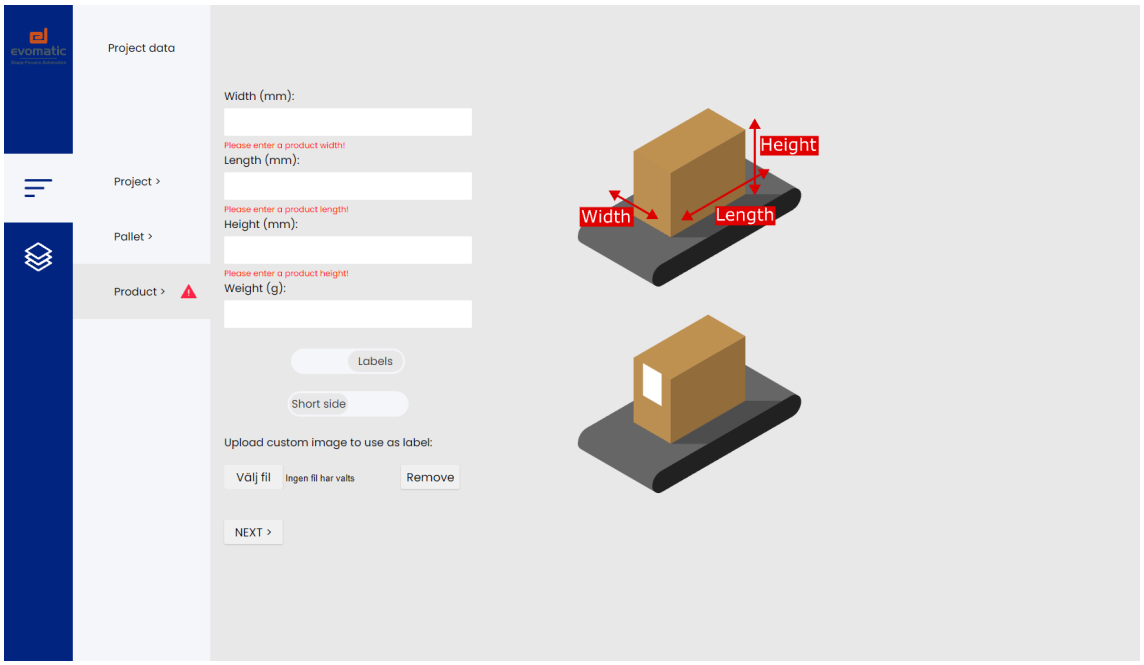

Figure 3.3. The page for the user of the application to enter dimensions for the product and optionally add labels, select label side and upload custom label image.

The 2D vector images shown to the right in Figure 3.3 was created using Inkscape. The image at the top is intended to help indicate the dimensions that a user is expected to enter. The bottom image depicts what side of the product that the user have currently selected for the labels to be on. The image will change based on slider selection and does so with the following JS code that changes image for an element if the slider is changed e.g. clicked by the user.

```
/* Switch determining label side, change preview image */
$("#toggleS").on('change', function() {
    if ($(this).is(':checked')) {
        document.getElementById("previewLabelProduct").src=
        "./res/box_label_long.svg"
    }
    else {
        document.getElementById("previewLabelProduct").src=
        "./res/box_label_short.svg"
    }
});
```

In the three user interfaces developed thus far, project data has been selected and entered by the user of the web application. When the button at the bottom of each of these user interfaces is pressed by the user to proceed forth in the web application between these user interfaces, all the data retrieved that far is stored using the *localStorage* property mentioned in 3.1.1 using very similar JS code. The JS code for storing user input about the product shown in Figure 3.3 is shown below as an example.

```
/* Store user input product parameters and label selection */
$("#btnSubmit").unbind('click').click(function(){
    /* No errors should remain to be able to proceed */
    if(!wErr && !lErr && !hErr && !weErr && !pnErr){
        localStorage.setItem("PRODUCT_HEIGHT_STORED",
        JSON.stringify($("#product-height").val()));
        localStorage.setItem("PRODUCT_WIDTH_STORED",
        JSON.stringify($("#product-width").val()));
        localStorage.setItem("PRODUCT_LENGTH_STORED",
        JSON.stringify($("#product-length").val()));
        localStorage.setItem("PRODUCT_WEIGHT_STORED",
        JSON.stringify($("#product-weight").val()));
        localStorage.setItem("INCLUDE_LABEL_STORED",
        JSON.stringify($("#toggle").is(':checked')));
        localStorage.setItem("LABEL_SIDE_STORED",
        JSON.stringify($("#toggleS").is(':checked')));
        /* Jump to pallet page */
        window.location.replace('pallet.html')
    }
});
```

3.1.3 Layer pattern creation

The developed page for creating patterns on the selected pallet using the specified product is shown in Figure 3.4. On this page, the user of the application builds and saves custom pallet layers to be selected amongst later for the final rendering of a complete pallet. The products lined up in the near center of the page are dragged and dropped onto the pallet at desired position using the mouse pointer. Rotation and/or label orientation is simply selected by grabbing and dragging the corresponding product.

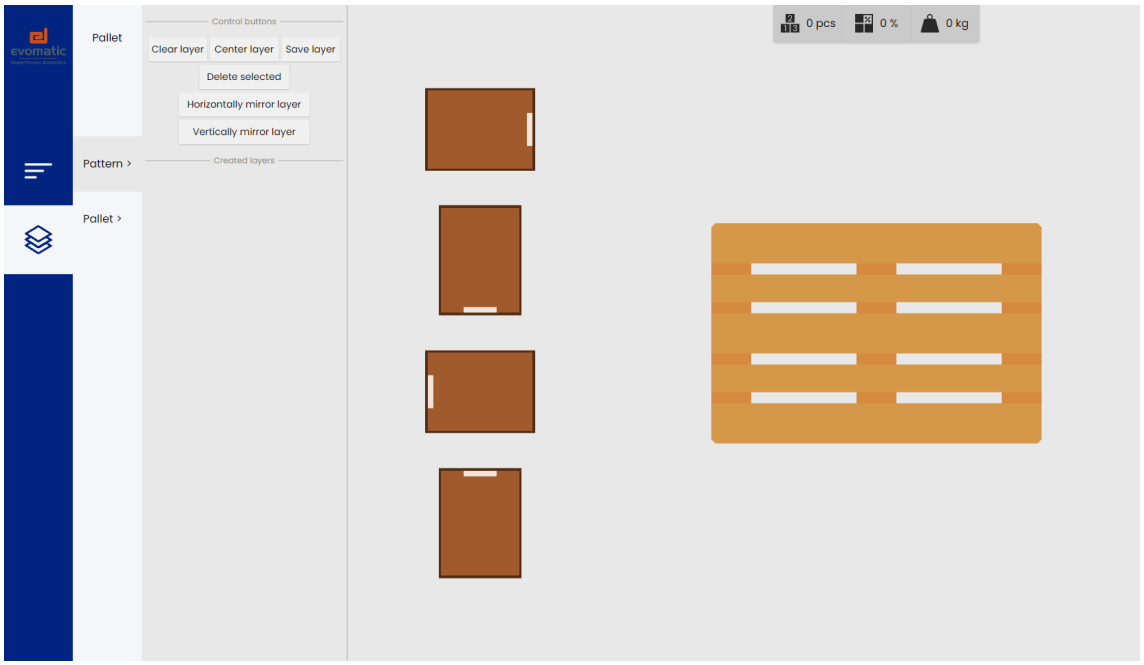

Figure 3.4. The page for the user of the application to create patterns by dragging and dropping products onto the pallet.

When the user of the application is satisfied with the current layer pattern, it can be saved using the saving button amongst the other buttons in the top left container. This container holds several buttons with a variety of functionality for controlling and helping with the workflow of creating patterns. The container below holds the saved layer patterns, a saved pattern is visualized by a smaller version of itself. In the top middle of the page is a container containing boxes displaying information about the current layer pattern being built by the user. The information that is being tracked is, in order of appearance from the left, total number of placed products on the layer, total area coverage percentage of the placed products on the pallet and total weight of all placed products combined if a weight were specified.

In Figure 3.4, arbitrary dimensions for the product were selected in the previous steps for the purpose of demonstrating. The pallet type was selected as Euro-pallet. Labels were selected to be included placed widthwise and no custom label was uploaded. The image for the pallet representing the droppable area for products will change depending on the selected pallet type for the project to depict the pallets real world appearance. The lined up products for dragging will also change appearance depending on previous choices e.g. might not include small white label indication or label might be placed lengthwise instead.

Figure 3.5 shows an example of the application where three layers have been created ans saved by a user. They have also marked the top left most saved layer, marked by an black outline, which is necessary and makes it possible to delete it. The layer pattern that is being built can be cleared by pressing the button for clearing a layer or the products on the pallet can simply be dragged and dropped outside of the

pallet one by one manually, which also deletes them. In the container with information, we can see that five products have been placed. It is also calculated that 62,5 % of the pallets area is being used and that the products weigh a combined 1 kg. In this example the product was assigned an arbitrary weight of 200 g.

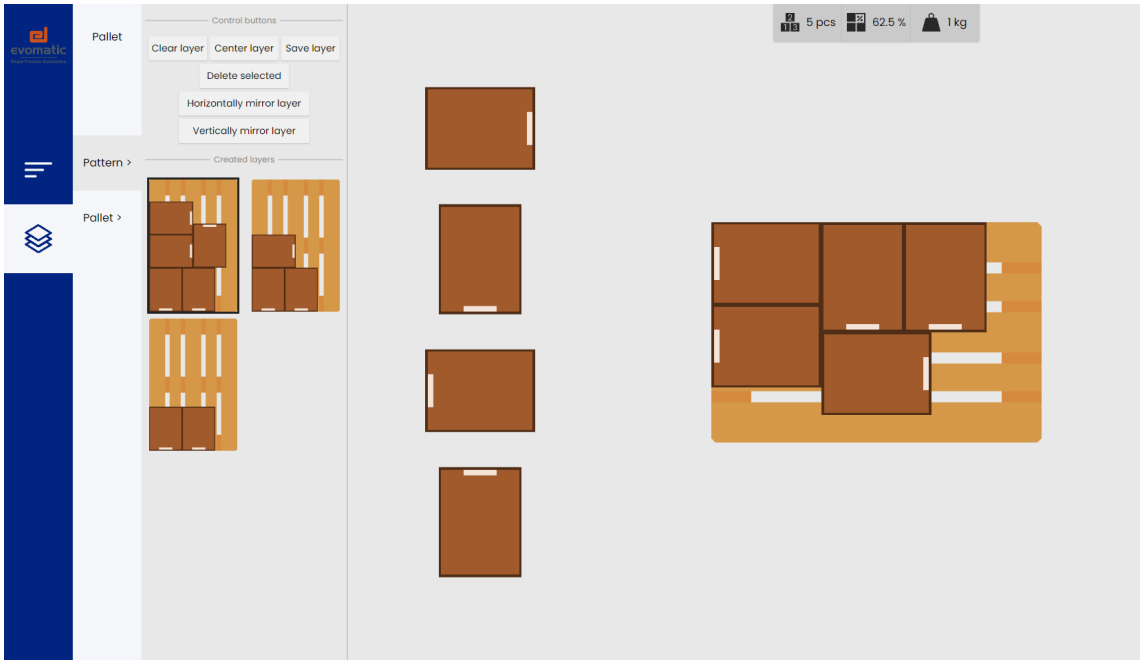

Figure 3.5. The user of the application have created and saved three different layer patterns. They have also marked the top left one, indicated by a black outline, making it possible to delete it.

The effect of the functionality of the rest of the control buttons available are shown in Figure 3.6. The previously marked layer pattern has been removed using the delete button. The layer pattern being built has also been centered on the pallet, horizontally mirrored and vertically mirrored. Each adjustment is performed with its corresponding button. Finally the changed layer pattern has been saved again. Invalid drop indication is also shown, visually telling the user that a product can not be dropped at this location. All products affected are marked with a red cross. Dropping a product while this indication is active will only result in the dropped product being deleted. The computer determines the validity of a drop by collision detection. While a product is dragged by the user, a helper function is called repeatedly that checks and detects if any overlapping has occurred between two rectangular shapes, i.e. products. The product being dragged is always passed to the helper function and the second product to compare against consists of looping through the products that have already been placed on the pallet. The JS code for the helper function repeatedly used for collision detection whilst a product is being dragged is shown below.

```
/* Helper function, checks collision between two rectangles */
function checkRectColl(rectOne, rectTwo){
    var r1 = $(rectOne);
    var r2 = $(rectTwo);
    var r1x = r1.offset().left -0.1;
    var r1w = r1.width()-0.1;
    var r1y = r1.offset().top-0.1;
    var r1h = r1.height()-0.1;
    var r2x = r2.offset().left -0.1;
    var r2w = r2.width()-0.1;
    var r2y = r2.offset().top-0.1;
    var r2h = r2.height()-0.1;
    if( r1y+r1h < r2y || r1y > r2y+r2h ||
        r1x > r2x+r2w || r1x+r1w < r2x ){
        return false;
    }else{
        return true;
    }
}
```

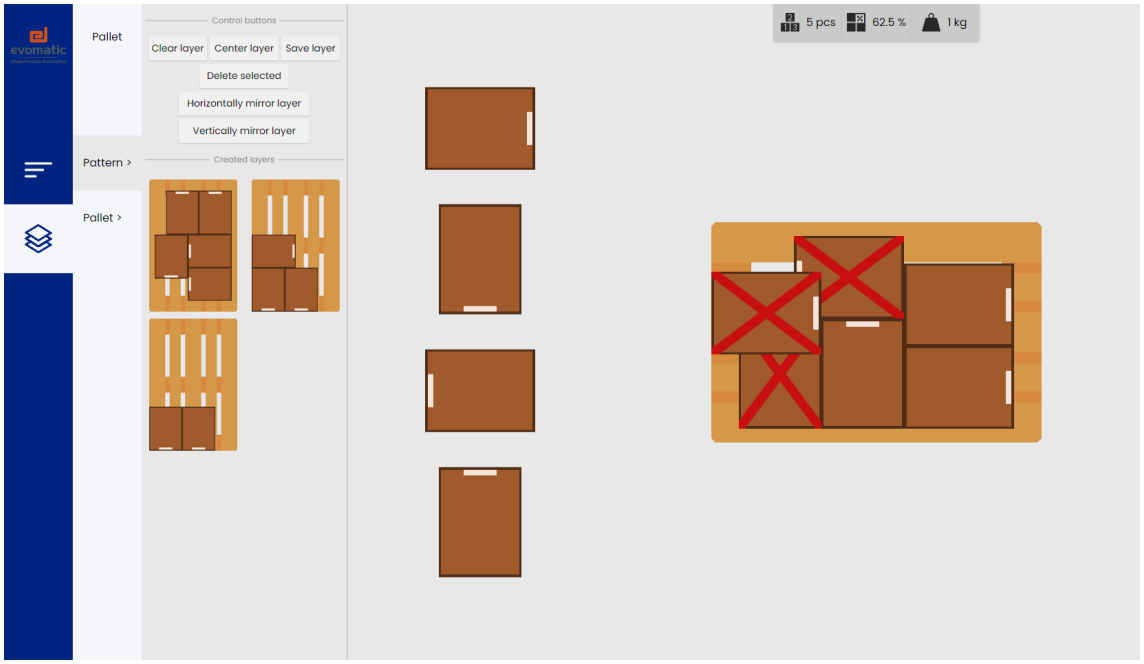

Figure 3.6. The user of the application have deleted the previously marked layer pattern. They have also centered, horizontally- and vertically mirrored the layer pattern being built and saved it again. Invalid drop indication is also shown.

3.1.4 Building and visualizing the pallet

The developed page for creating a custom pallet using the saved layer patterns is shown in Figure 3.7. This is the final stage and page of the web application. On this page, the user of the application builds a custom pallet by selecting layers to include amongst the saved layer patterns. Likewise as before, a saved layer is selected by clicking on it marking it with a black outline. A selected layer can then be added to the pallet by using the button for adding a selected layer amongst the control buttons. There's also a button for adding spacers to the pallet. Spacers are often thin sheets of paper added between layers. Their primary purpose is to increase stabilization of the pallet when multiple layers are stacked. The result of both these buttons functionality is shown in Figure 3.8. The user have added the selected layer to the pallet and they have also added a spacer to the pallet. This is indicated by corresponding placeholders being added to the "pallet stack" in the middle of the page. The pallet stack decides what should be 3D rendered. By default, layer patterns and spacers are appended to the top of the pallet stack.

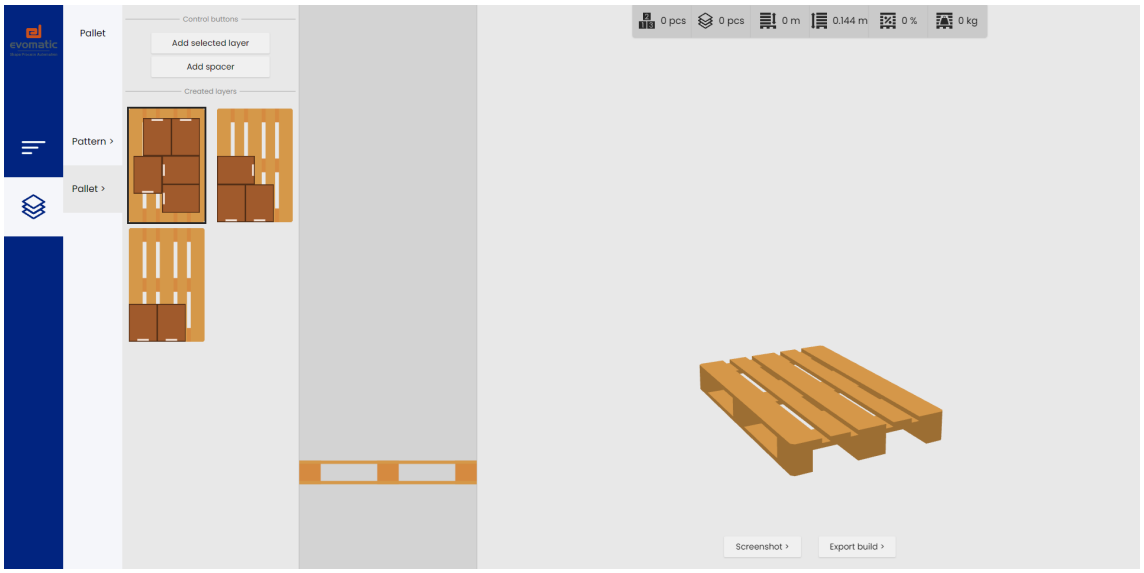

Figure 3.7. The page for the user of the application to build a custom pallet using the saved layer patterns. The pallet being built is also depicted by a corresponding 3D rendering.

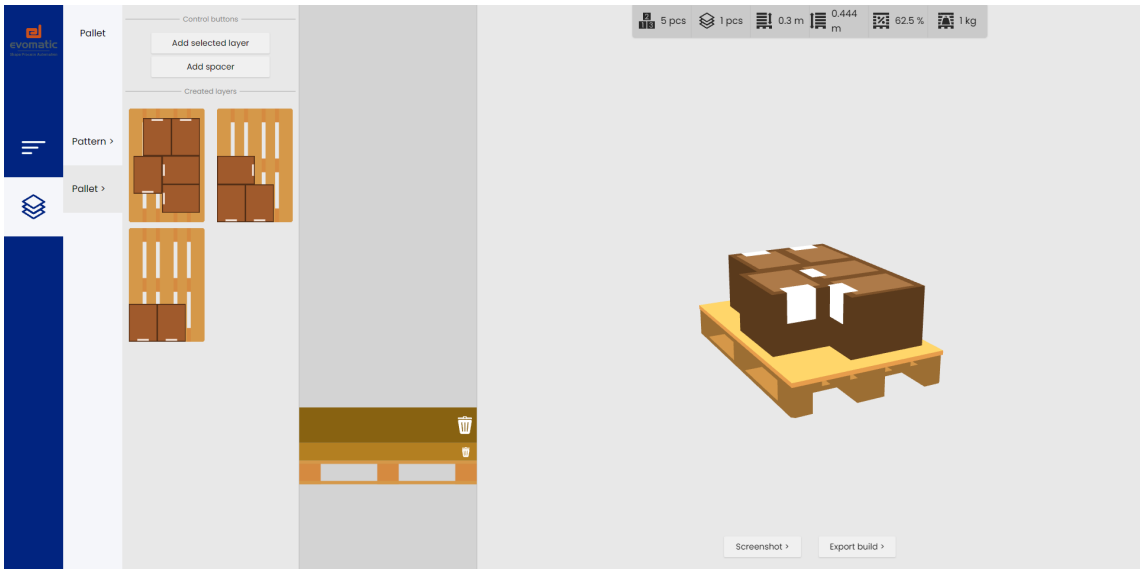

Figure 3.8. The user have added a selected layer pattern to the pallet. They have also added a spacer and placed it at the bottom. The 3D rendering automatically shows the pallet being built.

The order of layer patterns and spacers added to the pallet stack can be changed however the user requires by dragging and dropping the placeholders to desired position in the stack using the mouse pointer. Dragging and dropping a layer pattern or a spacer placeholder outside of the stack simply returns it to its original position or last registered position. A placeholder for a layer pattern or a spacer can be removed from the layer stack, thus also its corresponding representation in the rendering. This is done by clicking on the small trash can symbol that is located at the right side of each placeholder. Whenever the user adds to, removes from or updates the order of the layer stack, the rendering is updated and executes anew to consider all changes that might have occurred.

Similarly to the previous page, there's a container with boxes of information about the pallet being built at the top center above the 3D rendering. The data that is displayed is also recalculated and updated whenever the user performs any actions on the layer stack. The data that is being tracked is explained in order of appearance from left to right. Total number of products on the pallet. Total number of layers on the pallet, spacers are not included as layers in this count. Total height of the layers on the pallet, spacers and pallets own height are not included. Total height of the layers on the pallet including the pallets own height, spacers are not included. Percentage filled space of maximum available space utilized on the pallet for all layers combined, i.e. how much of the potential pallet space that is actually being used. Total weight of all layers on the pallet, spacers and pallets own weight are not included.

Figure 3.9 shown how the user of the application has advanced further in building a pallet by adding layers and spacers to it. Notice how they have also rotated the 3D rendering to get a different view of the pallet being built. The view of the

pallet can be rotated freely around the current center point of the rendered pallet. This is achieved by left clicking and dragging in desired direction with the mouse pointer anywhere within the 3D rendering. It is also possible to zoom in and out of the current center point of the rendered pallet using the mouse wheel.

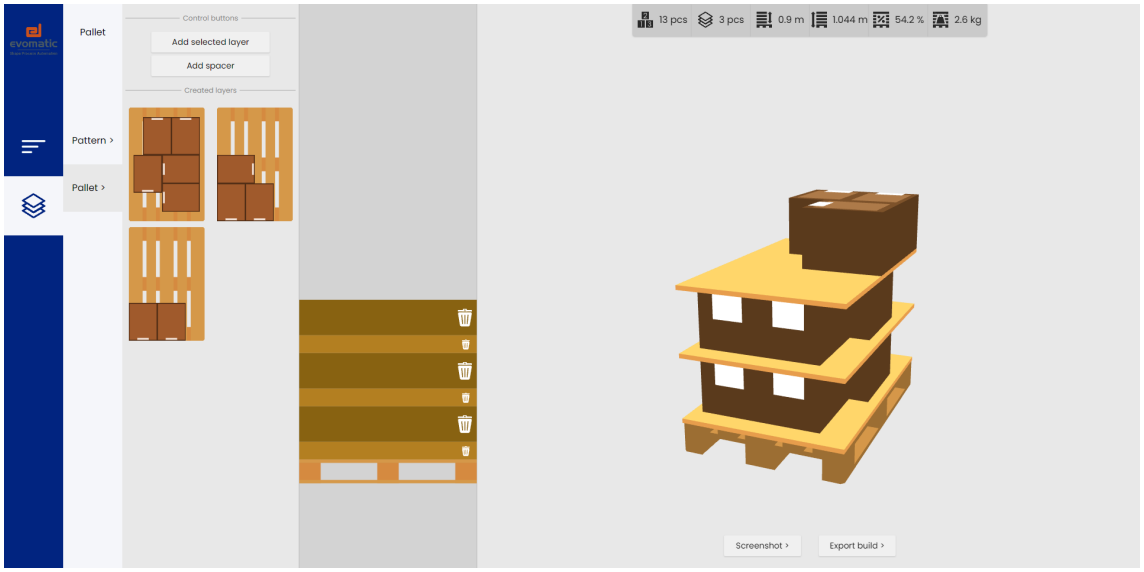

Figure 3.9. The user have added more layer patterns and spacers to the pallet. They have also rotated the view of the pallet that is 3D rendered.

The 3D models available for the 3D rendering in this user interface was created using Blender. The 3D models created was a blank box, a labeled box, a spacer and a pallet for each corresponding included pallet standard mentioned at the beginning. The 3D rendering part of the user interface was programmed using Three.js. To 3D render anything with Three.js in a browser, at least three components are required: a scene, a camera and a renderer. The scene refers to the place where everything that is going to be included in the rendering is placed. This includes for example loaded 3D models of selected pallet and layers. The camera determines how we see the scene, e.g. field of view. The renderer renders the scene for us by redrawing it, meaning all items inside, with a set interval in an animation or rendering loop. Example JS code that includes all these essential components are shown below.

```
/* Example code for creating a scene from threejs.org */
import * as THREE from 'three';

const scene = new THREE.Scene();
const camera = new THREE.PerspectiveCamera( 75,
window.innerWidth / window.innerHeight, 0.1, 1000 );

const renderer = new THREE.WebGLRenderer();
renderer.setSize( window.innerWidth, window.innerHeight );
document.body.appendChild( renderer.domElement );
```

To get anything rendering we also need a so-called animation or rendering loop mentioned previously. Example JS code for that is shown below.

```
/* Example code for an animation loop from threejs.org */
function animate() {
        renderer.render( scene, camera );
}
renderer.setAnimationLoop( animate );
```

Every item in the scene is a 3D model that was first loaded using a so-called loader. A wide range of different loaders exists available to Three.js that each are suitable for different model formats. To find a suitable one is thus a matter of matching ones expected model format to the loaders available. Example JS code for how the loading of a EUR-pallet model was done in the project is shown below. All other models in the project was loaded in a very similar way.

```
/* Example code for loading a model using available loader */
/* Model loader */
const loader = new GLTFLoader();
/* 3D model object that we can load model to */
var europallet = new THREE.Object3D();
/* EUR-pallet model */
loader.load(
    './res/models/europallet.glb',
    function (gltf) {
        europallet = gltf.scene;
        /* Position based on center point from Blender */
        /* Here center point is bottom left corner so */
        /* divide by 2 for middle times 1000 for scale */
        europallet.position.set(PALLET_LENGTH/2000,
                                -(PALLET_HEIGHT/2000),
                                -(PALLET_WIDTH/2000));
        europallet.rotateY(Math.PI/2);
        /* Only add selected pallet type to scene */
        if(PALLET_TYPE == 'europallet'){
            scene.add(europallet);
        }
    },
    (xhr) => {
        console.log('EUR-pallet 3D-model loaded');
    },
    (error) => {
        console.log('Error loading EUR-pallet 3D-model');
    }
)
```

At the bottom center below the rendering is buttons for taking a screenshot of the built and rendered pallet and for exporting the data of the pallet. A resulting image of the screenshot button functionality is shown in Figure 3.10. The screenshot is generated in Portable Networks Graphics (PNG) format, this is to acquire a transparent background. This is so that the screenshot should be as conveniently further usable as possible where varied backgrounds might occur, e.g. a report. The button for exporting pallet data saves all data associated with a pallet project using JavaScript Object Notation (JSON) to a file which the user can download. This saved data can then be imported into an instance of the application allowing for interchangeable work between e.g. colleagues on the same pallet for a project.

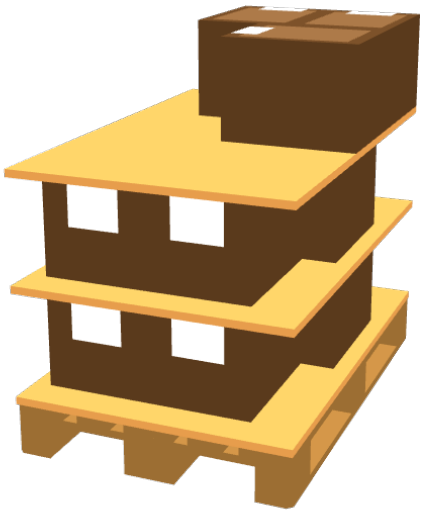

Figure 3.10. The transparent screenshot image taken of the 3D rendering of the pallet using the applications implemented functionality for taking screenshots with the press of a button.

3.1.5 Custom label image

As briefly mentioned previously in this chapter in the section for entering and selecting properties for the product to be used in the project, there existed an optional property to upload an image to be used as a custom label. This optional property is selected by uploading an image of supported format. When this property is set, the uploaded image will appear instead of the default white rectangle indication of a label on products in the 3D rendering of a pallet. An uploaded custom label behaves like a normal label meaning all other properties affecting labels works the same. This means that it is necessary to set the property to include labels for the project in the first place. The property toggle for specifying whether to include labels widthwise or lengthwise also applies to custom labels. The white indication of label orientation in the layer pattern builder also applies to custom labels. A screenshot of the rendering of an example project with a pallet whose layers consists of products utilizing an uploaded custom label is shown in Figure 3.11. The possibility for the user to upload images to be used as custom labels is intended as an ability to make pallets for projects be even more custom-made for e.g. a company or a report.

Figure 3.11. Screenshot image of 3D rendering of example pallet consisting of layers with products that utilize a user uploaded image as custom labels.

3.1.6 Overview stored project data

An overview of the most significant project data and its corresponding data type that is stored in *localStorage* when the application is used can be seen in Table 3.1. Ordered vertically based on occurrence in expected step through of the web application.

Table 3.1: The most significant properties, their corresponding data type and whether or not they are optional for a project, that is stored in the *localStorage* of the user during the session of using the application.

Property	Data type	Optional
Project name	String	No
Project description	String	Yes
Pallet width	Number	No
Pallet length	Number	No
Pallet height	Number	No
Pallet type	String	No
Product width	Number	No
Product length	Number	No
Product height	Number	No
Product weight	Number	Yes
Flag include label	Boolean	Yes
Flag label side	Boolean	Yes
Flag custom label	Boolean	Yes
Custom label image	String	Yes
Created layers	*Layer* Array	No
Selected layers	Array	No

The array of created layers that is being stored in *localStorage* is a linear data structure which holds *Layer* objects. The properties and data types which make up a *Layer* object are shown in Table 3.2. The array of selected layers that is being stored in *localStorage* is a linear data structure which holds IDs copied from certain *Layer* objects in the created layers array. The IDs contained in the array of selected layers represent the layers which has been added by the user to the layer stack in the rendering phase. It's used to tell the application which layers that should be 3D rendered on the pallet.

Table 3.2: The properties and data types which makes up a *Layer* object, including explanation.

Layer		
Property	Data type	Explanation
Flag spacer	Boolean	Whether or not layer is a spacer
Id	String	Randomly generated 13-character ID
Points	*Point* Array	Holds *Point* objects, information of products in layer

The array of points is a linear data structure which hold *Point* objects. A *Point* object is associated with a single product and holds information about a particular product on a pallet. The properties and data types which makes up a *Point* object are shown in Table 3.3.

Table 3.3: The properties and data types which makes up a *Point* object, including explanation.

Point		
Property	Data type	Explanation
Left	Number	X-coordinate
Top	Number	Y-coordinate
Height	Number	Z-coordinate
Flag blank	Boolean	Whether or not to generate product blank
Rotation	Number	Base rotation in degrees (0, 90, 180, 270)
Label side	String	What side the label should be on
Applied rotation	Number	Additional rotation applied by user

3.2 Concept validation

The foundation for the concept validation was an Apache HTTP Server instance that was run on a Microsoft Windows machine where the developed web application was deployed. Clients used the same LAN as the server and could connect to it and test the developed web application. The web application was then tested for its intended purpose by having two different instances, each on different machines, simultaneously use the web application and build a pallet of their choice. The created projects were then exported, the save files were exchanged and finally imported in each others instances of the web application. This was a validation of how the web application could be applied in a real life scenario. The intended purpose of the developed web application to be accessible to multiple users simultaneously worked. The intended purpose of the developed web application of a user being able to build a custom pallet whilst getting information about the current pallet and also getting it 3D visualized worked. Finally the intended purpose of the developed web application to be able to export and exchange projects between users to import into their own instances of the web application worked. All of the intended purposes of the developed web application which is a combination of the multiple objectives for the project presented in 1.3 was thus satisfied.

4
Discussion

The following chapter discusses certain decisions made during the project and future development possibilities for the developed web application.

4.1 Decisions

When looking for an appropriate solution for storing user entered and selected project data, the simplest and fastest method considering the relative small size of the application was found to be the *localStorage* property. This allowed storage of limited data (although more than enough for the scope of the application) in the users own browser during the session i.e. interaction with the application. Another possible solution could have been to store the data in a backend for the application, e.g. a database solution. However, the alternative that the project went with allowed the web application to remain smaller which suited the limitations in the extents of the project better.

4.2 Future development

The author sees positively at future additional functionality and further improvements to the developed web application. Quite a few possibilities for future development were found during development. They will be proposed and discussed here.

The use of a database solution for storage of user-entered-and-created project data, i.e. layer patterns and constructed pallets. This would have the potential to make storage and retrieval of projects simpler and more straight forward for the user of the application instead of importing and exporting files with the project data as it currently is. Consider e.g. a company with multiple employees using the developed web application at arbitrary times. The ability to store a created project in a local database or library-like solution so that anyone can continue work on it or maybe use it as a base template for a new project would greatly increase ease of operations. However, an implementation like this is something that would require further research to able to strike a balance between actual usage of the web application contra the potentially gained ease of use. This would be necessary to determine e.g. if development and implementation would be worth it compared to cost.

To enable a project to consist of products with different dimensions. A layer pattern to be included on a pallet in a project is currently only supported to consist of products of one set of dimensions. The only variation supported is different rotations on the products. The proposed addition would allow, if the user of the application wanted to, build and render a pallet where the ingoing layers consists of products of a varied dimensions. An implementation like this would greatly enhance the customizable aspect of the web application. However, an implementation like this would require a pretty substantial amount of rework and development of the currently developed web application.

Dynamic feedback checking if there exists enough support to hold a layer of products on top of another. In other words somehow implement functionality so that it is calculated and indicated whether or not the potential addition of a layer can be structurally supported by the layer beneath it. This would have to consider the coverage of the layer beneath accumulated by its products and compare if these are at the necessary positions to hold the selected layer that the user theoretically wants to add on top. No checks of this kind are performed in the developed application. This is mostly for practical ease of use and to not limit the ability of creating pallets for the user, but also delimits in the extent of the project. This checking might be implemented as switchable option to either build a pallet in free-mode or limit-mode. An implementation like this would certainly aid in the creation of stable, secure and well structured pallets. However it might require a considerable amount of work for an implementation like this to work as intended. The reason being that there would be multiple factors to consider which could potentially quickly become overwhelming if multiple complex layers are to be checked for compatibility.

5
Conclusion

This degree project have incrementally developed a web application which allows users to build, calculate information of interest and visualize stacked products on pallets. A suitable distribution alternative in the form of an Apache HTTP server to host the developed web application to make it accessible to users at the company have been tested. A concept validation of the developed web application has been performed at the company satisfying the objectives set out for the project. The software used throughout the project worked great and proved suitable for the objectives. The languages used for development of the web application also proved suitable for the project. The chosen libraries helped a lot to get more things done quicker in terms of development, especially Three.js that proved very suitable for the 3D rendering that the project had set out as an objective.

References

[1] "EPAL EURO PALLET (EPAL 1)", Accessed: Jun 17, 2024. [Online]. Available: https://www.epal-pallets.org/eu-en/load-carriers/epal-euro-pallet

[2] "Visual Studio Code - Code Editing. Redefined", Accessed: May 21, 2024. [Online]. Available: https://code.visualstudio.com/

[3] "Live Server | VSCode Extension", Accessed: May 29, 2024. [Online]. Available: https://ritwickdey.github.io/vscode-live-server/

[4] "HTML: HyperText Markup Language - MDN Web Docs", Accessed: May 21, 2024. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/HTML

[5] "JavaScript - MDN Web Docs - Mozilla", Accessed: May 21, 2024. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/JavaScript

[6] "CSS - Cascading Style Sheets home page - W3C", Accessed: May 21, 2024. [Online]. Available: https://www.w3.org/Style/CSS/Overview.en.html

[7] "Three.js - JavaScript 3D Library", Accessed: May 21, 2024. [Online]. Available: http://www.threejs.org

[8] "jQuery", Accessed: May 21, 2024. [Online]. Available: https://jquery.com

[9] "jQuery UI", Accessed: May 21, 2024. [Online]. Available: https://jqueryui.com

[10] "blender.org - Home of the Blender project - Free and Open 3D Creation Software", Accessed: May 21, 2024. [Online]. Available: https://www.blender.org

[11] "Inkscape: Draw Freely", Accessed: May 21, 2024. [Online]. Available: https://inkscape.org

[12] "Apache HTTP Server Project", Accessed: May 21, 2024. [Online]. Available: https://httpd.apache.org/

[13] "GitHub", Accessed: Mar. 15, 2024. [Online]. Available: https://github.com/

[14] "Scalable Vector Graphics (SVG) 2", Accessed: May 29, 2024. [Online]. Available: https://www.w3.org/TR/SVG/

[15] J. Persson, "Utveckling av skalbar PLC-styrning till palleteringscell", Jun. 2023. Accessed: May 29, 2024. [Online]. Available: https://odr.chalmers.se/items/b66885fa-0887-4df7-a759-3cf7ce4e0608

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY