CLAYSTER: 3D clay stamp tool

Design project

Evelyn V. van der Sar
Nikita Chulkov
Hakan Isakhanli
Timo van den Berg
Rutger Westgeest
Ali Kaan Uysal

November 6, 2026

Abstract

This report describes the design and development of a web based tool that allows non-technical users to create custom 3D printable clay stamps. It covers the complete design process, including domain analysis, requirements specification, interface design, system architecture, implementation, and testing. The requirements and interface were refined through regular client feedback and usability testing. The resulting system provides stamp customisation, 2D and 3D previews, printability validation, STL export, and an order management dashboard. The final product is delivered to the client in its current state, together with its known limitations and recommendations for future development.

Contents

1 Introduction **4**

2 Domain Analysis **5**
2.1 Introduction to the Domain . . . 5
2.2 Client, Users and interested Parties . . . 5
2.3 Stamps available . . . 5
2.4 Analysing existing solutions . . . 6
2.4.1 stamp3D . . . 6
2.4.2 Mud Stamp . . . 6
2.4.3 CutClay . . . 7
2.4.4 BakePress . . . 7
2.4.5 Cookiecad . . . 8
2.5 Software development environment . . . 8
2.6 LLM usage . . . 8
2.7 Conclusion . . . 8

3 Requirements Specification **9**
3.1 Evolution of the Requirements . . . 9
3.2 Requirements . . . 10
3.2.1 Must have . . . 10
3.2.2 Should have . . . 10
3.2.3 Could have . . . 11
3.2.4 Won't have . . . 11
3.3 Requirements Website . . . 11
3.3.1 Must have . . . 11
3.3.2 Should have . . . 11
3.3.3 Could have . . . 11
3.4 Minimal viable product . . . 12
3.5 LLM usage . . . 12

4 Requirement Analysis **13**
4.1 Activity Diagram . . . 13
4.2 Use Case Diagram . . . 15
4.3 Class Diagram . . . 16
4.4 LLM usage . . . 18
4.5 Conclusion . . . 18

5 Architectural Design **19**
5.1 System Structure . . . 19
5.2 Technology Choices . . . 19
5.3 Stamp-Generation Pipeline . . . 20
5.3.1 Shared Design Representation . . . 20
5.3.2 Validation, Construction, and Export . . . 21
5.3.3 Keeping the Editor Responsive . . . 21

5.4 Orders and Data Storage . . . 21
5.4.1 Data Model . . . 21
5.4.2 Order Submission and Production . . . 22
5.4.3 Storage and Limitations . . . 22
5.5 LLM usage . . . 22
5.6 Conclusion . . . 22

6 Detailed Design **23**
6.1 Interface design process . . . 23
6.1.1 Design Goals . . . 23
6.1.2 Initial Wireframes . . . 23
6.1.3 Client Feedback and Design Revisions . . . 24
6.2 Revised Customer Interface . . . 25
6.2.1 Stamp Editor . . . 25
6.2.2 Design Review . . . 25
6.2.3 Order Details . . . 25
6.2.4 Order Confirmation . . . 27
6.3 Frontend Implementation . . . 28
6.3.1 Editing and Validation . . . 28
6.3.2 2D and 3D Previews . . . 28
6.3.3 Review and Ordering . . . 30
6.3.4 Responsive Design . . . 30
6.3.5 Accessibility Considerations . . . 30
6.4 Client Interface . . . 32
6.4.1 Dashboard Overview . . . 32
6.4.2 Order Management . . . 32
6.4.3 Product and Production Management . . . 32
6.5 Design Limitations . . . 34
6.6 LLM usage . . . 34
6.7 Conclusion . . . 34

7 Testing the system **35**
7.1 Test plan . . . 35
7.1.1 Usability testing . . . 35
7.1.2 Unit tests . . . 36
7.1.3 Client testing . . . 40
7.1.4 Acceptable risk . . . 40
7.2 Test results . . . 40
7.2.1 Usability testing results . . . 40
7.2.2 Unit testing results . . . 42
7.2.3 Client testing results . . . 42
7.3 LLM usage . . . 43

8 Future Planning **44**
8.1 Deployment . . . 44
8.2 Manual . . . 44
8.3 Future Improvements . . . 44
8.4 Maintenance . . . 44
8.5 Conclusion . . . 44

9 Evaluation **45**
9.1 Planning . . . 45
9.2 Responsibilities . . . 45
9.3 Team Evaluation . . . 46
9.4 Final result . . . 46
9.5 Conclusion . . . 46

A AI-disclosure **48**

B Risk analysis **49**

Chapter 1

Introduction

This project follows the longstanding tradition of impression seals, which have been used since at least the sixth millennium BC [2]. Although these seals have existed for thousands of years, the basic idea behind them has remained largely the same. By pressing a shape or pattern into a material to leave an impression. Today, stamps are still commonly used in pottery and ceramics. Where artists use them to add letters, logos, patterns, and other decorative details to their work.

Our project also has some similarities to the development of the Linotype machine. Which mechanized the assembly of movable type by casting complete lines of text as single printing slugs [4]. Similarly, our goal is to replace the need for individual letter stamps with a complete, personalized stamp. Instead of having to arrange and use several separate stamps to create a word or design. The customers should be able to create everything as one piece. This also provides more flexibility, as the size, arrangement, and appearance of the stamp can be customized according to the customer's needs.

This project is offered by Clayster, an upcoming online ceramic shop that aims to provide creative tools and materials for ceramic artists. Clayster wants to offer a tool that allows customers to design their own 3D-printed clay cutters and pottery stamps. Ceramic artists can use these tools to create small but detailed shapes and impressions that would otherwise be difficult or time-consuming to produce by hand. Since ceramics is a creative hobby, different artists often have different preferences and specific requirements for their designs. For example, one customer might want a simple stamp containing their initials, while another might want a more complicated logo or a decorative pattern.

The main idea is to make the process of creating these customized stamps as simple and accessible as possible. Customers should not need experience with 3D modelling software to design their own stamps. Instead, they should be able to use a website where they can enter text, choose a font, adjust its size, and change additional properties such as letter thickness, spacing, and depth. The tool also supports working with uploaded designs, allowing customers to create stamps that are not limited to text alone. A 2D editor and a 3D preview help users see what their design looks like before producing it.

An important part of the project is making sure that the resulting designs can actually be 3D-printed. A design that looks good on a screen might still have problems when printed, especially if some parts are too thin or contain complicated geometry. Therefore, the tool also needs to check the design for potential printing issues and provide feedback to the customer. Once the design is finished and validated, the tool should generate a file that can be used by a 3D printer to produce the personalized stamp.

In this report, we will go over the different steps of the design project, from the initial ideas and requirements to the development and testing of the tool. We will explain the decisions made during the process, the problems we encountered, and how we approached solving them. We will also discuss the role of Large Language Models (LLMs) throughout the project, including which parts of the design and development process they were used for and how much they contributed to the final result.

The source code of Clayster is stored in the University of Twente GitLab repository `https://gitlab.utwente.nl/design-project-2026/clayster`. Instructions for building and running the system with Docker are in the `README` of the repository.

Chapter 2

Domain Analysis

In this chapter, we will discuss the domain of the system and look and compare existing solutions. This way we can understand the context of our project better. This helps with avoiding making the same mistakes, and helps with development.

2.1 Introduction to the Domain

Client makes ceramic products using the process of pottery. This means firing clay and other raw materials at high temperature. But first the client shapes the clay in the form that they desire. This can make different kinds of objects. These objects sometimes have engraving, like letters or words. Or have different forms on them.

2.2 Client, Users and interested Parties

Our client makes ceramic products for users. Client offers a service to add a custom engraving into the ceramic. To make it more custom. The client and the users should be able to make a custom stamp with our project. We also take nature as a central stakeholder. This can help with understanding their operational impact on earth natural systems and the planetary boundaries. [3, 1]

2.3 Stamps available

We had some stamps available from the client as reference point. See figure 2.1.

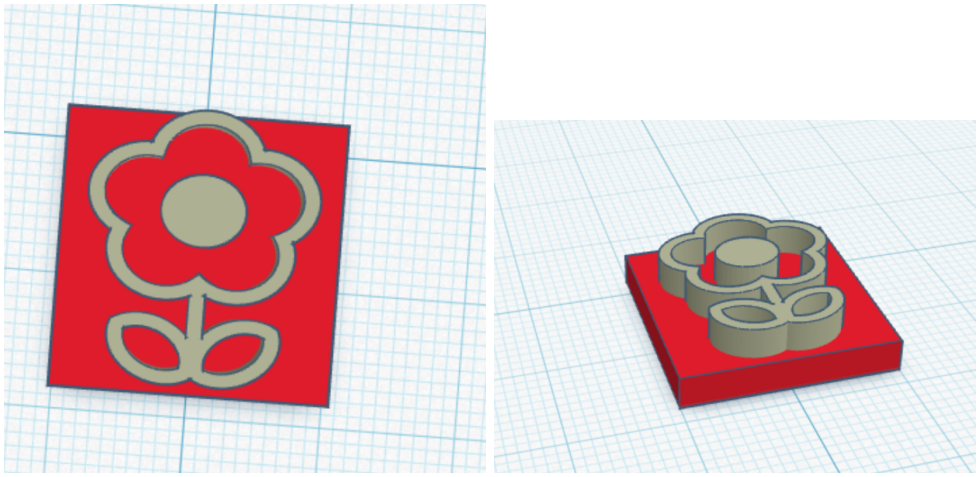


Figure 2.1: Stamps available

2.4 Analysing existing solutions

There are many existing solutions to make 3D file for a stamp or a cutter. We found 2 free and 3 freemium solutions. We will analyse these solutions with pros and cons. None of these solutions fit the wishes of the client.

2.4.1 stamp3D

pros

- Works well with uploaded pictures
- Simple but powerful website
- Free
- Has guides

Cons

- No 3D preview
- Too much vertical scrolling
- Too much textual information as you scroll through the website

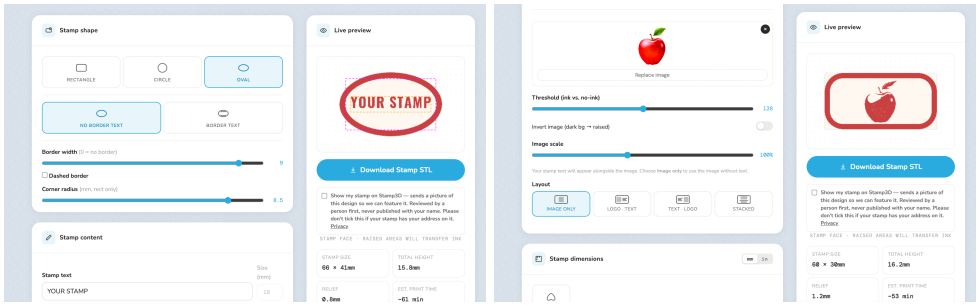


Figure 2.2: Stamp3D website

Licence: "Provided as-is for personal and hobby use. No warranty expressed or implied. You own everything you make with this tool." https://stamp3d.com

2.4.2 Mud Stamp

pros

- 3D preview on the end of the stamp
- Simple to understand
- Free

cons

- Can't move around the design on the stamp, clicking and dragging changes the camera angle

There is no licence on this website. https://mudstamp.com



Figure 2.3: Mud Stamp website

2.4.3 CutClay

pros

- Very professional looking
- A lot of shape types to choose from
- Great 3D preview
- Can create Cutters and imprint cutters
- Provides extensive manual

cons

- Allows users to create up to 10 cutters for free, after that it is paid
- Low usability due to too much information
- Very easy to get lost, too many details on the screen at once

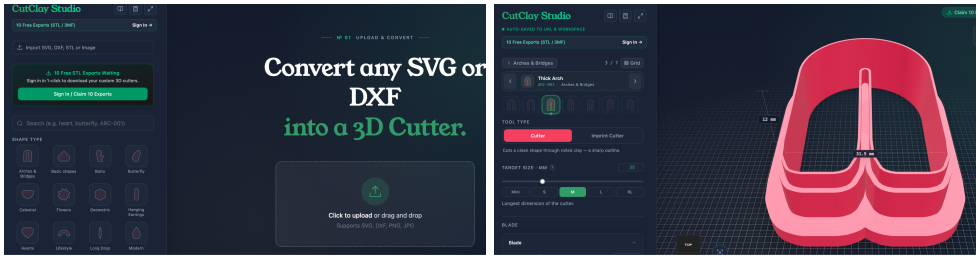


Figure 2.4: CutClay website

There is no licence on this website. https://cutclay.com

2.4.4 BakePress

pros

- Simple, easy to use
- Clean output
- Can create cutters and stamps

cons

- Account needed
- Freemium

There is no licence on this website. https://bakepress.com

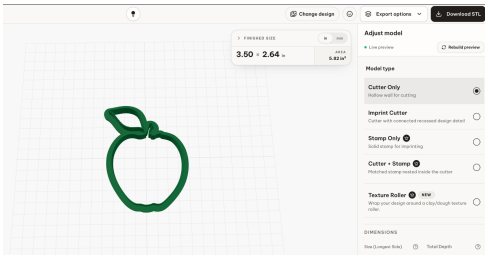


Figure 2.5: BakePress website

2.4.5 Cookiecad

pros

- Can create a lot of types of cutters and stamps
- Wide range of available custom designs to choose from

cons

- Freemium, some advanced features are locked
- Options to just have text not immediately visible to the user.

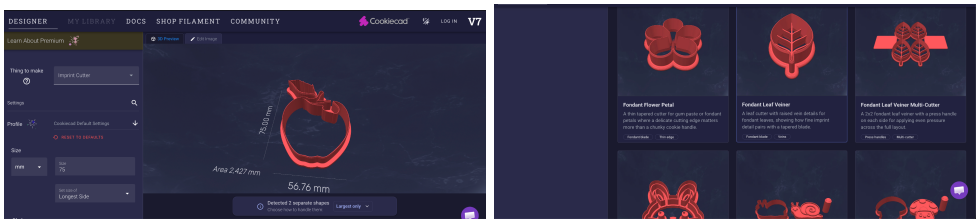


Figure 2.6: CookieCAD website

There is no licence on this website. https://app.cookiecad.com

2.5 Software development environment

From the first meeting it was clear that the client didn't have wishes about the technologies used. Therefore, we choose technologies that have existing solutions for our requirements. A short overview of the technologies used: we run the system in a docker on Linux. Django is used for the web framework, and we used JavaScript for the tool. For the database we use SQLite. More details about these choices and these technologies can be found in section 5.2.

2.6 LLM usage

For the research of this part we only used level 1, we did our research with search engines.

2.7 Conclusion

By analysing the domain processes are identified, and existing solutions are analysed. We found that none of the existing solutions satisfy all the wishes of the client. Moreover, the stakeholders are identified and taken into account for the development.

Chapter 3

Requirements Specification

In this chapter, we will discuss the requirements of the system. We have compiled these throughout the whole design project and updated where needed. During the meetings that we had with the client, we have got a lot of feedback and wishes. We tried to include as much in the requirements within the scope of the project.

3.1 Evolution of the Requirements

The requirements in this chapter are the final, approved set. These are not the requirements the proposal started with. We wrote the first version at the start of the project, before we knew the precise priorities of the client, and before we had tried to generate 3D geometry ourselves. During the weekly client meetings (Section 7.1.3) we learned which wishes mattered most, which ideas were out of scope, and which of our assumptions did not hold. We therefore treated the requirements as a changing document and updated them after meetings.

A change on the requirements was always one of the following:

- **Added**: a new wish from the client, or a need that became clear while building.
- **Removed**: a requirement that was out of scope, replaced by a solution or no longer wanted
- **New Priority**: a requirement that moved to another MoSCoW category.

Table 3.1 lists the changes recorded, together with the original category and the reason. In the last 4 weeks of the project we froze the requirements, so that we could deliver a stable version with a clear manual.

Table 3.1: Recorded changes to the requirements during the project

Requirement	**Original category**	**Change**	**Reason**
Real-time rendering	Could	Added	*Real-time rendering for 2D was added, whilst 3D is almost instantaneous but not real-time for performance improvement.*
Logo file upload for the stamp	Could	Added	*Uploading a logo file turned out to be a feature that would really improve the tool.*
Email the 3D printer file to the client	Must	Removed	*Client mentioned that this would not be necessary any more*
Filtering on the dashboard	Should	Removed	*Only by status now. Implementation and deployment time*
Notify the client of a new order	Should	Removed	*Email not deployed*
Email the STL file to a customer	Could	Removed	*Client want to only have STL files*

Requirement	Original category	Change	Reason
Suggest existing models to the user	Could	Removed	*Out of scope for time*
AI model to assist in generating designs	Could	Removed	*Out of scope for time*
Universal handle for all stamps	Could	Removed	*Client print quality lowered with it.*
Automatically add a handle to the print	Could	Removed	*Client print quality lowered with it.*
Customer can create an account	Should	Removed	*Not needed for ordering; orders are opened through a private link*
Customer can view past orders	Should	Removed	*Depends on customer accounts*

3.2 Requirements

In this section, we will discuss general requirements of the project. We used MoSCoW to sort these requirements and to see what we will prioritize.

3.2.1 Must have

1. The system must have basic printability validation (minimum wall thickness/depth, max size).
2. The system must warn for impossible to print combinations or avoid them entirely.
3. The system must allow users to enter a letter, name, or short text.
4. The system must allow for the user to specify the space in between the letters, thickness of letters, alignment, and size of letters.
5. The system must allow users to choose from a predefined set of supported fonts.
6. The system must export to an STL file.
7. The system must be available on Firefox and Chrome.
8. The system must have a 2D/3D preview of the stamp on the clay item.
9. The system must have a 2D/3D preview of the stamp itself.
10. The system must support receiving a custom font from the user.

3.2.2 Should have

1. The system should have a dashboard.
2. The system should allow the client to download the STL file from the dashboard.
3. The system should display all orders on the dashboard.
4. The system should allow filtering for the dashboard.
5. The system should allow to give orders a status (e.g. complete, printing, not started).
6. The system should have a database which records orders.

3.2.3 Could have

1. The system could have an option to provide clay cutters.
2. The system could be optimized for mobile devices.
3. The system could have real time rendering.
4. The system could have a support to autosave the user's work, so progress is not lost when site is abruptly closed.
5. The system could have support for logo file upload for the stamp.

3.2.4 Won't have

1. The system will not support every possible 3D printer or printing technology.
2. The system will not support complex uploaded 3D models.
3. The system will not have an option to choose the colour.

3.3 Requirements Website

Based on the first set requirements and our expected timeline we decided to add a website dashboard that will continue from the order requirements.

3.3.1 Must have

1. The website must have an overview of all orders.
2. The order must have an option to mark as finished.
3. The order screen must have a button to download the STL file.
4. The order must show the delivery details.
5. The order must show the contact details of the customer.
6. The customer must be able to order a stamp using the tool.
7. The client must be able to log in.

3.3.2 Should have

1. The website should have an option to filter orders based on their status.
2. The customer should be able to view past orders.

3.3.3 Could have

1. The system could have an option to pay.
2. The system could generate a custom invoice per order.
3. The system could have an option to edit the password.
4. The system could show an update for the delivery.

3.4 Minimal viable product

Together with the client, we agreed on the following requirements for the minimal viable product (MVP). The requirements are ranked on priority.

- A canvas/input field where the customer can freely type the text they want, with a choice of font (from a small preset library) and size.
- One style: stamp (raised/recessed relief) with a fixed, printed-in handle.
- Preview of both the expected result in clay and the printable object itself.
- Basic printability validation (minimum wall thickness/depth, maximum size).
- Export to an STL file.

We added additional requirements for the MVP to include a dashboard for the client. The following list contains the minimum requirements.

- A dashboard that shows all orders in an overview, where orders can be marked with several relevant status codes.
- An option to download the STL file.
- Orders contain all necessary information needed to ship the order (e.g. name, address, and contact information).

3.5 LLM usage

No AI has been used with interviewing the client, nor making the requirements and ranking the requirements.

Chapter 4

Requirement Analysis

This chapter examines how the requirements translate into the behaviour of Clayster. We used three diagrams to discuss the workflow with the client: an activity diagram for the sequence of tasks, a use case diagram for the user interactions, and a class diagram for the structure of the backend data.

These diagrams record the main workflow considered during requirements analysis. Some technical responsibilities changed during implementation. These differences are explained below, while the resulting architecture is described in Chapter 5.

4.1 Activity Diagram

Figure 4.1 shows the proposed workflow from creating a stamp to making its file available for production. The swim lanes distinguish the customer, the interface, server processing, data storage, and the client.

The customer starts by entering text or uploading artwork and adjusting the design. Validation provides feedback on problems that require attention. If a blocking error occurs, the customer must change the design before continuing. Once a preview is available, the customer can inspect it and either return to editing or proceed with the order.

This return path is important because a design can pass the implemented checks without meeting the customer's expectations. Validation and customer approval therefore have different purposes: validation checks configured constraints, while the customer checks the content and appearance.

After submission, the system stores the order information and connects the generated stamp file to the order item. The client can then open the order, view the customer and delivery details, and download the STL file for production.

The diagram reflects the original allocation of technical responsibilities. It places validation and STL conversion in the backend lane. In the current implementation, the browser performs the geometry checks and generates the STL before submission. The server checks the submitted information and file, calculates the price, and stores the order. The STL downloaded by the client is therefore the file submitted with the order, rather than a model generated when the client selects it for printing.

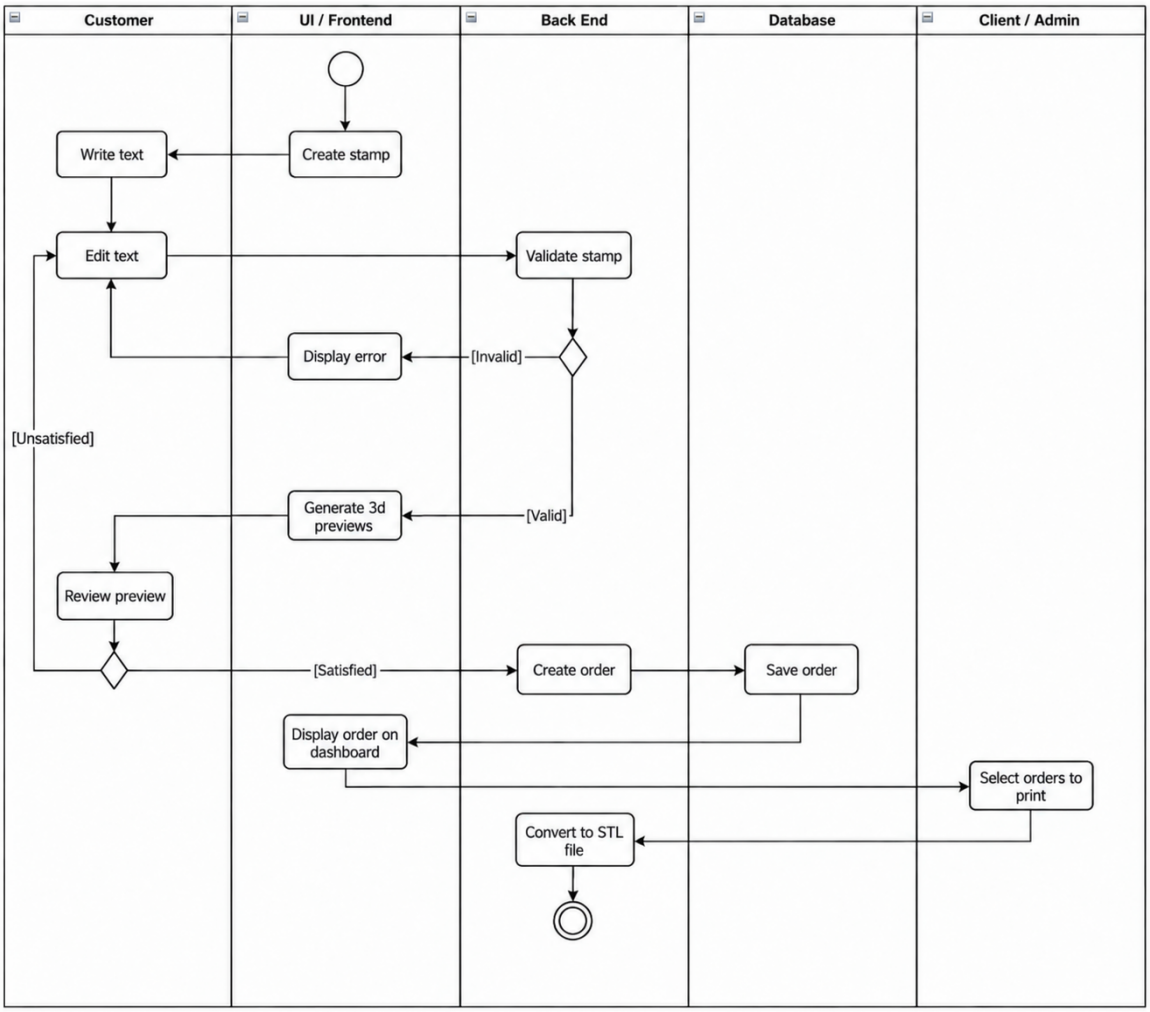


Figure 4.1: Activity diagram used during requirements analysis

4.2 Use Case Diagram

Figure 4.2 shows the main interactions between the two human roles and Clayster. The customer creates a stamp, while the client manages submitted orders and accesses the files needed for production.

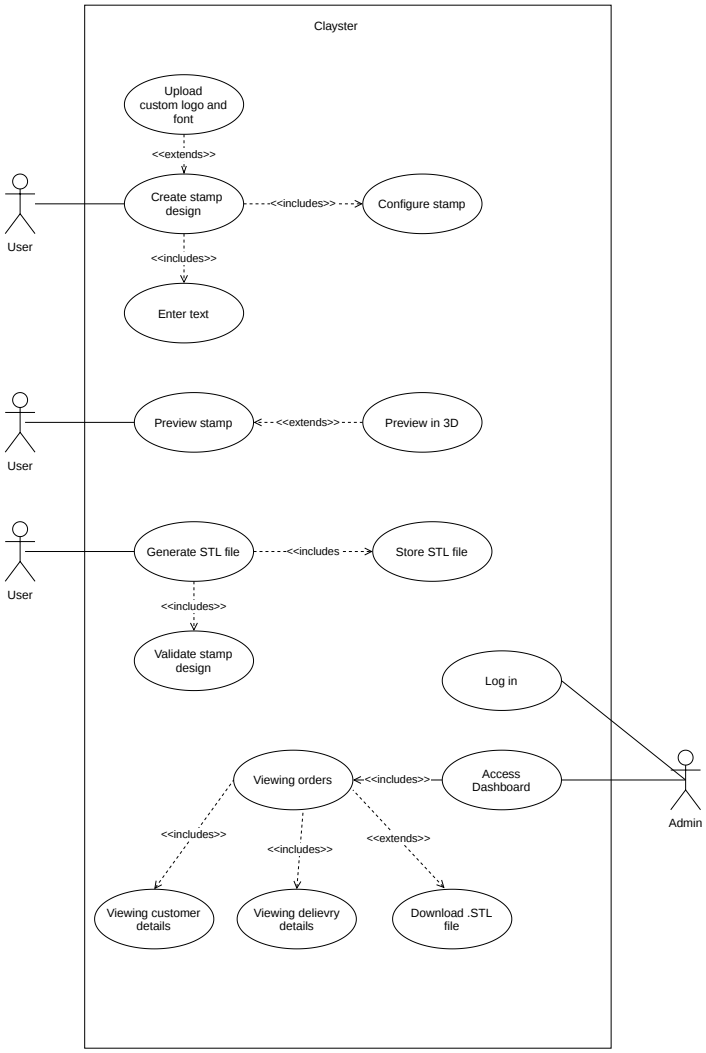


Figure 4.2: Use case diagram used during requirements analysis

The customer's main tasks are creating and configuring a design, inspecting the preview, and generating a printable file. Text can be configured through its font, size, spacing, alignment, and thickness. Uploaded artwork and custom fonts provide additional ways to personalise the stamp.

The diagram models the 3D preview as an optional extension of the standard preview. Customers can use it to inspect the physical stamp and its expected impression on clay. Validation is required before a file can be accepted for export or ordering.

The client logs in to access the dashboard and order information. Opening an order provides access to its items, customer details, delivery address, and generated STL files. Keeping this information together connects the customer's design to the client's production and delivery tasks.

The diagram focuses on design creation and access to submitted files. Two distinctions are needed when relating it to the implementation. First, text entry is one possible input route: a design can also consist of uploaded artwork without text. Second, generating an STL does not always require server

storage. Only placing an order submits the file for storage with the order.

The complete ordering interaction includes reviewing the design, entering the required details, and submitting the order. Its interface is described in Chapter 6.

4.3 Class Diagram

Figure 4.3 provides the original planned class diagram, which has changed throughout the project development.

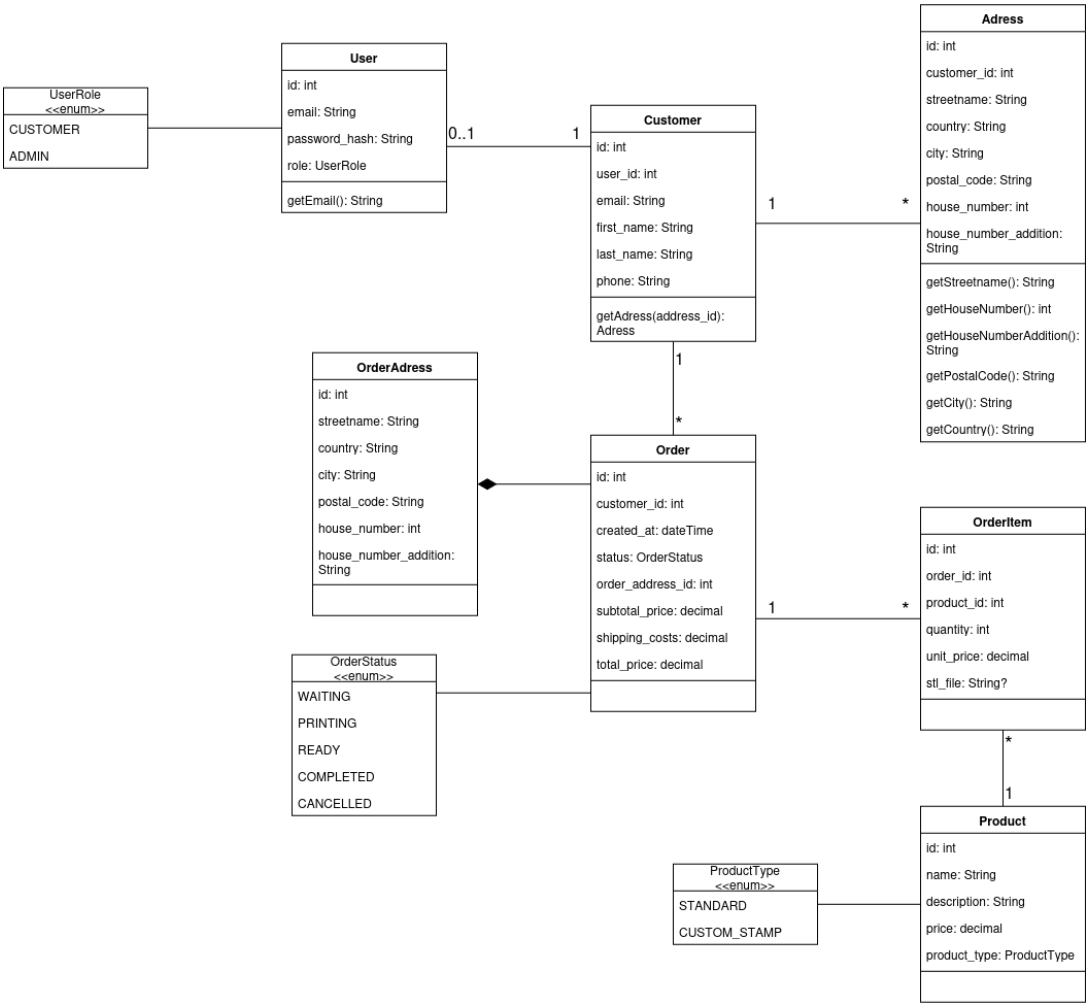


Figure 4.3: Original planned class diagram for backend

Figure 4.4 shows how the final class diagram looks after further development. This diagram has been generated from the code base using Django-extensions (graph models) and Graphviz. The version that was generated from the code base was then refined to include the proper relations to display the final diagram below.

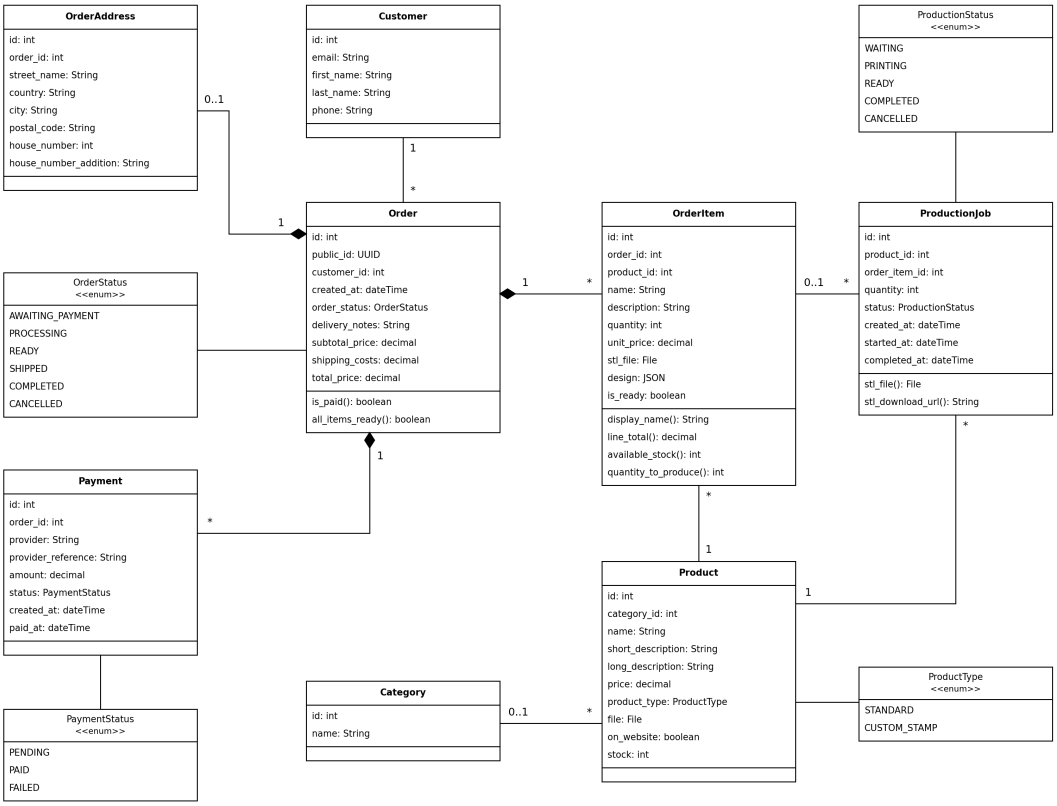


Figure 4.4: Final class diagram for backend generated from code base

Our first class diagram was made before we started working on the backend. It was meant to be as guide that we can use to see how the data shall fit together. But during development, the class diagram changed. Some of the classes ended up differently or removed. A couple of new classes were introduced as well. These were introduced as feedback from the client, and with the implementation of other requirements.

These changes include removing customer accounts, as they were not part of our must-have requirements. So, this means that there is no longer a UserRole and User class included for logging in. Also, we no longer save the address with the users log in but rather with the order itself in OrderAddress.

We introduced a new ProductionJob class, which tracks the printing of the item separately from the order. Payment which records the payment attempts of the user. Although the payment module is still a placeholder. And finally Category, which groups the products together, which allows the client to filter through the orders using the production status.

In some cases the relations have changed as well. Order still contains many OrderItems, however it is now a composition since the order items get deleted with the order. The same applies for address and payments. Also, ProductionJob is now connected with both OrderItem and Product, with the link to OrderItem being optional. It is optional because the job can also be made for stock.

4.4 LLM usage

No AI tools were used to create the diagrams in the requirements analysis. ChatGPT was only used to assist with LaTeX formatting, such as positioning the diagrams correctly within the document. Claude was also used for guidance, on how to generate the class diagram from the code base using Django-extensions and Graphviz.

4.5 Conclusion

The analysis identified three important conditions for the workflow. Customers need a way to return to editing when a design fails validation, they must be able to review an accepted design before ordering, and the client must receive the correct printable file together with the order information.

The diagrams helped describe these relationships from different perspectives. The activity diagram shows the sequence of work, the use case diagram identifies the main user tasks, and the class diagram was used as a guide for the development of the backend. The following chapters explain how these requirements were translated into the architecture and interfaces.

Chapter 5

Architectural Design

This chapter explains how Clayster turns a customer's design into a printable stamp and connects it to an order. It covers the browser and server responsibilities, geometry processing, and storage decisions.

5.1 System Structure

Clayster uses a JavaScript editor in the customer's browser and a Django application on the server, as shown in Figure 5.1. The browser converts artwork into geometry, produces previews, validates the stamp, and generates the STL. Django serves the website and manages orders, submitted files, products, and production jobs.

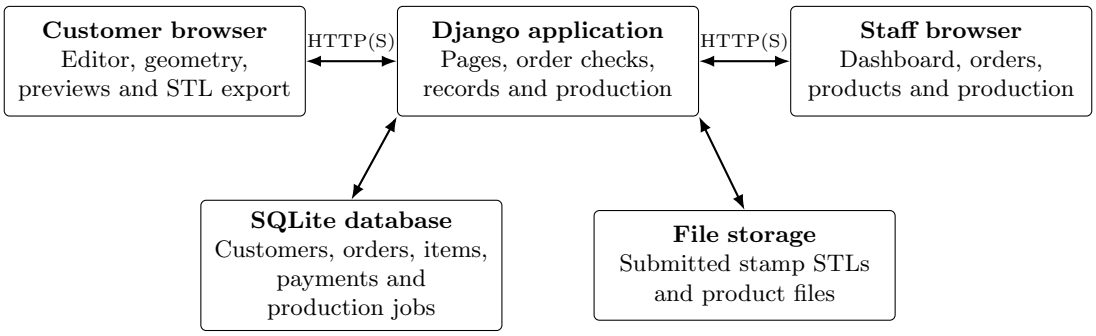


Figure 5.1: Main components and communication paths in Clayster

Processing the stamp on the browser-side avoids a server request for every edit and reduces the server's workload. The trade-off for this is that we assume that the customer's computer can handle all the calculations.

The code makes use of Django applications, creating different sections for customers, products, orders, production, the dashboard, and the stamp editor. This is common when using with Django, and improves the maintainability. The applications share one database and one deployment.

5.2 Technology Choices

Table 5.1 lists the technologies used for the main processing and storage tasks.

The printable stamp uses solid operations and export checks because a mesh that renders correctly is not necessarily suitable for printing. The clay preview is display geometry and uses a separate construction path.

Django serves the JavaScript modules through an import map pointing to local dependencies. Node.js prepares these dependencies and runs tests. The Docker image runs Django through Gunicorn, with WhiteNoise serving static assets.

Table 5.1: Main technologies and their architectural roles

Technology	Role
Django	Handles requests, templates, form validation, staff authentication, and database access in Python.
Three.js	Renders 3D previews and provides STL loading and export tools.
OpenType.js	Converts text and fonts into glyph outlines.
polyclip-ts	Combines and repairs 2D polygons before stamp construction.
Manifold	Constructs printable solids through extrusion, offsets, and boolean operations, using WebAssembly.
three-bvh-csg	Performs mesh boolean operations for the clay-impression preview.
three-mesh-bvh	Supports mesh operations and intersection checks used in export validation.
SQLite	Stores relational data without a separate database service.

Django was chosen because some people in the group already had experience with it and everyone in our group is able to use python. Flask was considerd but Django was choisen because of its full-stack framework and easy to use built in tools. Thus saving time in development. The stamp needs to be generated in the browser so it is able to give instant feedback while editing and requires no server load. SQLite was used because of it's easy implementation with Django and there was a small expected order volume. Also maintainability was considered here, where SQLite is preferred for the client.

5.3 Stamp-Generation Pipeline

Figure 5.2 shows how text and images become stamp geometry. Stamp settings determine the backing, relief, mirroring, and handle. Only the stamp is exported.

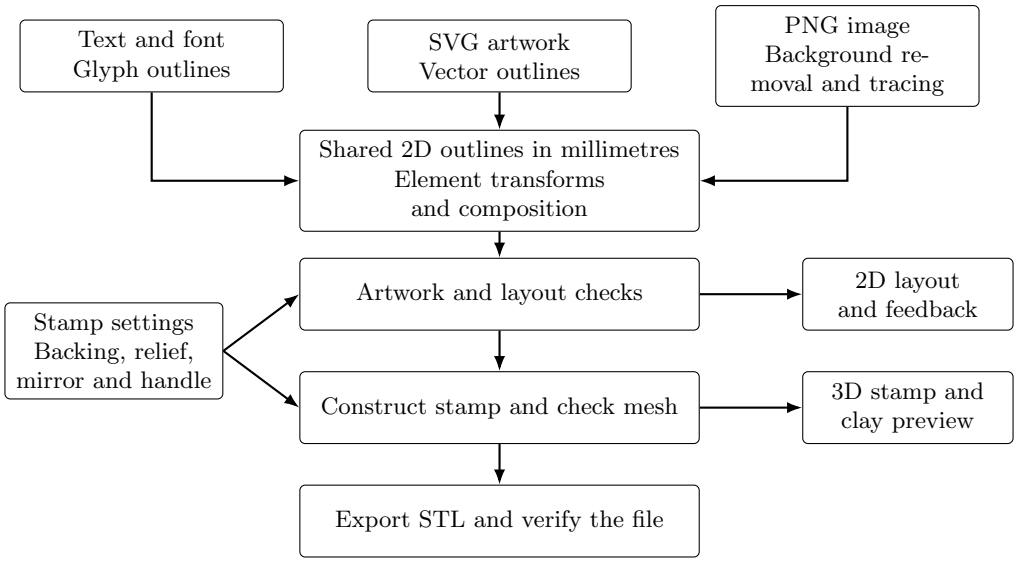


Figure 5.2: Main geometry stages; full 3D construction can be deferred during 2D editing

5.3.1 Shared Design Representation

The editable design stores each element's source, position, scale, rotation, and other settings separately from the generated mesh. This allows changes to regenerate the model without editing its triangles directly.

Visible elements are converted into a shared polygon representation in millimetres, called canonical geometry. Text becomes glyph outlines, SVG becomes filled outlines, and PNG images are traced after

background processing. Element transforms are then applied and the artwork is combined. This shared representation preserves holes and keeps size and placement consistent between the 2D editor and the 3D stamp.

5.3.2 Validation, Construction, and Export

The pipeline rejects unsupported input and checks and repairs the combined 2D artwork where possible. Feature-width and gap checks identify small details, while layout checks detect overflow and excessive stamp dimensions. Blocking errors prevent export; warnings communicate possible printing concerns.

Manifold constructs the backing, relief, and optional handle. The resulting mesh is checked for open or inconsistently oriented edges, collapsed triangles, invalid shell connections, and non-positive volume. Export writes the stamp as binary STL, reloads it, and repeats the surface checks while verifying its dimensions and triangle count. Local downloads and order submission use the same export function.

The clay preview approximates an impression on a mug and is excluded from export. It does not simulate clay deformation. Physical printing and use in clay remain necessary to assess the result.

5.3.3 Keeping the Editor Responsive

The pipeline reuses element conversion and measurement results when their inputs are unchanged. During 2D editing, it updates layout and feedback without constructing the full 3D stamp. A full build is requested for the 3D view, review, or export. Rapid edits are grouped through a short delay, and older asynchronous runs cannot overwrite newer results.

5.4 Orders and Data Storage

The data is stored in a local SQLite database generated by Django. The database tables are generated using Django Models, which allows the system to communicate with the database using objects in Python.

5.4.1 Data Model

Figure 5.3 shows the main database relationships. An order groups items, each referring to a product and recording its quantity and price. Payments and the delivery address are stored separately.

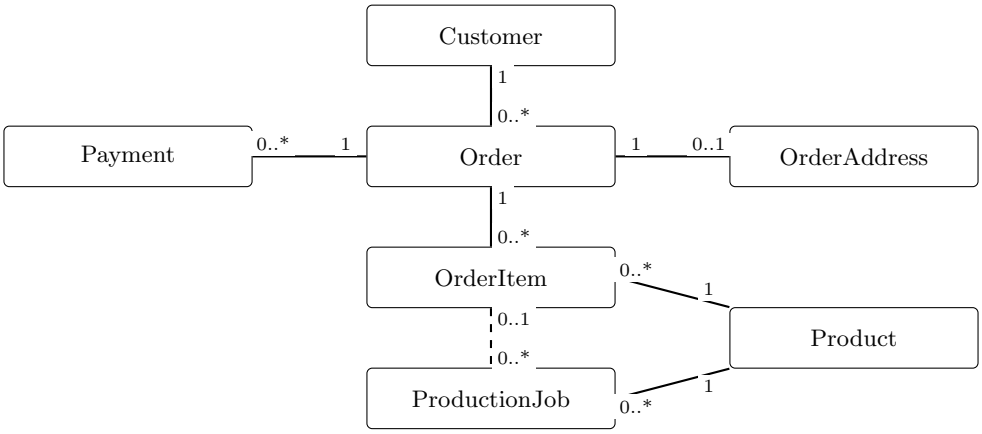


Figure 5.3: Main database relationships; 0..* means zero or more. Product categories are omitted.

Each custom order item holds its personalised STL and design settings. Standard products can use a shared product file and stock records. Production jobs link to a product and optionally to an order item, supporting both customer orders and stock production.

The diagram describes the schema: an order can exist without items or an address, but public submission creates both. The schema also permits several orders per customer; the current public API creates a new customer record for each submission.

5.4.2 Order Submission and Production

The browser submits order data as JSON together with the STL. Django validates the form fields and relevant stamp settings and calculates the price independently. It also checks the STL's structure, coordinates, dimensions, and 10 MiB file-size limit. These checks do not repeat the browser's full geometry validation or regenerate the stamp from its source description.

Order creation groups the database writes in a transaction and sets the status to *Awaiting payment*. Accepted payment changes the status to *Processing* and creates production jobs for custom stamps. Payment and email modules provide connection points for external services, but currently simulate payment and log confirmation emails without delivering them.

5.4.3 Storage and Limitations

SQLite stores the records, while file storage holds the STLs. Custom-stamp uploads receive random filenames. Both the database and files are preserved in a Docker volume and must be included in backups. A random order link gives customers access to limited personal information; anyone holding it can access the page.

The order retains design settings and source filenames, but not uploaded SVG, PNG, or font assets. This prevents complete design restoration. The editable design remains in browser memory and is lost on reload.

5.5 LLM usage

AI tools such as ChatGPT were used to assist in creating and formatting the graphics used as figures in the LaTeX report. The architectural content, design decisions, and accuracy of the figures were determined and verified by the Clayster team.

5.6 Conclusion

Browser-side generation supports interactive editing, while Django connects the resulting file to orders and production. Shared geometry and export checks support consistency between the design and its STL. The remaining architectural limitations are incomplete design persistence, narrower server-side geometry checks, and placeholder payment and email services.

Chapter 6

Detailed Design

This chapter describes how we developed the customer and client interfaces of Clayster. We explain the initial wireframes and the changes implemented after client feedback, and how these designs were implemented.

The customer interface supports the creation, review, and ordering of a custom stamp. The client interface supports order management and production. The design process considered both workflows, since the stamp created by the customer has got to reach the client together with the information needed to produce and deliver it.

6.1 Interface design process

We followed an iterative process to design the customer and client interfaces. We first translated the project requirements into the main tasks that each user needed to complete. For customers, these tasks included, but were were not limited to, creating a stamp, reviewing the result, entering order details, and confirming the order. For the client, they included reviewing submitted orders and managing the production workflow.

Based on these tasks, we divided each workflow into separate screens and created low fidelity wireframes in Balsamiq. The wireframes were used to Explore the layout, the order of the steps, and the information required at each stage. Desktop and mobile layouts were considered for the customer workflow because the interface should remain usable at different screen sizes.

The initial wireframes were presented to the client for feedback. Based on the feedback session, we then revised the designs to clarify validation messages and prevent customers from continuing with blocking errors, and remove repeated information from the review screen. The revised wireframes formed the basis for the frontend implementation. However, multiple adjustments were made when the interface was connected to the stamp generation and ordering functionality.

6.1.1 Design Goals

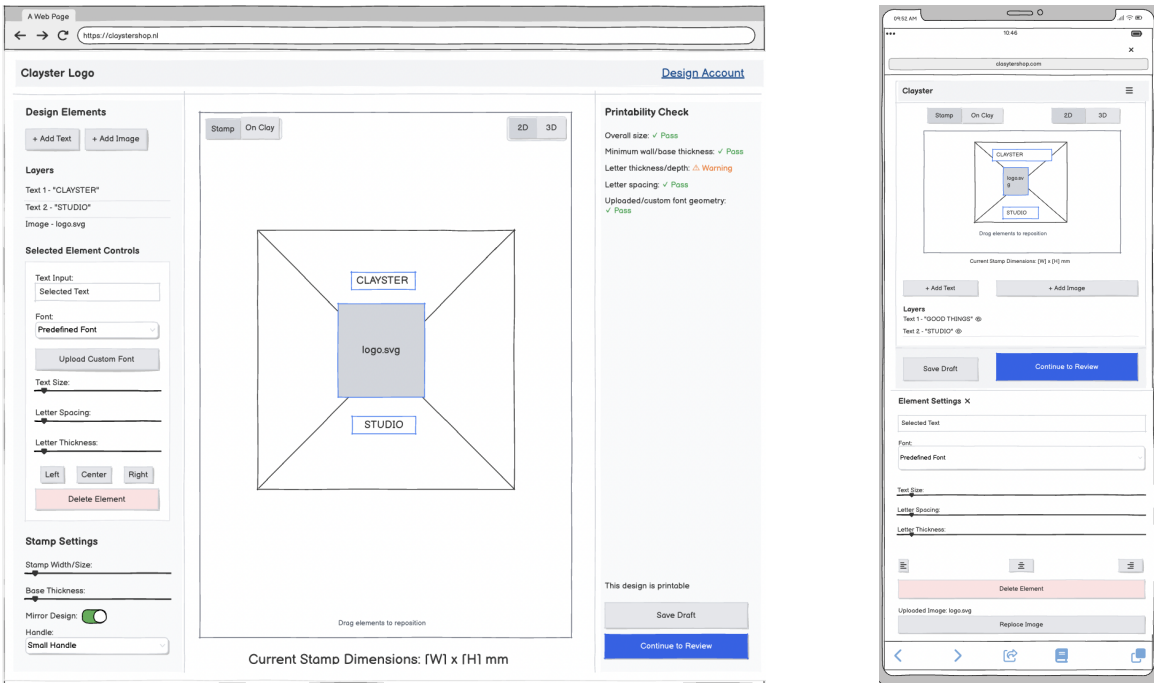
The tool is intended for customers without experience in 3D modelling. We therefore based the editor on familiar actions, such as entering text, uploading images, and dragging or resizing elements. The tool handles the geometry required to turn this artwork into a stamp.

A second goal was to provide feedback during editing. Customers should be able to identify and correct problems before placing an order. The interface therefore distinguishes between blocking errors and warnings about possible printing problems. This was one of the direct client requirements, in which she demanded that progression into the review screen should be disabled when errors were present.

6.1.2 Initial Wireframes

We created the initial wireframes in Balsamiq to discuss the layout and workflow before implementation. They showed where customers would enter their design, inspect the preview, receive validation feedback, and continue to ordering.

Figure 6.1 shows the initial desktop and mobile editors. The desktop version places the controls beside the preview, while the mobile version arranges them vertically.



(a) Desktop (b) Mobile

Figure 6.1: Initial stamp editor wireframes

The initial review screen repeated the printability checks alongside the preview and design information. This gave the customer another opportunity to inspect the result, but also repeated information already shown in the editor. Figure 6.2 shows this first version.

6.1.3 Client Feedback and Design Revisions

We presented the wireframes to the client and revised them based on the feedback. Three changes affected the customer workflow.

First, the client wanted errors that prevent a valid stamp from being generated to block further progress. The revised editor therefore makes these errors visible and prevents the customer from continuing until they are corrected. Warnings remain available for concerns that do not prevent model generation.

Second, the client suggested allowing customers to enter the available width and height on their clay object. These fields were included in the revised wireframe to support a comparison with the stamp dimensions.

Third, the review wireframe was simplified by removing the repeated printability checks. Its main purpose became checking the final preview and design summary before entering the order details.

These revisions formed the second design iteration. The next section describes the revised wireframes; Section 6.3 explains the implementation and the remaining differences.

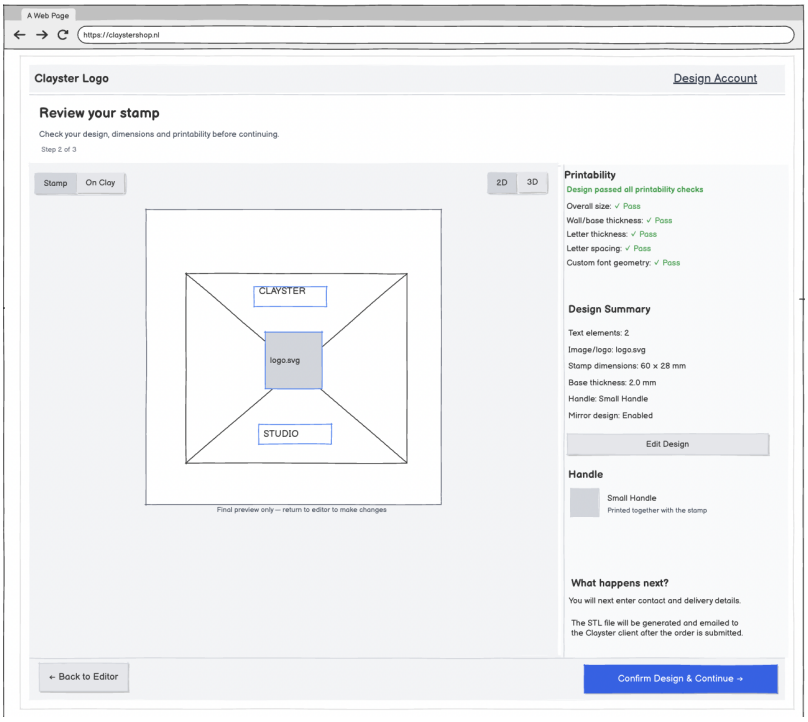


Figure 6.2: Initial design review wireframe

6.2 Revised Customer Interface

After the reviews we created these revisions.

6.2.1 Stamp Editor

The revised editor keeps the design controls, preview, and validation feedback together. Customers can enter text, select or upload a font, add artwork, and adjust its appearance. Problems are shown while they work, rather than only when they attempt to continue.

The proposed fields for the available space on the clay object appear alongside the design settings. This was intended to help customers judge whether the stamp would fit their object. These fields belong to the wireframe design and are not implemented as a separate fit check. Figure 6.3 shows the revised editor.

6.2.2 Design Review

The revised review screen focuses on the stamp preview, dimensions, and selected settings. Editing controls are removed, so the customer can check the result before continuing. If changes are needed, the customer can return to the editor. Following client feedback, the detailed validation checks were removed from this wireframe. Figure 6.4 shows the resulting design.

6.2.3 Order Details

The order details screen collects the quantity, contact details, and delivery address. A stamp summary remains visible, so the customer can check what they are ordering while completing the form.

The desktop wireframe places the form and summary beside each other. The mobile version arranges the same information vertically. Figure 6.5 shows both layouts.

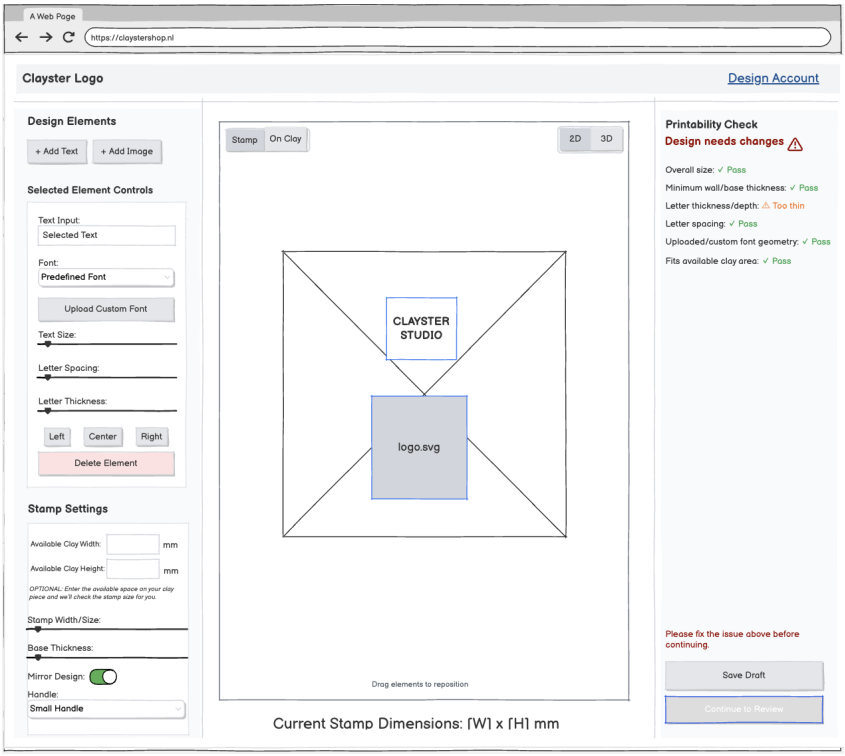


Figure 6.3: Revised stamp editor after client feedback

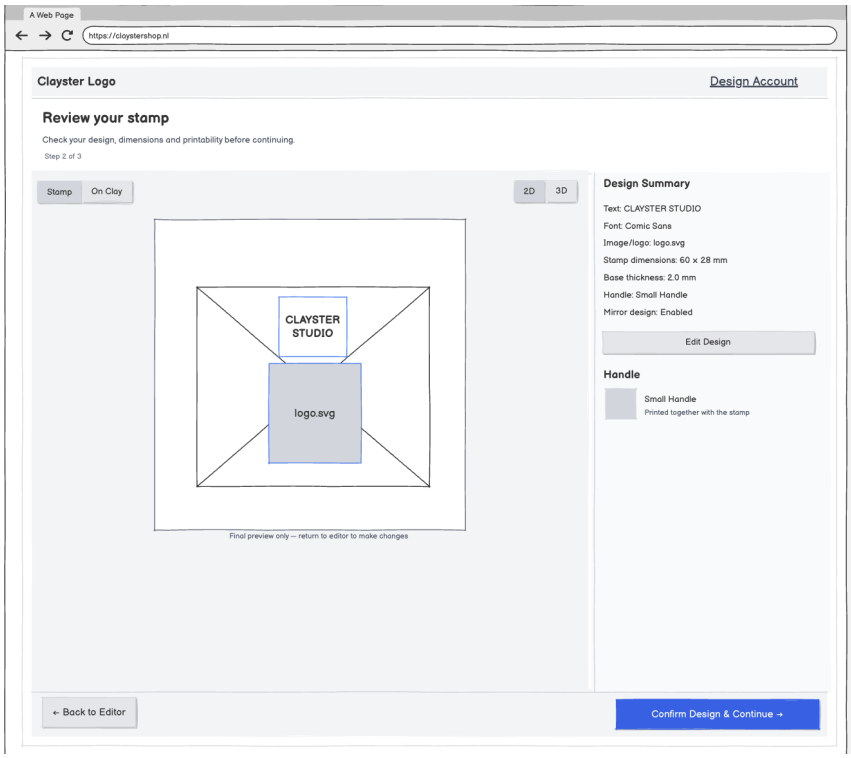
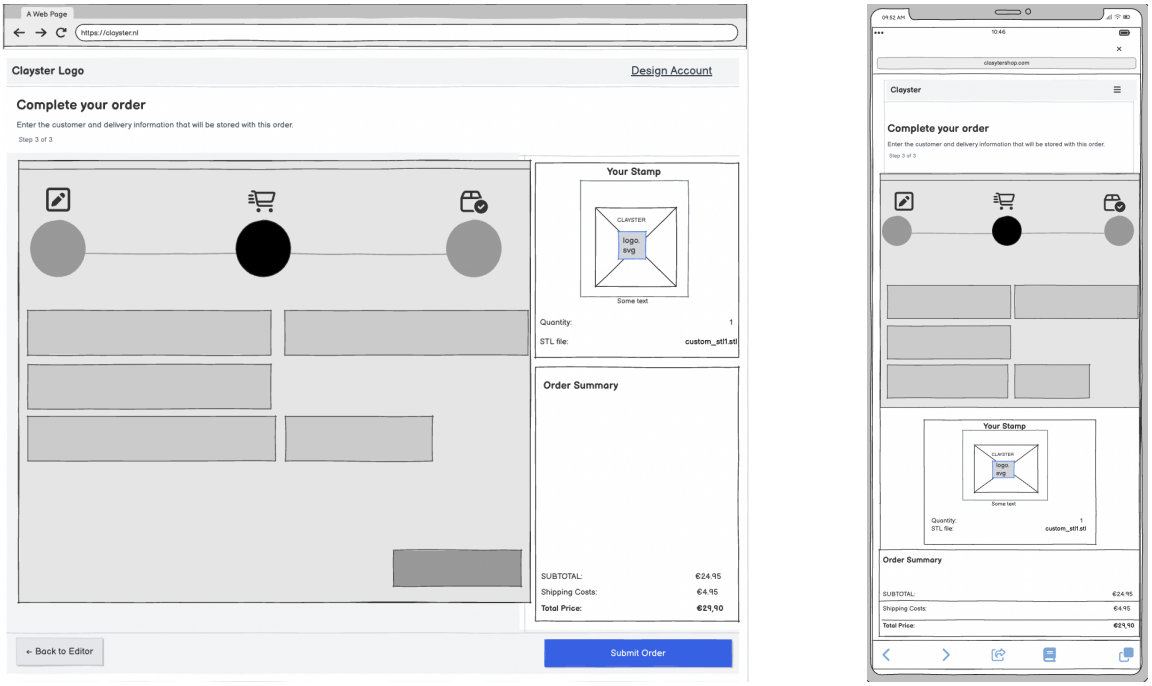


Figure 6.4: Revised review wireframe with the preview and design summary



(a) Desktop (b) Mobile

Figure 6.5: Order-details wireframes

6.2.4 Order Confirmation

The confirmation screen gives the customer a clear result after ordering. It shows the order reference, current status, and submitted items which lets the customer check that the order has been received and identify it when contacting the client. Figure 6.6 shows the revised confirmation wireframe.

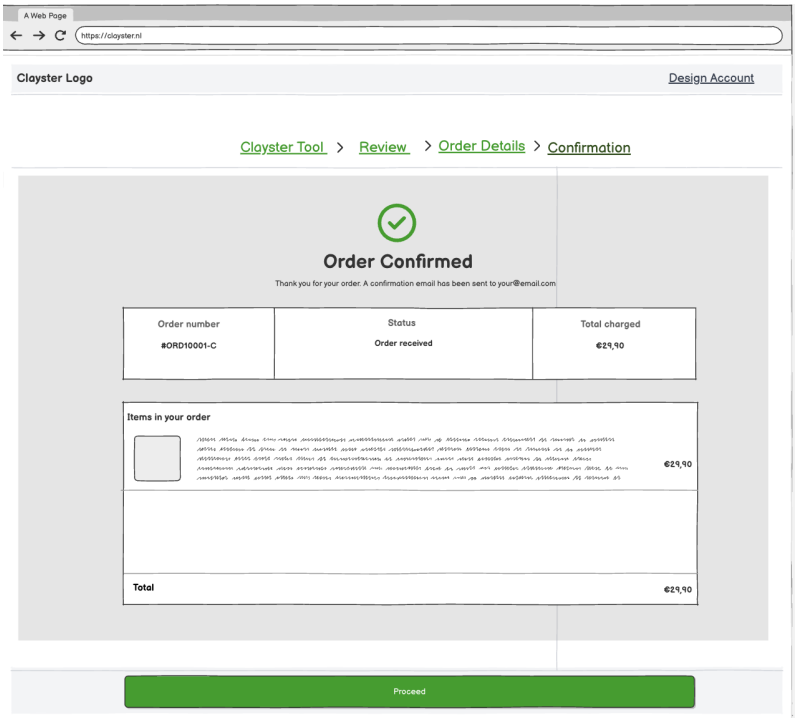


Figure 6.6: Revised order-confirmation wireframe

6.3 Frontend Implementation

The implementation follows the main workflow established in the wireframes. Further changes were made when the interface was connected to the stamp-generation pipeline, such as direct editing, validation feedback, and the transition from a design to a stored order.

6.3.1 Editing and Validation

The desktop editor uses three columns: design controls on the left, the preview in the centre, and printability results on the right.

Customers can combine text, SVG, and PNG elements in one design. Selecting an element reveals its controls. Elements can be dragged, resized, duplicated, or deleted. Undo and redo buttons allow customers to reverse changes to the artwork.

The stamp settings control the backing shape, dimensions, relief, mirroring, and handle. The default *Minimal* backing follows the rectangular bounds of the artwork with a margin on each side. The *Round* and *Square* options use fixed dimensions. Separate options control rounded plate edges and corners.

Figure 6.7 shows the implemented layout.

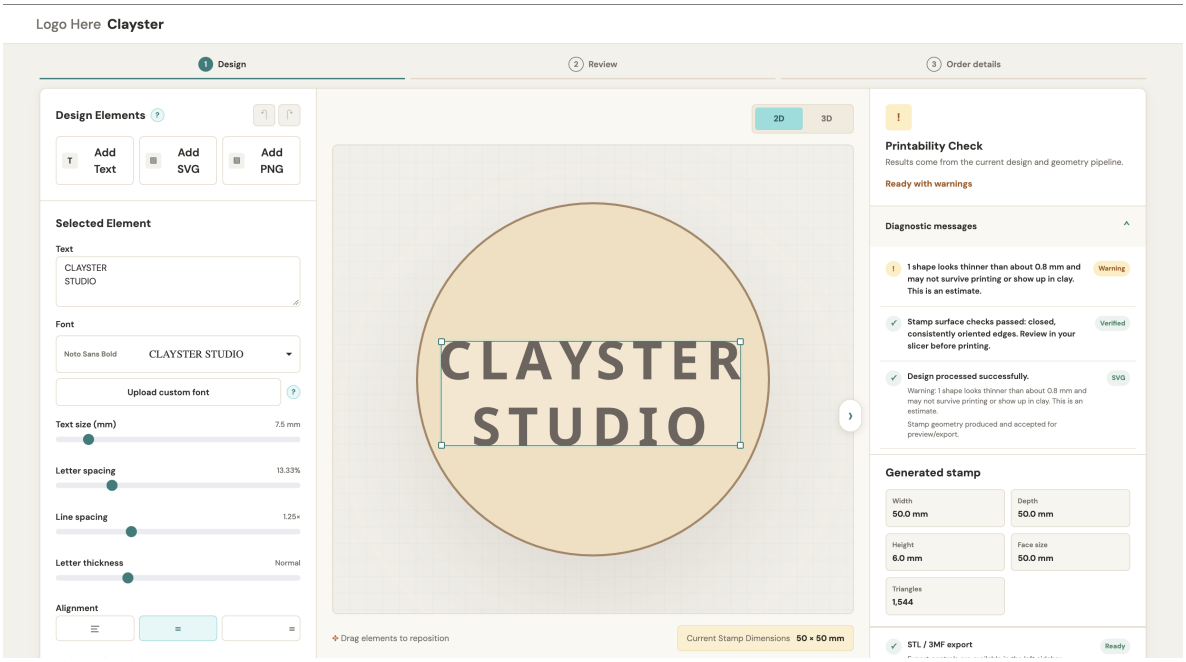


Figure 6.7: Implemented editor with controls, preview, and validation feedback

The validation panel distinguishes between processing, successful checks, warnings, and blocking errors. For example, artwork outside the usable stamp area is highlighted in red and accompanied by a message. A *Fit to stamp* action helps the customer correct this problem. Thin features and narrow gaps are reported as warnings.

Blocking errors prevent export and ordering. This includes a plate larger than 200x200 mm. Figure 6.8 shows feedback for artwork extending outside the usable area.

6.3.2 2D and 3D Previews

The 2D view supports editing, while the 3D view shows the generated stamp, including its relief, backing, and handle. Customers can also view the expected impression on clay. Both views use the same design state, keeping the artwork and generated model consistent.

The readable 2D artwork differs from the physical stamp when mirroring is enabled. By default, the stamp is mirrored so that its impression reads correctly on clay. Help text beside the mirror setting explains this difference. Figure 6.9 shows the generated stamp.

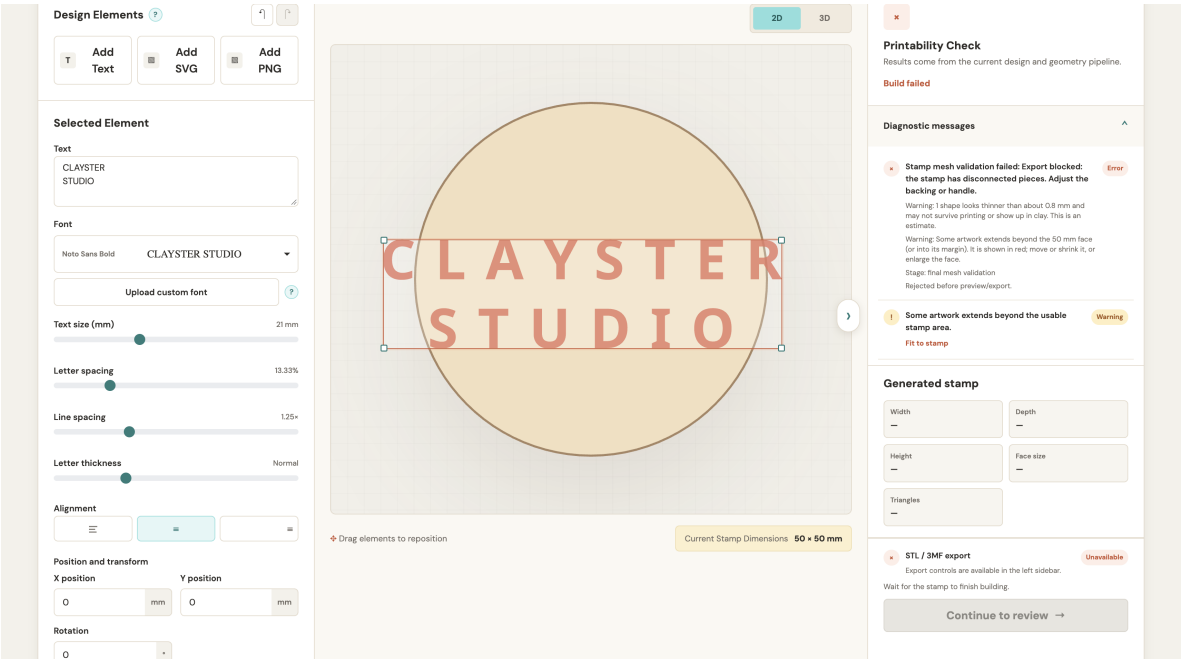


Figure 6.8: Validation feedback and highlighting for artwork outside the usable stamp area

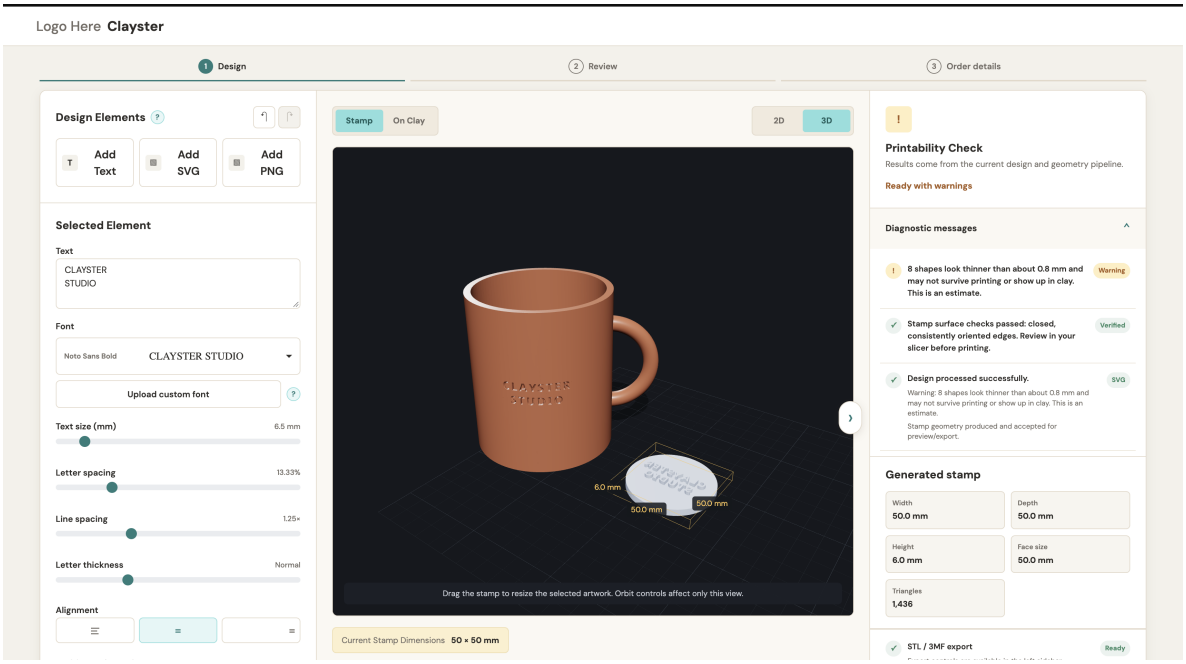


Figure 6.9: Implemented 3D stamp preview

6.3.3 Review and Ordering

The implemented review screen shows the final preview, dimensions, and stamp settings. Unlike the revised wireframe, it also retains the validation results. Warnings therefore remain visible when the customer confirms the design. Editing still takes place in the editor, which can be reopened from this screen. Figure 6.10 shows the implemented review.

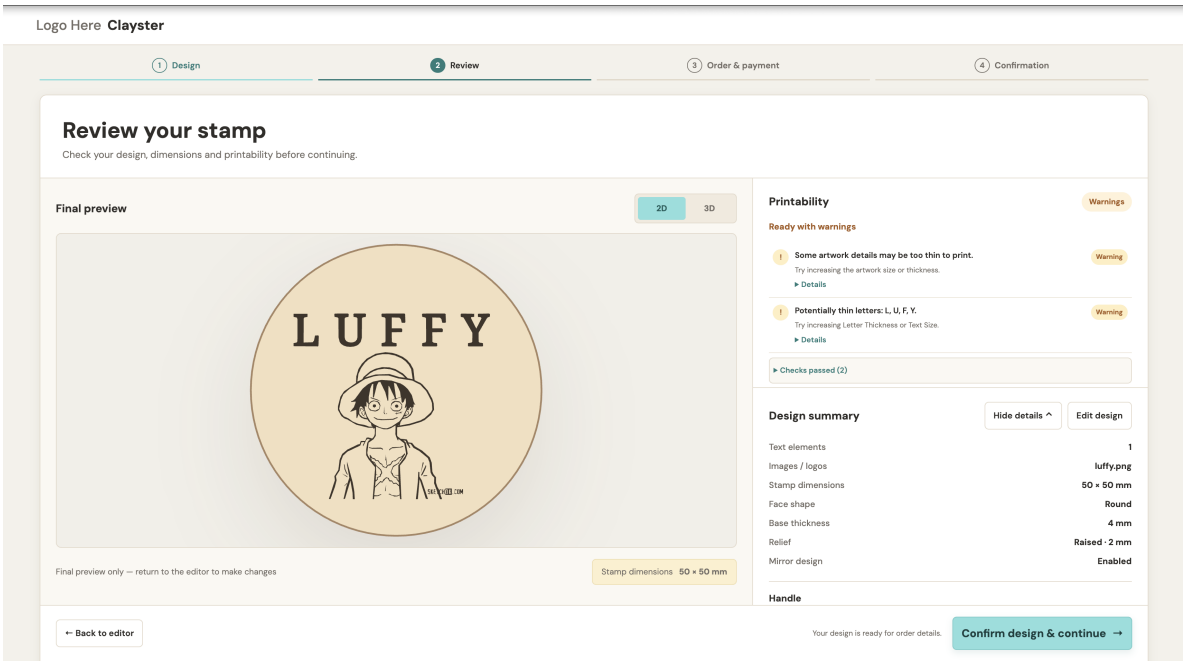


Figure 6.10: Implemented review with the stamp preview, settings, and validation results

The order form collects the stamp name, quantity, contact details, and delivery address, with optional descriptions and notes. The preview, dimensions, and price estimate remain visible beside the form. Errors are displayed near the fields that require correction.

On submission, the browser sends the order details, design settings, and generated STL file to the server. The server checks the submitted information, calculates the price independently, and stores the order as *Awaiting payment*.

Figure 6.11 shows the implemented order form.

The customer then continues to a payment page. Once payment is accepted, the order changes to *Processing* and a production job is created. The confirmation page provides the order number, status, items, total, and a private link for viewing the order later.

Payment and email are placeholders in the current implementation. The payment action accepts an order without charging money, and the The confirmation email is written to the server log. Real services still need to be connected, although for our project we decided that these features were out of scope.

6.3.4 Responsive Design

The interface uses one responsive layout. On wide screens, the editor displays its three columns side by side. On narrower screens, these areas are arranged vertically. The order form also changes to a single column. layout on small screens. Figure 6.12 shows the narrow-screen layout. This demonstrates how the interface adapts.

6.3.5 Accessibility Considerations

Design elements can receive keyboard focus and have a visible focus indicator. Selected elements can be moved with the arrow keys and deleted with Delete or Backspace. Alignment, undo, redo, and help controls have descriptive labels.

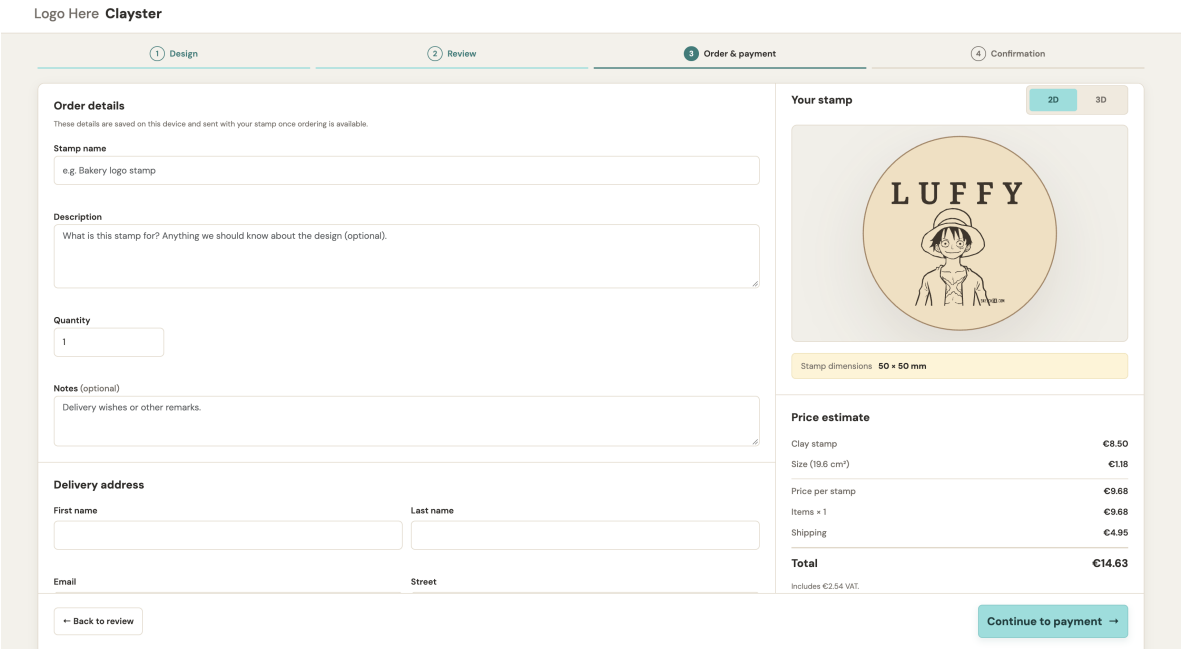


Figure 6.11: Implemented order form with the stamp summary and price estimate



Figure 6.12: Implemented editor at a narrow screen width

Validation messages use text and status indicators alongside colour. These measures support keyboard use and access to feedback, but full accessibility compliance has not been evaluated.

6.4 Client Interface

The client interface connects submitted orders to the files and tasks needed for production. A shared navigation menu provides access to the dashboard, orders, products, and production pages.

6.4.1 Dashboard Overview

The dashboard summarises orders received that day, open orders, production jobs waiting, and orders ready to ship. These summaries link to the relevant management pages.

It also highlights work requiring attention, shows production counts and the current printing job, and lists the five most recent orders. This gives the client an overview of the workload before opening individual orders or production jobs. Figure 6.13 shows the implemented dashboard homepage.

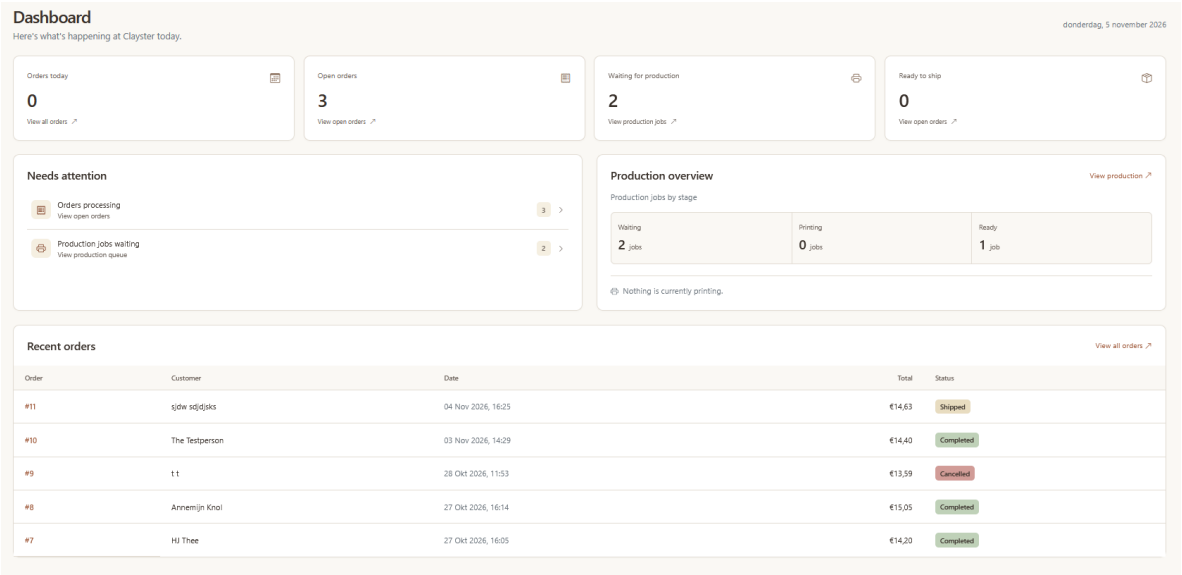


Figure 6.13: Overview of the dashboard homepage

6.4.2 Order Management

The order overview can be filtered by status and searched by order number or customer details. Opening an order shows its items, quantities, prices, contact details, and delivery address. The selected filter and search are retained when returning to the overview.

Each custom stamp has its own STL file attached to the order item. The client can download this file from the order details, keeping the personalised model connected to the correct customer and quantity.

Items can be marked as ready individually. When all items are ready, the order becomes ready for shipment. The client can also update the overall order status to record later stages of fulfilment. Figure 6.14 shows how the orders page looks like on the dashboard.

6.4.3 Product and Production Management

The product pages allow the client to manage descriptions, prices, visibility, and STL files. Standard products have stock information, while custom stamps use the personalised files stored with their orders.

The production page groups jobs into waiting, currently printing, ready, and completed work. The client can download the corresponding STL file and update the job as production progresses. A job can also be stopped or cancelled.

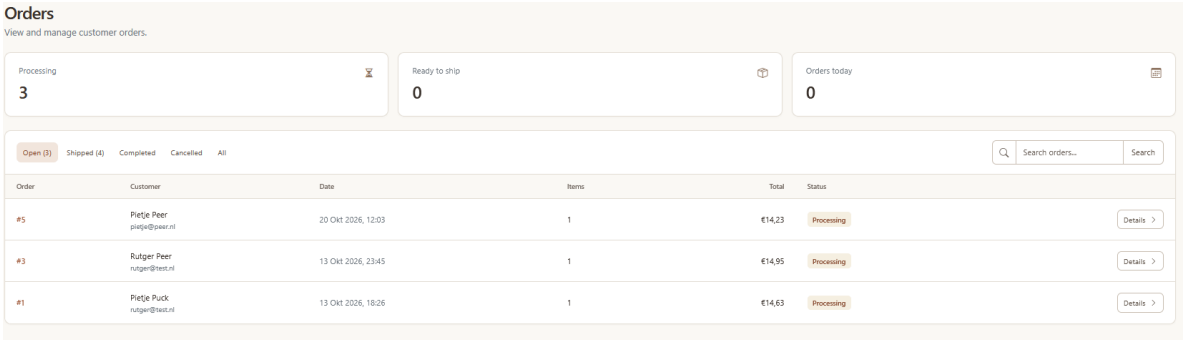


Figure 6.14: Overview of the order page in the dashboard

These actions record production progress; they do not operate the printer. The client still prepares and starts the physical print separately. Figure 6.15 and Figure 6.16 display the production and product pages on the dashboard.

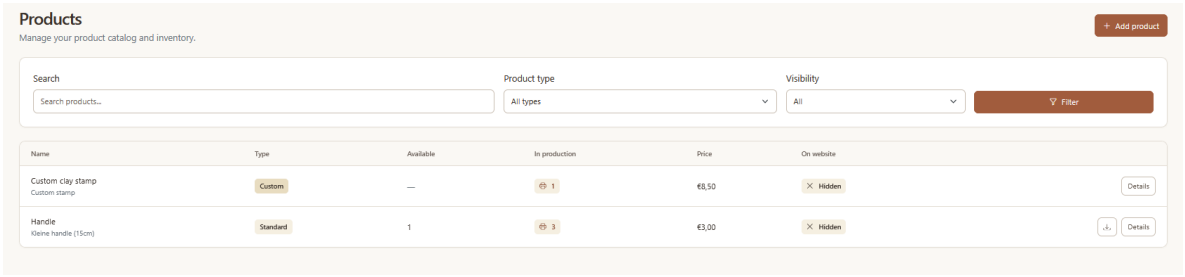


Figure 6.15: Overview of the product page on the dashboard

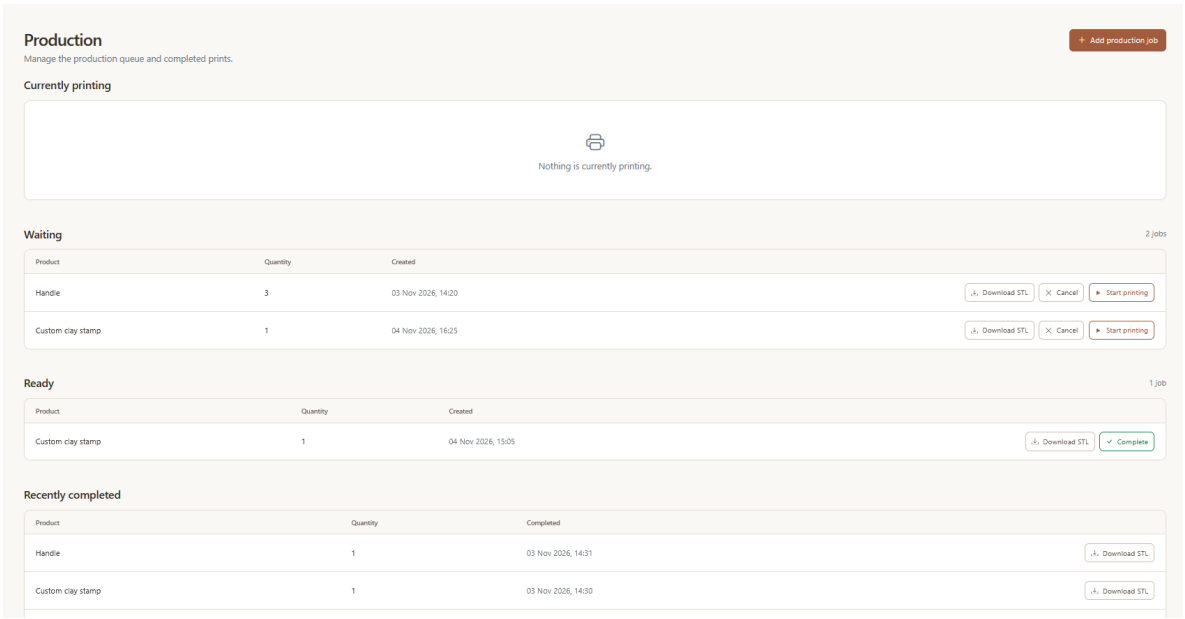


Figure 6.16: Overview of the production page on the dashboard

6.5 Design Limitations

Some differences between the wireframes and implementation remain. The proposed check against the available space on a clay object is not implemented. Customers must compare the displayed stamp dimensions with their object themselves.

The design is lost when the page is reloaded. Although the order form keeps a local draft until successful submission, this does not restore the artwork. The editor, review, and order form also share one page without separate routes, which limits expected browser back and reload behaviour. Undo and redo currently cover design elements, but not stamp settings.

Some technical terms still need explanation, and several selection buttons do not yet expose their selected state to screen readers.

The clay preview is a visual indication rather than a simulation of clay behaviour. Geometry checks cannot establish the quality of a physical impression. Test prints remain necessary, particularly for the latest rounded edges and corners.

6.6 LLM usage

OpenAI Codex was used to assist in generating frontend mockups based on wireframes created by the project team. All design concepts, design decisions, and original wireframes were independently developed by the team. AI tools were used to support the development process, while the team remained responsible for the final design decisions and implementations.

6.7 Conclusion

We developed the interfaces through initial wireframes, client feedback, revised designs, and implementation. Client feedback led to blocking validation errors and a simpler review wireframe. Implementation added direct artwork editing and connected the customer workflow to stored orders and production jobs.

The resulting interfaces support the main tasks of both customers and the client. The remaining work concerns saving designs, browser navigation, device and accessibility testing, physical validation, and the connection of real payment and email services.

Chapter 7

Testing the system

In this chapter, we describe the methods used to test and evaluate the system and present the corresponding results. The testing process was designed to assess the usability, technical reliability, and compliance with the client's requirements.

7.1 Test plan

The system was evaluated using three complementary testing methods: usability testing, client acceptance testing, and automated unit testing. Usability testing examined whether non-technical users could complete the stamp design and ordering process without assistance. Client acceptance testing was used to verify that the system met the client's requirements. Unit testing assessed the reliability of individual software components.

7.1.1 Usability testing

The usability test evaluates whether people with different levels of technical and design experience can create and order a custom clay stamp without prior training. The test focuses on the ordering flow. Its purpose is to identify controls that are difficult to find and steps in which a user requires assistance.

Participants

Four participants were selected as a small convenience sample. They were deliberately chosen with different ages and levels of technical confidence.

- Participant 1 was a 19-year-old university student with some experience using Canva for simple design tasks.
- Participant 2 was in his fifties and had a technical background but limited experience with digital design tools.
- Participant 3 was in her forties and had little confidence or previous experience with technology and design software.
- Participant 4 was a master's student at Delft University of Technology and had experience using digital tools.

None of the participants was part of the development team. Participants are assigned numbers instead of being identified by name to maintain their anonymity, as their personal identities are not relevant to the findings.

Procedure and Measurements

Each participant completed an individually moderated test using the same task order. The moderator introduced Clayster as a website for designing and ordering a custom clay stamp, but did not demonstrate the interface. Participants were asked to think aloud and explain what they were looking for. The moderator stayed silent where possible and only gave a few hints when a participant could no longer continue independently.

For every task, the moderator recorded whether it was completed independently, completed with assistance or not completed. The completion time and a rating on a seven point scale were also recorded, together with errors, hesitation, and comments made by the participant. The test guide defined 1 as very easy and 7 as very difficult. The ratings are therefore reported together with the written observations and are not combined into an average.

After completing the tasks, each participant was asked which parts they found easiest and most difficult, if any terms or controls were unclear, whether the previews and ordering process were easy to understand, and how confident they felt about using Clayster again without assistance. Dummy delivery details were used, and the test stopped before any real payment.

Tasks

The participants were asked to complete the following tasks:

1. Create a round stamp containing two lines of centered, bold text;
2. Make the design as large as possible while keeping it printable;
3. Configure the stamp size, relief, handle and mirroring;
4. Inspect the Stamp and On Clay results in both 2D and 3D;
5. Continue to the review page, return to the editor, change the relief and review the design again;
6. Enter the details for a test order of two copies and proceed to the payment step without completing the payment.

7.1.2 Unit tests

For unit tests, our original test coverage was 80%, and since the system is not highly critical, 80% was considered to be sufficient at first. However, after further testing, we had realized that some modules are on the critical side as they can affect the customers and the client. Hence, we decided to categorize our unit tests into 3 categories: critical, non-critical and normal. For the non-critical and normal modules, we have decided that 80% is sufficient. Whilst for the critical modules we settled on 90%.

We consider critical modules to be ones that specifically affect the price, stock, order status, customer data or access control. These include anything to do with products, including production, pricing of items, payment page, order API, the customer order page, staff views and STL file upload checks. More details about the tests for critical modules can be found in the Table 7.1 and Table 7.2.

In our unit tests, we avoid comparing error messages directly, as Django is currently set to Dutch. Instead, we check which fields contain errors. This ensures that our tests do not break if Django's default language is changed. During Docker build, we make sure that all the unit tests pass successfully, otherwise the build is ceased.

Table 7.1: Test coverage of critical order modules (target: 90%)

Module	Test class	What is tested
`orders/pricing.py`	`PricingTests`	The server calculates the same price as the browser for every shared test case, including shipping per country, the maximum stamp size of 200 × 200 mm and refusing input that cannot be priced.
`orders/stl.py`	`StlTests`	Reading the size of uploaded STL files, accepting files made by the stamp generator, rejecting files that are not stamps, and matching the file size to the stamp settings.
`orders/api.py`	`CreateOrderTests`	Placing an order from the website: the order is saved as awaiting payment, the price comes from the server and not the browser, invalid fields, files and oversized stamps are rejected, and the CSRF check is enforced.
`orders/payments.py`, `orders/services.py`	`PaymentFlowTests`	Payment flow and handing orders to production: the pay page shows the amount, paying sends the confirmation email and starts production, paying twice changes nothing, and a failing email does not undo the payment.
`orders/public_views.py`	`CustomerOrderPageTests`	The customer's order page behind a secret link: only the random link works, personal details are partly hidden, discounts are shown, the page stays out of caches and search results, and typed text is escaped.
`orders/views.py`	`StaffOrderViewTests`	Staff can download the customer's stamp file and nobody else can, an item without a file has no download, and new orders appear on the staff pages.
	`StaffOrderPermissionTests`	Visitors and logged-in customers are sent to the login page for every staff order page and action, and nothing in the order or stock changes.
	`StaffOrderListTests`	The order list shows the newest orders first, counts items and orders per status, and finds orders by customer name, email or order number.
	`OrderOverviewTests`	The order list shows open orders by default, each status filter shows the right orders, the counters stay correct while filtering, and the back link remembers the filter and search.
	`ChangeStatusTests`	Staff change an order's status, invalid statuses are refused, shipped and completed orders cannot be cancelled, and cancelling stops waiting production jobs and returns reserved stock.
	`ItemReadyTests`	Marking items ready reserves stock and is refused when there is not enough, custom stamps use no stock, the order becomes ready after its last item, finished orders cannot change, and undoing returns the stock.
	`AddItemToProductionTests`	Sending an order item to production creates one job with the item's product and quantity, never a second one, unless the earlier job was cancelled.

Table 7.2: Test coverage of critical production and product modules (target: 90%)

Module	Test class	What is tested
`production/views.py`	`ProductionPermissionTests`	Visitors and logged-in customers are sent to the login page for every production page and action, and no job changes.
	`ProductionIndexAndAddTests`	The production overview groups jobs by status, and staff can add a job by hand, while invalid input saves nothing.
	`ProductionStatusTests`	Jobs move from waiting to printing, ready and completed, only one job prints at a time, completing a job adds its quantity to stock exactly once, and jobs can be stopped or cancelled.
	`ProductionStlTests`	Each job downloads its own STL file, custom stamps never fall back to a generic file, downloads are staff-only, and missing files give a "not found" page.
`products/views.py`	`ProductPermissionTests`	Visitors and logged-in customers are sent to the login page for every product page and action, and actions only work through a form submission (POST).
	`ProductListTests`	The product list sorts by name, shows how many units are in production, and can be searched and filtered by type and visibility. The product page shows the units ordered.
	`ProductAddEditDeleteTests`	Staff add and edit products, invalid input such as a non-STL file saves nothing, products used in orders or jobs cannot be deleted, and the STL file can be downloaded.
	`ProductStockTests`	Staff raise or lower stock by one without going below zero, custom stamps have no stock, and products can be sent to production for stock with a valid quantity.

Table 7.3: Test coverage of normal modules (target: 80%)

Module	Test class	What is tested
`products/forms.py`	`ProductFormTests`	The product form accepts valid input with or without a file, accepts STL files including uppercase `.STL`, rejects other file types, and requires a name and a price.
`products/models.py`	`ProductTests`	A saved product can be loaded again from the database with its details.
`production/forms.py`	`ProductionJobFormTests`	The production job form accepts a valid product and quantity, and rejects a missing or unknown product and a negative quantity.
`dashboard/views.py`	`HomeDashboardTests`	The staff home page shows correct counts of open and ready orders, orders placed today (in Amsterdam time, including daylight saving changes), jobs per production stage and the five most recent orders. It requires staff access, shows development tools only in debug mode, and keeps the number of database queries the same as orders grow.
	`DashboardShellTests`	The staff navigation and the mobile menu show their state accessibly, error messages are visible, and text typed by users is escaped.
`stamp_generator/views.py`	`IndexViewTests`	The stamp editor loads at the site root without JavaScript from other websites, uses the library files stored in the project, and gives the order page its settings and CSRF token.
	`StlExportAccessTests`	Anyone can export an STL file while developing; on the live site customers cannot, but staff still can.

Table 7.4: Test coverage of non-critical modules (target: 80%)

Module	Test class	What is tested
`orders/utils.py` and the debug views in `orders/views.py`	`TestOrderButtonTests`	The buttons that create test orders return "not found" when debug mode is off. With debug mode on, a custom test order gets a production job, a standard test order uses the chosen product, and a custom product is refused for a standard test order.
`products/utils.py`	`ProductCreate`	The helper that creates a dummy product makes one custom stamp product with the expected name and price, and calling it twice reuses the same product.

7.1.3 Client testing

Client testing verifies that the project meets the client's needs and expectations. It takes place during weekly meetings with the client throughout the project. Each meeting starts with a review of the improvements from the previous week, where the client confirms what they think about the new progress. The current state of the project is then demonstrated, focusing on new or changed features, and any open design choices are discussed and decided together with the client. Finally, all feedback is collected into an improvement list, which is prioritised together with the client. The highest-priority improvements are implemented in the following week. An improvement is considered complete once the client approves it in a following meeting; otherwise it remains on the list.

7.1.4 Acceptable risk

We analyse the acceptable risk with the risk matrix, we made this at the beginning of the project. It can be found in the appendix in B chapter.

7.2 Test results

Here we have gathered all the test results.

7.2.1 Usability testing results

Individual results

Participant 1 completed five of the six tasks independently. Creating the main design took 45 seconds. The participant understood how to enter and position text, noticed when it exceeded the stamp boundary and corrected it without help. Assistance was needed during the printability task because the participant initially believed that the design was ready even though the text was highlighted in red. Configuring the physical stamp and inspecting the result were completed quickly. The review-and-edit task took 24 seconds, although the participant first had to discover that the review button was below the visible part of the page. The order task took 1 minute 16 seconds and was considered straightforward. The main recommendations were to make the review button more visible and explain the STL export option.

Participant 2 completed most tasks independently but needed help when configuring the physical stamp. The participant created the design without difficulty, although editing text in the left settings panel was less direct than expected. Making the printable design took 1 minute 6 seconds. The participant noticed the warnings but found them too small and did not immediately understand where to make the correction. The terms handle, raised and recessed were unfamiliar, and the participant did not initially realize that the left panel could be scrolled. Inspecting the result took 2 minutes 12 seconds and the difference between Stamp and On Clay was unclear. The review-and-edit task was the longest task in this session at 5 minutes 38 seconds. The participant could not initially find the review button, got stuck in the 3D view and did not realize that editing had to continue in 2D. The order task took 1 minute 2 seconds and was completed independently, although a validation error appeared outside the currently visible area.

Participant 3 required assistance during all six tasks. Creating the main design took 5 minutes because the participant could not find the stamp-shape controls and did not realize that the left panel was scrollable. Resizing the text and using Fit to stamp also required an explanation. Configuring the physical stamp took 3 minutes 15 seconds and required help to locate the width control. The participant could switch to the 3D preview but needed assistance to change between Stamp and On Clay. The review button itself was easy to recognize once visible, but the complete review-and-edit instruction was difficult because it required several steps across different parts of the interface. In contrast, the participant entered the order details and corrected an incomplete field with little difficulty. The participant described resizing the design as the most difficult part and the order form as the easiest.

Participant 4 created the main design independently in 32 seconds and rated the task 5 out of 7. Making the design printable took 3 minutes and also received a rating of 5. The participant initially overlooked the diagnostic messages and found the thin-letter warning difficult to resolve. Configuring the physical stamp was the most difficult task in this session: it took 7 minutes and received a rating

of 1. The terms face size, raised and recessed were unclear, and the participant interpreted the red text as a font colour before recognising it as an error. Inspecting the Stamp and On Clay results in 2D and 3D did not cause a recorded difficulty. The review-and-edit task took 52 seconds, received a rating of 7 and produced no errors. The order form was clear, although the participant recommended adding a back button and showing the item details during checkout so that customers can confirm what they are ordering.

Cross-participant findings

The first repeated issue was the discoverability of controls outside the visible area. Participants 2 and 3 needed help to discover that the left settings panel could be scrolled. Participants 1 and 2 also had difficulty finding the button used to continue to the review page because it was below the visible part of the screen. These observations show that the interface does not communicate clearly when more settings or actions are available outside the current view.

The second repeated issue concerned printability feedback. Participants 1, 2 and 4 did not immediately understand that red text indicated an invalid or unprintable design. Participant 3 needed assistance with resizing and Fit to stamp. In one test, an oversized design could still continue to the review page. This is a important issue because a user may believe that an invalid design is ready to order. The warnings needed to be more prominent. This problem was fixed after the usability test.

The physical stamp settings also contained unfamiliar terminology. Face size was interpreted as something different from the stamp width, while raised, recessed, base thickness, handle and STL were not understood by every participant. Help icons were noticed, but the explanations did not always provide enough information for a person without experience in design or 3D printing.

The 2D and 3D previews were useful, but the relationship between their modes was not always clear. Participants could generally switch between 2D and 3D, but Participants 2 and 3 had difficulty distinguishing Stamp from On Clay. Participant 2 also did not understand that changes had to be made in 2D and checked in 3D. The interface should make the purpose of each mode and the current editable view clearer.

The order flow was the strongest part of the system. All participants who commented on this step described entering the delivery details as easy or completed it with little difficulty. Participants could change the quantity and correct incomplete fields. The remaining problems were the placement of validation messages, the absence of a clear back action and the need for a stronger order summary before payment.

Resulting improvements

Based on the four tests, the following changes have the highest priority:

1. Make panels on the left and right scrollable and visually recognisable, keeping the main area actions visible;
2. Prevent an invalid design from continuing and make printability errors larger and more actionable;
3. Explain the difference between 2D, 3D, Stamp and On Clay and provide a clear route back to editing;
4. Move validation feedback close to the incorrect field and automatically focus or scroll to the first error;
5. Add a clear back action and a complete stamp summary to the order and payment flow;
6. Provide an optional short introduction for first time users without requiring every user to complete a tutorial.

Limitations

The test used a small convenience sample. The variation in age and technical confidence was useful for finding problems, but the sample is not representative of all potential Clayster customers or people who regularly work with clay. The moderator also knew all four participants, which may have influenced their

behaviour and comments. A future test should use the same interpretation of the rating scale, include people from the intended customer group and repeat the test after the main usability improvements have been implemented.

7.2.2 Unit testing results

All the backend unit tests pass, and the 3 test categories meet their target coverage. Table 7.5 shows the final coverage for the identified sections and shows their target percentage.

Models, forms and emails do not have their own category and target, however they are tested through other tests.

Table 7.5: Test coverage per category

Category	**Coverage**	**Target**
Critical	99%	90%
Normal	92%	80%
Non-critical	100%	80%
Models, forms and emails	97%	–
Configuration code (admin, apps, URLs)	100%	–

Table 7.6: Test results and coverage per category

Category	**Module**	**Result**	**Category coverage**
Critical	`orders/views.py`	Pass	99%
	`products/views.py`	Pass	
	`orders/api.py`	Pass	
	`production/views.py`	Pass	
	`orders/public_views.py`	Pass	
	`orders/pricing.py`	Pass	
	`orders/stl.py`	Pass	
	`orders/services.py`	Pass	
	`orders/payments.py`	Pass	
Normal	`dashboard/views.py`	Pass	92%
	`products/models.py`	Pass	
	`stamp_generator/views.py`	Pass	
	`products/forms.py`	Pass	
	`production/forms.py`	Pass	
Non-critical	`orders/utils.py`	Pass	100%
	`products/utils.py`	Pass	
Models, forms and emails	`orders/models.py`	Pass	97%
	`orders/forms.py`	Pass	
	`production/models.py`	Pass	
	`orders/emails.py`	Pass	
	`customers/models.py`	Pass	

7.2.3 Client testing results

The weekly meetings had a clear influence on the development of Clayster. Because the client could try each new version of the tool herself, feedback was based on actual use. And several features changed direction as a result. The most valuable outcome came from the design discussions. For example, the client would like to use the least plastic possible, but still make it printable. To figure out the exact specification of this we had some front and backs. The improvement lists helped us with keeping the current problems organised, where the first problems were mostly functional, later on it shifted towards usability and visuals. Overall, client testing confirmed that Clayster meets the client's expectations and that the most important design decisions were supported by the client.

7.3 LLM usage

ChatGPT was used to assist with the structure of the usability test guide. The usability tests and reporting themselves were fully conducted by the Clayster team, who moderated the sessions, observed the participants and recorded the results.

During the development of unit tests, Claude Code was used to assist the Clayster team with creating and reviewing tests. Initially, the team wrote unit tests independently, after which Claude Code was used to analyze and verify the logic of some of these tests. For certain tasks, team members provided the intended testing logic, and Claude Code generated the corresponding unit tests based on their instructions.

In the final iteration, Claude Code was also used to suggest additional test cases that the team had not initially considered. These tests were carefully reviewed by the team to understand their purpose and ensure that the results were genuine rather than artificially made to pass. Additionally, Claude was used in Overleaf to assist with formatting the unit test tables in the report.

Chapter 8

Future Planning

In this chapter, the future of this product will be discussed. The development from the team will stop, but the real usage will continue. Therefore, it is important to plan for this to ensure the product has a long life span.

8.1 Deployment

The current system has not yet been deployed to a production environment. Before public use, the client will need to arrange hosting and configure the server, database, domain, HTTPS, and backups. The deployed system should also be tested with real orders and printed stamps.

8.2 Manual

A separate manual is delivered together with this report. It is written in Dutch for the client and her staff. Using step-by-step user stories, it explains how a customer designs and orders a stamp, and how staff handle an order in the dashboard, from production to shipping. It also states that the product is provided as-is.

8.3 Future Improvements

Payment and automatic email delivery were not completed during the project. These features need to be implemented before deployment so that the tool is ready for practical use. A future development team could implement these features and make further usability improvements based on the findings in Chapter 7.2.1.

8.4 Maintenance

The client will be responsible for arranging future maintenance. This includes security updates, backups, error monitoring, and resolving technical problems. The source code will be provided to support future developers. Some members of the team will maintain contact with the client for a short period after handing over the code to handle any questions the client might have.

8.5 Conclusion

Completing the remaining integrations and deploying the system will make it ready for practical use. The project delivers a working foundation that can be deployed and extended by the client or a future development team.

Chapter 9

Evaluation

In this chapter, we discuss our means of collaboration, the team evaluation and discussing the final result of the project.

9.1 Planning

We planned our project in big lines in a Gantt-chard. From there every week, we did one or two stand-up meetings. We used Trello board to make it clear and have an overview. Every week we had a meeting with the client to show different proposals and request feedback, please refer to Section 7.1.3 for more details Furthermore, we also had weekly meetings with Vadim, our supervisor, to show the process and ask questions.

We also have deliverables, were we often set internal deadlines to ensure quality of our products. This also helps with equally distributed workload throughout the module and over people in the team.

9.2 Responsibilities

We divided the responsibilities in 6 roles. This does not mean that each person is expected to handle their assigned area entirely on their own, nor does it prevent them from contributing to other parts of the project.

Instead, each person is responsible for overseeing their assigned area, making sure it receives sufficient attention, that relevant tasks are addressed, and that progress remains on track. The whole team can still contribute to and make decisions across all areas of the project.

The roles are:

- **Project lead**: Planning, meetings, progress, blockers
- **Client and requirements lead**: client contact, requirements, acceptance criteria
- **Technical lead**: Architecture, technical choices, code quality
- **Product design lead**: Wireframes, UI/UX, design decisions, and visual consistency
- **Backend lead**: Backend, APIs, database
- **Delivery lead**: Latex, Report quality, presentation, manual

The roles were assigned in the corresponding order as follows:

- **Timo** - Project lead
- **Rutger** - Client and requirements lead
- **Hakan** - Technical lead
- **Ali** - Product design lead

- **Nikita** - Backend lead
- **Evelyn** - Delivery lead

9.3 Team Evaluation

From the beginning we discussed the expectations and the means of collaboration. We were quite on one line with this. Furthermore, we discussed the use of LLM's, not everyone had the same level in mind so we discussed how to make it work for everyone. This is why we chose level 1 for the report and level 2 for the programming. Level 1 is chosen for the report as we didn't want the report be slop. Level 2 is chosen for the programming because it can greatly increase the speed of implementation and some teammates felt that LLM's is the future of software.

One teammate moved during the project, but they did compensate it by working more before and after moving. We also did a team building activity.

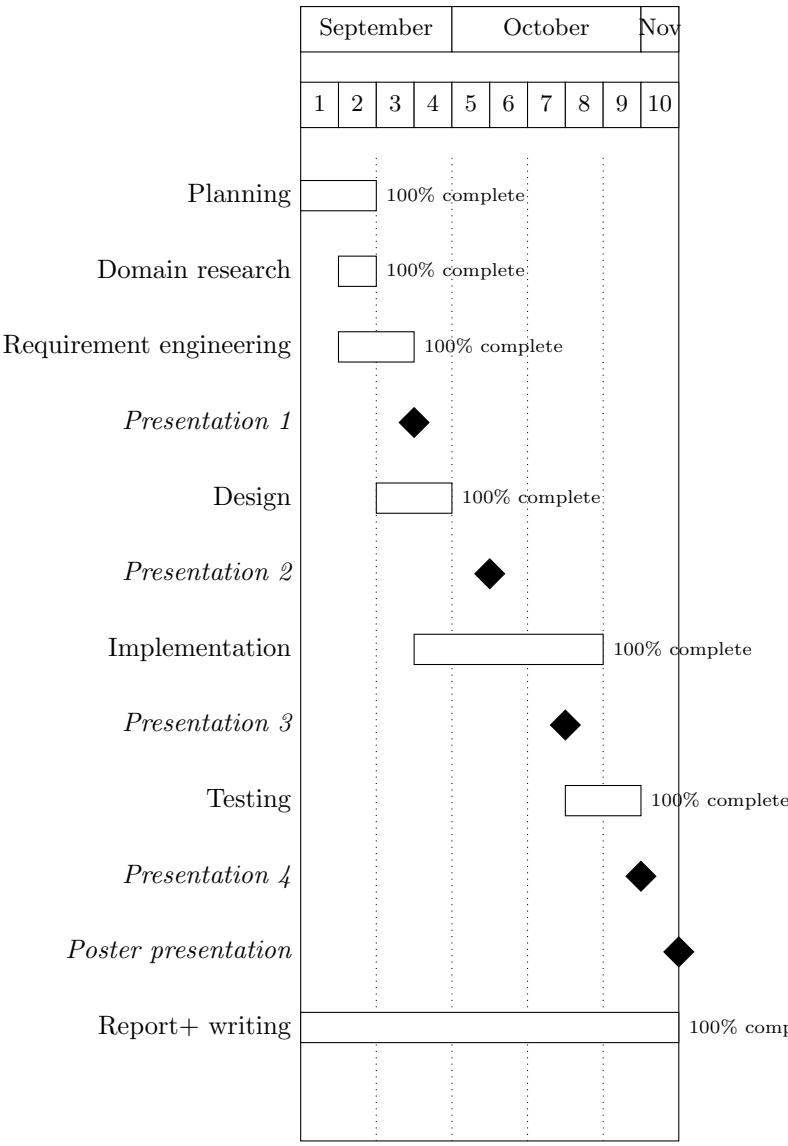


Figure 9.1: Ganttchart planning

9.4 Final result

The final result is a web tool that users can use to design custom 3D-printable clay stamps. Using a user interface that we have tried to make as intuitive as possible, users can design stamps with their own .svg and .png files, and personalized texts. The web tool offers a 2D preview for designing, and a 3D preview to quickly see the resulting stamp. It not only generates an STL-file for the 3D printer; it also verifies that the generated stamp is actually printable by the 3D printer based on printing requirements that have been tested in the test phase.

A customer can order a custom stamp using the connected order system. The client can manage the orders in the dashboard view that was built. In this dashboard, the client can manage all open orders, download the STL files of the ordered stamps, and manage the production process. The main workflow was completed, although payment, automatic email delivery, and deployment were not included in the final version.

9.5 Conclusion

Overall, the project produced a usable foundation that addresses the most important requirements. Future work should focus on deployment, additional usability improvements, payment and email integration, security review, and testing the generated stamps in a complete production environment.

Bibliography

[1] Al-Olaimy, T. Nature must be the most important stakeholder this decade.

[2] Brown, B., and Feldman, M. *Critical Approaches to Ancient Near Eastern Art*. EBL ebooks online. De Gruyter, 2013.

[3] European Parliament, and Council of the European Union. Directive (eu) 2022/2464 of the european parliament and of the council of 14 december 2022 amending regulation (eu) no 537/2014, directive 2004/109/ec, directive 2006/43/ec and directive 2013/34/eu, as regards corporate sustainability reporting.

[4] Renckens, M., and other contributors. Linotype wiki.

Appendix A

AI-disclosure

During the preparation of this work the authors used Claude/ChatGPT/Gemini in order to correct grammatical, spelling errors, polish writing and formatting. After using this tool/service, the author reviewed and edited the content as needed and takes full responsibility for the content of the work.

Appendix B

Risk analysis

At the beginning of the project we set up a risk analysis. For each risk we estimated the probability (P) and impact (I) on a scale from 1 to 5. An impact of 1 means no real effect, 3 means a few days of delay or a missed should have, and 5 means a must-have is not met. The score (S) is P $\times$ I: 15 or higher is high, 7 to 14 is medium and 6 or lower is low. When the risk is higher than 15 we go and try to reduce it to a lower value. Since we have no access to a 3D printer, we assume that a correct STL or 3MF file is enough, as the client prints the stamps.

ID	Risk	P	I	S	Measure
R1	**Inconsistent requirements.** Some requirements overlap or contradict each other, so we may build the wrong things.	4	3	12	Make one prioritized list and check it with the client.
R2	**No printability values.** We do not know the minimum wall thickness, depth, and maximum size the checks should use.	4	4	16	Research common values ourselves, confirm them with the client and make them easy to change.
R3	**Too much scope.** There are many must haves and the client has more ideas, while the deadline is 4 November.	4	4	16	Finish the must-haves first. New ideas are discussed before we add them.
R4	**Generating the 3D model.** Letters can overlap, be disconnected or too thin, the export can contain errors, and a stamp must be mirrored.	3	4	12	Build a prototype early, use a baseplate, test many texts and check exports in a slicer.
R5	**Fonts.** Uploaded fonts may not load or give broken shapes, and not every font may be used commercially.	4	3	12	Accept only certain formats, check fonts on upload and only offer fonts with a suitable licence.
R6	**Preview performance.** The preview may be slow or look different on phones or in Firefox and Chrome.	3	3	9	Test on both browsers and on phones from the start.
R7	**Email delivery.** Emails with the file can end up in spam or be too large.	3	3	9	Send a download link and keep the file available in the dashboard.
R8	**Hosting.** It is not decided where the app runs and who maintains it after the project.	3	3	9	Discuss this with the client soon and document the deployment.
R9	**Security and privacy.** The site stores personal data, has logins and accepts file uploads.	3	4	12	Store only what we need, use HTTPS, hash passwords and limit uploads.
R10	**No printer.** Problems that only show when printing may be found late.	4	3	12	Ask the client to print early versions when possible.

ID	Risk	P	I	S	Measure
R11	**No 3D experience.** Nobody in the team has worked with 3D geometry before, while the MVP is due in week 5.	4	4	16	Plan research time at the start and use existing libraries.
R12	**Team coordination.** Knowledge may sit with one person, or parts may not fit together at the end.	3	4	12	Work in pairs, review code and merge work regularly.
R13	**Client availability.** We depend on the client for decisions. We meet weekly and they reply fast.	1	3	3	Keep the weekly meetings.

Table B.1: Risk register

Table B.2 shows the risks in a matrix. The three high risks are R2, R3 and R11. R2 and R3 affect whether we can deliver the must-haves, and R11 affects the MVP, which is why we start with a prototype of the 3D generation.

P \ I	1	2	3	4	5
5					
4			R1, R5, R10	R2, R3, R11	
3			R6, R7, R8	R4, R9, R12	
2					
1			R13		

Table B.2: Risk matrix