VILNIUS GEDIMINAS TECHNICAL UNIVERSITY

FACULTY OF FUNDAMENTAL SCIENCES

DEPARTMENT OF INFORMATION TECHNOLOGIES

Zehra İrem Kuyucu

INTERNETINĖ 3D SPAUDINIŲ UŽSAKYMO IR VALDYMO SISTEMA

WEB-BASED SYSTEM FOR ORDERING AND MANAGING 3D PRINTS

Final Bachelor Thesis

Engineering Informatics Study Programme, State code 6121BX033

Management of Information Technologies

Study Field of Informatics

Vilnius, 2025

VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS

FUNDAMENTINIŲ MOKSLŲ FAKULTETAS

INFORMACINIŲ TECHNOLOGIJŲ KATEDRA

PATVIRTINTA
Katedros vedėjo
doc. dr. Dmitrij Šešok

Zehra İrem Kuyucu

INTERNETINĖ 3D SPAUDINIŲ UŽSAKYMO IR VALDYMO SISTEMA

WEB-BASED SYSTEM FOR ORDERING AND MANAGING 3D PRINTS

Baigiamasis bakalauro darbas

Inžinerinės informatikos studijų programa, valstybinis kodas 6121BX033

Informacinių technologijų valdymas

Informatikos studijų kryptis

Vadovas Dr. Danylo Shkundalov
(Pedag. vardas, vardas, pavardė)

Konsultantas ______________________
(Pedag. vardas, vardas, pavardė)

Vilnius, 2025

VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS
Vilnius Gediminas Technical University
FUNDAMENTINIŲ MOKSLŲ FAKULTETAS
Faculty of Fundamental Sciences
INFORMACINIŲ TECHNOLOGIJŲ KATEDRA
Department of Information Technologies

PATVIRTINA / *APPROVED*
Katedros vedėjas / *Head of the department*
doc. dr. Dmitrij Šešok

BAIGIAMOJO BAKALAURO DARBO UŽDUOTIS
ASSIGNMENT OF BACHELOR THESIS
Vilnius

Studentei
For student ZEHRA İREM KUYUCU

Baigiamojo darbo (projekto) tema: INTERNETINĖ 3D SPAUDINIŲ UŽSAKYMO IR VALDYMO SISTEMA
Bachelor thesis topic: *WEB-BASED SYSTEM FOR ORDERING AND MANAGING 3D PRINTS*

Baigiamojo darbo užbaigimo terminas 2025-05-06
The deadline for bachelor thesis *2025-05-06*

BAIGIAMOJO DARBO (PROJEKTO) UŽDUOTIS:
Žiniatinklio sistema, skirta 3D spaudos užsakymui ir valdymui, yra internetinė platforma, sukurta supaprastinti 3D spaudos užsakymo ir valdymo darbo eigą. Sukurta sistema automatizuoja 3D spaudos valdymo procesą, įskaitant atsargų valdymą, spausdinimo parametrų nurodymą, atraminių konstrukcijų pasirinkimą ir siūlo tipo pritaikymą. Ji atlieka nuotolinį pjaustymą ir spaudos patvirtinimą, kad automatizuotų užsakymų patvirtinimą ir pateiktų momentinius kainos pasiūlymus. Be to, sistema realiu laiku rodo medžiagų sąnaudų įvertinimą ir 3D modelio bei sugeneruoto G-kodo vizualizaciją pagal vartotojo pasirinkimus, suteikdama patogią vartotojo sąsają, kuri palengvina 3D spausdinimo procesą net ir neturintiems techninių žinių vartotojams.

THE ASSIGNMENT OF BACHELOR THESIS:
Web-based System for Ordering and Managing 3D Prints is a web-based platform designed to streamline the 3D print ordering and management workflow. The developed system automates the 3D print management process, including inventory management, printing parameter specification, support structure selection, and filament type customization. It performs remote slicing and print validation to automate order approval and provide instant pricing. Additionally, the system displays real-time material cost estimation and visualization of both the 3D model and the generated G-code based on user selections, providing a user-friendly interface that makes the 3D printing process more accessible for non-technical users.

Baigiamojo bakalauro darbo (projekto) konsultantai:
The consultant(s) of bachelor thesis:
(Pedag. vardas, vardas, pavardė / *Academic title, name, surname*)

Vadovas
Supervisor Dr. Danylo Shkundalov
(Parašas / *Signature*) (Pedag. vardas, vardas, pavardė / *Academic title, name, surname*)

Vilniaus Gedimino technikos universitetas
Fundamentinių mokslų fakultetas
Informacinių technologijų katedra

Informacinių technologijų studijų programos baigiamasis bakalauro darbas
Pavadinimas **Internetinė 3D spaudinių užsakymo ir valdymo sistema**
Autorius **Zehra İrem Kuyucu** Vadovas **Dr. Danylo Shkundalov**

Kalba	
	lietuvių
x	anglų

Anotacija

Šis darbas pristato internetinę sistemą 3D spaudinių užsakymui ir valdymui, skirtą spręsti įgimtus 3D spausdinimo paslaugų sudėtingumus ir prieinamumo kliūtis, įskaitant sudėtingą mokymosi procesą, nenuoseklią kainodarą ir varginančias rankines darbo eigas, susijusias su failų patvirtinimu, konfigūravimu ir kainų nustatymu. Sukurta sistema automatizuoja ir supaprastina visą 3D spaudinių užsakymo ir valdymo procesą, pasižymėdama automatizuotais parametrų koregavimais, patikimu spaudinių patvirtinimu, nuotoliniu „slicing“ (pjaustymu) ir momentiniais kainų pasiūlymais. Naudotojams teikiami realaus laiko medžiagų sąnaudų įvertinimai, interaktyvūs 3D modelio ir G-kodo vizualizavimai per patogią sąsają, kuri supaprastina 3D spausdinimo patirtį netechniniams asmenims. Integruotas administratoriaus pultas palengvina išsamų atsargų ir užsakymų valdymą, o mokėjimų apdorojimo sistema užtikrina sklandžias operacijas su automatizuotais būsenos atnaujinimais. Šios sistemos praktinė vertė yra daugialypė: ji žymiai sumažina veiklos išlaidas paslaugų teikėjams, pagerina naudotojų prieinamumą ir suteikia tvirtą sistemą viešai prieinamų 3D spausdintuvų monetizavimui tokiose aplinkose kaip kūrybinės dirbtuvės ir universitetai. Be to, sistemos keičiamo mastelio architektūra palaiko ateities tinklo modelį, leidžiantį individualiems spausdintuvų savininkams valdyti atsargas ir teikti paslaugas, taip išplečiant jos pasiekiamumą už centralizuotų spausdinimo ūkių operacijų ribų į labiau paskirstytą ir dinamišką 3D spausdinimo ekosistemą.

Prasminiai žodžiai: 3D Spausdinimas, Elektroninė Komercija, 3D Grafika, Automatizacija, Golang, Vue.js

Vilnius Gediminas Technical University

Faculty of Fundamental Science

Information Technology department

Information Technology bachelor study program final thesis

Title **Web-based System for Ordering and Managing 3D Prints**

Author **Zehra İrem Kuyucu** Supervisor **Dr. Danylo Shkundalov**

Language

	Lithuanian
x	English

Annotation

This thesis introduces a Web-based System for Ordering and Managing 3D Prints, addressing the inherent complexities and accessibility barriers of 3D printing services, including steep learning curves, inconsistent pricing, and laborious manual workflows for file validation, configuration, and quoting. The system automates and streamlines the entire 3D print ordering and management process, featuring automated parameter adjustments, robust print validation, remote slicing, and instant price quotations. It offers users real-time material cost estimations, interactive 3D model, and G-code visualizations via a user-friendly interface. An integrated administrator panel manages inventory and orders, while a payment gateway ensures seamless transactions with automated status updates. This system significantly reduces operational overhead for service providers, enhances user accessibility, and offers a robust framework for monetizing public 3D printers. Its scalable architecture supports a future network model, allowing individual printer owners to manage inventories and offer services, thereby extending its reach beyond centralized operations to a distributed 3D printing ecosystem.

Keywords: 3D Printing, E-Commerce, 3D Graphics, Automation, Golang, Vue.js

Table of Contents

List of Images

List of tables

List of abbreviations and terms

API	Application Programming Interface
UI	User Interface
STL	Standard Tessellation Language
3MF	3D Manufacturing Format
FBX	Filmbox File Format
GLTF	Graphics Library Transmission Format
WebGL	Web Graphics Library
SQL	Structured Query Language
HTTP	Hypertext Transfer Protocol
SMTP	Simple Mail Transfer Protocol
JSON	JavaScript Object Notation
YAML	Yet Another Markup Language
G-code	Geometric code (3D printing programming language)
NPM	Node Package Manager
TLS	Transport Layer Security
CSS	Cascading Style Sheets
PLA	Polylactic acid (3D printing filament)
PETG	Polyethylene terephthalate glycol (3D printing filament)
TPU	Thermoplastic polyurethane (3D printing filament)
S3	Amazon Simple Storage Service

Introduction

Relevance of the Thesis

The entry into 3D printing comes with a steep learning curve, even as the technology gains popularity across various sectors, hobbyists, and increasingly, in public spaces such as makerspaces [1], libraries [2], and educational institutions [3]. Users in these settings face challenges not only in understanding the technology but also in navigating the often complex and inconsistent pricing models. While makerspaces might rely on subscriptions or donations, and universities might charge hourly, a standardized and transparent payment system is often lacking. Furthermore, inexperienced users in these public settings may inadvertently cause printing failures due to incorrect material or setting selections, leading to wasted resources and equipment downtime. Similarly, users engaging with 3D printing service companies face a different set of complexities. They encounter challenges in enduring a time-consuming back-and-forth communication process for file validation, setting configuration, material selection, quoting, and order finalization. This prolonged exchange slows production, increases costs, and complicates the process for those seeking even simple prototype prints.

Problem of the Thesis

This thesis addresses these challenges by developing the Web-based System for Ordering and Managing 3D Prints, a platform that automates the 3D print ordering and management workflow. By streamlining key processes such as parameter adjustments, print validation, slicing, and price quotations, the system minimizes the need for ongoing oversight by service providers. With integrated inventory information, users can access information on available materials and configuration, eliminating the need for extensive consultations. This automation-driven approach not only improves operational efficiency but also enhances user convenience, providing immediate feedback through real-time pricing, STL and G-code visualizations, and instant order evaluation.

Object of the Thesis

The developed system which includes a web frontend for users to upload 3D object files, place orders, and receive feedback and visualizations. An administrator panel manages inventory and orders. The backend server handles print evaluation and pricing, integrating with slicing software and a payment processor.

Objective of the Thesis

This thesis aims to develop the Web-based System for Ordering and Managing 3D Prints.

Tasks of the Thesis

The four main tasks, each detailed in its own chapter, are:

1. **Analysis of 3D Printing Services and Technologies.** Evaluate existing solutions and technologies to identify requirements.
2. **System Development.** Design architecture and implement the system.
3. **System Documentation.** Create comprehensive documentation and present implementation results.
4. **System Testing.** Validate the system's functionality and usability to ensure it meets the defined requirements.

Scientific Novelty

While individual software tools exist for aspects like slicing, real-time visualization, and cost estimation, these are primarily desktop or mobile applications. Notably absent is commercially available off-the-shelf software that integrates these functionalities with comprehensive order management, inventory control, and print validation specifically tailored for 3D printing services or public facilities. Although some 3D printing businesses may have internal systems, these proprietary solutions lack broad accessibility.

Practical Values

The developed system offers significant practical value by streamlining the 3D print ordering process for both service providers and users through automation. This reduces operational overhead and costs for providers while enhancing accessibility for non-technical users. Furthermore, the platform provides a framework for the monetization of publicly accessible 3D printers in settings like makerspaces and universities by offering transparent pricing and mitigating user-error-induced failures. The system's architecture also supports future scalability into a network model, enabling individual printer owners to manage their inventories and offer services alongside larger print farm operations.

Research Methods

The research methodology consisted of several stages. Market analysis was conducted using search engines to identify global and local 3D print service providers. Each identified platform was manually evaluated through direct interaction to assess functionality, user experience, and feature completeness.

Technical research involved systematic examination of open-source projects in code repositories to identify potential solutions and implementation approaches. Web-based 3D graphics technologies were evaluated against predefined criteria including learning curve, integration capabilities with modern frameworks, and support for relevant file formats.

Development followed an iterative approach informed by the research findings, with regular evaluation against the identified requirements. The validation methodology focused on comparing the developed system's capabilities against limitations identified in existing solutions, particularly regarding automation, visualization, and cost estimation features.

1. Analysis of 3D Printing Services and Technologies

This section presents an analysis of the current landscape and challenges within 3D printing services, alongside the theoretical considerations guiding the technological choices for the developed system.

1.1. Background for a 3D Print Order System

3D printing, also known as additive manufacturing, involves creating objects layer by layer from a digital model. Central to this process is slicing software, which transforms 3D models into machine instructions. The slicing process divides the model into horizontal layers, calculates the movement paths of the printer's nozzle, and generates G-code commands to control the printer. Users can manipulate object alignment, configure build plates, and add supports or infills for structural strength. This process begins with a 3D model in formats such as STL or OBJ, which are then converted into G-code containing detailed instructions for the printer. While users can manipulate object alignment, configure build plates, and add supports or infills for structural strength, the process is far from straightforward. Different filament materials necessitate specific and often critical combinations of print bed adhesion methods, precise temperature settings for both the bed and the print nozzle, and even the type and size of the nozzle itself [4]. Incorrectly configured parameters can lead to print failures, wasted material, and, in more severe cases, even damage to the printer components. Understanding these fundamental yet nuanced requirements highlights the complexity involved for new users and underscores the steep learning curve associated with achieving successful 3D prints.

1.1.1. Fields of Application

3D printing has diverse applications across industries and for individual users. In manufacturing, it is widely used for prototyping and producing small-scale parts. The healthcare sector relies on it for custom prosthetics, surgical tools, and implants, while education uses it for creating teaching tools and fostering hands-on experimentation. In architecture, 3D printing enables scaled models and design validation, and in consumer goods, it is popular for producing customizable products like jewelry and toys. Beyond industries, online 3D print ordering services cater to hobbyists who lack their own printers and businesses that require specialized production capabilities or materials. The wide range of applications underscores the need for efficient ordering and management systems that can cater to diverse user needs.

1.1.2. 3D Printing File Formats

Several file formats are central to 3D printing workflows, each serving a specific purpose. STL files remain a common choice for representing the surface geometry of a 3D model using a mesh of triangles. While widely compatible, STL's primary limitation is its inability to store information beyond geometry, such as material properties, color, or texture. G-code is another critical format, acting as the machine language that dictates the 3D printer's operations. It contains precise instructions for nozzle movement, temperature control of the nozzle and build plate, and material extrusion rates. In contrast to STL's geometric focus, the 3MF offers a more comprehensive approach. As a container format, 3MF can embed not only the 3D model geometry but also data related to materials, colors, textures, print settings, and even entire G-code files. While STL and G-code represent fundamental aspects of the 3D printing process, 3MF aims to streamline workflows and enhance data exchange by packaging more information within a single file.

1.2. Analysis of Existing Solutions

Local 3D print shops typically exhibit no automation in their order processes. Users usually need to contact these providers directly, leading to significant back-and-forth communication to clarify requirements, obtain quotes, and manage the order. Features like online file uploads, instant quoting, and dynamic 3D model visualization are often entirely absent [5], [6], [7].

In contrast, Table 1 presents a comparison of the user interfaces offered by several of the largest global online 3D print shops, which represent some of the most automated solutions currently available. As indicated in the table, while many existing platforms, such as JLC3DP [8] and Craftcloud [9], allow file uploads and provide instant quotes, they often lack a responsive, interactive user interface with dynamic 3D model visualization. For instance, Shapeways [10] displays a 3D object viewer, but color selection isn't applied to the visual representation. Relying on static images, as seen with Craftcloud, limits the user's understanding of the final printed object. Furthermore, these platforms generally do not offer an integrated G-code viewer for pre-print inspection.

The *Print Configuration* column in Table 1 highlights the varying degrees of user control over printing parameters. While some shops like Shapeways and JLC3DP allow users to specify finish and filament, crucial parameters like layer height and support usage are often absent. This lack of granular control can impact the final print quality and material usage.

Notably, the *Software Available* column indicates whether the platform's software is available for other businesses to acquire and deploy for their own 3D printing services. As the table shows, only one of the compared proprietary platforms offer their software for external deployment. This highlights a gap in the market for a comprehensive, readily deployable solution.

Furthermore, although one open-source project exists within the 3D printing ecosystem [11], it currently lacks crucial features such as integrated slicing capabilities, inventory management, automated print validation, and comprehensive option validation. The developed system aims to provide a more complete and user-friendly solution compared to both proprietary and existing open-source alternatives.

Table 1. *Comparison of features in online 3D print services with automation (created by the author)*

	3D Object Viewer	**Instant Quote**	**Print Configuration**	**G-code Viewer**	**License**	**Software Available**
Shapeways [10]	Yes, but the color selection isn't applied to the object.	Yes	Filament – yes Layer height – no Supports – no Infill – no Finish – yes	No	Proprietary	No
JLC3DP [8]	Yes, but a static image of the object is displayed.	Yes	Filament – yes Layer height – no Supports – no Infill – no Finish – yes	No	Proprietary	No
Craftcloud [9]	Yes, but a static image of the object is displayed.	Yes, at checkout.	Filament – yes Layer height – no Supports – no Infill – yes Finish – yes	No	Proprietary	No
Materialise [12]	Yes	Yes	Filament – yes Layer height – no Supports – no Infill – no Finish – yes	No	Proprietary	Yes [13]
Zaccord [14]	Yes	Yes, at checkout.	Filament – yes	No	Open Source	Yes [11]

			Layer height – yes Supports – no Infill – yes Finish – no			

1.3. Fragmentation of 3D Printer Technologies

The 3D printing industry is characterized by significant fragmentation in both hardware and software. Printers vary widely in terms of supported file formats, APIs, firmware, and other hardware-specific details. Some systems are open-source, allowing greater flexibility for integration, while others are proprietary, limiting compatibility [15]. As a result, targeting a specific printer vendor can often seem essential when designing a specialized solution.

While this project was initially conceived with a focus on Bambu Lab printers due to their growing popularity and advanced features, the system's architecture has evolved to support a wide range of printer models and brands. Bambu Lab printers utilize the 3MF format with the Production extension, which has limited compatibility with generic 3MF files used by other vendors like Ultimaker [16] and may not be directly supported by all software libraries such as the Three.js library used in this thesis [17].

However, this system overcomes the challenges posed by format fragmentation through a flexible approach to slicing. While the system is configured to integrate directly with the Bambu Studio slicer binary for streamlined file generation (as Bambu Studio itself supports a wide array of popular printer brands beyond just Bambu Lab), the system's design also acknowledges the compatibility of this binary with other slicers like Orca Slicer. The path to the slicer binary is configurable within the system, allowing for potential switching to alternatives with minor code adjustments if desired. Furthermore, recognizing the need for vendor-agnostic workflows, the system also accommodates standard G-code and STL files, allowing users to utilize their preferred slicing software and upload G-code directly for print management, ensuring compatibility with numerous other printer models and brands.

1.4. Web-Based 3D Graphics Technologies

There are several libraries available for rendering 3D objects and animations in web browsers. WebGL is a JavaScript API that enables 3D graphics rendering in compatible browsers without requiring plug-ins. All modern browsers support WebGL [18]. Its successor, the WebGPU

API [19], is more performant but is not yet supported in some widely used browsers. While supporting both APIs is ideal, this thesis focuses on WebGL.

Using WebGL directly requires working with its extensive low-level API. Two widely used frameworks, Three.js and Babylon.js, provide higher-level abstractions over the WebGL API. There are also higher-level frameworks built on top of Three.js or Babylon.js that add features. Some of these frameworks were evaluated for this thesis.

Table 2 lists the selection criteria and the evaluations of the considered technologies. The *Learning Curve* column assesses the quality of existing documentation and examples and the ease of running a simple example. The *Integration Support* column evaluates whether the framework can integrate with Vue.js 3 and TypeScript. The *File Formats* column evaluates support for commonly used 3D printing file formats. Cells are color-coded to indicate suitability: green (high), yellow (medium), and red (low).

Table 2. *Evaluation of web-based 3D graphics frameworks (created by the author)*

	Learning Curve	**Integration Support**	**File Formats**
Three.js [20]	Average	Average	3MF, G-code, STL
Babylon.js [21]	Average	Average	STL
model-viewer (Three.js) [22]	Excellent (simple API)	Average	Only supports glTF/GLB which are not commonly used in 3D printing.
TresJS (Three.js) [23]	Difficult (Documentation and examples are not straightforward)	Excellent (Vue.js components available)	No composables for 3D printing file formats.
TroisJS (Three.js) [24]	Excellent (simple API, documentation and examples targeting Vue.js)	Excellent (Vue.js components available)	FBX and GLTF only.

Among the evaluated frameworks, using Three.js directly is the most feasible option. While other frameworks offer simpler interfaces, they lack the extensibility and features needed for loading the file formats required for this project.

2. System Development

In this chapter the system requirements are specified, architecture is described in detail using various graphs, as well as any technical limitations encountered during the development phase.

2.1. Requirements

This section specifies the requirements that the system must fulfill. These are divided into functional requirements, detailing what the system must do, and non-functional requirements, defining how the system should perform.

2.1.1. Functional Requirements

Table 3 provides a detailed specification of the functional requirements for the developed system.

Table 3. *Functional requirements of the Web-based System for Ordering and Managing 3D Prints (created by the author)*

User Management (Administrator)	
Inventory Access	The administrator user shall have access to the system's inventory management features.
Order History Access	The administrator user shall have access to view and manage the complete order history.
File Handling and Validation	
File Upload	The system shall enable users to upload 3D model files in STL format.
File Validation	The system shall automatically validate uploaded files to ensure they are suitable for 3D printing.
Print Parameter Configuration	
Parameter Specification	Users shall be able to define key print parameters, including infill density, filament type, and support structures.
Real-time Cost Estimation	The system shall dynamically display a cost estimate based on the user's selected print parameters.
Visualization	

3D Model Visualization	The system shall provide an interactive, real-time 3D visualization of the uploaded model.
G-code Preview	The system shall display a preview of the generated G-code based on the chosen print parameters.
Slicing	
Remote Slicing	The system shall perform the slicing process on a remote server based on the user-configured parameters.
Order Management (User)	
Order Submission	Users shall be able to submit their print order once the slicing is validated.
Payment Processing	The system shall integrate with a payment gateway to process user payments.
Order Tracking (User)	
Order Lookup	The system shall allow users to enter a unique order code to view the details and current status of their order.
Inventory Management (Administrator)	
Filament Management	Admin users shall be able to add new filaments, edit existing filament details, and remove filaments from the inventory.
Process Management	Admin users shall be able to add, edit, and remove available print finishes.
Configuration Selection (User)	
Inventory Display	Users shall be able to view a real-time list of available materials and process profiles based on the current inventory.
Error Handling (User)	
Notification	The system shall clearly notify users of any errors encountered during

	file validation or slicing.

2.1.2. Non-functional Requirements

Table 4 provides a detailed specification of the non-functional requirements for the developed system.

Table 4. *Non-functional requirements of the Web-based System for Ordering and Managing 3D Prints (created by the author)*

Performance	
Operation Speed	The system shall process file uploads, slicing operations, and parameter updates within a reasonable time depending on the file size and network conditions.
Portability	
Browser Support	The system shall be compatible with the latest versions of major web browsers, including Chrome, Firefox, and Edge.
Usability	
Fast Checkout	Customers must be able to place orders in under five minutes, even without prior experience using the system.
Security	
Administrator Area	The system shall protect administrator functionality behind authentication.
Secure Web Application	The system shall employ measures to mitigate against OWASP Top 10 vulnerabilities [25].
Maintainability	
Modularity	The system shall be composed of discrete components with specific functionality.
Extensive Logging	The system shall log system events and errors to allow for debugging and maintenance.

Reliability	
Error Handling	The system must gracefully handle errors without crashing.

2.2. Use Cases

Use case analysis was performed as part of the requirements gathering stage. The detailed specifications of the use cases are provided in Table 5.

Table 5. *Use cases for the Web-based System for Ordering and Managing 3D Prints (created by the author)*

UC1	**Parameter Configuration**
	1. The user uploads a 3D model file and proceeds to the configuration interface. **2.** The system allows the user to specify print parameters such as infill density, support structures, and filament type. **3.** The system updates the cost estimate in real time based on the selected parameters. **4.** The user has the option to preview the sliced model as a 3D visualization. **5.** The user can accept the parameters or discard them and reconfigure.
UC2	**Remote Slicing and Validation Process**
	1. The user finalizes the parameters and submits the model for slicing. **2.** The system performs slicing remotely and validates the G-code for compatibility with the target printer. **3.** The user can preview the sliced G-code and receive feedback on print feasibility. **4.** If slicing fails, the system notifies the user. **5.** If slicing is successful, the system enables the user to proceed with the order.
UC3	**Instant Order System**
	1. The user uploads a file. **2.** The user confirms the print settings and submits the order. **3.** The system generates a unique order code, calculates the final price, including material costs and additional charges, and displays the information to the user.

	4. The user makes the payment via the integrated payment gateway. **5.** The system generates a confirmation with an estimated delivery timeline. **6.** If the payment fails, the system allows the user to retry or cancel the order.
UC4	**Inventory Management**
	1. The administrator user navigates to the material selection section of the web portal. **2.** The system displays available filaments and processes based on current inventory. **3.** The user manages inventory by adding or removing filaments and processes. **4.** The changes to the inventory are saved and reflected in the order interface.
UC5	**Order Tracking**
	1. The user submits an order code. **2.** The system displays the order details and current status.

2.3. System Architecture

The system features a web-based frontend application that provides interfaces for both users and the administrator. This frontend communicates solely with a backend API server to manage all system functionalities. The backend API server handles core logic, data management, and integrates with external services. For computationally intensive tasks such as 3D model slicing and cost estimation, the API server communicates with a separate processing agent. The separation of the agent from the main server enhances future-proofing and flexibility. This agent-based approach allows for potential future scaling through the deployment of multiple agents or the implementation of load balancing techniques, such as DNS round robin. The described system is illustrated in Figure 1.

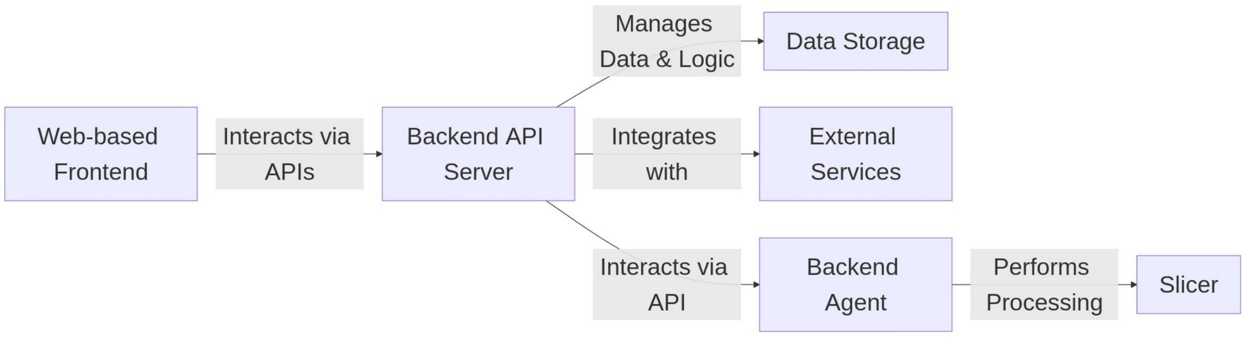


Figure 1. *The Web-based System for Ordering and Managing 3D Prints system architecture (created by the author)*

2.4. Technology Selection Rationale

The selection of key technologies for the developed system was driven by specific requirements and considerations for efficiency, scalability, ease of development, and maintainability of both the user interface and the backend services.

For the frontend implementation, Vue.js 3 and TypeScript were chosen. Vue.js 3's reactivity features enable dynamic updates to the user interface, providing a more engaging and responsive user experience. Its component-based architecture promotes modular development, making the codebase easier to manage and extend. TypeScript was adopted for its static typing capabilities, which enhance code maintainability and reduce the likelihood of runtime errors, particularly in a complex web application. The PrimeVue component library was leveraged to provide a set of pre-built, accessible UI components, significantly accelerating the development process and ensuring a consistent and user-friendly interface. Tailwind CSS was used for styling the user interface, offering a utility-first approach that allows for rapid and consistent styling with minimal custom CSS.

The Go programming language was selected for the backend components due to its strong support for concurrency and efficient handling of network operations, making it well-suited for building robust and scalable backend services and APIs.

For data storage, SQLite was chosen for the server due to its simplicity, file-based nature, and ease of setup, making it suitable for managing order information, user data, and configuration settings in the current phase of development without the immediate need for a complex relational database or high-concurrency writes. Redis was integrated for session management of administrator users, providing a fast and reliable in-memory data store for persisting session IDs and authenticated sessions across server restarts.

Both the server and the agent utilize MinIO for file storage. MinIO, which is compatible with the Amazon S3 API, offers scalable and reliable object storage, crucial for managing potentially large 3D model files, G-code files, and other print-related assets.

Stripe was selected for payment processing due to its secure and reliable platform and its official Go library (stripe-go), which facilitated smooth integration. Email communication, utilizing the standard SMTP protocol for sending order confirmations and updates, ensures interoperability with various email servers. The system also integrates with the Bambu Studio slicer to generate print instructions.

2.5. Database Schema

The database schema consists of four main tables: orders, panel_logins, available_filaments, and available_processes. Within the orders table, the print_config column utilizes a JSON structure to store details about the specific print configuration for that order. This JSON object contains information identifying the selected filament (corresponding to an entry in the available_filaments table) and the selected print process (corresponding to an entry in the available_processes table). The database schema is illustrated in Figure 2.

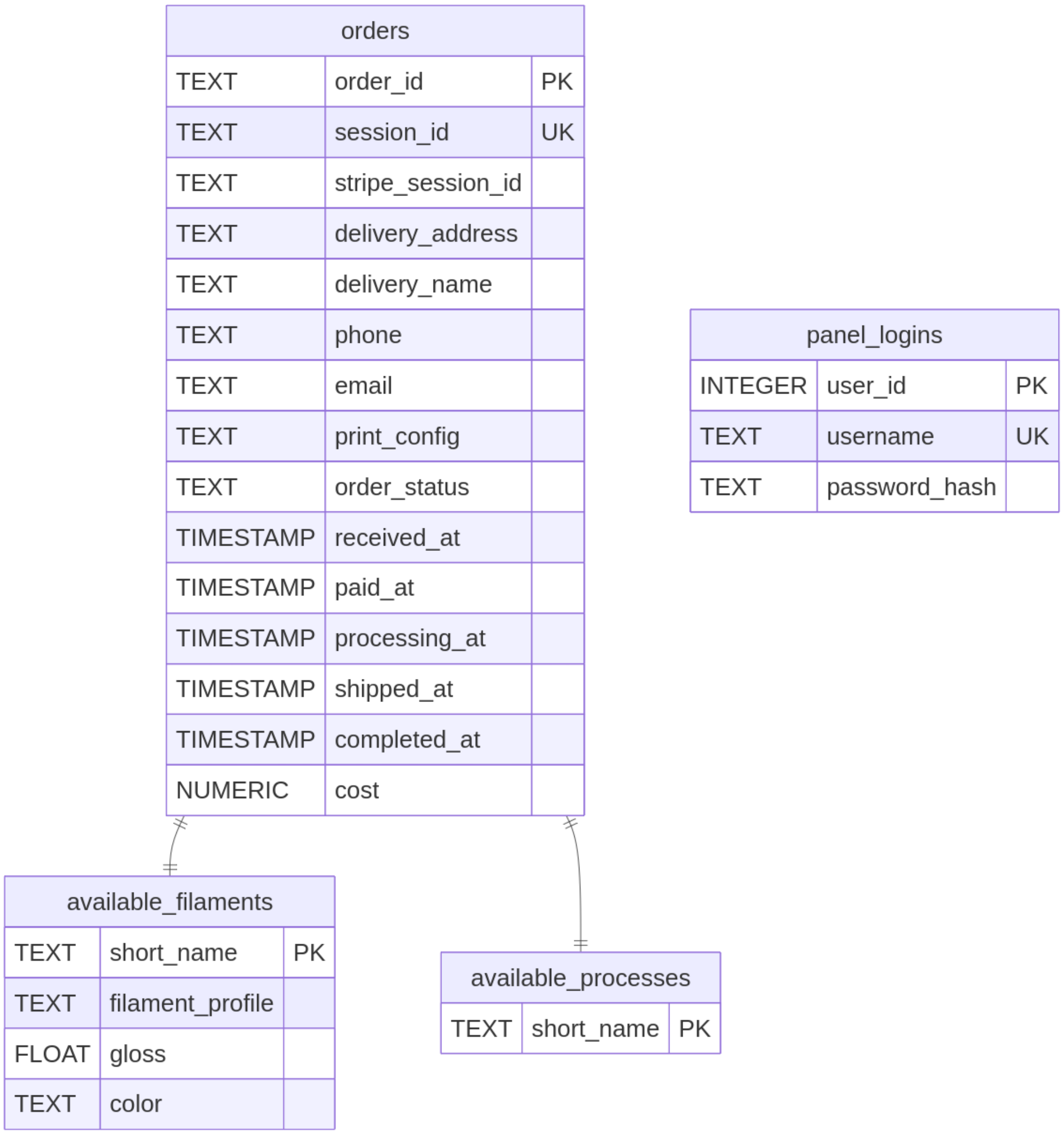


Figure 2. *Database schema (created by the author)*

2.6. Component Diagram

The system is divided into a frontend and a backend. The frontend has an order platform and an admin panel, both communicating with the backend via HTTP APIs.

The backend architecture includes a server and an agent. Initially, the system's design aimed to incorporate all backend functionalities, including print processing, within the server component.

However, to enhance future-proofing and flexibility, especially considering the resource-intensive nature of slicing, a separate agent component was introduced to handle print processing via its own HTTP API, interacting with a slicer.

The server provides order and admin APIs and manages data storage, email communication, payment processing, and session management. Both the server and agent use file storage. External services integrated into the system include a file storage service (MinIO), an email communication service (SMTP), a payment processing service (Stripe), and the slicer (Bambu Studio). The component diagram is provided in Figure 3.

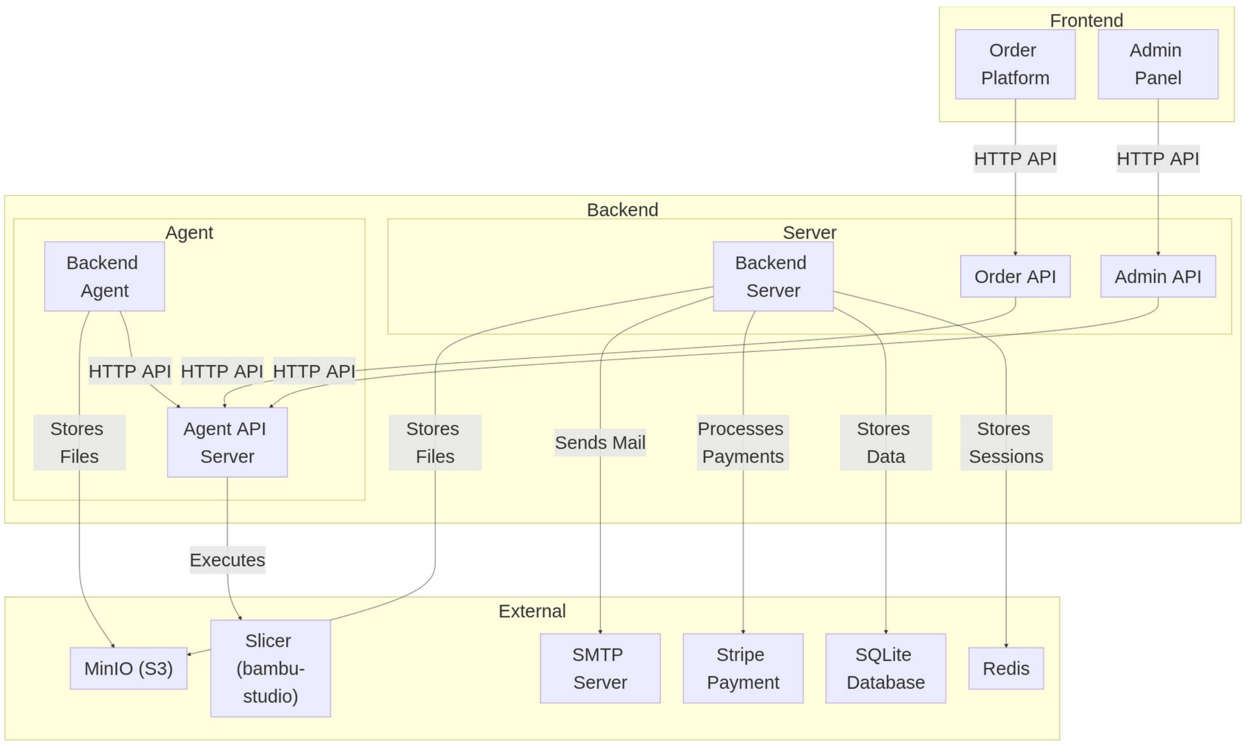


Figure 3. *Component diagram (created by the author)*

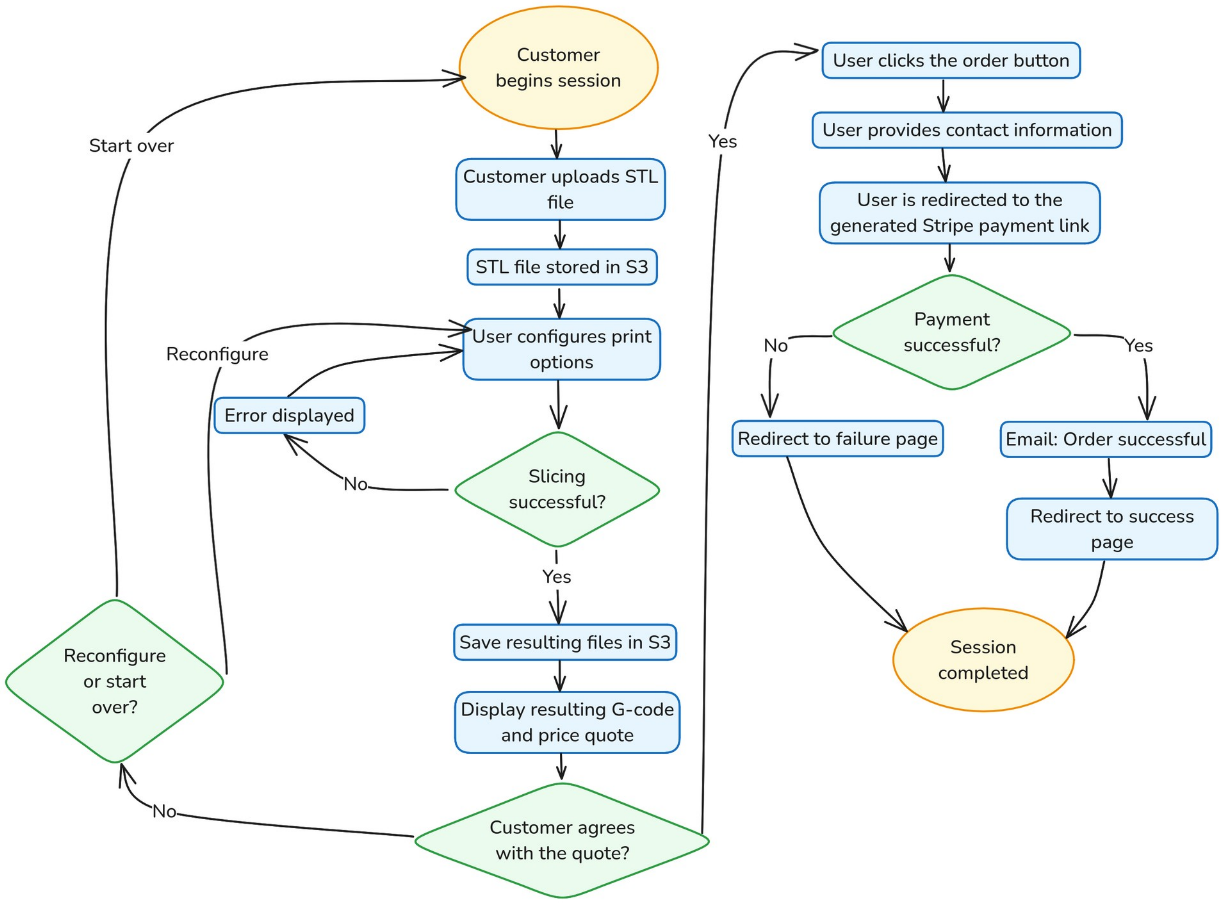
2.7. Print Order Process

The print order process outlines the steps a customer takes to place a 3D print order using the platform. As illustrated in the accompanying flow diagram, the process begins with the customer initiating a session and uploading an STL file, which is then stored in the system's storage. Following the upload, the customer configures their desired print options. Throughout this configuration phase, the system provides real-time feedback, including a live price quote and, crucially, an indication of whether the slicing process (performed in the background) is successful. If slicing is unsuccessful, an error is displayed to the user, allowing them to reconfigure their print options or start over. If slicing is successful, the generated 3MF and G-code files are stored, and the live price is presented. The customer then has the option to either agree with the quote and proceed to the checkout process or reconfigure their settings.

Upon agreeing to the quote, the customer clicks the “Order” button and provides their contact information. The system then creates an order with a unique tracking ID and generates a payment link via Stripe. The customer is redirected to Stripe to complete their payment. The outcome of the payment determines the next steps: if successful, the customer receives an order confirmation via email, and the order session is complete. If the payment fails, the customer is redirected to a payment failure page.

2.7.1. Flow Diagram

The flow diagram provided in Figure 4 visually represents the state transitions and decision points within the print order process. It details the progression from the initial file upload through configuration, slicing validation, payment processing, and final order completion. The diagram highlights validation checks, particularly the success of the slicing operation and the payment status, and illustrates the alternative paths users may take based on system feedback and their choices. This flow diagram, in conjunction with the sequence diagram, provides a comprehensive understanding of the user's journey and the system's behavior during the order placement process.



Customer begins session
Customer uploads STL file
STL file stored in S3
User configures print options
Slicing successful?
No
Error displayed
Reconfigure
Yes
Save resulting files in S3
Display resulting G-code and price quote
Customer agrees with the quote?
No
Reconfigure or start over?
Start over
Yes
User clicks the order button
User provides contact information
User is redirected to the generated Stripe payment link
Payment successful?
No
Yes
Redirect to failure page
Email: Order successful
Redirect to success page
Session completed

Figure 4. *Flow diagram of the order process (created by the author)*

2.7.2. Sequence Diagrams

In this section the successful order path through the system is explained using four sequence diagrams.

Figure 5 describes the initial customer interaction with the system, including beginning the session, uploading the STL file, storing it in MinIO (S3 storage), and rendering it in the frontend for the customer to visualize.

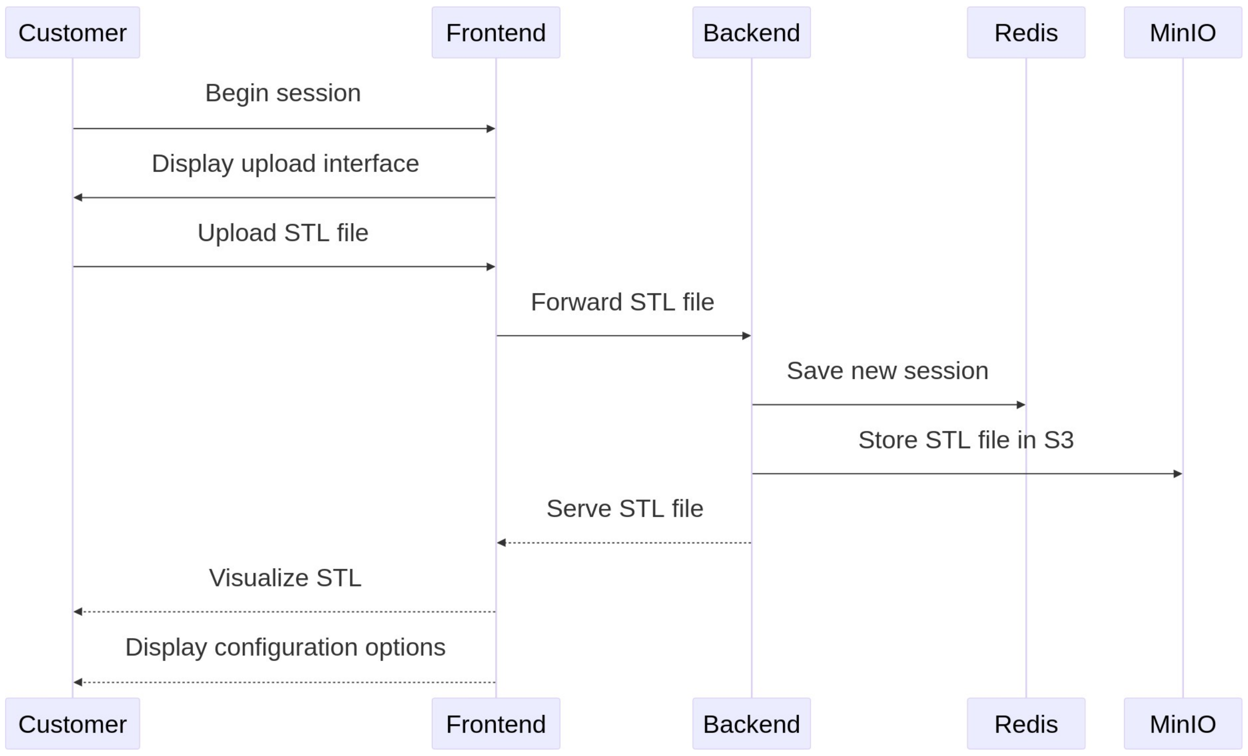


Figure 5. *Sequence diagram of the initial file upload flow (created by the author)*

Figure 6 demonstrates the technical core of the system where the customer configures print options, which are sent to the agent component. The agent retrieves the file from storage, processes it through the slicer, generates the required files (3MF and G-code), and calculates the cost estimate for the customer.

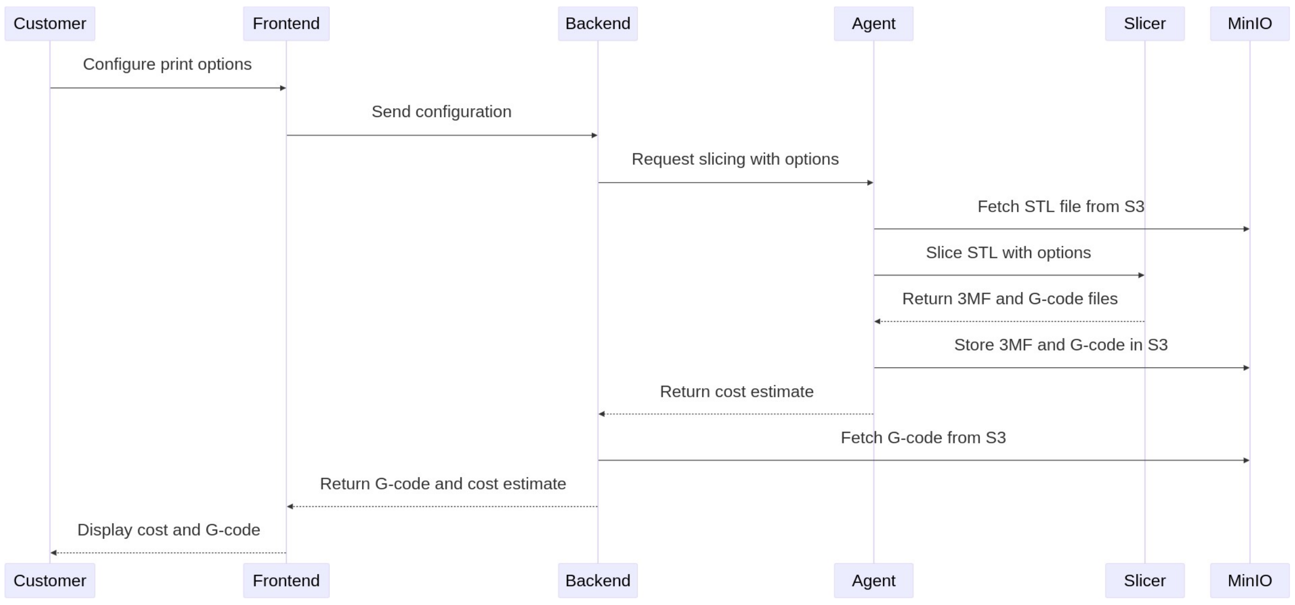


Figure 6. *Sequence diagram of the slicing step (created by the author)*

Figure 7 covers the business transaction flow, from the customer deciding to place an order through providing contact information, creating the order record in SQLite, generating a payment link via Stripe, and completing the payment.

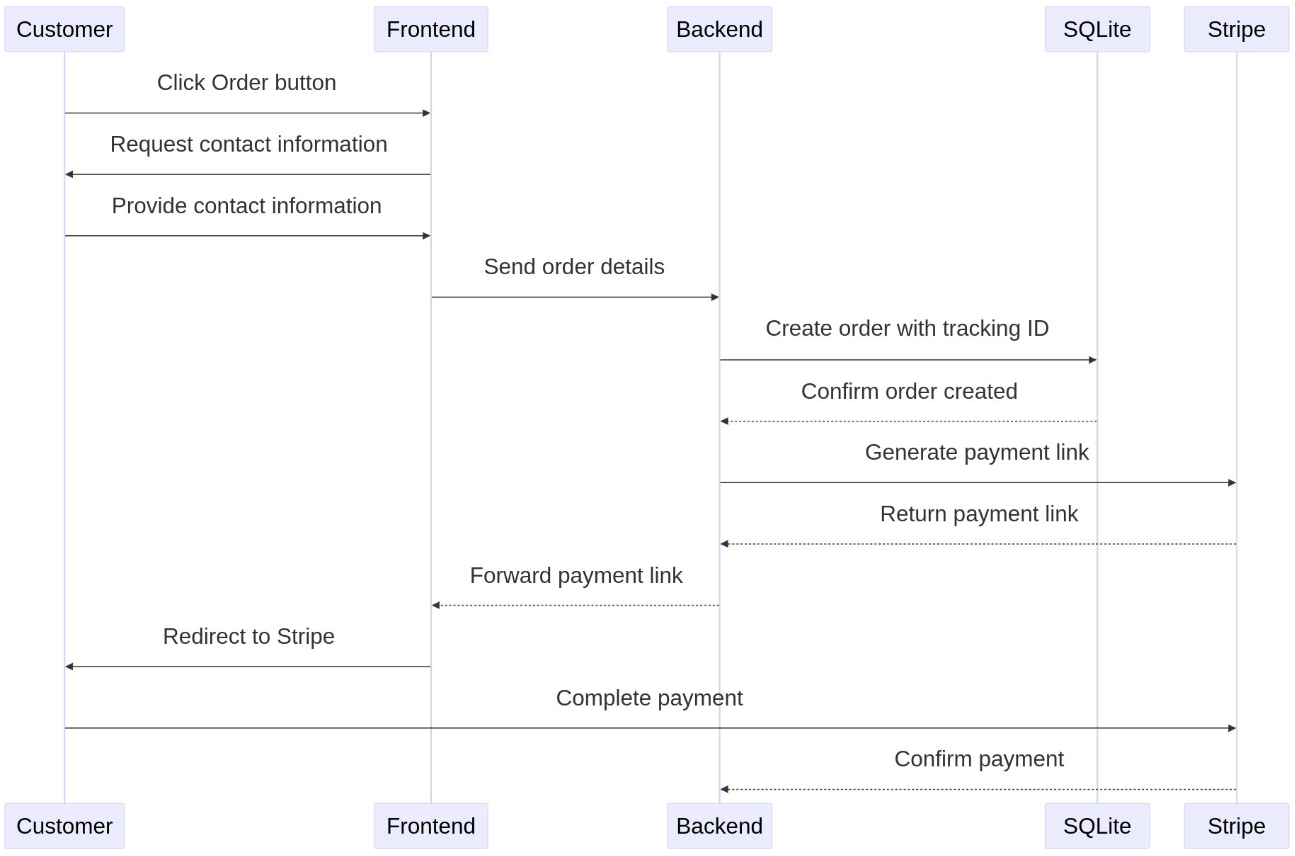


Figure 7. *Sequence diagram of the order submission step (created by the author)*

The final diagram shown in Figure 8 shows what happens after successful payment: updating the order status in the database, sending confirmation emails, cleaning up the session data in Redis, and redirecting the customer to a success page.

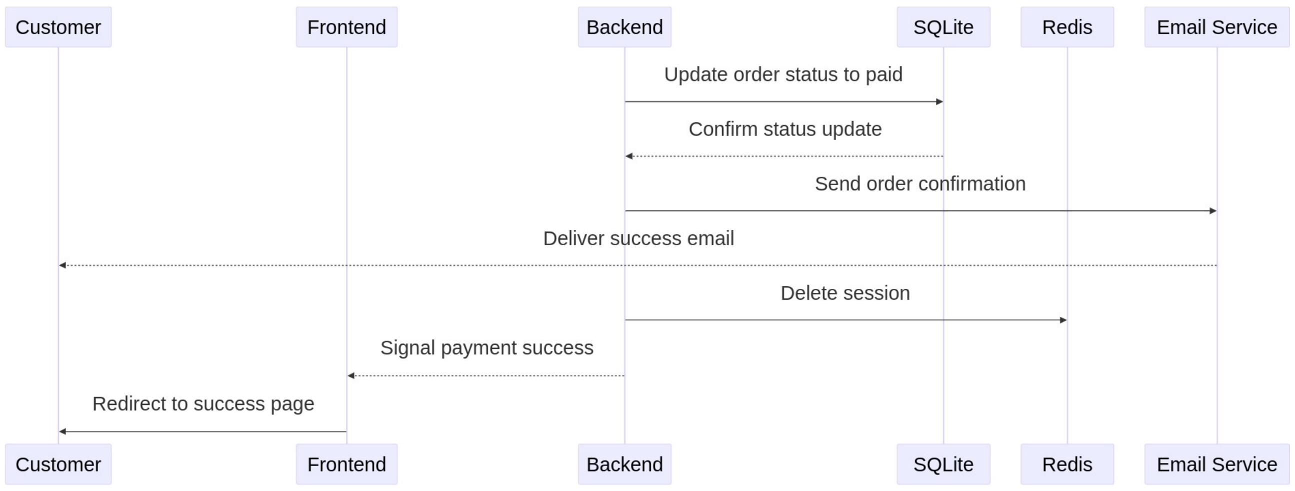


Figure 8. *Sequence diagram of a successful payment flow (created by the author)*

2.8. Inventory Management

The developed system enables the management of available printer processes and filaments. To facilitate this, the agent maintains profile files for the machine, processes, and filaments, which are then passed on to the slicer binary. The structure of the agent's profiles directory is described in Figure 9.

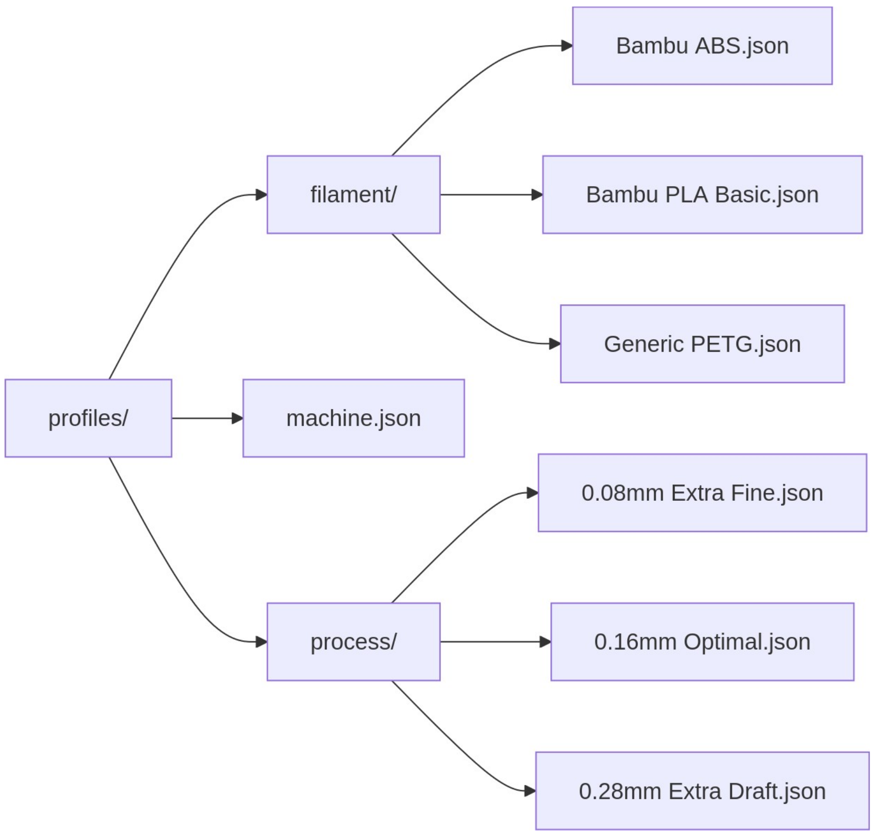


Figure 9. *Profile directory structure (created by the author)*

A key aspect of filament management is handling color variations within a filament type (e.g., Red PLA, Blue PLA). To address this, a single file is used as a template to create one or more filaments in the database. When an administrator views available templates for filament creation through the frontend, the backend queries the agent API to retrieve these profile files.

When the administrator creates a new filament using the frontend, the backend saves the template name along with display-specific properties: color and gloss percentage. These color and gloss details are used for visualization purposes in the frontend. During the slicing process, the agent relies on the template name sent from the backend to select the appropriate profile.

No templating occurs with process profiles; they are simply enabled or disabled in the database.

2.9. Print Validation and Cost Estimation

The agent performs print validation by analyzing the slicer's output. When invoking the slicer binary, the agent automatically enables features such as orientation optimization, part arrangement on the build plate, and support structure generation. The agent also dynamically selects the appropriate build plate based on the requested filament type. This selection is performed by evaluating the filament name and mapping it to a suitable bed type. For instance, standard PLA is assigned the "Cool Plate", while TPU utilizes the "Engineering Plate" and PETG is assigned the "Textured PEI Plate". Figure 10 contains the code section where the agent executes the slicer binary and passes the required arguments.

```
      cmd := exec.Command(
            config.Config.Slicer.BinaryPath,
            "--slice", "0",
            "--orient", "1",
            "--arrange", "1",
            "--load-filaments", "filament.json",
            "--export-3mf", sessionID+".3mf",
            "--curr-bed-type", buildPlate,
            "--load-settings", "process.json;"+config.Config.Profiles.Machine,
            sessionID+".stl",
      )
```

Figure 10. *Agent code calling slicer binary with flags (created by the author)*

Cost estimation is calculated based on the total mass of the printed object (in grams) and a pricing model that includes a base price and a price multiplier per gram of the selected filament. Specifically, the final cost is determined by the formula: Final Cost = Base Price + Total Mass in Grams * Price Multiplier. Consequently, enabling support structures or using a higher infill density will increase the estimated cost due to the greater material consumption, which is then scaled by the price multiplier and added to the base price.

The G-code file generated by the slicer contains information about the estimated material usage. An example of a G-code header block providing this information is shown in Figure 11.

```
; HEADER_BLOCK_START
; BambuStudio 02.00.01.50
; model printing time: 1h 7m 16s; total estimated time: 1h 15m 1s
; total layer number: 240
; total filament length [mm] : 4000.39
; total filament volume [cm^3] : 9622.05
; total filament weight [g] : 12.12
; filament_density: 1.26
; filament_diameter: 1.75
; max_z_height: 48.00
; HEADER_BLOCK_END
```

Figure 11. *G-code header (created by the author)*

This G-code header block includes the “total filament weight [g]” parameter, which the system uses for cost calculation in conjunction with a base price and a per-gram multiplier. While each filament profile file stores a “filament_cost” attribute (representing a base price per kilogram), the final cost calculation incorporates a fixed base price and a price multiplier applied to the estimated filament grams from the G-code.

2.10. File Visualization

To visualize G-code and STL files, two Vue.js components, “GCodeViewer” and “STLViewer”, were developed. Initially, the 3MF format was considered for model visualization due to its smaller file size and inclusion of color information. However, at the time of writing, the Three.js library lacked support for 3MF files with the Production extension produced by Bambu Studio. While future support is anticipated, STL was chosen as the alternative. To accurately represent the intended appearance, the color and glossiness attributes of the selected filament are retrieved and applied to the material properties (color and shininess) of the STL mesh within the Three.js scene.

2.11. Automatic Slicing

The design of the frontend prioritizes efficient resource utilization by automating the slicing process. To this end, users cannot manually initiate slicing; instead, it is triggered automatically when print configuration parameters are altered. This automation is coupled with a debouncing mechanism to prevent excessive slicing requests due to rapid UI interactions. The Vue.js watch function observes changes in printing parameters (selected filament, layer height, infill density, support usage, and brim). The implementation of automatic slicing is detailed in Figure 12.

```
watch([selectedFilament, layerHeight, infillDensity, useSupports, useBrim],
async () => {
  // Only proceed if a file has been uploaded
  if (uploaded.value && !isLoading.value) {
    // Check if any options have changed compared to the last slice
    if (areOptionsChanged()) {
      // Set options changed flag to true
      optionsChanged.value = true;

      // Add a small debounce to prevent multiple rapid slices
      if (window.sliceDebounceTimer) {
        clearTimeout(window.sliceDebounceTimer);
      }
      // Debounce the slicing operation
      window.sliceDebounceTimer = setTimeout(async () => {
        // Trigger reslicing with new parameters
        await sliceStl();

        // Fetch updated cost
        await fetchSessionData();
      }, 1500);
    }
  }
}, { deep: true });
```

Figure 12. *Watching print configuration changes and debouncing (created by the author)*

However, the "POST /slice" API endpoint can be used directly to bypass the frontend measures to manually trigger a reslice. To safeguard against potential denial-of-service attacks in a production environment, the implementation of a rate limiter on this endpoint is a necessary security measure.

2.12. Authentication

For client-server interactions, the application utilizes Redis-backed session management with a cookie named "p3d" to maintain user state. When the server needs to query the agent API for tasks such as slicing or fetching available profiles, it authenticates via a secret key mechanism, transmitting the pre-configured secret value in an HTTP header called "x-secret". This authentication mechanism is illustrated in Figure 13.

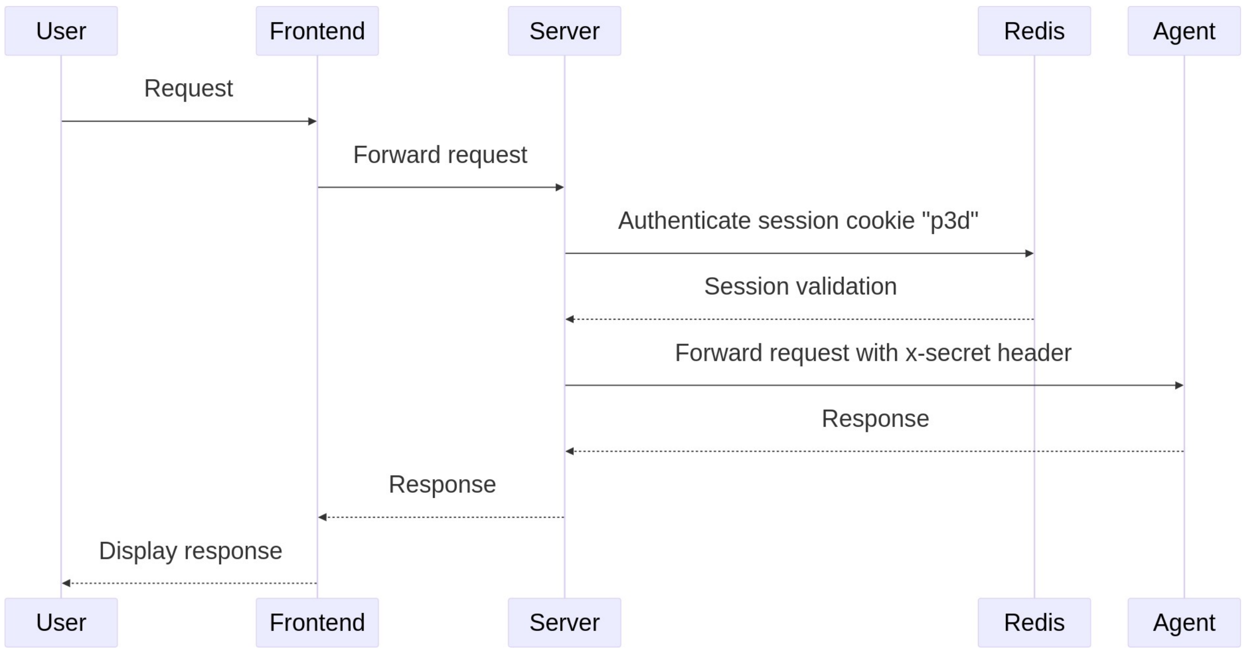


Figure 13. *Authentication mechanisms in the developed system (created by the author)*

3. System Documentation

3.1. Installation

The system comprises a frontend and a backend, each managed within separate Git repositories. The frontend utilizes NPM for dependency management and compilation, while the backend employs the Go toolchain for similar processes.

3.1.1. System Configuration

The system handles configuration for its frontend and backend parts in different ways, making it possible to adjust settings for development or production environments.

For the frontend, settings like the backend API URLs and locale preferences for displaying dates and currency are set using environment variables. Depending on the target environment, these variables are loaded from either an .env.development file or an .env.production file before the frontend application is built. This way, settings can be changed for development or production without needing to modify the actual frontend code.

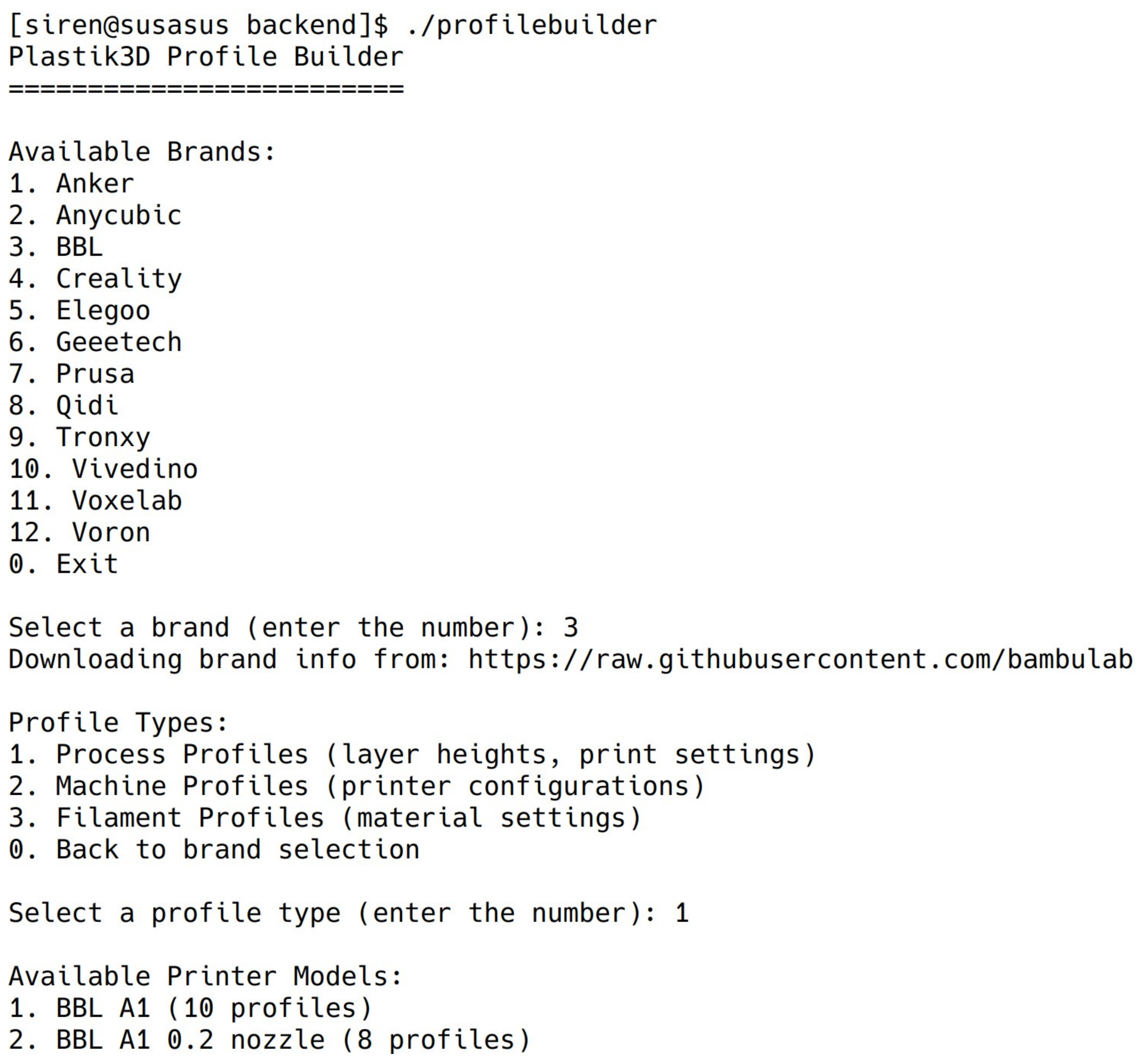
The backend's configuration uses two main YAML files: agent-config.yaml and server-config.yaml. The server-config.yaml file holds the settings for the main server application. This includes details for the database connection, internal API configurations, credentials for MinIO object storage, payment gateway keys, and email service settings.

The agent-config.yaml file is used to configure the agent application. It sets the necessary paths for filament and process profiles, defines the machine profile to use, includes API settings, provides the MinIO credentials needed by the agent, and contains various parameters for the slicing engine.

3.1.2. Profile Builder Helper Utility

Profile files are necessary for the backend agent to function correctly. Since these files are not included with the codebase, the system administrator must acquire them from repositories such as Bambu Studio's or other slicer software. A further complexity arises from the inheritance structure within these profile files, a feature not directly processed by Bambu Studio when invoked through the command line. This inheritance requires users to manually merge options from inherited files into their parent files.

To simplify this process, a helper script has been developed. This utility assists users in downloading the appropriate profile files for their specific printer brand and model and automatically composes complete profiles, resolving external inheritance without manual merging. The interface of the helper script is presented in Figure 14.


```
[siren@susasus backend]$ ./profilebuilder
Plastik3D Profile Builder
=========================

Available Brands:
1. Anker
2. Anycubic
3. BBL
4. Creality
5. Elegoo
6. Geeetech
7. Prusa
8. Qidi
9. Tronxy
10. Vivedino
11. Voxelab
12. Voron
0. Exit

Select a brand (enter the number): 3
Downloading brand info from: https://raw.githubusercontent.com/bambulab

Profile Types:
1. Process Profiles (layer heights, print settings)
2. Machine Profiles (printer configurations)
3. Filament Profiles (material settings)
0. Back to brand selection

Select a profile type (enter the number): 1

Available Printer Models:
1. BBL A1 (10 profiles)
2. BBL A1 0.2 nozzle (8 profiles)
```
Figure 14. *Profile builder interface (created by the author)*

3.1.3. Docker Compose for Rapid Deployment

A Docker Compose configuration has been created to facilitate rapid deployment of the backend and its self-hosted dependencies. This setup allows for the entire backend environment to be launched with a single command. The docker-compose.yaml file defines the services required to run four containers: agent, server, MinIO, and Redis, employing Docker volumes to ensure data persistence.

3.1.4. Web Server Configuration

A web server is required to deploy the system effectively, handling several key tasks. It serves the compiled static files that constitute the frontend application, and it also acts as a reverse proxy for the backend APIs. A critical reason for using a reverse proxy in this manner is to serve both the frontend application and the backend APIs under the same domain. This unification is necessary to prevent Cross-Origin Resource Sharing (CORS) errors that web browsers would otherwise enforce if the frontend code tried to access APIs on a different domain, protocol, or port.

Furthermore, the system relies on the web server to provide TLS encryption, ensuring secure HTTPS communication. The same-domain architecture enabled by the reverse proxy is also integral to the application's authentication flow. The backend sets a HttpOnly cookie named p3d upon successful login. Because the frontend and backend share the same origin thanks to the reverse proxy, the browser automatically sends this p3d cookie with subsequent requests the frontend makes to the backend API, thereby authenticating those requests.

Figure 15 provides an example configuration using the Caddy 2 web server that fulfills these requirements. This configuration demonstrates reverse proxying for the API endpoints (/api/ and /admin_api/), serving static frontend files, and implicitly handling automatic TLS certificate provisioning for the specified domain.

```
plastik3d.net {
    handle_path /api/* {
        uri strip_prefix /api
        reverse_proxy localhost:3000 {
            header_up Host {upstream_hostport}
            header_up X-Forwarded-Host {host}
            header_up X-Forwarded-Proto {scheme}
            header_up X-Real-IP {remote}
        }
    }

    handle_path /admin_api/* {
        uri strip_prefix /admin_api
        reverse_proxy localhost:3001 {
            header_up Host {upstream_hostport}
            header_up X-Forwarded-Host {host}
            header_up X-Forwarded-Proto {scheme}
            header_up X-Real-IP {remote}
        }
    }

    handle {
        root * /srv/www/plastik3d
        try_files {path} {path}/ /index.html
        file_server
    }
}
```

Figure 15. Example Caddyfile showing reverse proxy and static file serving (created by the author)

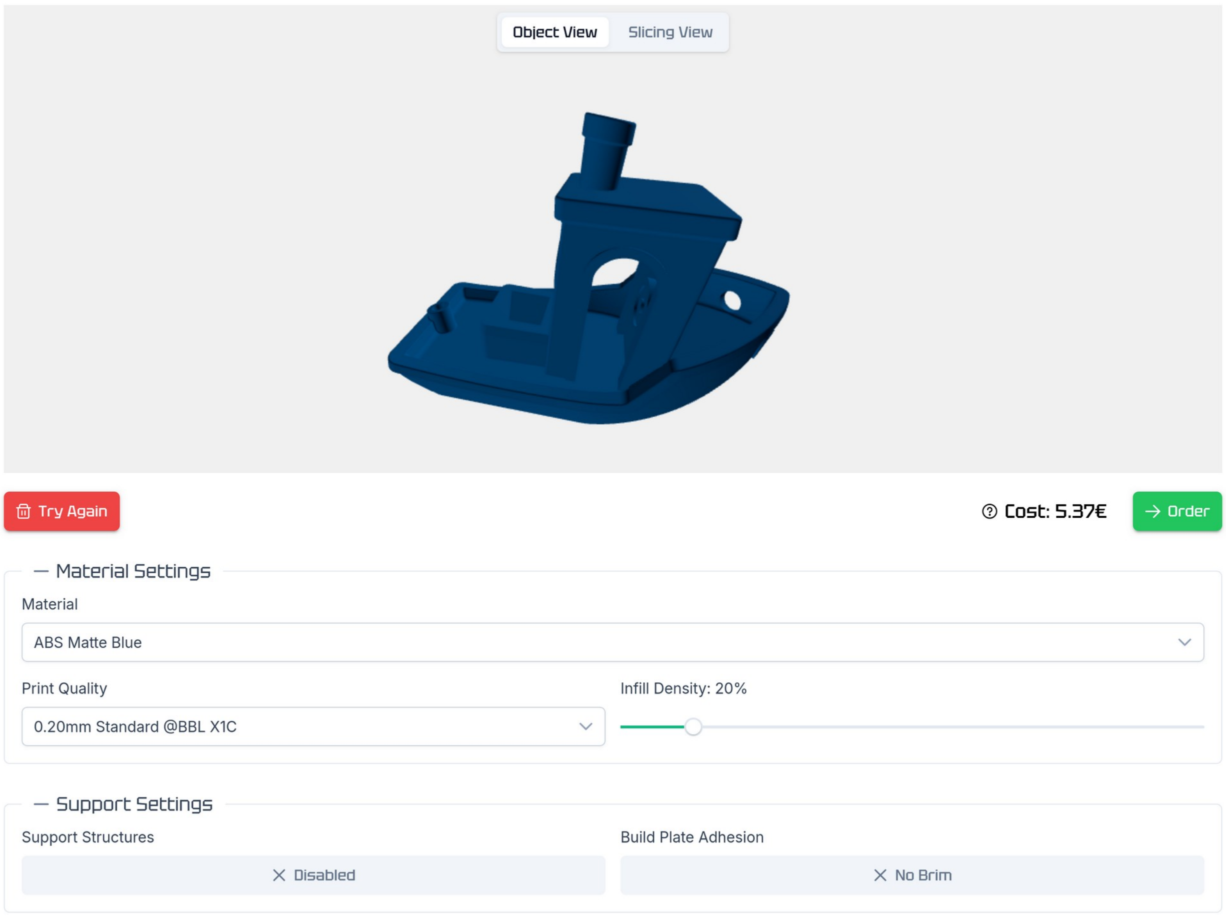
3.2. User Guide

This section details the system usage from the customer's perspective, providing a comprehensive overview of the available features and interfaces.

3.2.1. Print Studio

The Print Studio serves as the workbench for customers to prepare models for 3D printing. Within this interface, users can upload STL files, visualize models in different views, configure print settings, and place orders. The Print Studio provides two distinct visualization modes:

As shown in Figure 16, Object View displays the rendered 3D model, allowing users to examine the overall structure and appearance of their design before printing. The 3D model can be rotated and examined from all angles.



— Material Settings

Material: ABS Matte Blue

Print Quality: 0.20mm Standard @BBL X1C

Infill Density: 20%

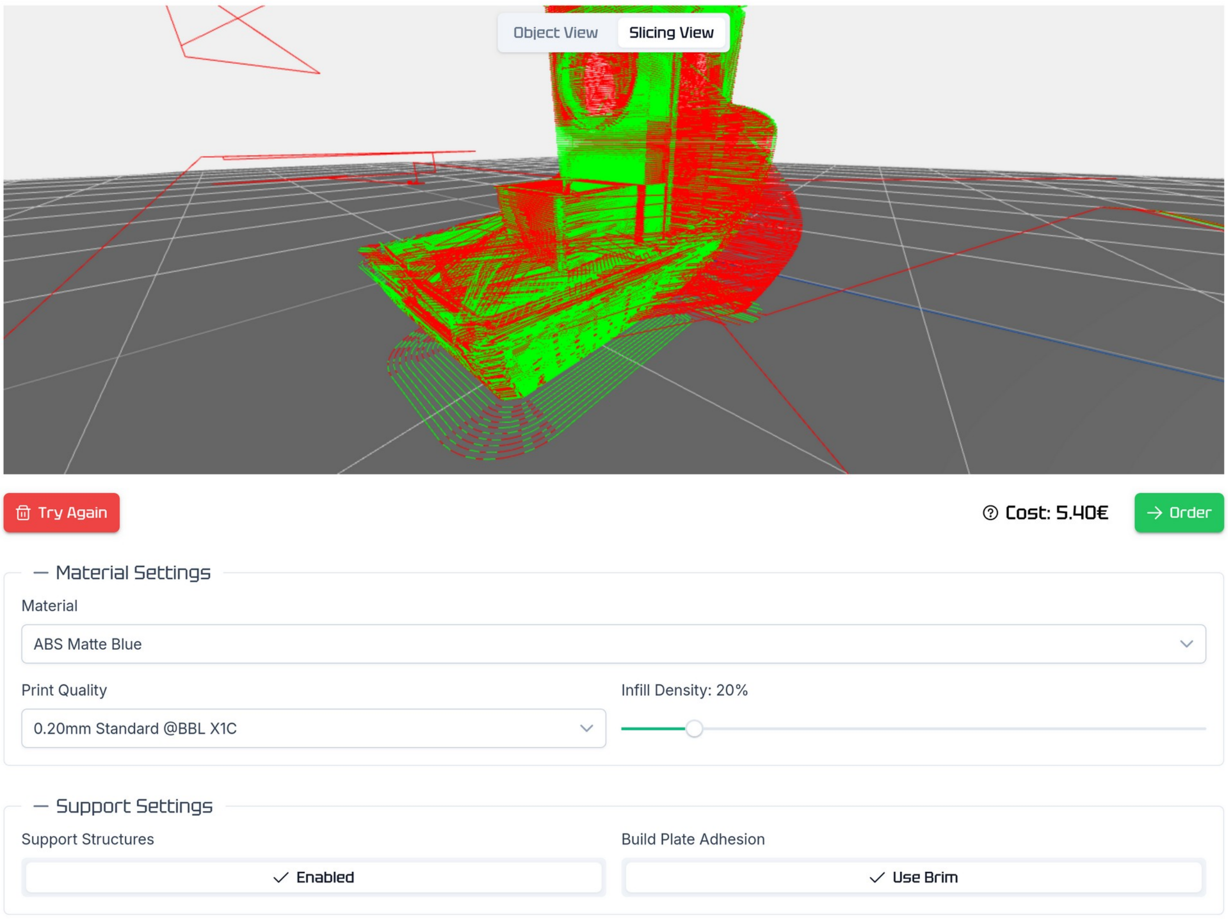
— Support Settings

Support Structures: Disabled

Build Plate Adhesion: No Brim

Figure 16. *STL visualization (created by the author)*

As shown in Figure 17, Slicing View displays the G-code visualization with support structures and printing paths, giving users insight into how the model will be printed layer by layer.



Cost: 5.40€

— Material Settings

Material

ABS Matte Blue

Print Quality

0.20mm Standard @BBL X1C

Infill Density: 20%

— Support Settings

Support Structures

✓ Enabled

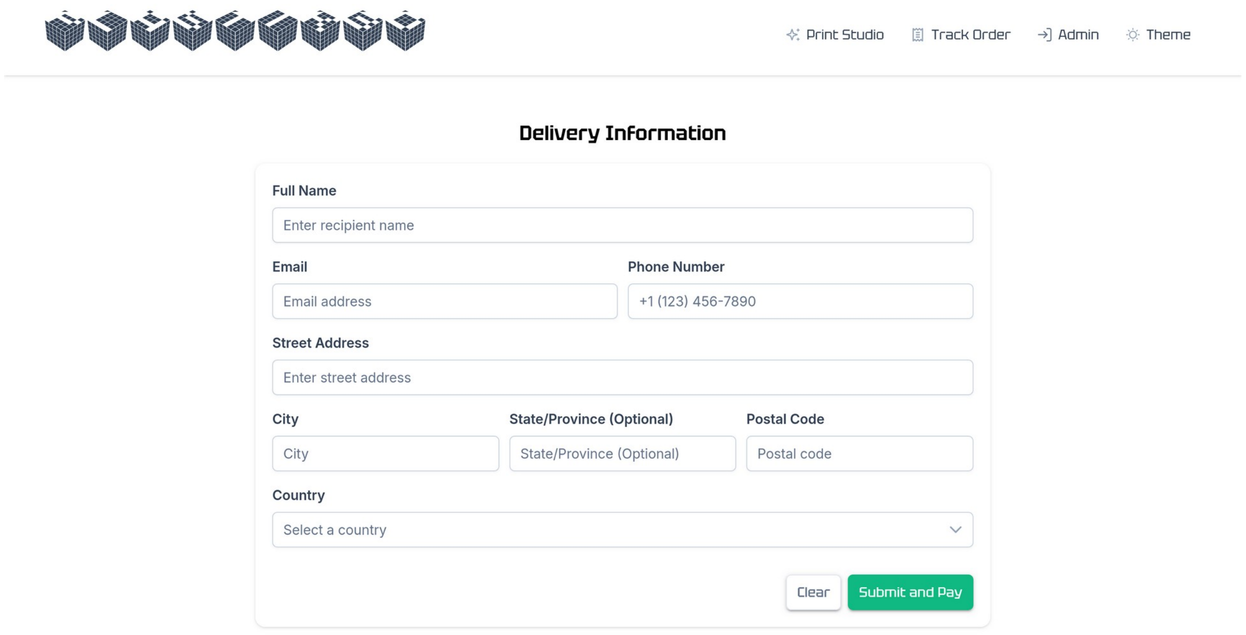
Build Plate Adhesion

✓ Use Brim

Figure 17. *G-code visualization with supports enabled (created by the author)*

3.2.2. Placing Orders

Once the print configuration is finalized, the user initiates the order placement process by providing their contact information via the form illustrated in Figure 18. This form requires the following mandatory fields: full name, email address, phone number, street address, city, postal code, and country. A field for state or province is also included, but it is only mandatory when the selected country is the United States.

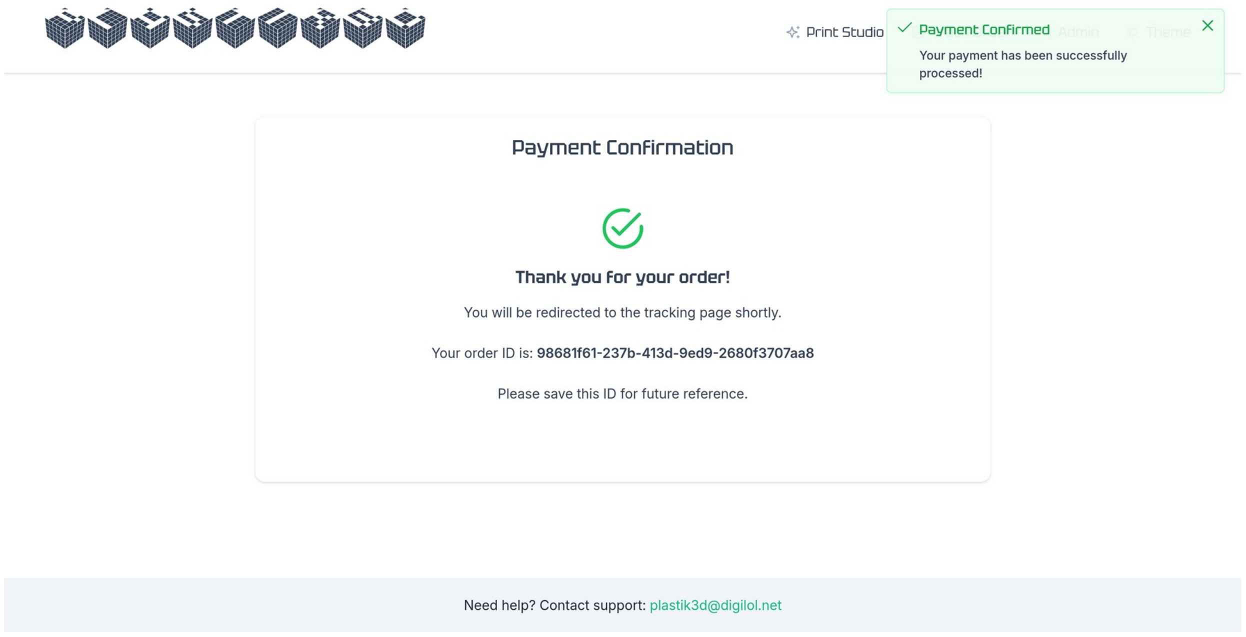


Delivery Information

Full Name		
Enter recipient name		
Email	Phone Number	
Email address	+1 (123) 456-7890	
Street Address		
Enter street address		
City	State/Province (Optional)	Postal Code
City	State/Province (Optional)	Postal code
Country		
Select a country		

Figure 18. *Form for collecting delivery information (created by the author)*

Upon submitting their contact details, the user is redirected to Stripe, a payment gateway, to enter their card information and complete the payment. Following successful payment authorization, the user is initially directed to a confirmation page as shown in Figure 19 and subsequently automatically redirected to the order tracking page.



Payment Confirmed

Your payment has been successfully processed!

Payment Confirmation

Thank you for your order!

You will be redirected to the tracking page shortly.

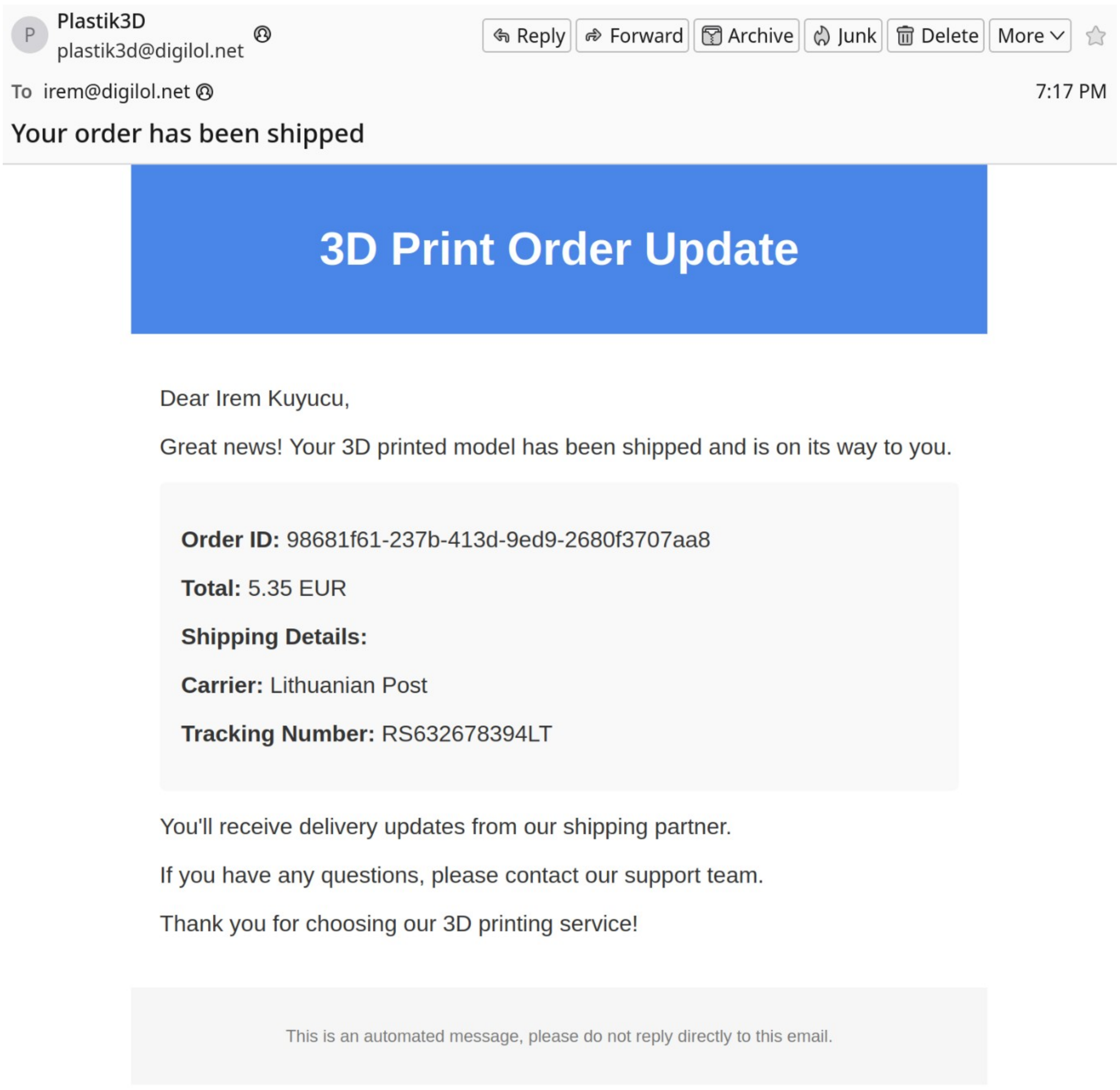
Your order ID is: **98681f61-237b-413d-9ed9-2680f3707aa8**

Please save this ID for future reference.

Need help? Contact support: plastik3d@digilol.net

Figure 19. *Payment confirmation page (created by the author)*

The order tracking page displays a unique order ID, which the user can utilize to check the real-time status of their order. Additionally, the user receives an immediate email confirmation upon successful order placement. This email also includes the crucial order ID for reference. Throughout the order fulfillment process, the user will continue to receive subsequent status updates via email, keeping them informed of their order's progress. Figure 20 contains an example email a customer would receive once their order is marked as shipped.



Your order has been shipped

3D Print Order Update

Dear Irem Kuyucu,

Great news! Your 3D printed model has been shipped and is on its way to you.

Order ID: 98681f61-237b-413d-9ed9-2680f3707aa8

Total: 5.35 EUR

Shipping Details:

Carrier: Lithuanian Post

Tracking Number: RS632678394LT

You'll receive delivery updates from our shipping partner.

If you have any questions, please contact our support team.

Thank you for choosing our 3D printing service!

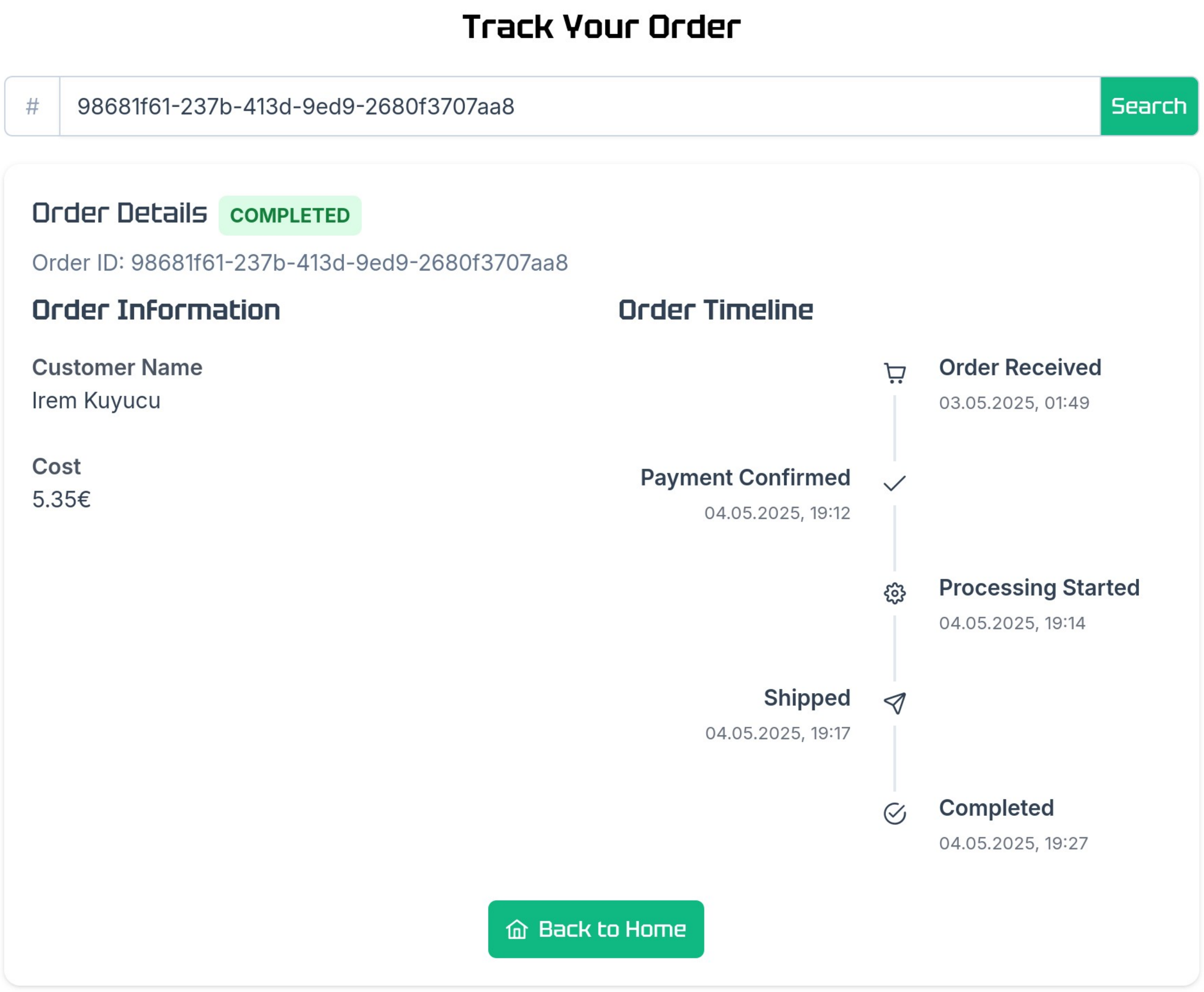
This is an automated message, please do not reply directly to this email.

Figure 20. *Contents of the order update received via email once an order has been shipped (created by the author)*

3.2.3. Tracking Orders

Users can monitor the status of their placed orders by entering their unique order ID into the order tracking page. This functionality is particularly useful for users who may not be receiving email notifications for any reason, providing an alternative way to stay informed about the progress of their print jobs.

As shown in Figure 21, when queried, this page contains information on the cost, the current status, and the order timeline, detailing the history of order updates.


Track Your Order

98681f61-237b-413d-9ed9-2680f3707aa8 Search

Order Details COMPLETED

Order ID: 98681f61-237b-413d-9ed9-2680f3707aa8

Order Information

Customer Name
Irem Kuyucu

Cost
5.35€

Order Timeline

Order Received
03.05.2025, 01:49

Payment Confirmed
04.05.2025, 19:12

Processing Started
04.05.2025, 19:14

Shipped
04.05.2025, 19:17

Completed
04.05.2025, 19:27

Back to Home

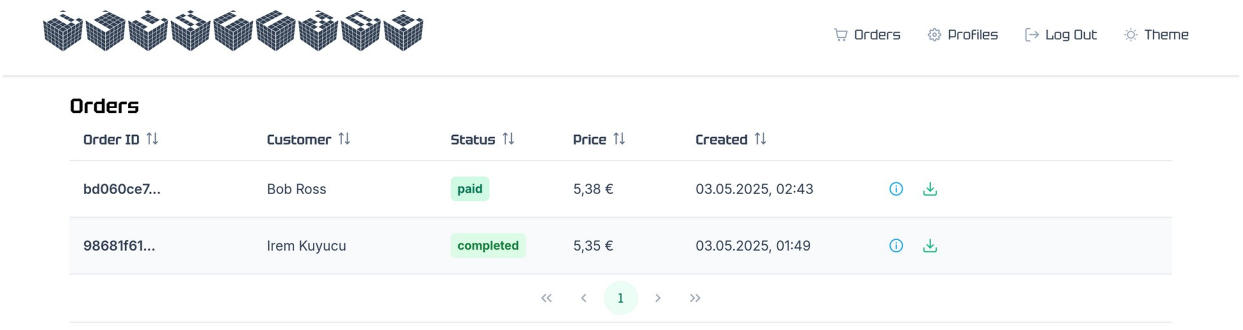
Figure 21. *Order tracking page (created by the author)*

3.3. Administrator Guide

This section outlines the system functionalities accessible to the administrator user. All administrative pages are available after successful authentication.

3.3.1. Managing Orders

The administrator can access and monitor all orders and their current statuses via the Orders page as presented in Figure 22.



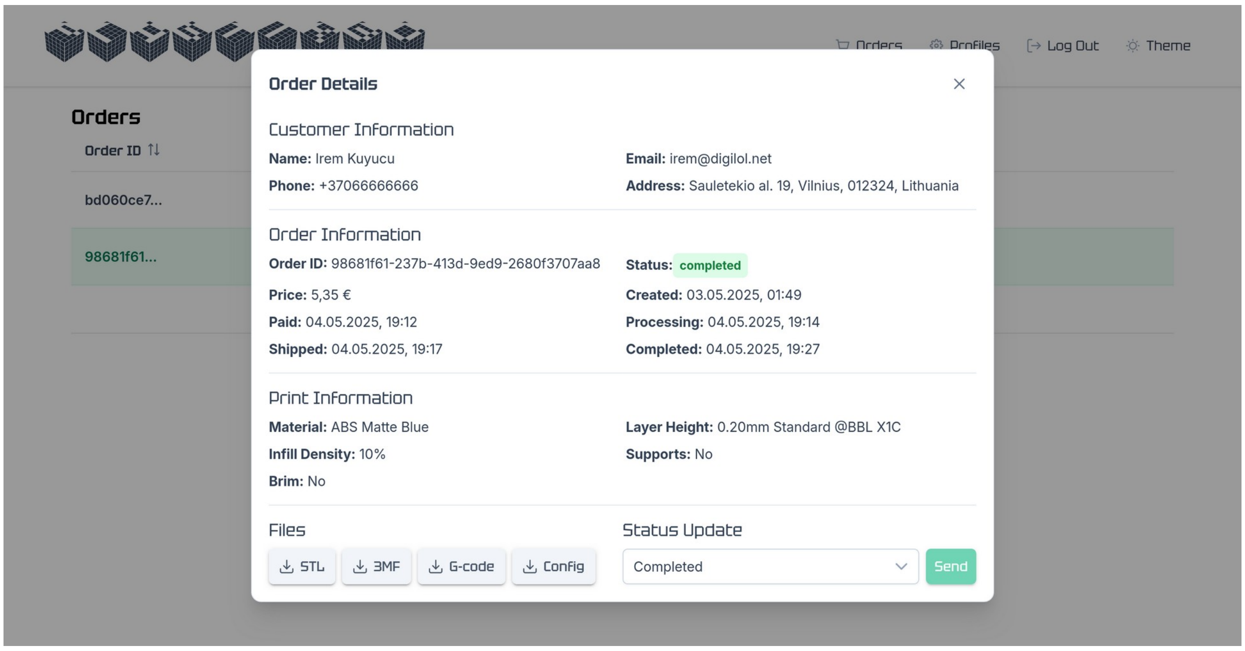
Orders

Order ID	Customer	Status	Price	Created
bd060ce7...	Bob Ross	paid	5,38 €	03.05.2025, 02:43
98681f61...	Irem Kuyucu	completed	5,35 €	03.05.2025, 01:49

« ‹ 1 › »

Figure 22. *Orders list (created by the author)*

On the Orders list page, each order entry provides quick actions: an information button and a download button. Clicking the information button opens an Order Details modal as shown in Figure 23, while the download button initiates an immediate download of the associated 3MF project file for that order.



Orders
Order ID
bd060ce7...
98681f61...

Order Details

Customer Information
Name: Irem Kuyucu
Email: irem@digilol.net
Phone: +37066666666
Address: Sauletekio al. 19, Vilnius, 012324, Lithuania

Order Information
Order ID: 98681f61-237b-413d-9ed9-2680f3707aa8
Status: completed
Price: 5,35 €
Created: 03.05.2025, 01:49
Paid: 04.05.2025, 19:12
Processing: 04.05.2025, 19:14
Shipped: 04.05.2025, 19:17
Completed: 04.05.2025, 19:27

Print Information
Material: ABS Matte Blue
Layer Height: 0.20mm Standard @BBL X1C
Infill Density: 10%
Supports: No
Brim: No

Files
Status Update
Completed

Figure 23. *Order details modal (created by the author)*

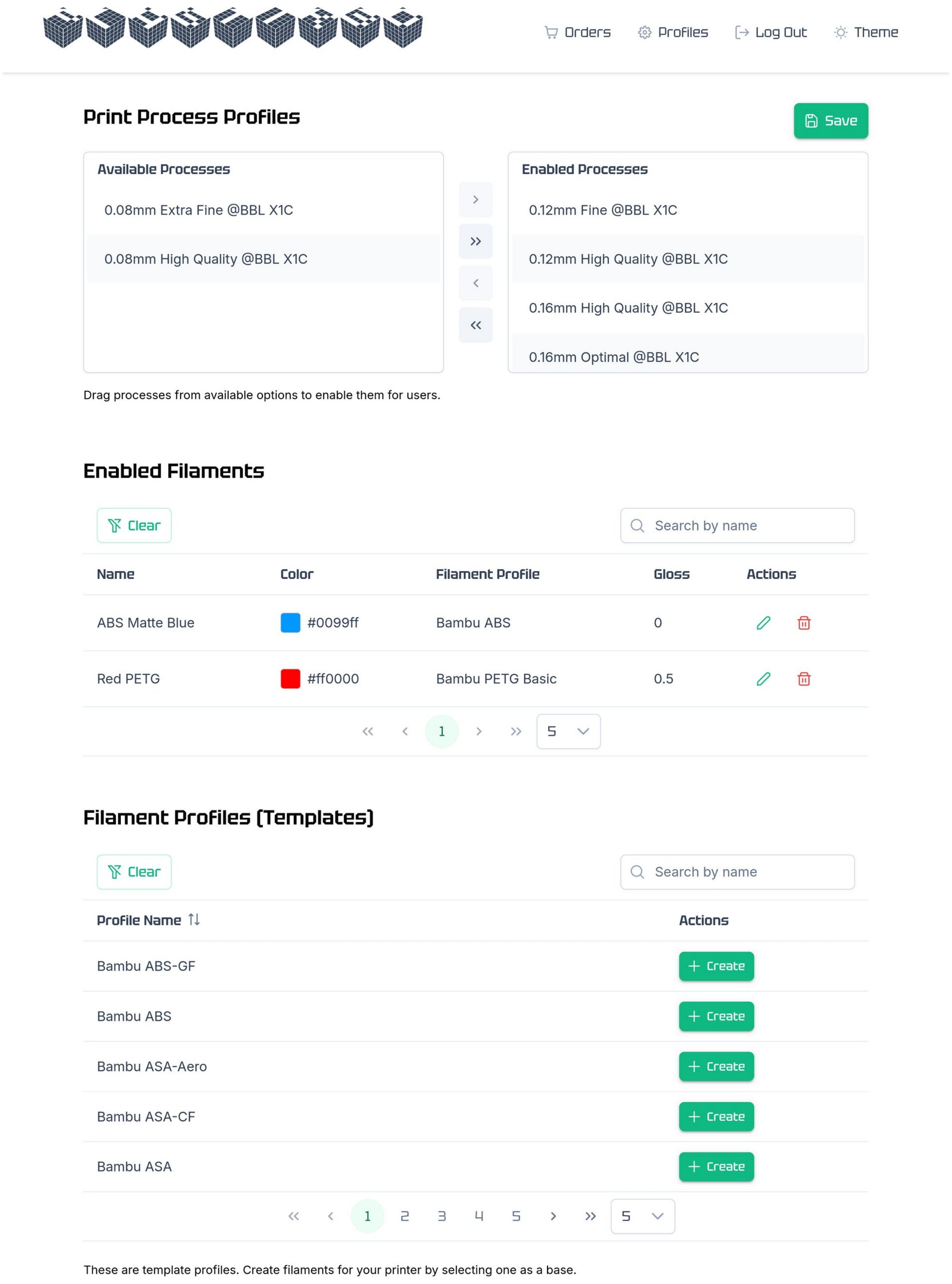
The Order Details modal provides information about a specific order. Within this modal, the administrator can download the sliced, ready-to-print files. Additionally, the administrator can update the order's status using the buttons. Each status update performed here automatically triggers a corresponding email notification to be sent to the customer, keeping them informed of their order's progress.

3.3.2. Managing Inventory

The page shown in Figure 24 provides administrators with tools to manage the profiles used by the agent for print processing and the available filament options.

The system lists all available process profiles, which govern the layer height parameter. Administrators can enable or disable these profiles as needed, controlling which options are available for print jobs.

Administrators can create new filament entries within the system. This process often involves basing the new filament on existing profiles, allowing for efficient setup. When creating or editing filaments, administrators can specify color and gloss settings. These visual attributes are used to enhance STL visualizations within the system, providing a preview of how the printed object might appear with the selected filament.



Print Process Profiles

Available Processes	Enabled Processes
0.08mm Extra Fine @BBL X1C	0.12mm Fine @BBL X1C
0.08mm High Quality @BBL X1C	0.12mm High Quality @BBL X1C
	0.16mm High Quality @BBL X1C
	0.16mm Optimal @BBL X1C

Drag processes from available options to enable them for users.

Enabled Filaments

Name	Color	Filament Profile	Gloss	Actions
ABS Matte Blue	🟦 #0099ff	Bambu ABS	0	
Red PETG	🟥 #ff0000	Bambu PETG Basic	0.5	

Filament Profiles (Templates)

Profile Name	Actions
Bambu ABS-GF	+ Create
Bambu ABS	+ Create
Bambu ASA-Aero	+ Create
Bambu ASA-CF	+ Create
Bambu ASA	+ Create

These are template profiles. Create filaments for your printer by selecting one as a base.

Figure 24. *Profiles page (created by the author)*

4. System Testing

The system underwent manual testing to evaluate core functionalities and identify potential issues across diverse scenarios. Testing was conducted within the Docker Compose environment described in Section 3.1.3, ensuring consistent and reproducible backend service evaluation. Frontend testing employed Firefox and Chromium browsers to assess the user interface and interaction flows.

4.1. Testing Methodology

The testing methodology comprised exploratory testing, use case validation, and regression testing following code modifications. Static analysis tools complemented manual testing through TypeScript type checking and integrated development environment extensions, including Vue and Go plugins for Visual Studio Code. This approach facilitated early detection of potential type errors and syntax issues during development, though it cannot replace runtime testing for logical or integration defects.

4.2. Test Cases and Results

Test cases were derived from the use cases defined in Section 2.2, covering critical system workflows such as STL file upload, print configuration, order placement, and administrative operations. The testing process verified the system's ability to handle standard operations while identifying areas requiring refinement. Most core functionalities performed as intended, with certain edge cases requiring additional error handling to improve user feedback quality. Performance constraints were observed during slicing operations for geometrically complex models. The test cases and the results are presented in Table 6.

Table 6. *Details of test cases (created by the author)*

ID	**Description**	**Steps**	**Expected Result**	**Actual Result**	**Status**
TC01	Basic STL Upload	**1.** Navigate to Print Studio. **2.** Upload a valid STL file.	STL visualization appears, configuration options become available.	As expected.	Pass
TC02	Print Configuration Changes	**1.** Upload an STL file. **2.** Change filament type. **3.** Adjust layer	System debounces changes, triggers slicing, updates visualization and price estimate.	As expected, slight delay observed with large files.	Pass

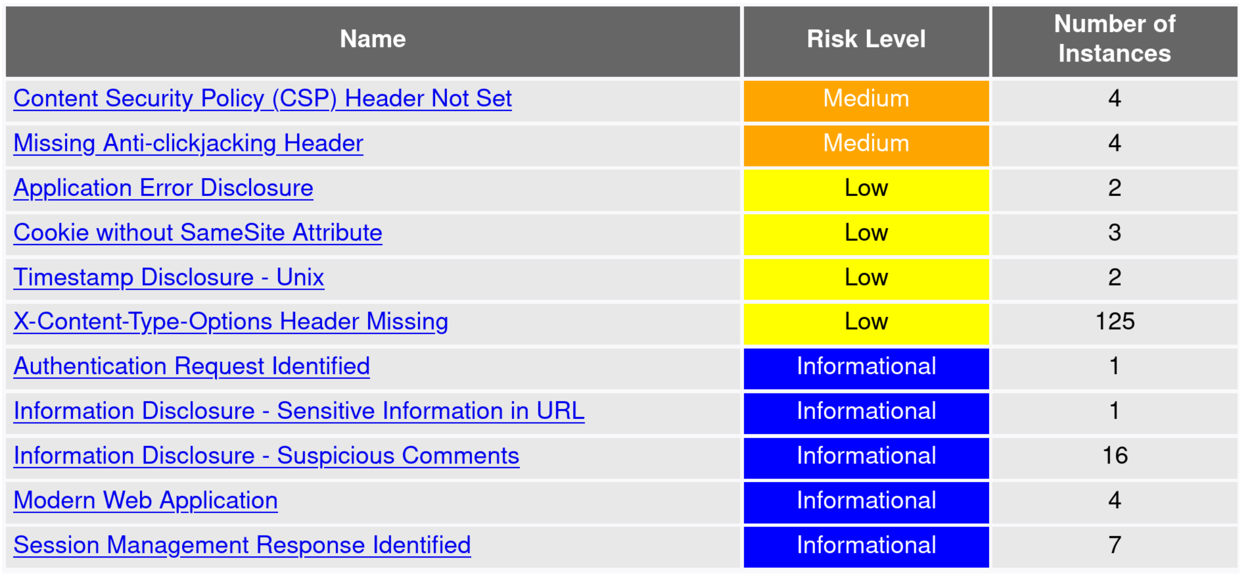
		height. **4.** Toggle supports. **5.** View G-code.			
TC03	Invalid File Upload	**1.** Attempt to upload a corrupted STL file.	System displays a meaningful error message.	Basic error shown.	Partial
TC04	Order Placement Flow	**1.** Configure print. **2.** Click order. **3.** Enter delivery info. **4.** Proceed to payment. **5.** Complete Stripe test payment.	Order created, confirmation email sent, redirection to tracking page.	As expected.	Pass
TC05	Order Status Update	**1.** Log into the administrator panel. **2.** View order list. **3.** Change order status to “Processing”.	Status updates in database, notification email sent to customer.	As expected.	Pass
TC06	Inventory Change	**1.** Upload an STL file. **2.** View available filaments and layer heights. **3.** Log into the administrator panel.	Changes to the filaments and layer heights in the administrator panel are reflected in Print Studio.	As expected.	Pass

		4. Change available filaments and layer heights. **5.** View available filaments and layer heights in Print Studio.			

4.3. Security Testing

Security assessment of the system incorporated both manual penetration testing techniques and automated scanning. Zed Attack Proxy (ZAP) [26] was employed to identify potential vulnerabilities across the system's attack surface. Manual security review focused on authentication mechanisms, session management, input validation, and cross-site scripting prevention.

This hybrid approach identified minor security considerations that were subsequently addressed in the implementation. Security testing confirmed adequate protection mechanisms for the administrative interface, proper implementation of Stripe's secure payment processing, and resilient API endpoints. Figure 25 details the low-severity findings from the ZAP scan, which primarily concerned recommendations for cookie attributes and HTTP security headers.



Name	Risk Level	Number of Instances
Content Security Policy (CSP) Header Not Set	Medium	4
Missing Anti-clickjacking Header	Medium	4
Application Error Disclosure	Low	2
Cookie without SameSite Attribute	Low	3
Timestamp Disclosure - Unix	Low	2
X-Content-Type-Options Header Missing	Low	125
Authentication Request Identified	Informational	1
Information Disclosure - Sensitive Information in URL	Informational	1
Information Disclosure - Suspicious Comments	Informational	16
Modern Web Application	Informational	4
Session Management Response Identified	Informational	7

Figure 25. *ZAP findings (generated by Zed Attack Proxy)*

Conclusions

This thesis addressed the challenges of opaque pricing models and complex workflows associated with 3D printing services by developing the Web-based System for Ordering and Managing 3D Prints. This system provides an integrated solution that automates critical processes, including file validation, parameter adjustment, slicing, and cost estimation, thereby simplifying and streamlining the entire 3D printing workflow.

The development followed four main tasks outlined earlier in this work. Task 1, the Analysis of 3D Printing Services and Technologies, was successfully completed through market research and technical evaluation, which established the core requirements driving the system's design. Task 2, System Development, encompassed the design of the system architecture and the full implementation of the system. Subsequently, Task 3, System Documentation, was fulfilled by creating comprehensive guides for installation, users, and administrators, alongside presenting key implementation results. Finally, Task 4, System Testing, was executed using the methodology and test cases described, validating that the system's functionality and usability meet the defined requirements.

Key achievements resulting from this work include automated workflow management and integrated inventory control, significantly improving the efficiency of 3D printing operations. Furthermore, the system offers an intuitive web interface with real-time 3D model visualization, enhancing the user experience and reducing errors. While initially conceived with Bambu Lab printers in focus, the system's flexible architecture supports a wide range of printer models and brands through its integration with Bambu Studio, and the configurable slicer path allows for adaptation to alternatives like Orca Slicer, ensuring future-proofing.

Future development could extend the system's capabilities in several directions, addressing some of the limitations encountered during development and introducing new features to further optimize the user experience. While the current system underwent manual testing, including security assessments and specific test cases, future work should incorporate a more comprehensive testing strategy. This includes acknowledging the limitations inherent in manual testing alone and establishing a robust automated testing framework to ensure greater reliability and facilitate regression testing. Technical enhancements might include expanded payment gateway options and broader file format support. Functional expansions could incorporate more advanced configuration options like infill pattern selection, a field for additional user comments, support for multiple objects and build plates, customer account management, and managing multiple printers with individual inventories to support print farm operations, enabling more complex and large-scale printing projects. A further enhancement to improve efficiency within a single user session would be to implement caching of sliced files based on previously submitted print configurations. Currently, modifying a setting overwrites the previously sliced files. By retaining these files, the system could avoid redundant slicing if a user reverts to a prior configuration, significantly reducing processing time and improving responsiveness.

In conclusion, this project delivers a comprehensive and impactful solution addressing significant challenges in 3D print ordering and management. Through automation, real-time feedback, and integrated management tools, the system enhances efficiency and accessibility. Its modular architecture and flexible design position it as a valuable contribution to the 3D printing field, adaptable to evolving technologies while offering immediate practical benefits for both service providers and end-users.

References

[1] “Printing of complex-shaped objects from thermoplastics,” Services for business | Vilnius Tech “LinkMenų fabrikas.” Accessed: Mar. 31, 2025. [Online]. Available: https://vilniustech.lt/linkmenu-fabrikas/business/services-for-business/3d-printing--fdm/360713

[2] “PATS SAU Makerspace,” PATS SAU Makerspace - Martynas Mažvydas National Library of Lithuania. Accessed: Apr. 01, 2025. [Online]. Available: https://www.lnb.lt/en/spaces/creative-and-leisure-spaces/pats-sau-workshop

[3] “Print center,” Print center | Art and Design Lab | Vilnius Academy of Arts. Accessed: Apr. 01, 2025. [Online]. Available: https://www.vda.lt/en/art-and-design-lab/print-center

[4] “Filament guide - Compatibility to printer, nozzle, AMS, build plate, glue and required parameters,” Bambu Lab Wiki. Accessed: Apr. 02, 2025. [Online]. Available: https://wiki.bambulab.com/en/general/filament-guide-material-table

[5] “3dpro,” 3dpro. Accessed: Dec. 19, 2024. [Online]. Available: http://3dpro.lt

[6] “3D PRINTING,” Reklamos gamyba ir 3D spausdinimas. Accessed: Dec. 19, 2024. [Online]. Available: https://duv.lt/en/3d-spausdinimas

[7] “Orders,” 3D Creative. Accessed: Dec. 19, 2024. [Online]. Available: https://3dcreative.lt/en/orders

[8] “Online 3D Printing Instant Quote,” JLC3DP. Accessed: Dec. 20, 2024. [Online]. Available: https://jlc3dp.com/3d-printing-quote

[9] “The Streamlined 3D Printing Service,” Craftcloud®. Accessed: Dec. 20, 2024. [Online]. Available: https://craftcloud3d.com

[10] “3D Printing Service Online,” Shapeways. Accessed: Dec. 20, 2024. [Online]. Available: https://www.shapeways.com

[11] “squancy/zaccord: A service for ordering STL files and lithophanes & a webshop for selling 3D printed products.,” GitHub. Accessed: Dec. 12, 2024. [Online]. Available: https://github.com/squancy/zaccord

[12] “Your 3D Printing Service,” i.materialise. Accessed: Dec. 20, 2024. [Online]. Available: https://i.materialise.com

[13] “Order Management - 3D Printing Quoting Software,” Order Management - 3D Printing Quoting Software. Accessed: Dec. 20, 2024. [Online]. Available: https://www.materialise.com/en/industrial/software/order-management

[14] “3D Nyomtatás,” Zaccord. Accessed: Dec. 20, 2024. [Online]. Available: https://www.zaccord.com

[15] “Bambu Lab Authorization Control System,” Consumer Action Taskforce. Accessed: Mar. 21, 2025. [Online]. Available: https://wiki.rossmanngroup.com/wiki/Bambu_Lab_Authorization_Control_System

[16] “Bambu Studio 3mf Compatibility,” Bambu Lab Wiki. Accessed: Mar. 12, 2024. [Online]. Available: https://wiki.bambulab.com/en/software/bambu-studio/3mf-compatibility

[17] “3MF Production Extension support · Issue #30174 · mrdoob/three.js,” GitHub. Accessed: Dec. 22, 2024. [Online]. Available: https://github.com/mrdoob/three.js/issues/30174

[18] “WebGL: 2D and 3D graphics for the web - Web APIs: MDN,” MDN Web Docs. Accessed: Dec. 19, 2024. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/WebGL_API#browser_compatibility

[19] “WebGPU API - Web APIs: MDN,” MDN Web Docs. Accessed: Dec. 19, 2024. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/WebGPU_API#browser_compatibility

[20] “Three.js – JavaScript 3D library,” Three.js – JavaScript 3D Library. Accessed: Dec. 19, 2024. [Online]. Available: https://threejs.org

[21] “Powerful, Beautiful, Simple, Open - Web-Based 3D At Its Best,” Babylon.js. Accessed: Dec. 19, 2024. [Online]. Available: https://www.babylonjs.com

[22] “3D model-viewer embed,” <model-viewer>. Accessed: Dec. 19, 2024. [Online]. Available: https://modelviewer.dev

[23] “TresJS,” TresJS. Accessed: Dec. 19, 2024. [Online]. Available: https://tresjs.org

[24] “TroisJS,” TroisJS. Accessed: Dec. 19, 2024. [Online]. Available: https://troisjs.github.io

[25] “OWASP Top Ten,” OWASP Top Ten | OWASP Foundation. Accessed: May 05, 2025. [Online]. Available: https://owasp.org/www-project-top-ten

[26] “The ZAP Homepage,” ZAP. Accessed: May 05, 2025. [Online]. Available: https://www.zaproxy.org