VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

Jesse Båtman

TULUS CLOUD MANUFACTURING

Technology
2022

ACKNOWLEDGEMENTS

I would like to express my utmost gratitude to my current employer, Jubic Oy, who gave me a chance to learn exponentially during four years of studies. The people's knowledge and various projects that Jubic offered were highly crucial in becoming an independent and proficient software developer. No other place could have provided the same experience.

I would also like to extend my appreciation to people who worked on Tulus Cloud Manufacturing. Thanks to Mr. Jesse Ikola, a project manager from Jubic, who managed this project professionally and provided invaluable feedback about technical decisions. I am also grateful for Mr. Mikko Ylihärsilä, a CAM developer/consultant during this project, who developed the NC Express service and mathematics for part processing. The project would not be without them.

I am also very thankful to the people who help write and validate this thesis. I am especially grateful for Mr. Aleksi Männistö, who tirelessly gave me feedback about the language of this thesis during various stages of writing.

Vaasa, 24.5.2022

Jesse Båtman

VAASAN AMMATTIKORKEAKOULU
Tietotekniikka

TIIVISTELMÄ

Tekijä	Jesse Båtman
Opinnäytetyön nimi	Tulus Cloud Manufacturing
Vuosi	2022
Kieli	englanti
Sivumäärä	56
Ohjaaja	Ghodrat Moghadampour

Työn tavoitteena oli kuvailla Tulus Cloud Manufacturing-sovelluksen toteutettua arkkitehtuuria ja integraatioita. Painopiste toteutetuista ominaisuuksista oli näiden tietoturvassa sekä testaamisessa. Lisäksi työssä käydään läpi mahdollista jatkokehitystä, integraatioiden laajuutta ja rakennetta.

Tulus Cloud Manufacturing on tarjousportaalisovellus Prima Powerin asiakkaille, eli valmistajille. Sovellus mahdollistaa ohutmetallilevyistä työstettyjen osien automaattisen käsittelyn. Cloud Manufacturing eroaa muista tarjousportaalista siten, että asiakkaat saavat itse ladata haluamansa suunnittelutiedostot, määritellä erityisprosessit ja seurata tarjouksen kulkua tilauksen läpi lähes reaaliajassa. Käsitellyistä osista sovellus osaa laskea arvioidun tarkan prosessointiajan, kulut ja hinnat asiakaskohtaisesti, joita valmistaja pystyy käyttämään hyväksi.

Työssä kuvaillaan neljän integraation toteutusta, sekä niiden tietoturvaa ja testaamista. NC Express-palvelun integraatio mahdollistaa ladattujen suunnittelutiedostojen käsittelyn ja sovelluksen ydintoiminnan. Authentication Service-palvelun integraatio vastaa valmistajien eli ylläpidon käyttäjien hallinnasta. Palveluun voidaan esimerkiksi liittää aktiivisia sovelluslisenssejä.

Integraatio toiminnanohjausjärjestelmään mahdollistaa tarkan ja oikean tiedon saamisen alustalle. Sähköpostipalvelun integraatio mahdollistaa muiden ominaisuuksien toiminnallisuuksien toimivuuden, kuten sisäisen käyttäjänhallinnan.

Työssä testattiin näiden integraatioiden toimivuutta yksikkö- ja integraatiotesteillä usealla tasolla. Testeillä voidaan automatisoida testaamista ja taata ominaisuuksien toimivuus jatkokehityksissä.

Avainsanat React, TypeScipt, SaaS, ASP.NET Core, API

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Tietotekniikka

ABSTRACT

Author	Jesse Båtman
Title	Tulus Cloud Manufacturing
Year	2022
Language	English
Pages	56
Name of Supervisor	Ghodrat Moghadampour

The purpose of this thesis was to describe the general architecture and developed integrations of Tulus Cloud Manufacturing. The primary focus was on information security and testing. The thesis also briefly describes the scale and structure of possible further integration development.

Tulus Cloud Manufacturing is a quotation platform for manufactured sheet metal parts. The software is designed for manufacturers who are Prima Powers customers. The software allows users to upload computer-aided design files that contain instructions for manufacturing parts to be uploaded to the platform. Cloud Manufacturing can calculate accurate estimates on processing times, costs, and prices that the manufacturer can utilize in the quotation process.

The thesis describes the structure of four integrations and how they are secured and tested. The core integration in Cloud Manufacturing is NC Express Service. The service parses the design files and enables the core business logic of the software to work. Integration to Authentication Service allows Prima Power to manage admin access to specific Cloud Manufacturing instances. This access can be controlled manually or with license keys.

Integration to Enterprise Resource Planning (ERP) enables Cloud Manufacturing to blend into the manufacturers’ process. ERP integration provides accurate information, such as material costs. Mailing service integration enables other application features, such as internal user management, to work.

Developed integrations were tested on multiple levels using the unit and integration testing. Written tests ensure the feature functionality in further development.

Keywords React, TypeScript, SaaS, ASP.NET Core, API

CONTENTS

LIST OF FIGURES, TABLES, AND CODE SNIPPETS

LIST OF ABBREVIATIONS

CM	Cloud Manufacturing
CAM	Computer-Aided Manufacturing
CAD	Computer-Aided Design
NC	Numerical Code
ASP.NET	Active Server Pages.NET
IoT	Internet of Things
MVC	Model-View-Controller
API	Application Programming Interface
JSON	JavaScript Object Notation
CLR	Common Language Runtime
EF Core	Entity Framework Core
ORM	Object-Relational Mapper
CRUD	Create, Read, Update, Delete
VDOM	Virtual Document Object Model
JSX	JavaScript XML
HTML	Hypertext Markup Language
MS SQL	Microsoft SQL Server
OIDC	OpenID Connect
OAuth	Open Authorization
JWT	JSON Web Token
SMTP	Simple Mail Transfer Protocol
CI/CD	Continuous Integration / Continuous Deployment
AWS	Amazon Web Services

1 INTRODUCTION

Competition in sheet metal manufacturing is fierce. Manufacturers are fighting for ways to make the manufacturing process cheaper and faster than the competition. As computer-aided manufacturing became more efficient and generally more available, the improvements had to come from somewhere else. Traditionally these improvements were based on factors, such as cheaper materials or shipping costs. As a leading sheet metal machinery provider, Prima Power wants to enhance the order process for the manufacturers by simplifying and hastening the quotation process.

As a result, Prima Power started developing Tulus Cloud Manufacturing. Cloud Manufacturing is an online quotation and ordering platform for manufacturers and customers. It aims to enhance the speed of turnover time between quotes and orders. Cloud Manufacturing offers customers multiple requested features, such as live status updates on orders, instant quotes, and part modifications. Cloud Manufacturing gives manufacturers an entry point to the quotation process and multiple tools to fasten the process.

Manufacturers often communicate with customers, especially smaller ones, by email and phone. The customer would send a packet of design files to the manufacturers. The manufacturer must manually calculate the processing times, costs, and prices. Cloud Manufacturing reduces the extra communication between the parties. The manufacturer’s customer can upload the design files directly into Cloud Manufacturing, which would process them into parts. These parts can be used in the new quotation with instantaneous estimated pricing. The customer would be able to test different materials and define any other processes that could not be detected from the file.

Cloud Manufacturing is part of the Tulus software family. Tulus contains tools to manage and enhance the experience of Prima Power machinery. Logically Cloud

Manufacturing is mainly targeted at manufacturers who own Prima Powers machinery.

Integrations between different services are required to create flexible, fast, and accurate software. Cloud Manufacturing uses multiple integrations to various power features ranging from core business to messaging. This thesis primarily describes the developed integrations, focusing on information security and testing. The integrations enable the manufacturers to integrate the Cloud Manufacturing software into their existing processes more efficiently.

Cloud Manufacturing software is distributed using a Software as a Service (SaaS) model. The SaaS model ensures that Prima Power manages the installation and upkeep of the instance. This makes Cloud Manufacturing even more accessible for manufacturers with Prima Power machinery.

Prima Power bought the development for this project as subcontracting work from Jubic Oy.

1.1 Jubic Oy

Jubic is a modern software company located in Vaasa, Finland. Jubic was founded in 2015 and currently consists of around 18 employees. Jubic specializes in the development of industrial software solutions. Alongside development, Jubic offers cloud and consultation services to its customers. Consultation may consist of information security, project management, or software acquisitions. Many of Jubic's customers are from the industrial sector, like Prima Power. (Jubic, 2022)

1.2 Prima Power

Prima Power specializes in the manufacturing of different types of sheet metal machinery. Worldwide, Prima Power is one of the leading operators in this field. Prima Power's machinery can operate a wide range of tasks such as bending, punching, shearing, and lasering. These operations can run sequentially in complex machinery combinations to maximize production throughput.

On February 4, 2008, Prima Industrie Group acquired Finnish Finn-Power Group. Before the acquisition, Finn-Power Group produced primarily punch manufacturing machines, whereas Prima Industrie Group owned Prima Electro produced laser machines. Together with Prima Industrie Group brands, creating Prima Power.

Today Prima Power has facilities worldwide, including Finland, Italy, China, and the United States, and serves customers in over 70 countries. Prima Power has more than 400 employees in their Finnish unit located in Seinäjoki. (Prima Power, 2022)

1.3 Tulus

Tulus is a software family around Prima Powers machinery. Tulus software is tied closely with sheet metal machinery provided by Prima Power. The software family contains tools and software to expedite and improve different tasks regarding the machinery. Tulus software is designed to work with each other seamlessly. (Prima Power, Tulus, 2022)

Tulus Office is a software package for planning and managing machinery capacity. Production planning, tracking machine states, and continuous manufacturing are its primary uses. Office has three tiers: Basic, Classic, and Premium. These tiers give users more advanced features such as two-way ERP integration and dynamic nests. (Prima Power, Tulus Office, 2022)

Tulus Cell is software for managing machinery directly from the machine. Machine operators mainly communicate with sheet metal machinery using Cell. The software displays valuable information, such as task lists and helps with operations for example, managing tools, sorting, and stacking parts. (Prima Power, Tulus Cell, 2022)

Tulus Analytics is a web-based analytics application. Analytics provides an easy way to track and analyze production and machinery performance. The application aims to enhance product quality and reduce downtime. (Prima Power, Tulus Analytics, 2022)

Tulus Visual Monitoring is a web application for the machine operator. The application gives real-time information about the state of the machine. The application is used for identifying error reasons and location. (Prima Power, Tulus Visual Monitoring, 2022)

1.4 NC Express

NC Express (NCX) is a CAM/CAD-like programming software for sheet metal drafting and tooling. NC Express is highly integrated with Prima Powers machines and other tools like Tulus Office. The integrations create a robust environment for efficient and well-optimized NC-code programming. NC-code is a collection of instructions for machinery to manufacture a given part. (Prima Power, NC Express e3, 2022)

2 RELEVANT TECHNOLOGIES

This section contains detailed information about technologies that power Cloud Manufacturing.

2.1 ASP.NET Core

The application back-end is built using a free and open-sourced web framework called ASP.NET Core which Microsoft originally developed. ASP.NET Core was initially released on June 7, 2016. ASP.NET Core is a modern successor of ASP.NET and aims to be a lightweight, modular, and developer-friendly programming framework.

ASP.NET Core applications are cross-platform and support compatibility with most used operating systems, for example Windows, Linux, and Android. ASP.NET Core provides tools and libraries to create internet-based web and IoT applications implementing MVC architecture. ASP.NET Core applications are built on top of Common Language Runtime, allowing developers to use any programming language that supports CLR. Most commonly, ASP.NET applications are written with C#, but languages like F# and Visual Basic are also supported. (Microsoft, 2022)

MVC is shorthand for Model, View, Controller. MVC architecture is one of the most common approaches to building a web application. Model stands for abstraction over persistence layer, such as a database. This typically comes in the form of an object-relational mapper or query builder. The controller is used to route specific resources into URIs. These resources can be anything from raw JSON data to HTML pages. View implements a way to add information to HTML directly. (Mozilla, 2022)

ASP.NET Core implements common design patterns such as dependency injection. Dependency injection is an abstraction over accessing resources and services. Dependency injected services referenced during compile time are guaranteed to be accessed correctly during runtime. If implemented correctly, dependency injection can save valuable performance. (Microsoft, 2022)

2.2 RESTful API

Application programming interfaces (API) allow communication between two components or modules. APIs are the mediator between clients and services. Application programming interfaces handle more specific details such as caching and create standardization for communication security. Typical API implementations are RESTful and SOAP. (MuleSoft, n.d.)

RESTful is an API architectural style that allows interaction between web services. REST defines constraints that developers can implement. REST APIs implement endpoints that clients can communicate with. REST endpoints can be separated into multiple uniform resource identifiers (URI) and have various types. The most common request types are POST, GET, PUT and DELETE. REST supports numerous communication formats, but JSON and XML are the most popular.

RESTful HTTP requests contain headers that have metadata. This metadata can include authorization information, caching options, and cookies. Additional information can be passed in query parameters or URI. This information is in text format. (Redhat, 2020)

2.3 JSON

JavaScript Object Notation (JSON) is a lightweight and language-agnostic file format. JSON’s popularity comes from its readability and performance. JSON is commonly used as a RESTful APIs information format. JSON consists of a collection of keys and values, where keys are strings. Values support six types: string, number, Boolean, list, object, and null. (Mozilla, 2022)

2.4 Entity Framework Core

Entity Framework Core (EF Core) is an open-source Object-Relational Mapper (ORM). EF Core is the successor for Entity Framework, originally designed to work with .NET Framework. The first version of Entity Framework Core was released on June 27, 2016, as part of Microsoft's goals to modernize the .NET stack.

ORM is an abstraction over databases and allows the developer to convert programming language code to SQL that the database understands. EF Core implements the most needed database operations like creating, reading, updating, and deleting (CRUD) like other Object-Relational Mappers.

Entity Framework uses entities to create and track database tables. The entity class is an active description of the table in a database. EF Core provides tooling to create seamless migrations based on changes to entities. An instance of an entity is a database record. The Entity Framework Core abstract creates, updates, and deletes operations using tracked entities. Changes to tracked entities are reflected to the database as soon as the database context is saved.

Database context is an API provided within EF Core. This allows live access to all the registered entities and database operations. Together with ASP.NET Core, database context can be accessed using dependency injection. Dependency injected context handles database connection and disposition automatically. (Microsoft, 2021)

2.5 React

React is a free, open-source user interface library for JavaScript. React was released on May 29, 2013. Meta, formally known as Facebook, maintains React, but it is very community-driven with an active developer community. React was initially developed for creating client-side rendered single-page websites and web applications. Present-day React works as the backbone for multiple UI frameworks and libraries. These include mobile development such as React Native and server-side rendered web applications such as Next.js and Remix. (React, n.d.)

React builds user interfaces using virtual DOM. VDOM is a virtual structure of the application stored in memory. The state controls VDOM. React uses the state to store information programmatically. Internally, React reconciles past and current states to determine which parts of virtual DOM should be rendered. Only components that have access to the changed state will re-render. (React, n.d.)

Virtual DOM is separated into logical components. Components can contain state and return JSX. JSX is a syntax extension for JavaScript that allows HTML-like syntax. JSX code is constructed from native HTML elements for example, div, header, paragraph, and other React components. Components can pass state and functions top-down to children using props.

Props are defined by passing information to components attributes, such as HTML element attributes. Attributes in JSX cannot contain any hyphens. Element attributes with hyphens are converted using camelCase naming. For example, a "*class-name*" attribute is converted to "*className*". (React, n.d.)

2.6 TypeScript

TypeScript is a strongly typed, open-source programming language maintained by Microsoft. TypeScript language was first published on October 1, 2012, to add static types for JavaScript. TypeScript is a superset of JavaScript, and therefore any JavaScript code will work in TypeScript. TypeScript compiles into JavaScript. TypeScript improves code quality by enforcing types for variables, functions, and classes. (Typescriptlang, n.d.)

2.7 Microsoft SQL Server

Microsoft SQL Server (MS SQL) is a relational database management system. Microsoft initially released MS SQL on April 24, 1989, but it is still actively developed.

The latest version was published in November 2019. MS SQL is used to store and retrieve persistent data by other applications. MS SQL supports popular server operating systems like Linux and Windows Server. (SQLServerTutorial, n.d.)

A relational database is based on a relational model, which represents and connects data between tables. Relational databases have been the most used databases in modern history because they are resilient and scalable. Relational databases hold data in rows that are in tables. Rows contain columns that contain different types of data. Other database types are non-relational databases, such as document-oriented databases, key-value- and graph stores. (Oracle, n.d.)

Column types are database-specific, but SQL databases commonly contain basic types for example, integer, string, double, and date. Each database record consists of a unique identifier, typically an integer type. This identifier column can be referenced in other tables records, creating the relationship between two records.

The SQL language is used for communicating with MS SQL. The SQL language is a standardized relational database language developed by IBM. Most modern relational databases support and use SQL as the communication language. The most common SQL commands are SELECT, INSERT, UPDATE, and DELETE, which implement the basic CRUD operations. (Loshin, 2022)

2.8 OpenID Connect

OpenID Connect (OIDC) is an authentication protocol on top of the OAuth 2.0 framework. OIDC adds an identity layer on top of the OAuth protocol. OIDC allows client applications to verify and obtain basic information about users without direct access to identity providers. OIDC eliminates the need for separate identity management and enables the creation of a centralized identity provider. The identity provider and acceptor communicate via JSON using OAuth REST API. Identity is given to the acceptor using JSON Web Tokens. OIDC is widely adopted by the world’s biggest technology companies, for example Google, Microsoft, and Facebook. (Auth0, n.d.)

2.9 Docker

Docker is a software framework for building, managing, and running OS-level virtualized packages called containers. Docker was initially developed for Linux as an open-source virtualization solution but was later continued by Docker Inc. The first version of Docker was released on March 20, 2013. Docker’s popularity has risen in the last century for its ease of use when creating scalable and robust operating environments. Applications built and packaged using docker are guaranteed to run on any platform that supports Docker. (Docker, n.d.)

A Docker image is a file for executing code inside the container. It contains a set of instructions to build the application. The instructions define all the necessary dependencies and libraries for creating and executing the application. A Docker image is always immutable, and they can often be compared to a virtual machine snapshot. Docker image instructions are defined using the Dockerfile file. An image can be built from the file using the “docker build” command.

Docker images can be stored in centralized repositories. Repositories can be private or public, for example Docker Hub. Repositories allow easy distribution of the application from the built environment to production. (Gillis, 2021)

Containers are used to create an instance of an image. Containers create a stable and reliable runtime environment. Containerized application abstracts the underlying infrastructure to guarantee that applications run the same on different machines. Containers allow an easy way to modularize the program. Modular programs are split into multiple containers using container management software, such as docker-compose or Kubernetes. This is an underlying ideology of microservice architecture. (Rubens, 2017)

2.10 Kubernetes

Kubernetes is open-sourced system for managing containerized applications. Kubernetes was released on June 7, 2014, by Google. Kubernetes offers tools and architecture for managing, scaling, and deploying containerized application clusters. Clusters consist of the master machines and possible worker machines. The master machine coordinates the communication between nodes.

A node is a machine inside the cluster. The master machine inside the cluster controls nodes. The responsibility of a node is to define rules for pods running inside the nodes. These rules include networking settings, device and pod capacities, and pod information. Pod information consists of application and image versions and active statuses.

A pod is the smallest unit of a cluster. It represents a single instance of application deployment. A pod consists of one or multiple containers. Each pod has an IP address assigned, and they can communicate with each other using a local network. External communication outside the pod requires an exposed port. (Kubernetes, n.d.)

3 APPLICATION

This chapter explains general application information, requirements, and infrastructure.

3.1 Application Description

Tulus Cloud Manufacturing is an ordering platform for sheet metal manufactured parts. Cloud Manufacturing is intended to be a part of the Tulus software family and provide the manufacturer with an interface for managing customers' manufacturing orders. Cloud Manufacturing provides tools for the fast turn-over time between a quotation and order and gives precise estimates about processing times, costs, and pricing.

Cloud Manufacturing gives the customers the ability to upload custom design files. Cloud Manufacturing provides accurate pricing estimates for customers after the design file has been uploaded and processed. Integration with NC Express allows the initial estimations to be precise to genuine part processing calculations. Miscalculated estimates can be corrected easily with tools provided by Cloud Manufacturing.

Prima Power distributes Cloud Manufacturing independently from other Tulus software, using a SaaS model. Software as a Service model means that Prima Power maintains the Cloud Manufacturing instances and all the required services. Service is intended for manufacturers that use Prima Powers' sheet metal machinery.

3.2 Project Motivation

Traditionally, sheet metal manufacturing sales flow goes through the manufacturers' sales department via email and phone. After contact, the sales teams contact the CAM programmer, who calculates processing times for each given design file.

The sales will then contact the customer with the options for the part manufacturing. Customers might add or modify existing options, such as new material, which will require recalculation of processing times. This process is lengthy and costly and requires attention from multiple parties.

Cloud Manufacturing changes the traditional flow. Customers can upload design files directly into Cloud Manufacturing instead of contacting the sales. Customers can then freely adjust the materials and manufacturing thicknesses and possibly get an instantaneous price estimation. This process saves time and money for both parties. What traditionally took days takes minutes in Cloud Manufacturing.

3.3 Objectives and Features

A key objective of Cloud Manufacturing is to simplify quotation and order flow. Simplification reduces operation fees and ultimately adds revenue. To achieve the goal, the application implements the following features:

- Tools for uploading and managing design files.
- Accurate quotation estimation instantaneously after adding parts.
- Tools for managing quotations and orders for both parties.
- Tools for managing customer accounts and instance admin access via Authentication Service.
- Tools for managing context parameters, for example machines, materials, tools, and pricing configurations.

3.4 Application Infrastructure

Cloud Manufacturing uses external integrations to implement the key features. This infrastructure allows easy and scalable deployment plans for multiple instances of Cloud Manufacturing. The infrastructure enables customer-specific configurations. These configurations can easily be managed in a centralized deployment plan.

Cloud Manufacturing is designed to be platform-independent if needed. Regular deployment of Cloud Manufacturing contains a self-hosted database and volume, managed by Kubernetes. Cloud Manufacturing can be deployed independently from Kubernetes and configured to use hosted databases and data storage. An example of hosted data storage can be Microsoft's Azure Files or Amazon's S3 storage.

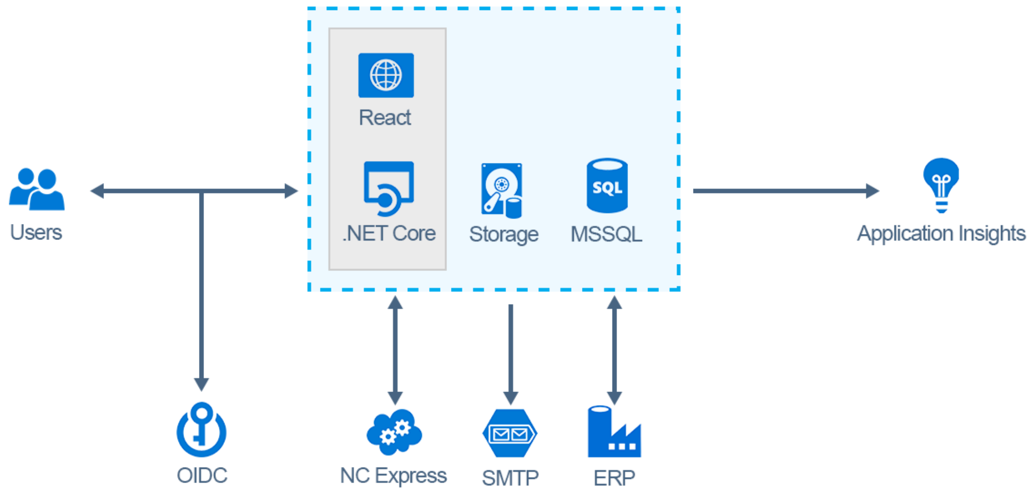


Figure 1. Application infrastructure schema.

Application infrastructure is described in Figure 1. The Grey zone in this figure illustrates the Cloud Manufacturing container pod. The container is based on an image built using CI/CD pipelines. The pipelines are tied to the Cloud Manufacturing version control. The Cloud Manufacturing user interface is built using React. The back-end is built using ASP.NET Core and serves the assets generated in the front end.

The light blue zone containing a grey zone describes one instance of Cloud Manufacturing. One instance of Cloud Manufacturing is the Kubernetes node which includes a Cloud Manufacturing container, storage, and MS SQL. The Kubernetes node can be characterized as one instance of Cloud Manufacturing. Storage is a volume mount for the node. This volume mount stores uploaded assets, such as images, resource files, and design files. The Cloud Manufacturing pod has read and write access to the storage.

The ingress controller of the pod exposes HTTP and HTTPS traffic to Cloud Manufacturing container. Only resources inside the pod can access the database. Cloud Manufacturing integrations and external services communicate via HTTP(s) protocol through the Cloud Manufacturing container.

Users in the infrastructure schema represent application end-users. OIDC is connected with users and Cloud Manufacturing. OIDC symbol in infrastructure references to Prima Powers Authentication Service. Authentication Service is a centralized identity provider that implements the OpenID Connect protocol. This service is used to authenticate manufacturers' admin users. Authentication Service is connected to license services. This service directly manages access to Cloud Manufacturing.

NC Express service is an API client with direct access to licensed NC Express e3 instances. NC Express is responsible for parsing manufacturing parts out of uploaded design files. The NC Express client enables the core functionality of Cloud Manufacturing to work. The NC Express client communicates with worker REST API using the HTTP protocol. NC Express Clients are instance independent, meaning one client can handle multiple instances simultaneously.

The SMTP symbol represents email integration. Cloud Manufacturing uses external mailing services to send account-related confirmation operations through email. These emails are constructed using HTML email templates. Email integration will be expanded to cover notifications and newsletters in the future.

The ERP symbol represents the ERP integration. Enterprise resource planning software manages essential parts of the company, for example manufacturing, supply chain, services, human resources, and procurements. ERP is critical for gathering accurate information such as material availability and costs. Cloud Manufacturing currently implements a CSV data import feature for materials. The ERP integration will be expanded in the future to be automatic and possibly two-way using REST APIs.

Application Insights is a monitoring tool in Azure. It provides tools to monitor live application performances, detect anomalies, track users, and diagnose issues. Application Insights has broad support for platforms for example .NET, Java, and Node.js. Cloud Manufacturing uses Application Insight to analyze general performance and log errors. One instance of Application Insights is used to gather all Cloud Manufacturing instances statistics. Cloud Manufacturing does not include instance-specific data, such as user information in Application Insight logging.

3.5 Backup Strategies

The software provider is responsible for data recovery when the software is distributed using the SaaS model. During the development, two data recovery strategies were designed. These strategies are based on how the instance has been set up.

The first backup strategy is to use managed services. Instead of using self-hosted services in the persistence layer, Cloud Manufacturing uses services provided by cloud platforms, such as Azure or Amazon Web Services. Cloud Manufacturing container would be deployed independently without persistence layer, and Cloud Manufacturing would use services Azure SQL Database as database and Azure Files for volume. These services offer data recovery as a service.

When deploying with Kubernetes, the primary backup strategy would be to develop a separate backup service. The backup plan is described in Figure 2.

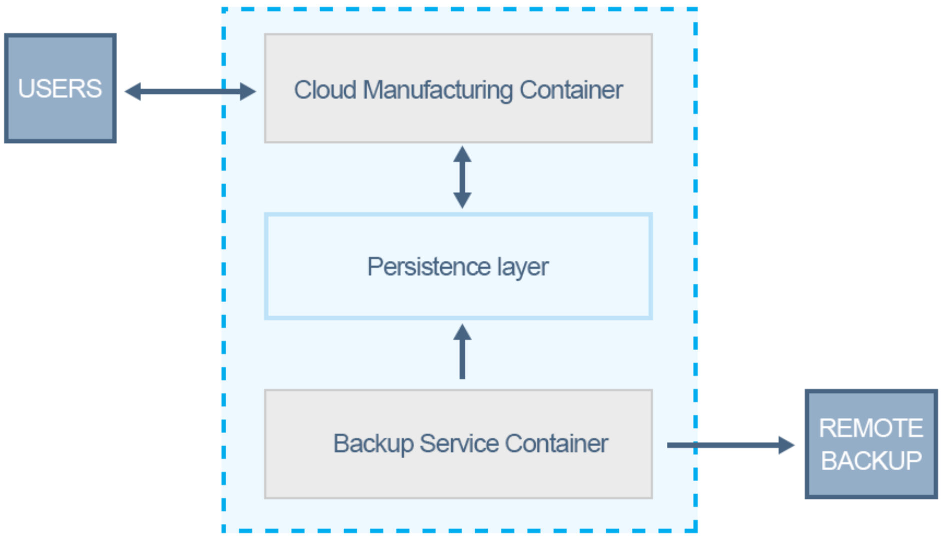


Figure 2. Self-managed backup strategy.

The backup service would be located inside the node's instance and would have access to the persistence layer. The service would be configured on pod level to run periodically. The service would pack the data from volume and database into a zip archive. This zip archive would be uploaded to managed remote data storage like an AWS S3 bucket. Data lifetime would be configured on the remote data storage level.

3.6 Connection to Tulus

Cloud Manufacturing is part of the Tulus software family. The development of Tulus integration is not within the scope of this thesis. Figure 3 describes the possible extent of integration.

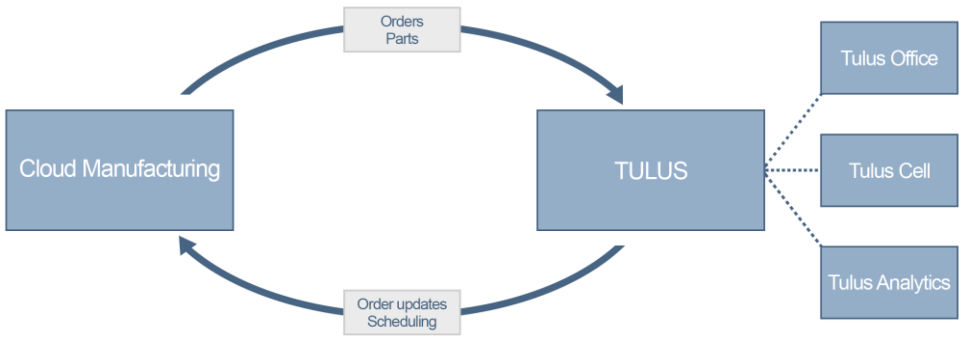


Figure 3. Possible Tulus integration.

The rest of the Tulus software could use Cloud Manufacturing data to enrich the experience. Tulus Office and Tulus Cell could use order data to schedule machinery based on needs. Tulus Office could update quotation and order statuses and delivery estimates. Tulus Office could automatically download the design file from Cloud Manufacturing and provide it directly to NC Express. The CAM programmer would not need to find the file separately since it is automatically linked.

4 INTEGRATIONS AND FEATURES

This chapter introduces requirements and descriptions about different internal and external services.

4.1 NC Express Service

NC Express service is part of the core functionality of Cloud Manufacturing. Nc Express service allows Cloud Manufacturing to process design files into parts. Parsed parts contain essential information about the part, such as the list of features, part image preview, and geometry. This functionality enables other internal services, for example costing and pricing, to work.

NC Express services are instance independent. One NC Express Instance can process multiple Cloud Manufacturing instances simultaneously. One Cloud Manufacturing instance is not limited to one NC Express Instance. Having multiple NC Express instances for one Cloud Manufacturing instance can significantly decrease processing times in heavy load times.

4.1.1 Uploading Design File

The first step in using the NC Express service is uploading and storing design files. Cloud Manufacturing users can upload design files into separate folders or under one general folder. The uploaded design files are stored in the part library. Each user has access to their own company's part library, whereas admin users have access to the admin-only part library and each company's library.

Table 1 describes the requirements of the design file upload process. Essential requirements are user accessibility, security, process swiftness, and user interface clarity. Requirement specifications define priority ranging from 1 to 3, from most to least important. All the requirements are listed as priority one and are "must-have" features. It is crucial that uploaded files can be directly used in new quotations after the upload is done and the NC Express service has processed the file.

Table 1. Design file upload requirement specification.

Number	Description	Priority
1	Users can upload one or many design files simultaneously	1
2	Users can drag and drop files to be uploaded	1
3	Users get instant feedback about file upload status	1
4	A newly added part can be used in the quotation after upload	1
5	Users can delete part(s)	1
6	Users can see parts within the company's scope	1
7	Admin users can see all parts and customers' parts	1
8	Simple to use	1

The design file upload process is shown in Figure 4. The file upload endpoint expects a file to be transferred as form-data instead of JSON request. After the Cloud Manufacturing back-end receives design files, they are immediately validated. Unfamiliar file formats are instantly rejected. NC Express Service can parse neutral CAD file formats such as STEP, STL, and DXF and native formats such as AutoCAD, Inventor, and Solid Edge. (Prima Power, NC Express e3, 2022)

A new processing job is created once the resource receives a valid file, and the file is successfully stored on the volume. The job indicates to worker API that the new file is ready to be processed.

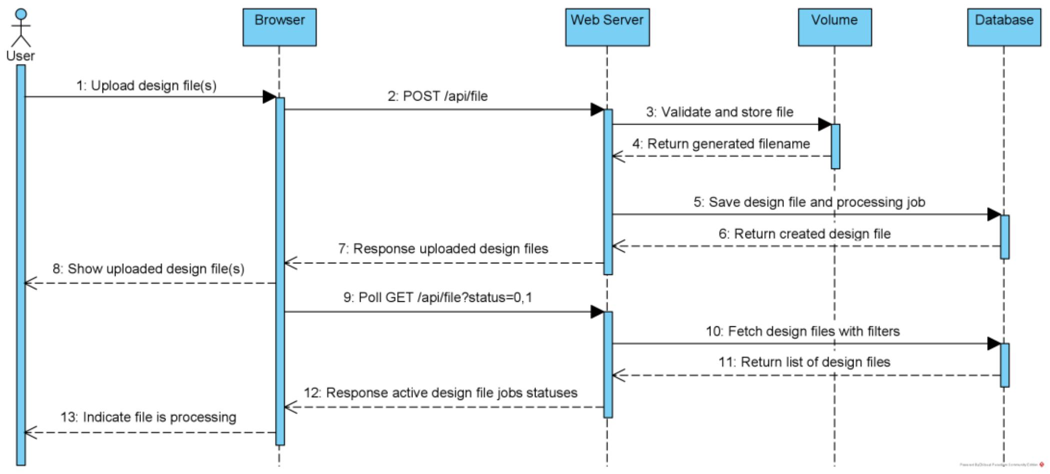

User
Browser
Web Server
Volume
Database
1: Upload design file(s)
2: POST /api/file
3: Validate and store file
4: Return generated filename
5: Save design file and processing job
6: Return created design file
7: Response uploaded design files
8: Show uploaded design file(s)
9: Poll GET /api/file?status=0,1
10: Fetch design files with filters
11: Return list of design files
12: Response active design file jobs statuses
13: Indicate file is processing

Figure 4. Upload design file sequence diagram.

While the NC Express service processes a new design file, the browser polls the design file API to get accurate information about the processing job status. Polling rate is around 3 seconds to match the average processing time.

4.1.2 Parsing Design Files into Parts

After design file validation, a new processing job is created. This job is used to track NC Express processing status. The Worker API is constructed to provide NC Express service access to design files and processing jobs.

Table 2. Worker API requirement specification.

Number	**Description**	**Priority**
1	NCX Service can communicate with CM using worker API	1
2	NCX requires an API key for authorization	1
3	API provides a method for listing available processing jobs	1
4	API provides a method for taking the next processing job	1
5	API provides a method for updating the processing job	1
6	API provides a method for downloading design file	1
7	NCX can finish or reject the active processing job	1
8	Worker API consumes JSON	1

Table 2 lists requirements around the worker API, which provides NC Express service with methods for reading and updating processing jobs and downloading design files based on processing job id. All the listed requirements are marked as priority one meaning it is a “must-have” feature. Figure 5 shows a detailed processing flow between NC Express service and Cloud Manufacturing.

The worker API authorizes incoming API requests using an API key. The worker API key is a randomly generated string. The key is stored in application configuration and is unique to a specific Cloud Manufacturing instance. Authorized users with a regular authorization token are not authorized to use the worker API.

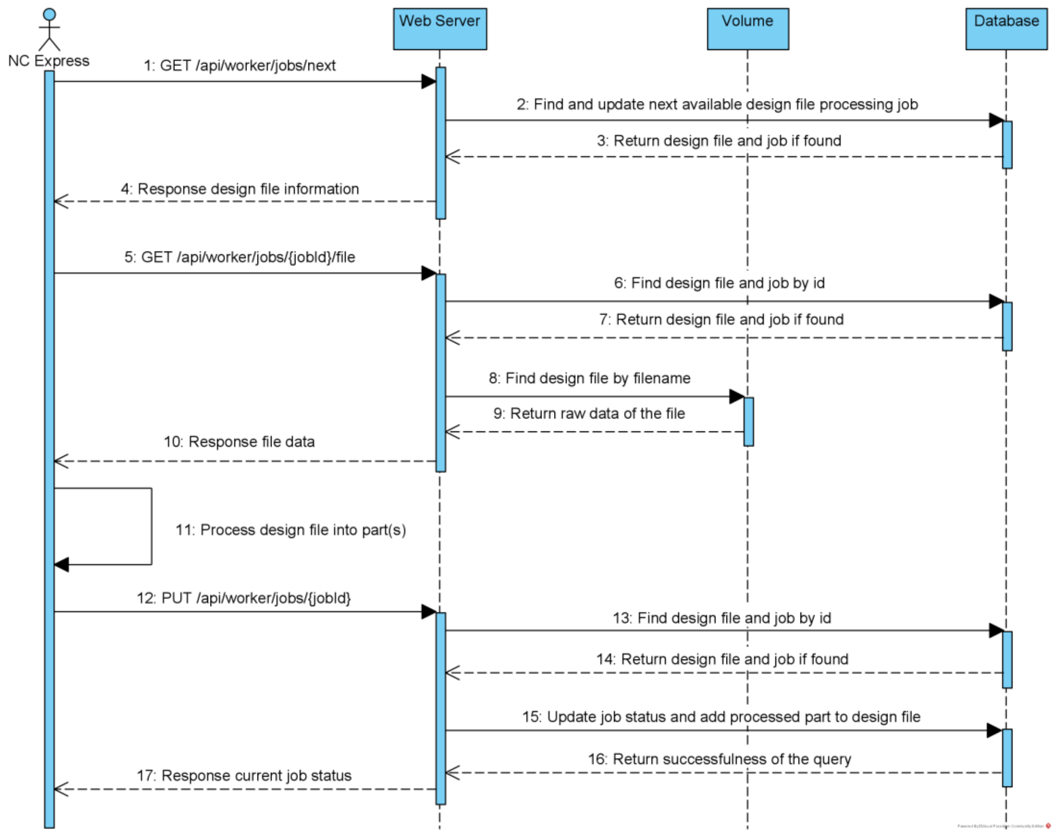

NC Express
Web Server
Volume
Database
1: GET /api/worker/jobs/next
2: Find and update next available design file processing job
3: Return design file and job if found
4: Response design file information
5: GET /api/worker/jobs/{jobId}/file
6: Find design file and job by id
7: Return design file and job if found
8: Find design file by filename
9: Return raw data of the file
10: Response file data
11: Process design file into part(s)
12: PUT /api/worker/jobs/{jobId}
13: Find design file and job by id
14: Return design file and job if found
15: Update job status and add processed part to design file
16: Return successfulness of the query
17: Response current job status

Figure 5. NC Express service flow sequence diagram.

After the processing job is finished, the worker API requires NC Express to provide a list of new parts from the design file. The NC Express parts include a list of features in part, part sizes, feature patterns, and base64 encoded PNG image. A PNG image is used for previewing the general shape of the elements. Cloud Manufacturing constructs detailed and interactive vector-based graphics using feature shapes provided by NC Express.

The NC Express parts are then parsed and transformed into Cloud Manufacturing parts. The transformation formats information from NC Express, adds additional meta information to the part and optimizes data into multiple database tables and volumes.

Cloud Manufacturing timeouts automatically process jobs if the NC Express service encounters a critical error during processing and cannot finish or reject the job. This timeout period can be configured per instance and defaults to 5 minutes. Admin users can rerun or delete denied processing jobs. Deleting only the processing job causes design file deletion.

4.1.3 Part Method Authorization

User actions around processed parts are complex. Many of the actions are limited to the context of the part and HTTP requester. The method authorization is based on the user role, while the context specifies which information can be accessed or modified. An example of context limitation would be when a user tries to calculate an estimation of the part while the part has no defined manufacturing material. Role-based actions for parts can be seen in Figure 6.

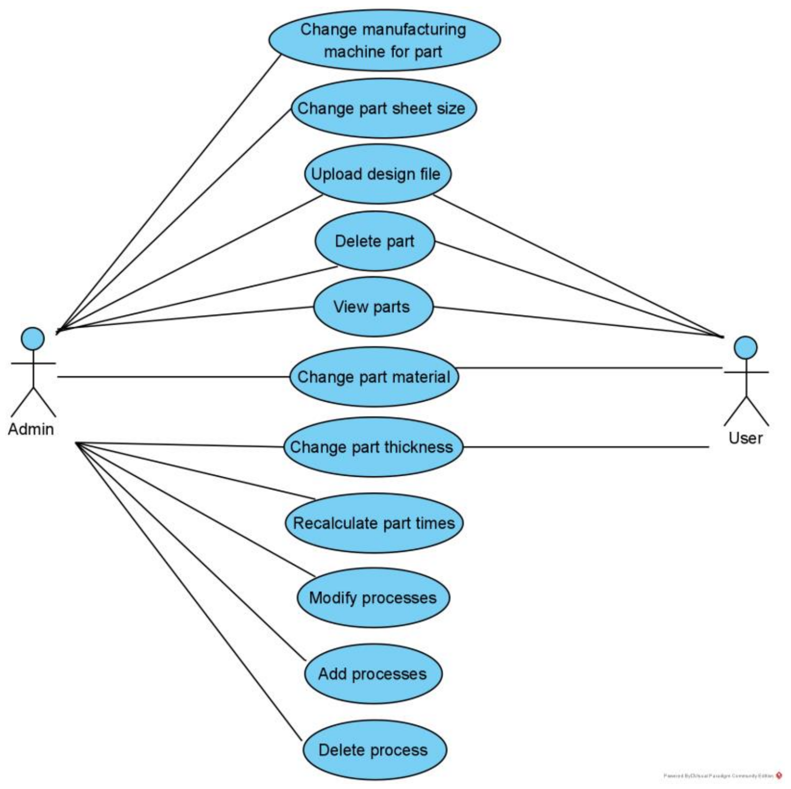


Figure 6. Part method authorization use case diagram.

Multiple basic activities are accessible by both admin and end-users. Admin users can modify processing times, override default manufacturing machines and material sheet sizes, and recalculate part processing times that end users can not. End-user actions are limited to parts owned by the company.

4.1.4 Part Processing Flow

Figure 7 describes part flow after it is received from NC Express service. Only initialized parts can be added to a new quotation. The initialized part has manufacturing material and thickness defined. Initialization can be done dynamically while creating a new quote.

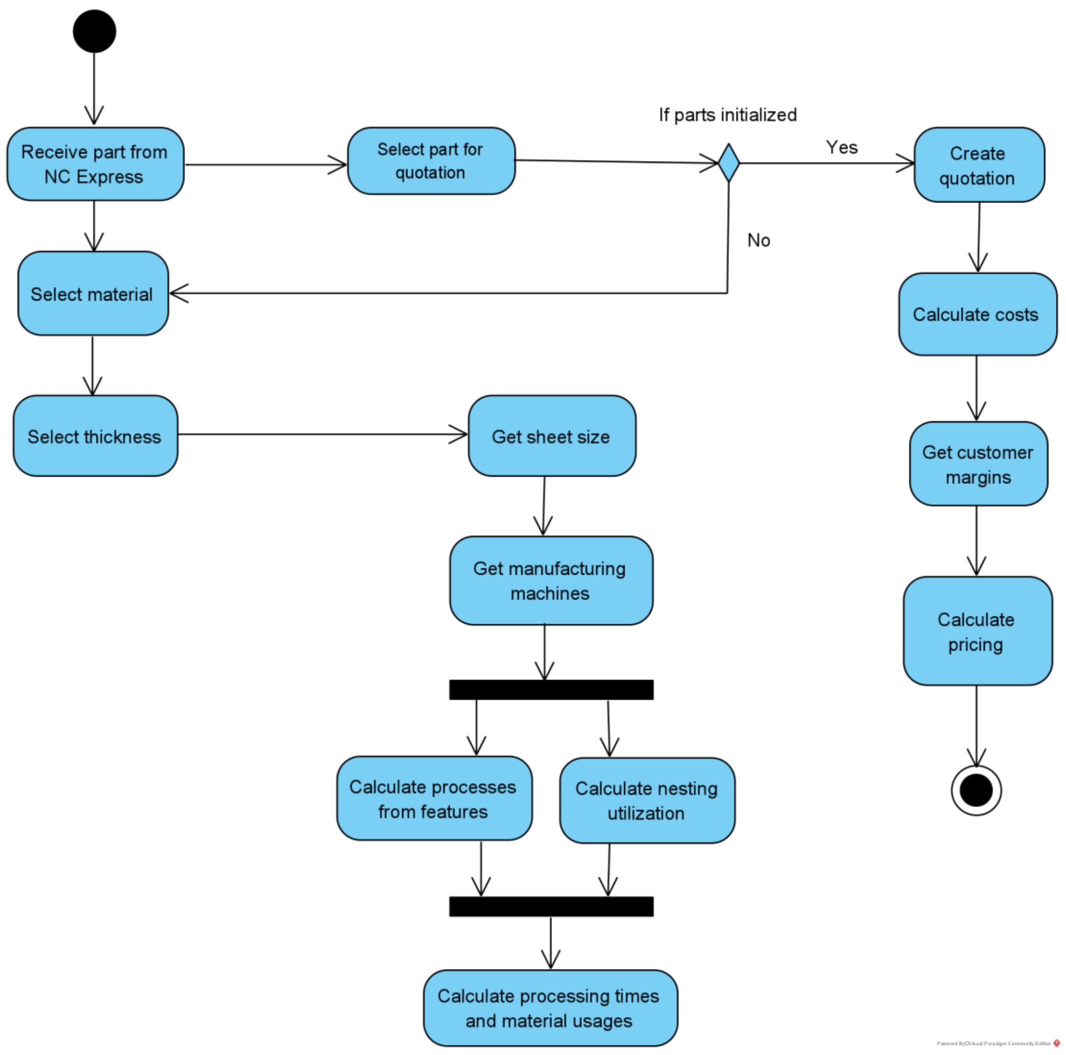

Receive part from NC Express
Select part for quotation
If parts initialized
Yes
No
Create quotation
Select material
Calculate costs
Select thickness
Get sheet size
Get customer margins
Get manufacturing machines
Calculate pricing
Calculate processes from features
Calculate nesting utilization
Calculate processing times and material usages

Figure 7. Part processing flow diagram.

Initialization allows Cloud Manufacturing to calculate correct processing times and add the right processes to parts. Initialized parts include all the necessary information to calculate quotation costs and prices. Initialized parts can be re-used in further quotations, allowing the software to estimate more accurately in the future.

4.2 Application Security

Application security is one of the most critical aspects of software development, mainly when the application handles business-critical information or customer information. Cloud Manufacturing is designed to be a secure web application. Application is secure when only the right amount of data is given out based on user authorization, resources require authentication, and there are no apparent design flaws.

Cloud Manufacturing uses JSON Web Tokens (JWT) as a primary identity provider. JWT is a compact and self-contained token that Cloud Manufacturing signs. Signed JWT is used to authenticate and authorize application APIs. Applications that use JWT as primary authentication solution can add custom information in the shape of claims. Usually, these claims consist of users' unique identifiers, names, and other relevant information. (Auth0, n.d.)

The application that signs JWT can quickly validate signed tokens and ensure that the given data is not modified. This certainty disposes of the need to query the database every time a token is authorized. (Auth0, n.d.)

Cloud Manufacturing has two user roles, normal and admin. Admin users are the manufacturer's employees who have bought a license for the software. Regular users are customers of the company that bought Cloud Manufacturing. Admin users are limited from Prima Powers Authentication Service. Therefore, Prima Power can manage how many users access admin privileges. The managing company can decide how many end-users they want to allow on the platform. Admin users have full access to manage end users.

4.2.1 Security Requirements

Table 3 lists the general security requirements for Cloud Manufacturing. Needs are targeted around authentication and authorization. Security requirements for other APIs are not listed here since they are all different based on use-case.

Table 3. Application security requirement specification.

Number	Description	Priority
1	Internal API is protected	1
2	User authorization uses JSON Web Tokens	1
3	A token is provided in a request authorization header	1
4	End-users authenticate using username and password	1
5	Admin-users authenticate using an authentication service	1
6	Admin user is limited to instances of CM defined in the authentication service	1
7	Admin users can add, update, or delete end users	1
8	API segregates responses based on user role	1
9	API segregates responses based on user company	1
10	End users can reset their password	2
11	Admin users can create passwordless mail invitations for the end-user	2

Application security requirement specification lists priorities for each item. Priority goes from 1 to 3, from the most to the least important. Items 10 and 11 are marked as a secondary priority because the application can be easily used securely without them. All the features listed in Table 3 are implemented. The implementation of item 10 is shown in Figure 11.

4.2.2 User Authentication

Cloud Manufacturing has internal user management for end-users. Admin users manage End-users. End users are customers of the company that bought an instance of Cloud Manufacturing. End-users use the email and password form on the login page for authentication. Figure 8 describes the end-user login flow.

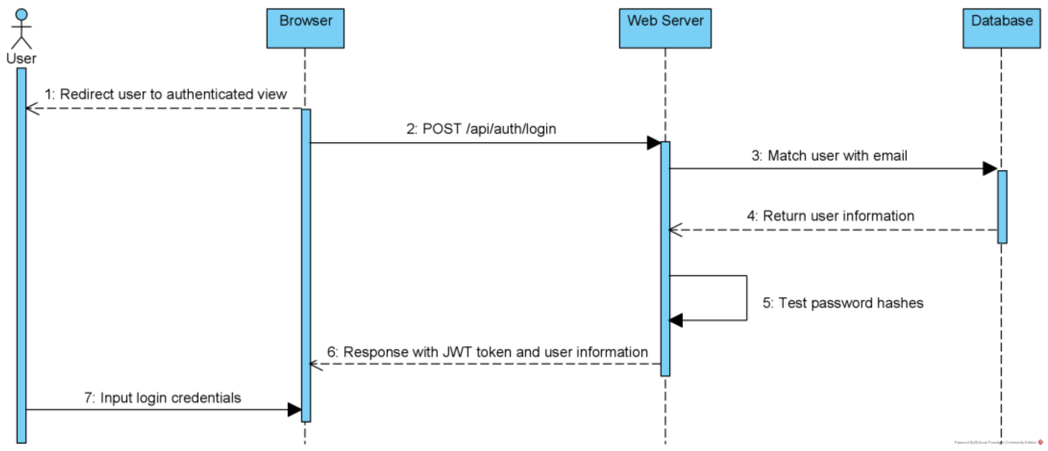

User
Browser
Web Server
Database
1: Redirect user to authenticated view
2: POST /api/auth/login
3: Match user with email
4: Return user information
5: Test password hashes
6: Response with JWT token and user information
7: Input login credentials

Figure 8. End-user login sequence diagram.

Cloud Manufacturing uses a pseudo-random encryption function, plain text password, and salt to create a secure encrypted password. Generated passwords and salts are stored in the database. This stored password is compared to the inputted password when the user tries to authenticate.

If the authentication is successful, the user gets JSON Web Token and user information. JWT is stored in the browser local storage and is added as an authentication header for every API request. If the authentication fails, API responds with an unauthorized HTTP error. Login API does not give detailed information about the failed request. This protects the API from user scanning vulnerabilities.

4.2.3 Admin Authentication

Admin users authenticate into Cloud Manufacturing using Prima Powers Authentication Service. The authentication Service is OpenID Connect Identity Service. Authentication Service allows Prima Power to allocate admin access to Cloud Manufacturing instances.

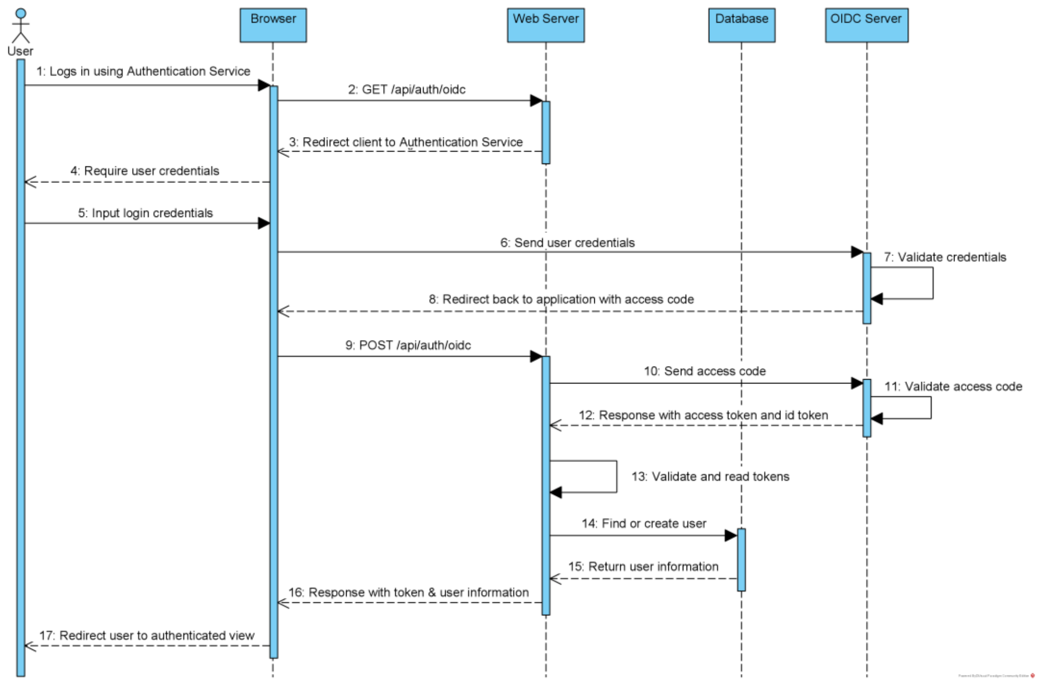

User
Browser
Web Server
Database
OIDC Server
1: Logs in using Authentication Service
2: GET /api/auth/oidc
3: Redirect client to Authentication Service
4: Require user credentials
5: Input login credentials
6: Send user credentials
7: Validate credentials
8: Redirect back to application with access code
9: POST /api/auth/oidc
10: Send access code
11: Validate access code
12: Response with access token and id token
13: Validate and read tokens
14: Find or create user
15: Return user information
16: Response with token & user information
17: Redirect user to authenticated view

Figure 9. Admin user login sequence diagram.

Figure 9 describes the authentication flow for admin users. Admins can access Authentication Service by accessing the Manufacturer tab on the login page and clicking the Authentication Service link. Redirection adds a state and nonce to browser cookies to protect users from cross-site request forgery attacks. These cookies will be validated further down the process. The user is redirected back to Cloud Manufacturing when the correct credentials have been input. Redirection adds access code to URI fragment.

The access code is then sent to the Cloud Manufacturing back-end, where the access token is validated using OpenID token endpoint. The Authentication Service responses with identity- and access tokens if the code is correct. The identity token contains all the necessary information about the users, such as email, first name, and last name. This information can be used to create or update the Cloud Manufacturing account and grant access.

4.2.4 Application Authorization

Cloud Manufacturing authorizes users to different resources based on multiple factors. User identity is obtained by decoding bearer tokens in the API request headers. Most API authorization follows the pattern seen in Figure 10, which shows the flow of an API request and response.

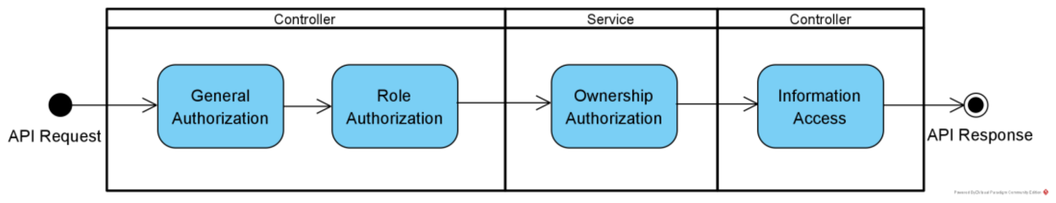


Figure 10. API data authorization diagram.

When the API controller receives the new request, the application first checks the general status of authorization. General Authorization indicates whether the user is logged in or not. This is the most basic authorization level and is on by default in all internal APIs. Role authorization checks whether the requestee's role matches the required role. Cloud Manufacturing commonly uses this to separate user access to specific API resources.

After the controller has authorized the request, it can reach the service. Ownership authorization is used to verify that the requestee has access to the requested data. Cloud Manufacturing uses ownership authorization to ensure that regular users only have access to the information they or their company owns. For admins, ownership authorization grants direct access to all the data.

Information access limits the API response data. This is based on role and ownership authorization. Data limitation is expanded with data transfer objects that ensure no confidential information is passed in response. An example of information access is a processed part. Regular users do not have access to processing times as admins do.

4.3 ERP Integration

Enterprise resource planning is a software system solution for business. ERP manages essential parts of the company, such as manufacturing, supply chain, services, human resources, and procurements. ERP often provides automation and intelligence to all day-to-day business operations. Most of the organization's data is stored in the ERP system. This provides a single source of truth across the business. (Oracle, n.d.)

4.3.1 Integration Requirements

ERP integration is an essential part of using Cloud Manufacturing in production. The integration allows Cloud Manufacturing to receive accurate, up-to-date information. Currently, Cloud Manufacturing has one-way integration in CSV file import. CSV was chosen as the form of data transfer because of its simplicity and general availability. Table 4 contains requirements for material import implementation.

Table 4. ERP integration requirement specification

Number	Description	Priority
1	Efficient and easy way to update materials	1
2	Admin user can upload CSV file	1
3	Admin can configure mapping from CSV fields	1
4	Admin can configure CSV fields unit and format	1
5	Admin can preview upcoming changes	1
6	The system adds new materials and updates existing materials	1
7	The system saves the used configuration for CSV format	1
8	System loads used configuration based on CSV format	1

The essential feature in implementing integration was simplicity and generalization of data. Data import should be easy to use to encourage users to use it while giving options to expand and customize the structure as much as possible. Data structure customization comes in the form of different units or currencies.

4.3.2 Material CSV Import

Material prices and availability are often very volatile. Therefore, it was required to have a way to add and update material availability and prices. Most ERP systems allow users to print a CSV file of the stock, allowing a way to update Cloud Manufacturing without having direct access to the ERP system.

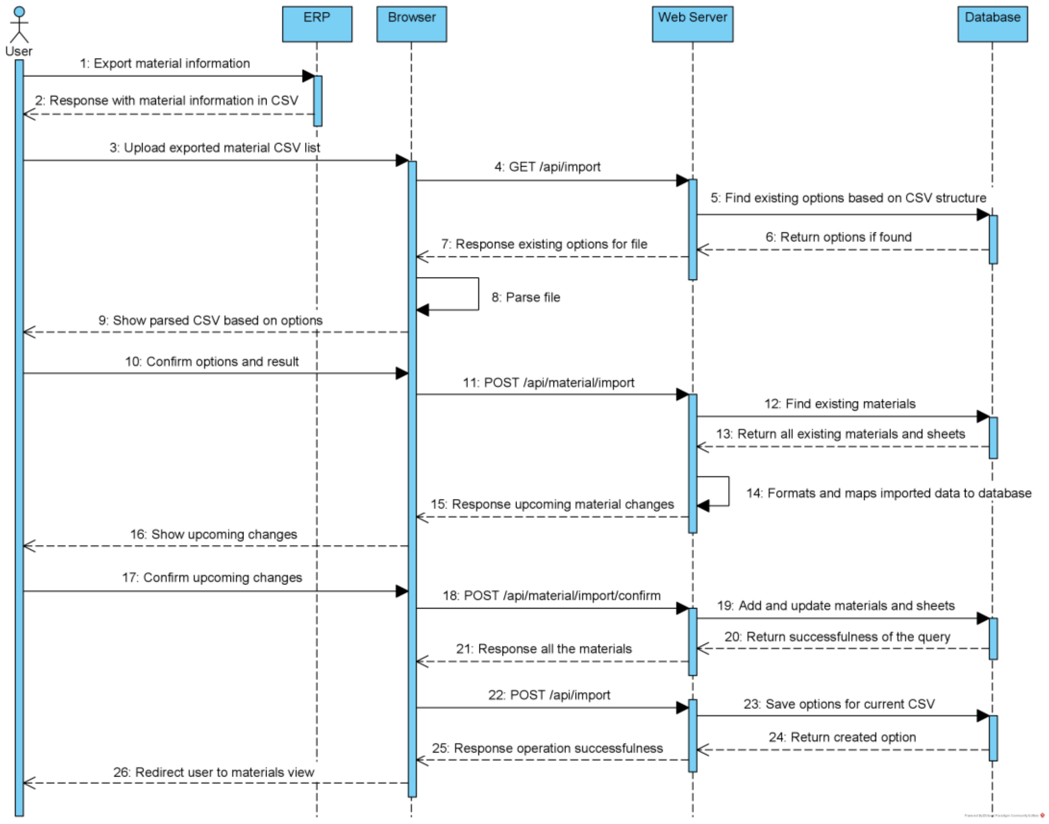
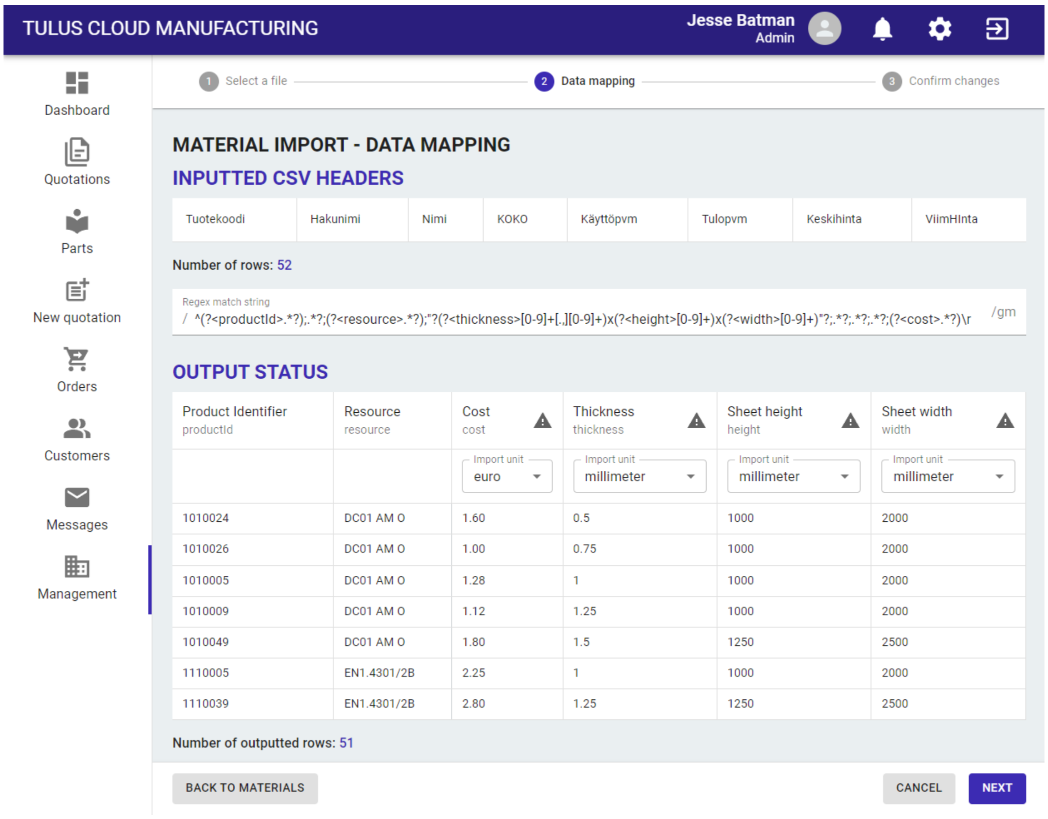

User
ERP
Browser
Web Server
Database
1: Export material information
2: Response with material information in CSV
3: Upload exported material CSV list
4: GET /api/import
5: Find existing options based on CSV structure
6: Return options if found
7: Response existing options for file
8: Parse file
9: Show parsed CSV based on options
10: Confirm options and result
11: POST /api/material/import
12: Find existing materials
13: Return all existing materials and sheets
14: Formats and maps imported data to database
15: Response upcoming material changes
16: Show upcoming changes
17: Confirm upcoming changes
18: POST /api/material/import/confirm
19: Add and update materials and sheets
20: Return successfulness of the query
21: Response all the materials
22: POST /api/import
23: Save options for current CSV
24: Return created option
25: Response operation successfulness
26: Redirect user to materials view

Figure 11. Material CSV import sequence diagram.

Figure 11 describes the ERP material CSV data import flow. Cloud Manufacturing analyses the CSV file after it has been uploaded. The header row is used to find older configurations from the API. If the configuration is found, it is used to parse the file client. The configuration consists of selector text that consists of regular expression and column-specific unit mappings. Units are pre-mapped to the wanted output format.

A regular expression (RegEx) is a pattern matching method for texts. Regex is often used to validate user input or find or replace text based on a pattern. Regular expressions are constructed between two slashes. (Mozilla, 2022)

When the Cloud Manufacturing instance is set up, import configurations are configured by the installation party. After initial setup, import requires no further user configuration. Material import configuration is shown in Figure 12.



MATERIAL IMPORT - DATA MAPPING

INPUTTED CSV HEADERS

Tuotekoodi	Hakunimi	Nimi	KOKO	Käyttöpvm	Tulopvm	Keskihinta	ViimHInta

Number of rows: 52

Regex match string

```
/ ^(?<productId>.*?);.*?;(?<resource>.*?);"?(?<thickness>[0-9]+[.,][0-9]+)x(?<height>[0-9]+)x(?<width>[0-9]+)"?;.*?;.*?;.*?;(?<cost>.*?)\r /gm
```

OUTPUT STATUS

Product Identifier productId	Resource resource	Cost cost	Thickness thickness	Sheet height height	Sheet width width
		Import unit: euro	Import unit: millimeter	Import unit: millimeter	Import unit: millimeter
1010024	DC01 AM O	1.60	0.5	1000	2000
1010026	DC01 AM O	1.00	0.75	1000	2000
1010005	DC01 AM O	1.28	1	1000	2000
1010009	DC01 AM O	1.12	1.25	1000	2000
1010049	DC01 AM O	1.80	1.5	1250	2500
1110005	EN1.4301/2B	2.25	1	1000	2000
1110039	EN1.4301/2B	2.80	1.25	1250	2500

Number of outputted rows: 51

Figure 12. Material CSV import options UI.

Admin can confirm data import after validating the output to be correctly formatted. After confirming CSV data import, it updates the software to use new materials. The new configuration would be saved for further usage if the configurations were changed. This approach gives the users an ability to use multiple CSV structures.

4.3.3 Plans for Future

Only having CSV data import as sole ERP integration limits the platform. There are plans to add support for automated and two-way ERP integrations when the platform gets more pilot customers. These integrations would significantly increase the platform value since no manual work would be needed after the integration is configured and connected.

The technical implementation of automated ERP integration would consist of integration API. The API would implement features to update targeted values, such as materials, machine parameters, or tools. ERP would implement the logic behind the integration, for example, how frequent the updates are and what information should be updated.

The primary use case for two-way ERP integration is ordered synchronization, where new orders would be added and updated to ERP. Technically, two-way integration would be implemented using an HTTP client that sends the wanted data to ERP API. The ERP would need to implement an API that accepts Cloud Manufacturing information in a standardized format.

4.4 Emailing Integration

Modern cloud services are highly connected to end-user devices and communication services. The connection is commonly used to verify user authentication, notifications, newsletters, and account recovery. This connection would engage users to be more active. Email integration is the minimum requirement for web applications.

Cloud Manufacturing implements mailing integration over SMTP protocol. The Simple Mail Transfer Protocol is TCP based communication protocol between mailing services. The protocol enables emails to be sent and received. Port 25 is commonly used in SMTP communication. (GeeksForGeeks, 2021)

Cloud Manufacturing implements an authentication method using email integration for end-users. These methods consist of invitation and reset password. Cloud Manufacturing could, in the future, implement features such as platform notifications and newsletters over email integration. Admin users can recover the account using Authentication Services recovery tools.

4.4.1 Password Recovery Example

Password recovery is the most used method when dealing with account restoration. A forgotten password can be restored by accessing the registered account's email. This pattern guarantees that the requestee is authorized to access the account. Figure 13 describes the integration between Cloud Manufacturing and SMTP mailing service with a password recovery example.

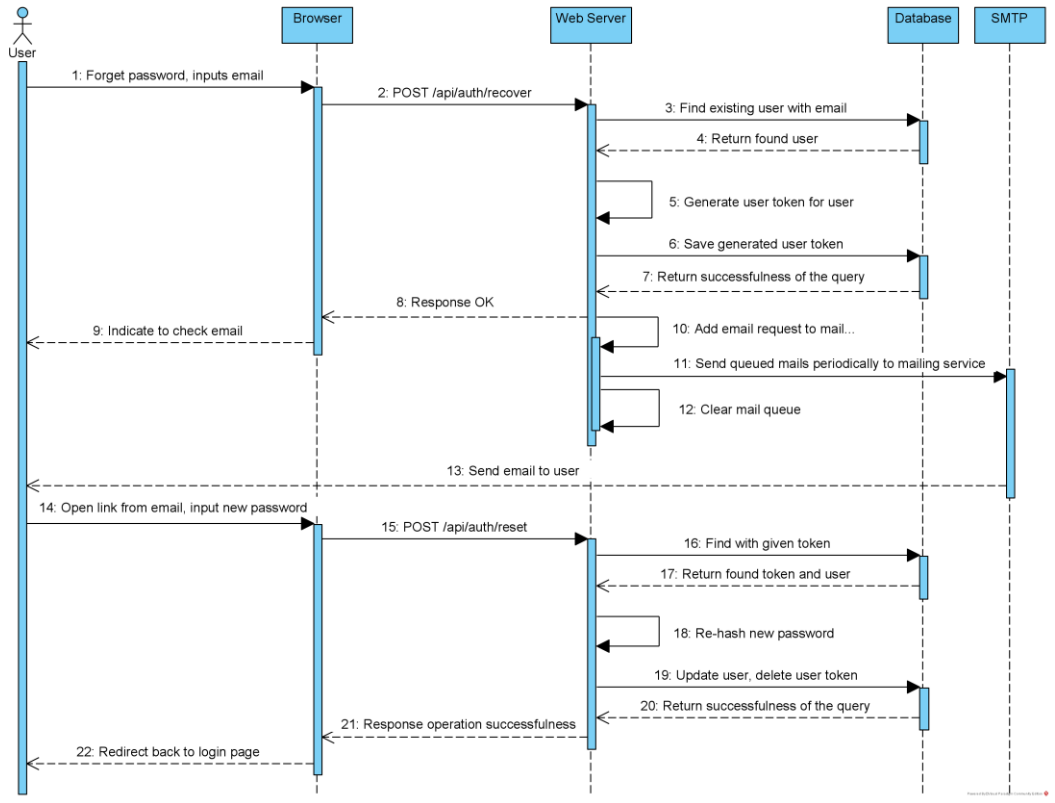
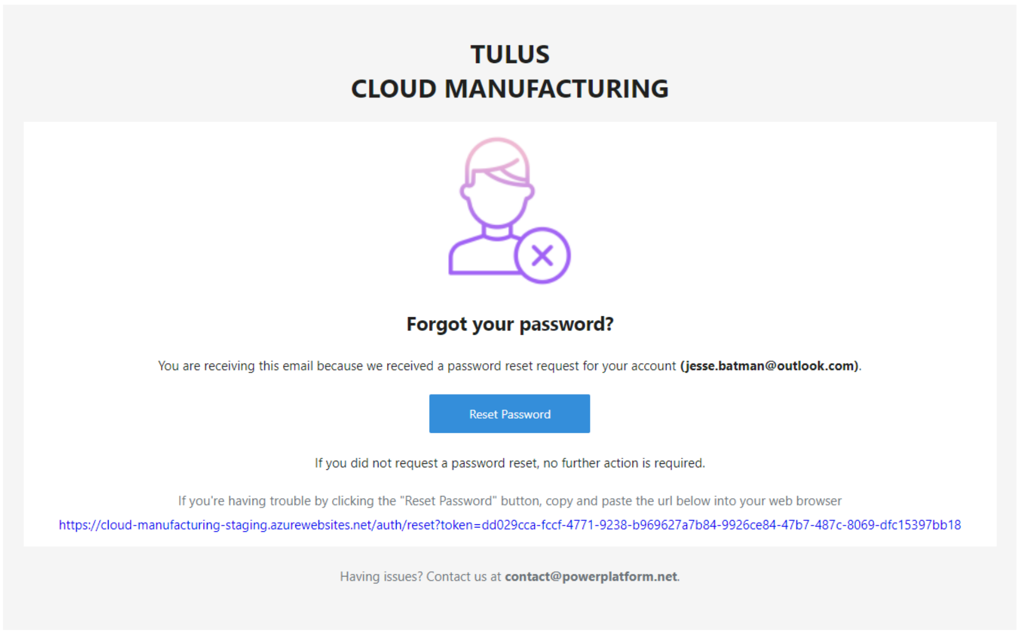

User
Browser
Web Server
Database
SMTP
1: Forget password, inputs email
2: POST /api/auth/recover
3: Find existing user with email
4: Return found user
5: Generate user token for user
6: Save generated user token
7: Return successfulness of the query
8: Response OK
9: Indicate to check email
10: Add email request to mail...
11: Send queued mails periodically to mailing service
12: Clear mail queue
13: Send email to user
14: Open link from email, input new password
15: POST /api/auth/reset
16: Find with given token
17: Return found token and user
18: Re-hash new password
19: Update user, delete user token
20: Return successfulness of the query
21: Response operation successfulness
22: Redirect back to login page

Figure 13. Password recovery sequence diagram.

Recovery and restoration API resources have no authorization requirements since users have no way to authenticate themselves. API allowing anonymous requests needs to be highly secured since bad actors often use them as attack vectors. Cloud Manufacturing enhances security by limiting the data given as a resource response and using request rate limiters. Even if a user with a given email is not found, the API returns HTTP ok code. This is to avoid user scanning using the API resource.

A randomly generated reset token is created when a valid and existing user email is input. This token is used to authenticate password reset requests further in sequence. Cloud Manufacturing uses email-supported HTML templates, which only support limited HTML tags and attributes. The templates are stored in the projects file system. HTML template contains variables that are programmatically populated. As seen in Figure 14, the email template for resetting password contains variables such as users' email and recovery URL.



TULUS
CLOUD MANUFACTURING

Forgot your password?

You are receiving this email because we received a password reset request for your account **(jesse.batman@outlook.com)**.

If you did not request a password reset, no further action is required.

If you're having trouble by clicking the "Reset Password" button, copy and paste the url below into your web browser
https://cloud-manufacturing-staging.azurewebsites.net/auth/reset?token=dd029cca-fccf-4771-9238-b969627a7b84-9926ce84-47b7-487c-8069-dfc15397bb18

Having issues? Contact us at **contact@powerplatform.net**.

Figure 14. Reset email template.

SMTP mails are constructed using the populated template as the body and email address as the recipient. The mail is added to the queue system, which manages the connection to SMTP service and sending.

The restoration email contains a link to a new password view where the user can input a new password. After the form submission, the newly added password is re-hashed and stored in the database, and the user token is deleted. The user can instantly log into Cloud Manufacturing using the new password.

Generated user tokens are invalidated after a day if they are not used. The clean-up happens as a recurring background task.

4.4.2 Email Queue Structure

SMTP mail requests are added to a queue to avoid unnecessary TCP connections to the mailing service. This dramatically increases the efficiency of the systems since the emails are sent in batches. The drawback of this implementation is that there will be a slight delay in sent mails. Cloud Manufacturing balances this by accessing and clearing the queue every 30 seconds. This was concluded to be an excellent middle ground between response times and performance. The email queue flow is described in Figure 15.

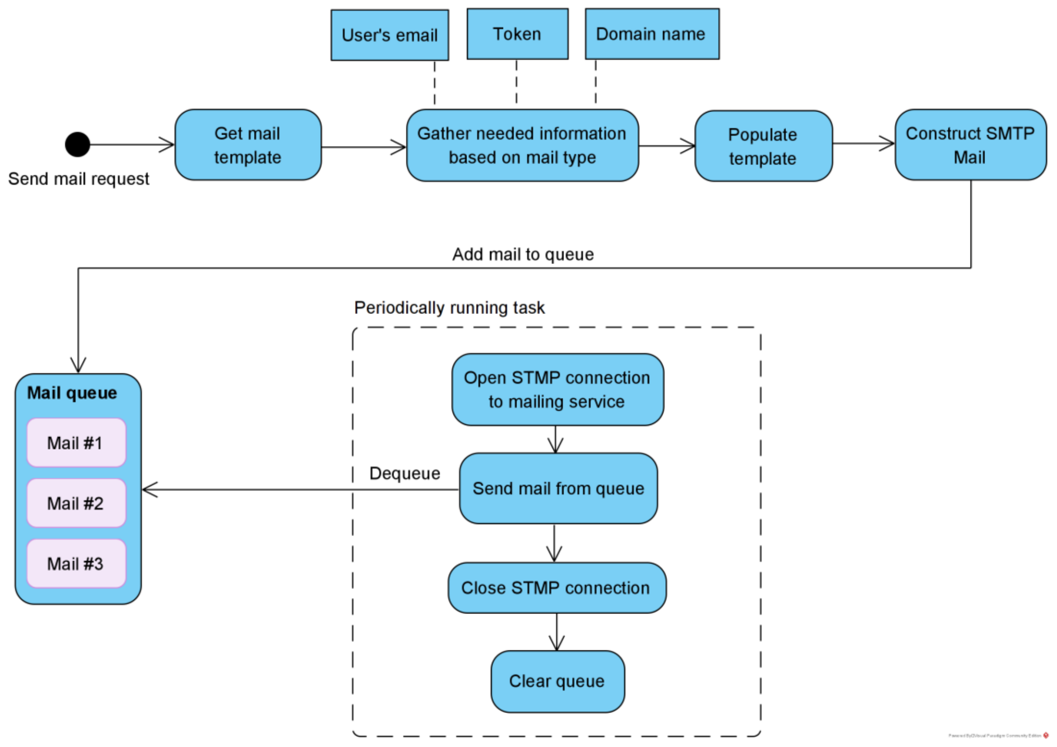


Figure 15. Email queue flow diagram.

5 TESTING

This chapter explains different testing levels and how they are implemented into Cloud Manufacturing.

5.1 White Box Testing

White box testing is a method for testing internal structures of the software. White box testing is available when the tester has access to internal components. Software developers and cyber security professionals do white box testing to find a bug or exploit from a software. White box testing is often pre-emptive, whereas black-box testing is often malicious. (Imperva, n.d.)

White box testing is commonly used in software development to write a test. Written tests can be used to create test coverage. Test coverage describes the scope of the testing. Test coverage should be remarkably high on critical software components. Multiple types of tests differ in the level of testing. Most used tests in software projects are unit and integration testing. Other types of tests consist of end-to-end, performance, and acceptance tests. (Imperva, n.d.)

White box testing is extensively used in Cloud Manufacturing. Integration tests are used in Cloud Manufacturing to test integrations, whereas singular functions and methods are tested with unit tests. Tests are both back-end and front-end. This thesis only contains a limited number of tests related to integrations and features described in the document.

5.1.1 Unit Testing

A unit test is a technique for testing an isolated method or function. Unit testing is a programmatic way to write test cases to determine the method’s functionality. Unit tests reduce bugs in new and further developed features, reduce needs for manual testing, and improve general code design.

Unit test, by definition, follows a typical pattern that is mostly universal between programming languages. The unit test contains a test case, which is a base function and expected result. Based on the expected value, a test is either failed or passed. (TutorialPoint, n.d.)

Cloud Manufacturing has unit tests for both back-end and front-end. Most back-end unit tests involve part costing logic, whereas front-end unit tests focus on UI components and business logic. Cloud Manufacturing uses the xUnit package for back-end unit tests, and the front-end uses jest and react testing-library.

5.1.2 Integration Testing

An integration test is a testing technique for testing two or more modules. The goal of integration testing is to determine whether two components communicate reliably. Integration testing can also be used to measure performance. Integration testing is often used to test client-server or server-database functionality. (TutorialPoint, n.d.)

Cloud Manufacturing uses integration testing to test internal and external RESTful APIs. The test confirms the functionality to be accurate with specs. Integration tests are also used for testing authorization levels on API.

5.2 Testing Setup

Tests run whenever the source control detects changes to the code base. The tests are used to determine whether the code is working as intended. All the tests are required to pass successfully before building the application image. Different types of tests run on different virtualized environments since they require different configurations and take additional time to finish. This setup allows the tests to run parallel.

Integration test run inside docker-compose. The environment creates a testing database and fixture API with access to controllers and service methods. The fixture

API allows every integration test to be in a stale state. The stale state ensures that every test is in an identical form before running the test. After each test, the API uncommits any changes from the database and starts the next test.

The unit tests are running in a separate virtualized environment. Unit tests do not require much processing power, and therefore they do not take long to finish. Front-end and back-end tests run on the same environment to save time on setup times.

5.3 NC Express Integration

Integration to NC Express is primarily tested with integration testing. The NC Express service integration test cases are shown in Table 5. Most of the specific tests are only defined one way. For example, in Test case 6, multiple tests are created to cover all the expected values.

Table 5. NC Express integration test cases.

Case	Objective
1	API token required for any worker API request
2	Correct API token returns an OK response
3	Invalid API token returns an unauthorized response
4	GET “jobs/next” Endpoint returns the next available processing job
5	GET “jobs/{id}/file” Endpoint returns blob
6	GET “jobs” Endpoint return list of jobs
7	PUT “jobs/{id}” Endpoint requires correct next status and values

```
private readonly string _ncxKey = "1676734cdb7111ec9d640242ac120002";
private readonly string _headerName = "x-api-key";

[Fact]
public async Task WorkerRequest_CorrectKey()
{
        Client.DefaultRequestHeaders.Add(_headerName, _ncxKey);
        var response = await Client.GetAsync("/api/worker/jobs");
        response.StatusCode.Should().Be(HttpStatusCode.OK);
}
```

Code Snippet 1. NC Express service integration Test case 2.

The integration test created from Test case 2 is shown in Code Snippet 1. The unit test uses an HTTP client to execute a GET request on worker API. The test provides the correct API key and incorrect HTTP header. The test expects the HTTP response to be OK.

5.4 Authentication and Authorization

Authentication and authorization are tested with integration and unit tests. Unit tests focus on smaller methods, such as password hashing and matching functions, whereas integrations focus on RESTful API and database. Table 6 contains the main objectives of testing authentication and authorization.

Table 6. Authentication and authorization test cases.

Case	Objective
1	Internal login accepts only the correct username and password
2	After login, given authorization token gives access to internal API
3	Internal API is not accessible without an authorization token
4	A normal user is not able to access admin API
5	A normal user does not receive admin-only information from API
6	OIDC login requires correct fields in the request
7	OIDC login does not accept incorrect code

```
private readonly string _username = "test@cloud-manufacturing.com";
private readonly string _password = "TestPassword1!";

[Fact]
public async Task LoginRequest_CorrectPassword()
{
		var request = new LoginRequest { Username = _username, Password = _password };

		var response = await Client.PostAsJsonAsync($"/api/auth/login", request);

		response.StatusCode.Should().Be(HttpStatusCode.OK);

		var body = await response.Content.ReadAsAsync<LoginResponse>();

		body.Token.Should().NotBeNull();
		body.User.Email.Should().Be(_username);
}
```

Code Snippet 2. Authentication and authorization integration Test case 1.

Tests mainly focus on internal authentication and authorization. Authentication Service integration is tested based on OpenID Connect 1.0 specification. The integration test for Test case 1 is shown in Code Snippet 2.

The test sends the correct username and password into the internal authentication API endpoint. The test expects an OK HTTP response and requires the response body to have a token field with a non-null value and email to match the given username. Test case 2 ensures that the API can find a user with a username and can compare password hashes correctly.

5.5 ERP Data Import

ERP integration is tested on integration and unit test level. Unit tests are located in the front-end, and integration tests are between the server and database. The front-end unit testing focuses on user interface rendering correctly. This is to test both the UI logic and the business logic behind UI logic. Table 7 contains a list of main objectives when testing ERP data import, and the table includes the scope of the tests.

Table 7. ERP data import test cases.

Case	Objective	Scope
1	The view shows the correct number of rows	Front-end
2	The view formats the data with regex selector	Front-end
3	The view contains the correct CSV header row	Front-end
4	The view fetches old configurations with the CSV header.	Front-end
5	The view saves the new configuration after submitting	Front-end
6	Backend updates new materials according to the request	Back-end
7	Backend update request requires correct parameters	Back-end
8	The backend allows saving and updating of import configurations	Back-end

```
test('loads correct output number', async () => {
        const contents = ['1, DC01, 2.0', '2, DC02, 3.0'];
        const regex = '^(?<productId>.*?),(?<resource>.*?),(?<cost>.*?)\r';

        render(createComponent(contents, regex));

        expect(screen.getByTestId('total-formatted')).toHaveTextContent('2');
        expect(screen.getByTestId('total-failed')).toBeNull();
});
```

Code Snippet 3. ERP data import Test case 1.

The code snippet for Test case 1 is shown in Code Snippet 3. The test defines contents and Regex variables to describe the CSV file contents and selector for formatting the output. After variable definitions, the test calls the "createComponent" function, a utility function for creating the react component. The utility function is used since multiple unit tests initialize the same component with the same props. The unit test expects total formatted rows to match with total rows and failed rows not to be found since they are not shown if the value is 0.

6 CONCLUSIONS

The primary purpose of this thesis was to describe the general architecture and developed integrations of Tulus Cloud Manufacturing. The key aspects of this thesis were information security and testing. Secure and functional Cloud Manufacturing was deployed for the first pilot customers as the result of this thesis. The development of this project will continue based on pilot customer feedback and planned features.

6.1 Project Evaluation

Based on given feedback from the client (Prima Power) and employer (Jubic Oy), the project was a success. After around 11 months of part-time and mostly solo development of the software, a working product was produced which can be used to enhance the quotation process. This project has been a great learning opportunity for everyone involved in the development process.

A few things could be improved based on this project experience. During the early stages of development, a lack of direct contact with an expert who understood the existing process of sheet metal manufacturers resulted in unnecessary work. In later stages, multiple refactors of data structures and UIs had to be done because they were impractical or outright wrong. In addition, the inclusion of pilot customers' knowledge should have been used more in the early stages of development.

More test cases could have been written during the development phase. There were multiple cases where the application did not work as intended by adding or updating a feature. This could have been prevented if there had been better testing coverage.

Every project has something to improve on. In general, the project was a success. This is partly because of excellent management from both development parties. Product owners at Prima Power were extraordinarily active during the development, resulting in faster and more precise development goals.

6.2 Future Development

Development in software, such as Cloud Manufacturing, never ends if it is in use. Multiple features have been planned, and some of them have specifications. The thesis lists some of the upcoming development, for example, the Tulus integration and two-way ERP integration. The development ranges from back-end to front-end, with many bug fixes from existing features.

Some of the future development will focus on application distribution. The current infrastructure supports tens of instances while adding some manual work. There are plans to make application deployment almost automatic with technologies such as Amazon EKS, Helm, and Terraform. The new distribution pattern enables Prima Power to manage hundreds of instances with minimal work.

REFERENCES

Auth0. OpenID Connect Protocol. Accessed 1.4.2022. https://auth0.com/docs/authenticate/protocols/openid-connect-protocol

Auth0. JSON Web Tokens. Accessed 31.3.2022. https://auth0.com/docs/se-cure/tokens/json-web-tokens

Docker. Accessed 30.3.2022. https://www.docker.com/

GeeksForGeeks. Simple Mail Transfer Protocol. 5.11.2021. Accessed 3.4.2022. https://www.geeksforgeeks.org/simple-mail-transfer-protocol-smtp/

Gillis A. Docker Image. May 2021. Accessed 2.4.2022. https://www.tech-target.com/searchitoperations/definition/Docker-image

Imperva. White Box Testing. Accessed 17.5.2022. https://www.im-perva.com/learn/application-security/white-box-testing/

Jubic Oy. Accessed 28.3.2022. https://jubic.fi/

Kubernetes. Accessed 30.3.2022. https://kubernetes.io/

Loshin P. What is SQL? February 2022. Accessed 3.4.2022. https://www.tech-target.com/searchdatamanagement/definition/SQL

Microsoft. ASP.NET Core Introduction. 26.3.2022. Accessed 29.3.2022. https://docs.microsoft.com/en-us/aspnet/core/introduction-to-aspnet-core

Microsoft. EF Core. 25.5.2021. Accessed 29.3.2022. https://docs.mi-crosoft.com/en-us/ef/core/

Mozilla. JSON. Accessed. 27.3.2022. 29.3.2022. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/JSON

Mozilla. MCV. 18.2.2022. Accessed 1.4.2022. https://developer.mozilla.org/en-US/docs/Glossary/MVC

Mozilla. Regular expressions. 7.5.2022. Accessed 1.4.2022. https://devel-oper.mozilla.org/en-US/docs/Web/JavaScript/Guide/Regular_Expressions

MuleSoft. What is an API? Accessed 29.3.2022. https://www.mulesoft.com/re-sources/api/what-is-an-api

Oracle. What is ERP? Accessed 1.4.2022. https://www.oracle.com/erp/what-is-erp/

Oracle. What is a relational database? Accessed 3.4.2022. https://www.oracle.com/database/what-is-a-relational-database/

Prima Power. Accessed 28.3.2022. https://www.primapower.com/fi/

Prima Power. NC Express e3. Accessed 28.3.2022. https://www.primapower.com/fi/technologies/software/machine-programming/nc-express-e3

Prima Power. Tulus Analytics. Accessed 28.3.2022. https://www.primapower.com/fi/technologies/software/production-reporting-and-monitoring/Tulus-analytics

Prima Power. Tulus Cell. Accessed 28.3.2022. https://www.primapower.com/fi/technologies/software/hmi/Tulus-cell

Prima Power. Tulus Office. Accessed 28.3.2022. https://www.primapower.com/fi/technologies/software/production-planning/Tulus-office

Prima Power. Tulus software. Accessed 28.3.2022. https://www.primapower.com/fi/technologies/software

Prima Power. Tulus Visual Monitoring. Accessed 28.3.2022. https://www.primapower.com/fi/technologies/software/hmi/Tulus-visual-monitoring

React.js. Accessed 29.3.2022. https://reactjs.org/

Reactjs. What is virtual DOM? Accessed 29.3.2022. https://reactjs.org/docs/faq-internals.html

Reactjs. Why JSX? Accessed 29.3.2022. https://reactjs.org/docs/introducing-jsx.html

Redhat. What is a REST API? 8.5.2020. Accessed 29.3.2022. https://www.redhat.com/en/topics/api/what-is-a-rest-api

Rubens P. What are containers, and why do you need them? 27.6.2017. Accessed 2.4.2022. https://www.cio.com/article/247005/what-are-containers-and-why-do-you-need-them.html

SQLServerTutorial. What is SQL Server? Accessed 29.3.2022. https://www.sqlservertutorial.net/getting-started/what-is-sql-server/

TutorialPoint. Unit Testing. Accessed 17.5.2022. https://www.tutorialspoint.com/software_testing_dictionary/unit_testing.htm

TutorialPoint. Integration Testing. Accessed 17.5.2022. https://www.tutorialspoint.com/software_testing_dictionary/integration_testing.htm

Typescriptlang. Accessed 29.3.2022. https://www.typescriptlang.org/