OULU UNIVERSITY OF
APPLIED SCIENCES

Aram Abbasi

Capability to Customize Elements in Forklift Application

CAPABILITY TO CUSTOMIZE ELEMENTS IN FORKLIFT APPLICATION

Aram Abbasi
Bachelor’s Thesis
Spring 2023
Information Technology
Oulu University of Applied Sciences

1 ABSTRACT

Oulu University of Applied Sciences
Degree Programme in Information Technology Option of Web Development

Author: Aram Abbasi
Title of the bachelor's thesis: Capability to customize element in Forklift application.
Supervisor: Pekka Alaluukas
Term and year of completion: Spring 2023 Number of pages: 42

This thesis presents the development of a new feature for a Forklift application that enables users to customize various displayable information, such as font type and size, and layout configurations. The new feature was ordered by a customer who required increased flexibility and customization in their software application.

The theoretical background of the research draws on principles of software development, user experience design, and industrial ergonomics. The methodology includes a literature review of software development and user experience design principles, as well as user research and prototyping.

The new feature is implemented using the .NET Framework and WPF (Windows Presentation Foundation) technology, which allows users to customize the forklift application by changing displayable information, such as font type and size, and layout configurations. Users can choose from pre-defined templates or create their own custom templates.

The results of the research demonstrate that the new feature significantly improves the usability and accessibility of the forklift application. Users appreciate the ability to customize the application to meet their specific needs and preferences. The conclusions drawn from the research suggest that customizability is a key factor in the adoption and effectiveness of software applications in industrial settings.

Future development could include the exploration of other customization options, such as custom icons and graphics, and the integration of predictive analytics to further enhance the user experience of the forklift application.

PREFACE

This thesis, which was completed at Oulu University of Applied Sciences in Information Technology, focuses on the ability to customize elements in forklift applications. The study began in 2017 and was completed in 2023.

The purpose of this thesis is to explore the possibility of customizing various elements in forklift applications to meet the specific needs of various users. This thesis also seeks to improve understanding of the customization process and its impact on the overall performance of forklift applications.

I would like to take this opportunity to express my gratitude to my supervisor, Pekka Alaluukas, for his invaluable guidance, support, and encouragement throughout this study.

I thank sincerely Toni Hämeenniemi, the Product Owner, and Elvis Punnek, the Timber Team Leader, for their assistance and feedback throughout the research process.

I would also like to thank all the individuals who have contributed to this thesis in various ways, including those who have provided technical support, research materials, and other resources. Without their help, this thesis would not have been possible.

Finally, I would like to express my heartfelt appreciation to my family and friends for their unwavering support and encouragement throughout my academic journey.

Oulu, 30.3.2023
Aram Abbasi

Contents

2 VOCABULARY

DOTNET FRAMEWORK

The Dot NET Framework (1) (.NET) is a software development platform created by Microsoft that provides developers with a consistent and comprehensive programming model for building applications. It includes a large library of pre-built code and a virtual machine that manages the execution of code written in multiple languages. The .NET Framework supports multiple programming languages, including C#, Visual Basic, and F# (Microsoft 2023a).

WPF

WPF (2) is an abbreviation for Windows Presentation Foundation, a user interface (UI) framework developed by Microsoft for the development of desktop applications on Windows. It provides an extensive array of tools and APIs for developing visually appealing and interactive user interfaces. WPF defines the UI with XAML (Extensible Application Markup Language), which allows for easy separation of the UI design from the application logic (Microsoft 2023b).

WPF USER CONTROLS

WPF user controls (3) are a type of custom control that can be created and used in Windows Presentation Foundation (WPF) applications. They are reusable UI components that contain a set of visual elements and functionality. User controls enable developers to create complex UI elements by combining simpler controls, as well as a means of separating the UI design from the application logic (Microsoft 2023b).

WPF LISTBOX

WPF ListBox (4) is a control that is used to display a collection of items in a list format. It provides various features such as data binding, item templates, selection modes, and grouping to enhance the user experience. The ListBox control can display any type of data, including strings, images, and complex objects. It is highly customizable and can be easily styled to match the look and feel of your application (Microsoft 2023b).

LISTBOX ITEMTEMPLATE

In WPF, the ListBoxItemTemplate (5) is used to define the appearance of each item in a ListBox control. It allows developers to customize the visual appearance of the list items by specifying a data template. The data template can include any combination of WPF controls, such as text blocks, images, and buttons, to display the data in the desired format. The ListBoxItemTemplate can be defined inline or as a separate resource (Microsoft 2023c).

DATATEMPLATE

A DataTemplate (6) in WPF is a way to define the visual appearance of data bound to a control. It provides a framework for displaying data in a specific format and can contain an unlimited number of WPF controls. A DataTemplate can be defined inline or as a resource and assigned to a control's ItemTemplate property to specify how the data should be displayed. DataTemplates are frequently used with ItemsControls, such as ListBox and ListView, to display a collection of items (Microsoft 2023d).

WCF

WCF (7) stands for Windows Communication Foundation, which is a framework developed by Microsoft for building distributed applications. It provides a unified programming model for building service-oriented applications that can communicate across various platforms and technologies. WCF supports multiple communication protocols, including HTTP, TCP, and named pipes, and can be used to build services that expose functionality to other applications, both on the same machine and over a network (Microsoft 2023e).

ERP

ERP (8) stands for Enterprise Resource Planning, a type of software that assists organizations in centralizing and integrating their core business processes. ERP software typically includes modules for accounting, human resources, inventory management, supply chain management, and customer relationship management. ERP systems can improve efficiency, lower costs, and improve decision-making by providing a unified view of business operations (Consulting, panorama 2021).

SQL

SQL (9) (Structured Query Language) is a programming language used to manage and manipulate relational databases. It provides a standard method of communicating with a database for tasks such as data retrieval, data modification, and user permission management. SQL is widely used in enterprise applications, data analytics, and data science, and it is supported by numerous database management systems. (Microsoft 2023f).

C#

C# (10) is a modern, object-oriented programming language that can be used to create a wide range of applications such as desktop, web-based, games, and mobile apps. Its main advantage is that it supports the.NET Framework, which provides a large library of pre-built code. C# is widely used in enterprise and commercial applications, and it is supported by Microsoft's Visual Studio IDE and the.NET Framework (Microsoft 2023g).

3 INTRODUCTION

The forklift program is a sub-project of Pinja's ERP and was developed over a five-year period. Its primary goal is to optimize the transportation and handling of raw materials and products in sawmills. The program is installed on a tablet and placed in a forklift, and it can connect to GPC to provide optimal tracking and routing capabilities. The program was designed using cutting-edge design techniques with the goal of reducing the number of clicks required to complete a process. It was developed in.NET format with WPF and the C# programming language, with SQL database.

This thesis discusses the project of adding a new feature to the program that allows the user to change the information display. The feature was developed to enhance the user experience and make it easier to use the program. In recognition of its innovation, the Forklift program won PuuMessuje's novelty product competition in 2019.

4 HOW THE FORKLIFT WORKS

To have better understanding of the customer’s request, it is better to explain how the Forklift works.

4.1 Software architecture

Forklift is a multi-tires windows-based application which is divided into the following layers:

4.1.1 Business logic layer

The business logic layer in the Forklift application is responsible for implementing the business rules and logic that govern the behavior of the system. This layer sits between the presentation layer and the data layer, and it acts as an intermediary between the two layers.

The primary purpose of the business logic layer is to encapsulate the business rules and logic of the system in a way that is independent of the user interface and the data access layer. This allows for greater flexibility and maintainability of the system, as changes to the business rules and logic can be made in the business logic layer without affecting the other layers of the system.

The Net Standard 2 was used to implement this layer. By using .NET Standard 2, we were able to write this layer of the Forklift software in a platform-agnostic way, which means that it can run on any .NET platform that supports .NET Standard 2.

4.1.2 User interface layer

The user interface (UI) layer in Forklift architecture is responsible for presenting information to and receiving input from the user. This layer is also known as the presentation layer or the client-side layer.

The primary purpose of the user interface layer is to provide a user-friendly interface for users to interact with the system. This includes displaying information in a way that is easy to understand and navigate. In addition, it allows users to enter data and interact with the system in a natural and intuitive way.

To implement this layer, we used WPF (Windows Presentation Foundation). WPF is a user interface framework for building desktop applications on the Windows operating system (Microsoft 2023b). It provides a rich set of controls and visual elements that can be used to create visually appealing and interactive user interfaces. WPF provides a consistent and modern look and feel for your application, which can help to improve the user experience. Additionally, WPF provides a flexible layout system that can adapt to different screen sizes and resolutions, which is important for creating responsive user interfaces (Microsoft 2023b).

4.1.3 Database layer

A database is a structured collection of data that is organized in a way that enables efficient storage, retrieval, and management of the data. In the software, a database is used to store and manage the data that is used by the system (Microsoft 2023h).

The database layer is responsible for managing the interactions between the software and the database. It provides a set of functions and APIs that allow the software to store, retrieve, and manipulate data in the database (Microsoft 2023h)

Using a database layer can improve the scalability and performance of the software by allowing data to be stored and accessed efficiently. Additionally, using a database layer can provide data consistency and integrity, which is important for ensuring the accuracy and reliability of the system (Microsoft 2023h). In this layer, we use SQL (Structured Query Language). SQL is a standard language used for managing relational databases. In Forklift, SQL is being used to interact with the database and manipulate data stored in it. Using SQL can provide a powerful and flexible way to interact with the database, allowing the system to store and retrieve data efficiently. Additionally, using SQL can provide advanced functionality such as filtering, sorting, and aggregation of data, which can be important for providing insights and analytics based on the data stored in the database.

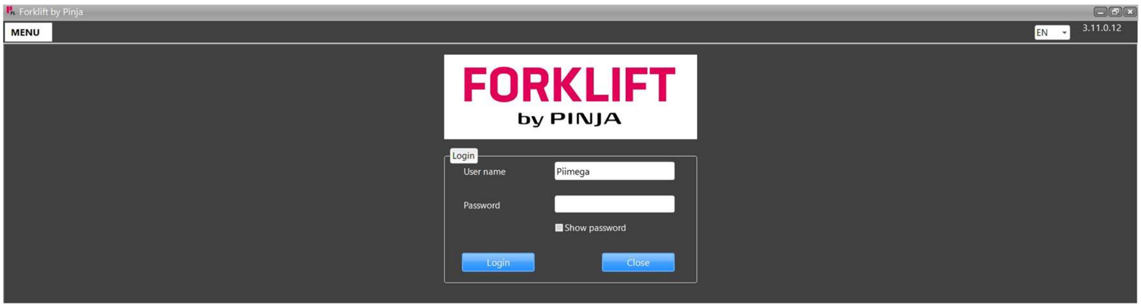
4.1.4 WCF conection sub-layer

This sub-layer in Forklift is responsible for making the connection between the client and WCF (Windows Communication Foundation). WCF is a Microsoft framework that provides a unified programming model for building service-oriented applications. In Forklift, WCF is being used to implement the service layer that allows the client to interact with the software.

This sub-layer is responsible for managing communication between the client and the WCF service layer. It provides a set of functions and APIs that allow the client to connect to the WCF service, send requests, and receive responses. Using this sub-layer can abstract away some of the complexity of managing client-WCF communication, making it easier to maintain and update the system. Additionally, using this sub-layer can provide additional security and scalability features that can be important for ensuring the reliability and availability of the system.

4.2 Layer interaction

Upon running the software, the business logic layer initializes primary objects such as setting files and network tools. Next, the UI layer displays the login form, which is depicted in Figure 1.



User name: Piimega

□ Show password

FIGURE 1. Login form

When the user presses the login button on the form, the UI layer populates an instance of the login class object and sends it to the business logic (BL) layer. The BL layer utilizes the Windows Communication Foundation (WCF) sub-layer to establish a connection to the WCF server to transmit the login data. Subsequently, on the WCF side, the BL layer establishes a connection with the database layer, retrieves the user data, and verifies its accuracy. If everything is correct, the WCF responds to the client BL layer, and the main data retrieval process commences.

The Forklift application requires extensive data collections such as collections of package data, work orders, and various other lists of collection data. Once the WCF grants authorization for logging into the system, the BL layer initiates a thread to retrieve data from the WCF side. This process repeats every 20 seconds for each bundle of data. After downloading each collection, the BL layer updates the client-side data list, and if there are any modifications to the data list, it calls upon the UI layer to update itself.

Additionally, the BL layer creates a buffer to store data change requests resulting from user actions, such as package movements. It also employs another thread to transmit the change data request from the buffer list. Once all the data has been retrieved and set, the UI layer displays the main page, as shown in the image below.

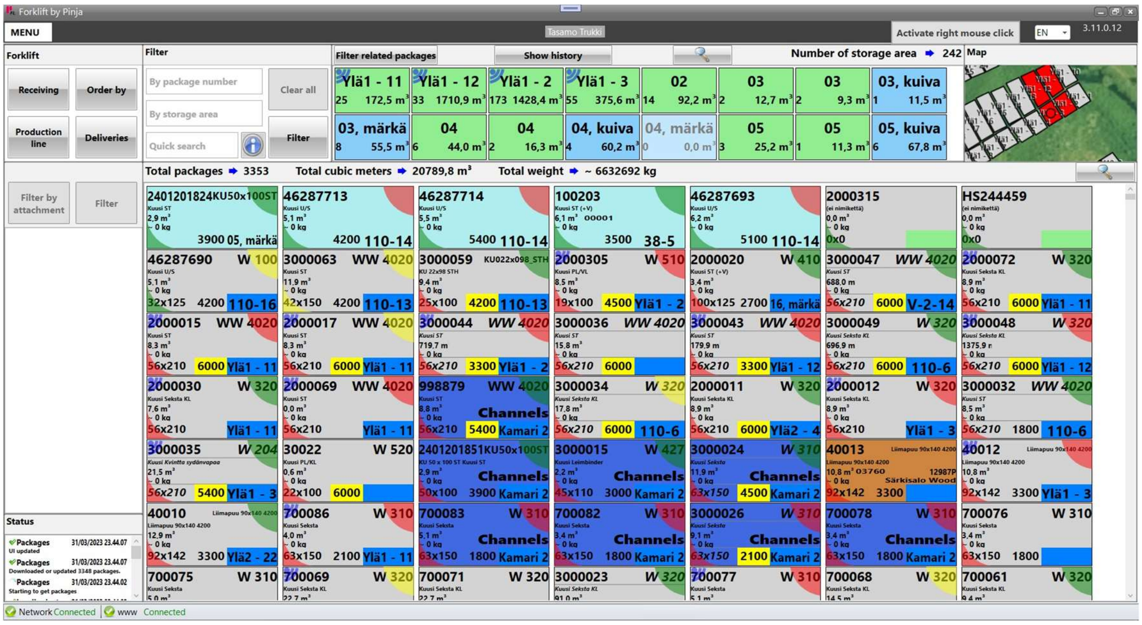
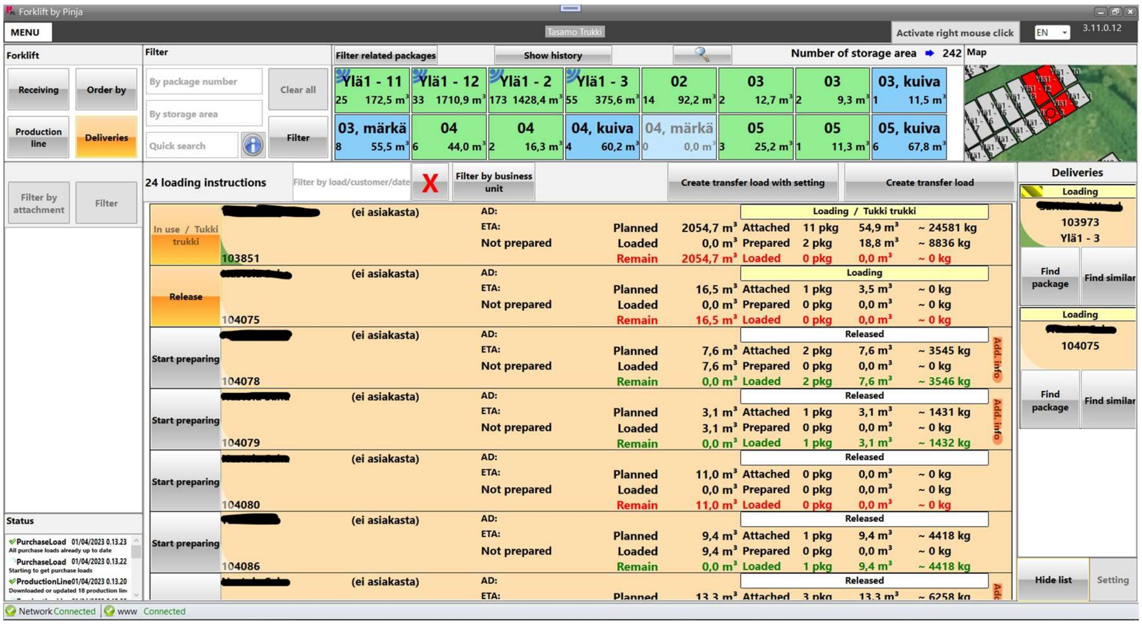


FIGURE 2. Main page

In the following section, UI layer is discussed, this is a crucial component of this subject.

As previously stated, the Forklift application necessitates extensive data collection. To update the data in the data lists, the user interface (UI) layer utilizes them as a data source for the list boxes and displays the information in a predetermined format known as a data template.

As illustrated in Figure 2, the user interface (UI) employs a list box on the main page to display packages in a horizontal list. The UI utilizes a data template to showcase the selected properties within the box. Additionally, there are two other list boxes that pertain to the subject matter of this document.



Number of storage area ➡ 242

Ylä1 - 11 25 172,5 m³	Ylä1 - 12 33 1710,9 m³	Ylä1 - 2 173 1428,4 m³	Ylä1 - 3 55 375,6 m³	02 14 92,2 m³	03 2 12,7 m³	03 2 9,3 m³	03, kuiva 1 11,5 m³
03, märkä 8 55,5 m³	04 6 44,0 m³	04 2 16,3 m³	04, kuiva 4 60,2 m³	04, märkä 0 0,0 m³	05 3 25,2 m³	05 1 11,3 m³	05, kuiva 6 67,8 m³

24 loading instructions

In use / Tukki trukki	[redacted] 103851	(ei asiakasta)	AD: ETA: Not prepared	Loading / Tukki trukki					
				Planned	2054,7 m³	Attached	11 pkg	54,9 m³	~ 24581 kg
				Loaded	0,0 m³	Prepared	2 pkg	18,8 m³	~ 8836 kg
				Remain	2054,7 m³	Loaded	0 pkg	0,0 m³	~ 0 kg
Release	[redacted] 104075	(ei asiakasta)	AD: ETA: Not prepared	Loading					
				Planned	16,5 m³	Attached	1 pkg	3,5 m³	~ 0 kg
				Loaded	0,0 m³	Prepared	0 pkg	0,0 m³	~ 0 kg
				Remain	16,5 m³	Loaded	0 pkg	0,0 m³	~ 0 kg
Start preparing	[redacted] 104078	(ei asiakasta)	AD: ETA: Not prepared	Released					
				Planned	7,6 m³	Attached	2 pkg	7,6 m³	~ 3545 kg
				Loaded	7,6 m³	Prepared	0 pkg	0,0 m³	~ 0 kg
				Remain	0,0 m³	Loaded	2 pkg	7,6 m³	~ 3546 kg
Start preparing	[redacted] 104079	(ei asiakasta)	AD: ETA: Not prepared	Released					
				Planned	3,1 m³	Attached	1 pkg	3,1 m³	~ 1431 kg
				Loaded	3,1 m³	Prepared	0 pkg	0,0 m³	~ 0 kg
				Remain	0,0 m³	Loaded	1 pkg	3,1 m³	~ 1432 kg
Start preparing	[redacted] 104080	(ei asiakasta)	AD: ETA: Not prepared	Released					
				Planned	11,0 m³	Attached	0 pkg	0,0 m³	~ 0 kg
				Loaded	0,0 m³	Prepared	0 pkg	0,0 m³	~ 0 kg
				Remain	11,0 m³	Loaded	0 pkg	0,0 m³	~ 0 kg
Start preparing	[redacted] 104086	(ei asiakasta)	AD: ETA: Not prepared	Released					
				Planned	9,4 m³	Attached	1 pkg	9,4 m³	~ 4418 kg
				Loaded	9,4 m³	Prepared	0 pkg	0,0 m³	~ 0 kg
				Remain	0,0 m³	Loaded	1 pkg	9,4 m³	~ 4418 kg
	[redacted]	(ei asiakasta)	AD: ETA:	Released					
				Planned	13,3 m³	Attached	3 pkg	13,3 m³	~ 6258 kg

Deliveries

Loading
[redacted]
103973
Ylä1 - 3

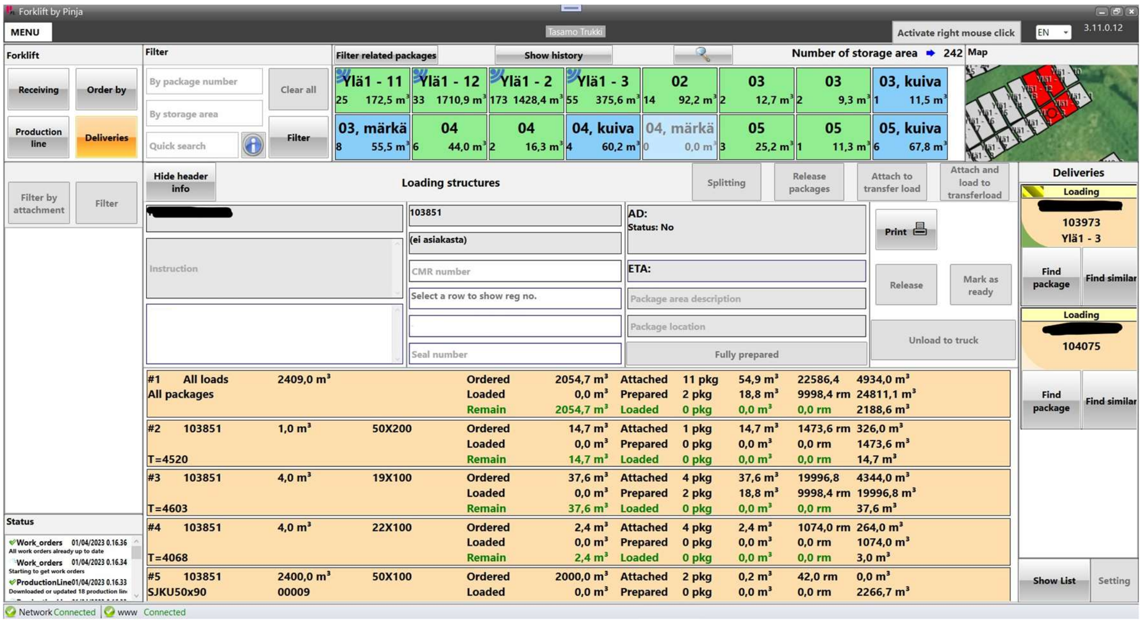
Loading
[redacted]
104075

Status

✓ PurchaseLoad 01/04/2023 0.13.23
All purchase loads already up to date
PurchaseLoad 01/04/2023 0.13.22
Starting to get purchase loads
✓ ProductionLine 01/04/2023 0.13.20
Downloaded or updated 18 production lin

✓ Network Connected | ✓ www Connected

FIGURE 3. Loading instructions list box



Number of storage area ➡ 242

Ylä1 - 11 25 172,5 m³	Ylä1 - 12 33 1710,9 m³	Ylä1 - 2 173 1428,4 m³	Ylä1 - 3 55 375,6 m³	02 14 92,2 m³	03 2 12,7 m³	03 2 9,3 m³	03, kuiva 1 11,5 m³
03, märkä 8 55,5 m³	04 6 44,0 m³	04 2 16,3 m³	04, kuiva 4 60,2 m³	04, märkä 0 0,0 m³	05 3 25,2 m³	05 1 11,3 m³	05, kuiva 6 67,8 m³

Loading structures

[redacted]

103851
(ei asiakasta)
Select a row to show reg no.

AD:
Status: No
ETA:

#1 All loads	2409,0 m³		Ordered	2054,7 m³	Attached	11 pkg	54,9 m³	22586,4	4934,0 m³
All packages			Loaded	0,0 m³	Prepared	2 pkg	18,8 m³	9998,4 rm	24811,1 m³
			Remain	2054,7 m³	Loaded	0 pkg	0,0 m³	0,0 rm	2188,6 m³
#2 103851	1,0 m³	50X200	Ordered	14,7 m³	Attached	1 pkg	14,7 m³	1473,6 rm	326,0 m³
			Loaded	0,0 m³	Prepared	0 pkg	0,0 m³	0,0 rm	1473,6 m³
T=4520			Remain	14,7 m³	Loaded	0 pkg	0,0 m³	0,0 rm	14,7 m³
#3 103851	4,0 m³	19X100	Ordered	37,6 m³	Attached	4 pkg	37,6 m³	19996,8	4344,0 m³
			Loaded	0,0 m³	Prepared	2 pkg	18,8 m³	9998,4 rm	19996,8 m³
T=4603			Remain	37,6 m³	Loaded	0 pkg	0,0 m³	0,0 rm	37,6 m³
#4 103851	4,0 m³	22X100	Ordered	2,4 m³	Attached	4 pkg	2,4 m³	1074,0 rm	264,0 m³
			Loaded	0,0 m³	Prepared	0 pkg	0,0 m³	0,0 rm	1074,0 m³
T=4068			Remain	2,4 m³	Loaded	0 pkg	0,0 m³	0,0 rm	3,0 m³
#5 103851	2400,0 m³	50X100	Ordered	2000,0 m³	Attached	2 pkg	0,2 m³	42,0 rm	0,0 m³
SJKU50x90	00009		Loaded	0,0 m³	Prepared	0 pkg	0,0 m³	0,0 rm	2266,7 m³

Deliveries

Loading
[redacted]
103973
Ylä1 - 3

Loading
[redacted]
104075

Status

✓ Work_orders 01/04/2023 0.16.36
All work orders already up to date
Work_orders 01/04/2023 0.16.34
Starting to get work orders
✓ ProductionLine 01/04/2023 0.16.33
Downloaded or updated 18 production lin

✓ Network Connected | ✓ www Connected

FIGURE 4. Loading instruction rows

The two list boxes also utilize a data template to display the selected properties within their respective boxes.

After getting a handle on how the system works, let us have a discussion about the proposed addition of a new feature.

The new feature entails the capability to personalize the information exhibited in each of the three lists. In essence, the information displayed in the lists can be tailored to each user's activity, resulting in different data being presented to each user.

In the following chapters, the requirement analysis, design, implementation, and testing of this new feature will be presented in this particular order. Each chapter will explain the necessary steps to be taken in a classical manner. Subsequently, a brief overview of how to execute each chapter in practice will also be provided.

5 REQUIREMENT ANALYSIS

Requirement analysis is a critical process that involves identifying, analysing, and documenting the needs and expectations of stakeholders for a new feature in a software system. It is a crucial part of the software development lifecycle, as it helps ensure that the resulting software system meets the needs of its users (Accessibility Requirements Tool 2021).

The goal of requirement analysis for a new feature in a software system is to identify the functional and non-functional requirements of the system. Functional requirements are those that describe what the software system should do, while non-functional requirements describe how well the system should perform in terms of factors such as usability, security, and performance.

Requirement analysis typically involves the following steps:

5.1 Elicitation

This involves gathering information from stakeholders, such as end-users, business owners, and developers, to understand their needs and expectations for the new feature.

5.2 Analysis

This involves analysing the gathered information to identify the key functional and non-functional requirements of the new feature.

5.3 Specification

This involves documenting the requirements in a clear and concise manner, using techniques such as use cases, user stories, and functional specifications.

5.4 Validation

This involves reviewing and validating the requirements with stakeholders to ensure that they accurately reflect their needs and expectations.

5.5 Management

This involves managing changes to the requirements over time, as well as ensuring that they are aligned with the overall goals and objectives of the software system.

Overall, requirement analysis is a critical process that helps ensure that a new feature in a software system meets the needs and expectations of its users and is delivered on time and within budget.

5.6 Practical approach

After installing the new generation of the Forklift program, it became apparent that different users utilized the program differently. This meant that each user needed access to specific details of the information presented. For example, in a sawmill, forklift drivers have varying tasks. Some are responsible for delivering packages to production lines, while others are responsible for loading and delivering finished products to customers. Consequently, the information that each user considers essential differs.

For instance, for the user tasked with delivering packages to customers, seeing customer information such as the customer's name and address is crucial. On the other hand, the user who moves packages between production lines prioritizes viewing the package code and product type. To address this issue, the design team held meetings with the client to discuss this matter. Ultimately, it was agreed that a new feature would be added to the system. The fundamental specifications of this feature were established, and the design team was tasked with completing both the functional and non-functional components.

The key features that were defined for this project are as follows:

1. Introduce a new form in the main program (ERP) that allows users to customize the displayed information in the system.

2. In this form, provide the option to select the desired information for the three main lists: packages, loading instructions, and loading instructions rows.
3. Enable customization of the displayed information for each user.
4. Allow users to change specifications such as font size, display location, and other required items for the selected information.
5. When Forklift starts, the system collects the defined specifications for each list based on the entered user and then displays the lists according to those specifications.

After further review by the research and development team, as well as considering the needs of the customer and the company, the list of features was finalized. This list includes:

1. Ability to copy settings for other users: This feature allows a user to copy their customized settings for each list and apply them to other users' accounts, saving them time and effort.
2. Default settings: If a user has not customized their list settings, the program will use default settings to display the information.
3. Ability to add new details for each list: This feature allows the user to add new details to each list, such as new fields or information that they want to display.
4. Activation and deactivation for each customer: This feature allows the user to activate or deactivate the list customization feature for each customer, depending on their needs.
5. Easy addition of new lists in the future: The program has been designed to make it easy to add new lists in the future, so that additional information can be displayed to the user as needed.

   These features are important as they enhance the customization options of the system and provide more flexibility to the users. The ability to copy settings for other users saves time and effort for administrators and allows for consistent settings across different users. The default settings ensure that users who have not customized their lists still have a functional interface. The ability to add new details for each list provides future proofing and allows the system to adapt to changing needs. The ability to activate and deactivate the feature for each customer allows for granular control over the customization options. Finally, the ease of adding other lists in the future allows for scalability and continuous improvement of the system.

The output of this phase moves to the design phase. In this phase, the design team will take the requirements and specifications gathered from the previous phase and create detailed designs for the new feature. The team will also ensure that the design aligns with the overall design of the forklift program and is scalable for future changes.

The design team will first create a high-level design document that outlines the overall architecture and flow of the new feature. This document will include diagrams and descriptions of each component, how they interact with each other, and the overall flow of information within the feature.

Next, the team will create a low-level design document that provides more details on each component and how they will be implemented. This document will include information such as class and method definitions, database schema designs, and any necessary API specifications.

During the design phase, the team will also create mock-ups of the new user interface to ensure that it meets the requirements and specifications gathered from the previous phase. The mockups will be presented to the customer for approval before proceeding to the implementation phase.

Once the design is complete and approved by the customer, the implementation phase can begin.

6 DESIGN

The design of a new feature in software involves creating a detailed plan or blueprint for how the feature will be implemented in the software system. The goal of the design process is to ensure that the feature is implemented in a way that is efficient, effective, and meets the needs of its users.

The design process typically involves the following steps:

6.1 System Architecture

This involves defining the overall system architecture for the software system, including the hardware and software components that will be used to implement the feature.

6.2 Detailed Design

This involves creating a detailed design for the new feature, including the user interface, algorithms, data structures, and software components that will be used to implement the feature.

6.3 Prototyping

This involves creating a prototype of the feature to test its functionality and usability, and to identify any potential design flaws or issues.

6.4 Testing and Validation

This involves testing and validating the feature to ensure that it meets the specified requirements and functions as expected.

6.5 Documentation

This involves documenting the design of the feature, including specifications, user manuals, and other documentation to support the implementation and maintenance of the feature.

Overall, the design process is critical for ensuring the success of a new feature in software. A well-designed feature is more likely to be efficient, effective, and meet the needs of its users, resulting in greater user satisfaction and better overall system performance.

6.6 Practical approach

Based on the findings of the request analysis, the process is divided into two parts: the first part involves defining box specifications in the main program, while the second part involves using these specifications in the forklift program as a user. These processes are carried out step-by-step as follows:

In the ERP section, the user opens the form and selects a forklift user. Next, one of the three requested samples is selected, and the list of items related to the selected sample is displayed. By selecting any item, the user can define the display characteristics of that item. This information is then stored in the database.

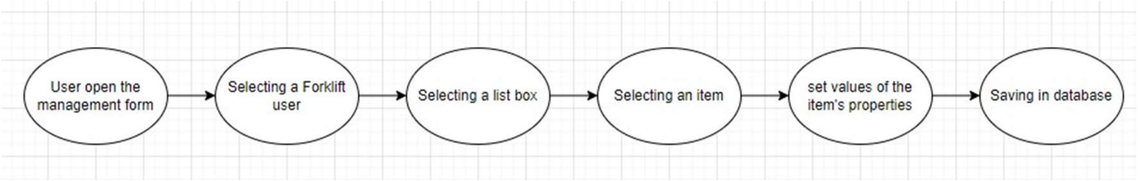

User open the management form
Selecting a Forklift user
Selecting a list box
Selecting an item
set values of the item's properties
Saving in database

FIGURE 5. Process steps in main program (ERP)

In the Forklift side, the user login and the settings of the boxes are taken from the database. The data model is created and attached to the corresponding lists, and information is displayed based on these specifications.

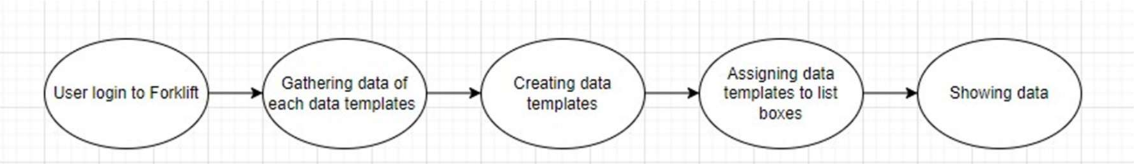
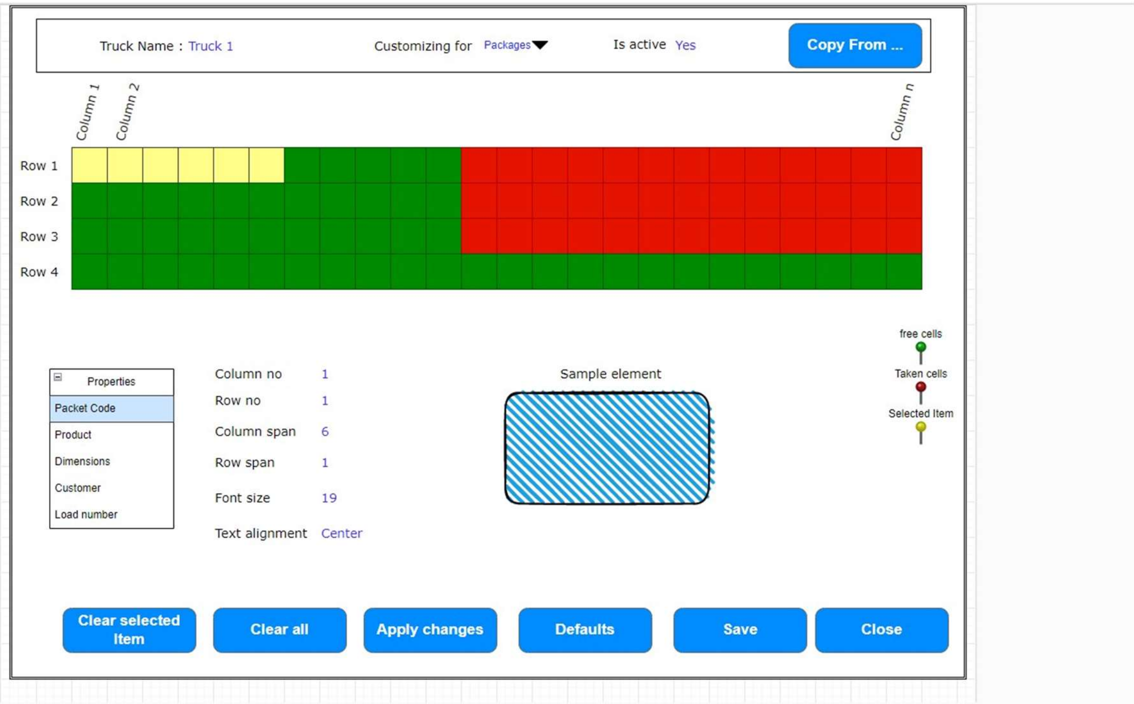

User login to Forklift
Gathering data of each data templates
Creating data templates
Assigning data templates to list boxes
Showing data

FIGURE 6. Process steps in Forklift

Based on the defined processes in the main program, a form for customizing the samples was created as follows:



Truck Name : Truck 1

Customizing for Packages

Is active Yes

Column no 1

Row no 1

Column span 6

Row span 1

Font size 19

Text alignment Center

FIGURE 7. Prototype of management form

The system has two tables to store data, the first one is called "Truck" table which includes the following columns:

ID (integer, not null)

TruckDescription (nvarchar(500))

The second table is called "ForkliftSetting" which includes the following columns:

ID (integer, not null)

TruckID (integer, not null, references Truck.ID)

Configtype (smallint) (Can be 0 for packages, 1 for loading instructions and 2 for loading instruction’s rows)

ConfigXML (XML type)

The "ForkliftSetting" table has a foreign key constraint on the "TruckID" column, which references the "ID" column of the "Truck" table. The " ConfigXML" column is an XML type, which stores the specifications of each item selected by the user.

And finally, the required classes, which are shown below, were designed.

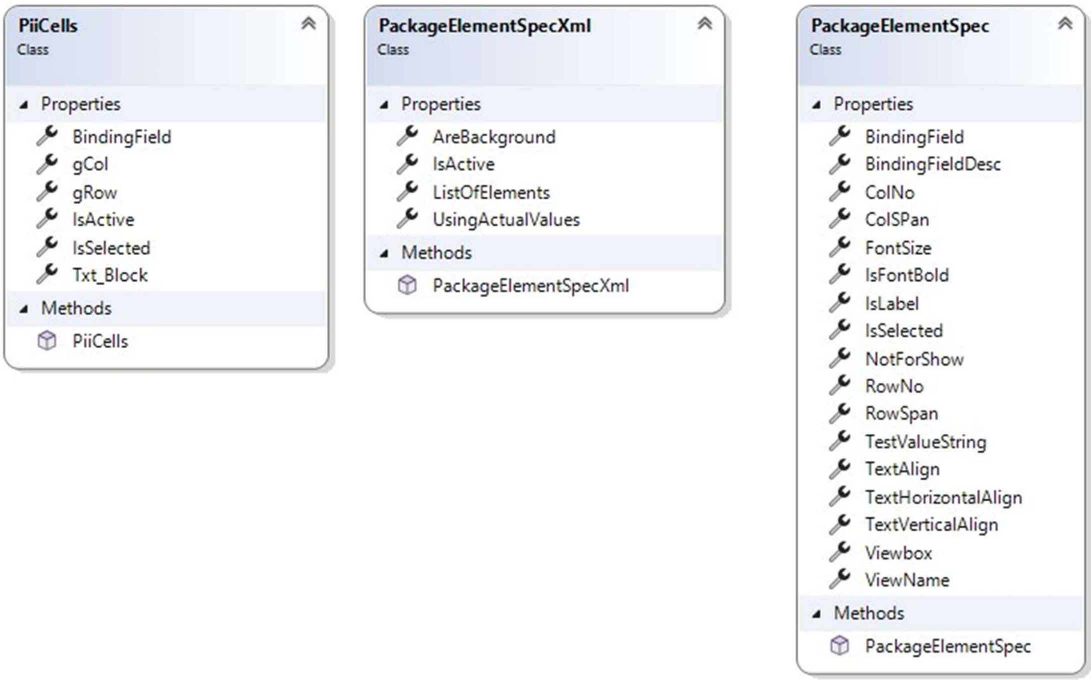


FIGURE 8. Class diagram

7 IMPLEMENTATION

The implementation of a new feature in a software system involves the actual coding and development of the feature, and the integration of the code with the existing software system. The goal of the implementation process is to bring the design of the feature to life and create a working feature that meets the specified requirements and works as expected.

The implementation process typically involves the following steps:

7.1 Coding

This involves writing the code for the new feature using the programming language and tools specified in the design phase.

7.2 Integration

This involves integrating the new code with the existing software system, ensuring that the feature works seamlessly with the rest of the system.

7.3 Testing

This involves testing the new feature to ensure that it works as expected, meets the specified requirements, and does not introduce any new bugs or issues into the system.

7.4 Debugging

This involves identifying and fixing any bugs or issues that are discovered during testing, ensuring that the feature works reliably and as intended.

7.5 Deployment

This involves deploying the new feature to the production environment, making it available to end-users.

Overall, the implementation process is critical for ensuring that a new feature is successfully integrated into the software system and works as expected. A well-implemented feature will be reliable, efficient, and meet the needs of its users, contributing to the overall success of the software system.

7.6 Practical approach

To start the implementation process, we need to first create SQL database objects as shown in the picture below.

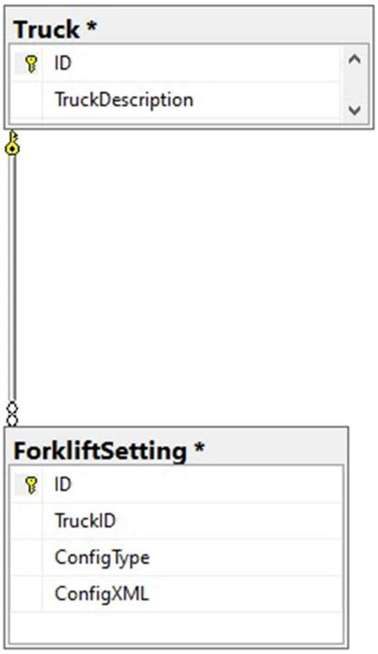
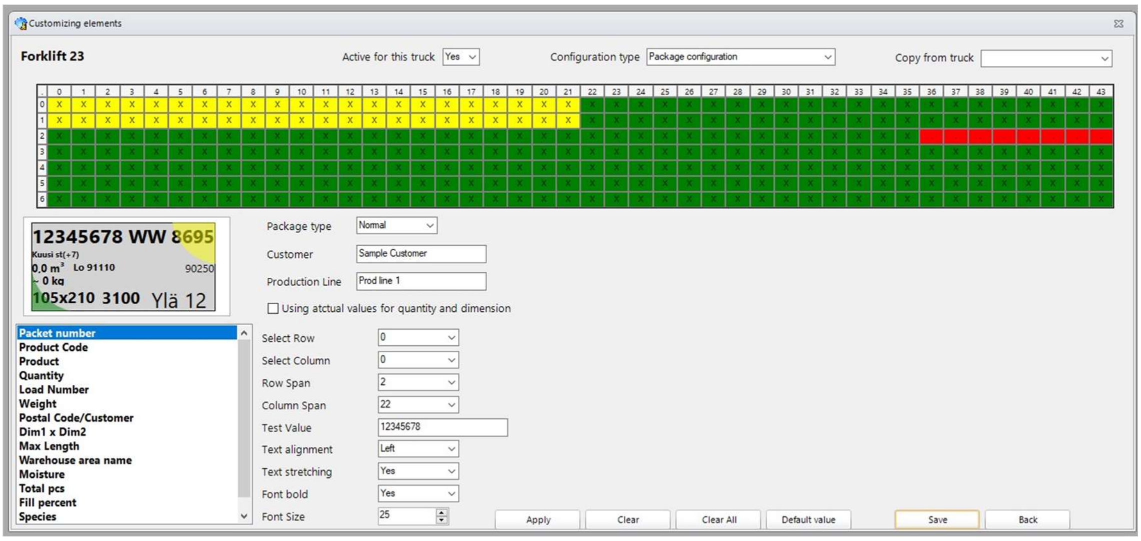


FIGURE 9. Sql Tables

The implementation is divided into two separate parts for the main program and for the forklift. On the main program side, there will be a form for customizing elements, as shown in the following figures.



Forklift 23

Active for this truck: Yes

Configuration type: Package configuration

Copy from truck:

Package type: Normal

Customer: Sample Customer

Production Line: Prod line 1

☐ Using atctual values for quantity and dimension

Select Row: 0

Select Column: 0

Row Span: 2

Column Span: 22

Test Value: 12345678

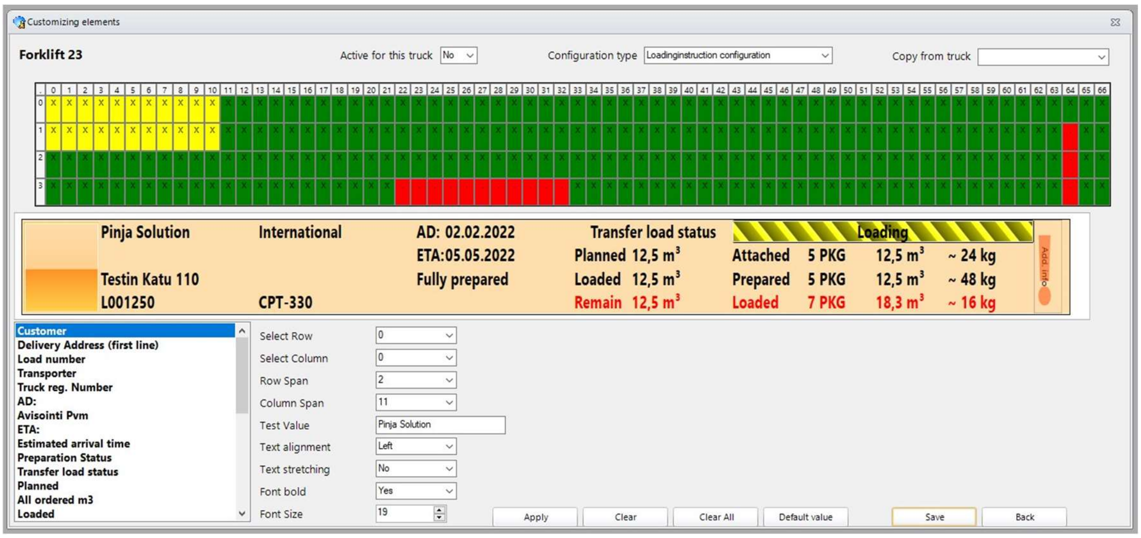
Text alignment: Left

Text stretching: Yes

Font bold: Yes

Font Size: 25

FIGURE 10. Customizing for the packages



Forklift 23

Active for this truck: No

Configuration type: Loadinginstruction configuration

Copy from truck:

Select Row: 0

Select Column: 0

Row Span: 2

Column Span: 11

Test Value: Pinja Solution

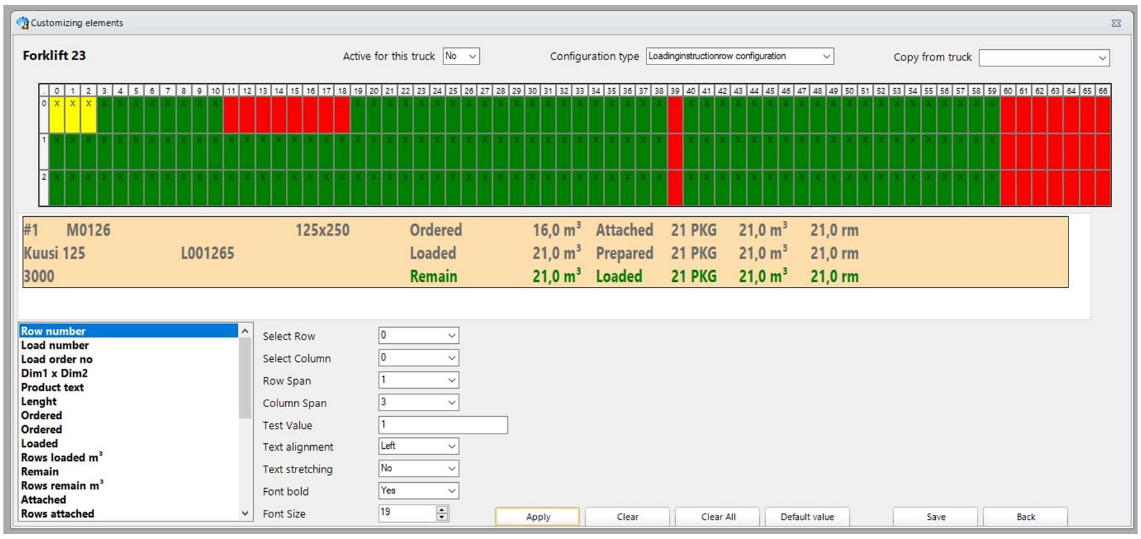
Text alignment: Left

Text stretching: No

Font bold: Yes

Font Size: 19

FIGURE 11. Customizing for the loading instructions



Forklift 23

Active for this truck: No

Configuration type: Loadinginstructionrow configuration

Copy from truck:

#1	M0126		125x250	Ordered	16,0 m³	Attached	21 PKG	21,0 m³	21,0 rm
Kuusi 125		L001265		Loaded	21,0 m³	Prepared	21 PKG	21,0 m³	21,0 rm
3000				Remain	21,0 m³	Loaded	21 PKG	21,0 m³	21,0 rm

Select Row: 0
Select Column: 0
Row Span: 1
Column Span: 3
Test Value: 1
Text alignment: Left
Text stretching: No
Font bold: Yes
Font Size: 19

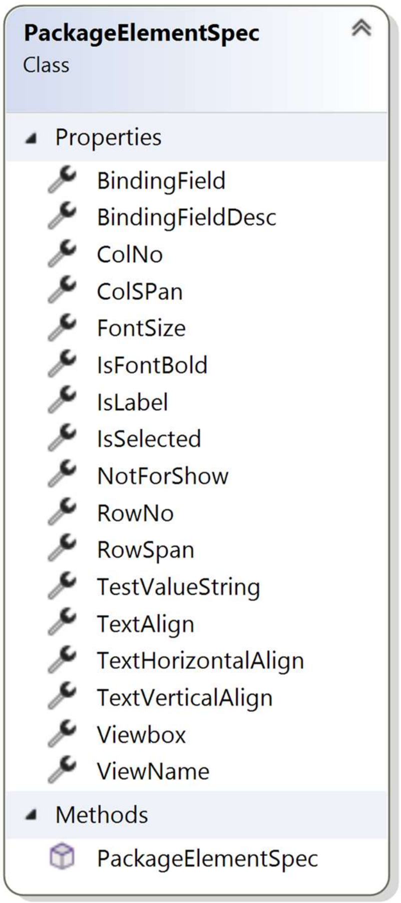
FIGURE 12. Customizing for the loading instruction's rows

As depicted in the images, each list consists of distinct properties; however, they have common features.

The properties are:

a) Row number
b) Column number
c) Row span
d) Column span
e) Test value
f) Text alignment
g) Text stretching
h) Font type
i) Font size

To store and retrieve these properties, it is necessary to define a class.



PackageElementSpec
Class

Properties

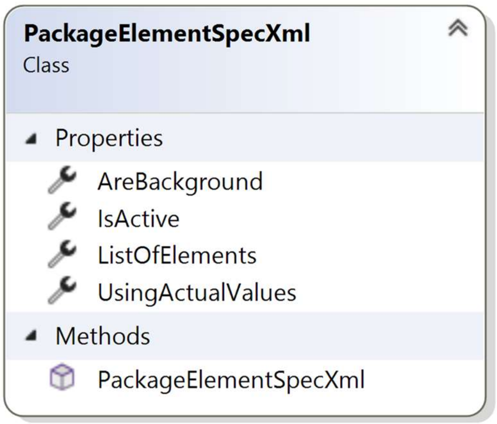
- BindingField
- BindingFieldDesc
- ColNo
- ColSPan
- FontSize
- IsFontBold
- IsLabel
- IsSelected
- NotForShow
- RowNo
- RowSpan
- TestValueString
- TextAlign
- TextHorizontalAlign
- TextVerticalAlign
- Viewbox
- ViewName

Methods

- PackageElementSpec

FIGURE 13. Package element specification class

However, since each list contains multiple items, it is necessary to define a new class that can hold a collection of these properties for each item. This new class should also have a property for enabling or disabling customization and other auxiliary properties.



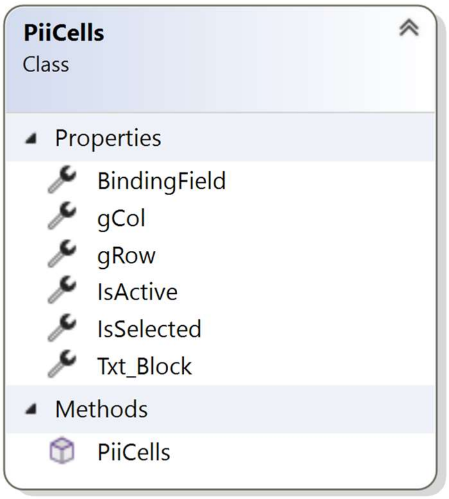
PackageElementSpecXml
Class

Properties
- AreBackground
- IsActive
- ListOfElements
- UsingActualValues

Methods
- PackageElementSpecXml

FIGURE 14. Package element specification XML class

To keep track of the data displayed in the grid at the top of the customization form, it is necessary to define a class to represent the cells in the box and their occupancy status. This class should contain properties such as cell coordinates, whether the cell is free or taken, and any other relevant information. This will allow the program to accurately track the status of the boxes and provide the necessary information to the forklift user during their operation.



PiiCells
Class

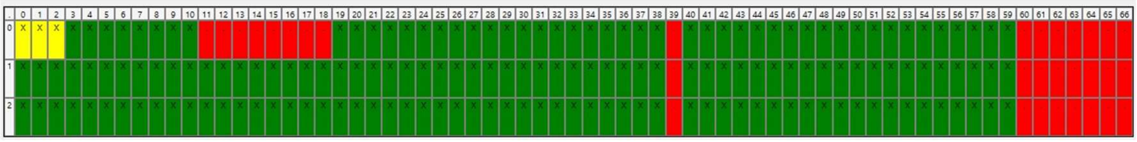
Properties
- BindingField
- gCol
- gRow
- IsActive
- IsSelected
- Txt_Block

Methods
- PiiCells

FIGURE 15. PiiCells class

When user selects one of the list boxes (Package, Loading instructions or Loading instruction's rows), a business logic function creates an object of the

PackageElementSpecXml class (as shown in Figure 14) and adds a list of items for the selected list box to the ListOfElements property. It then calls the database layer to retrieve data from the ForkliftSetting table for the selected list box and the selected forklift user. If there are saved data for this user, the ListOfElements is updated with new values. Another function is then called to create a list of PiiCells class and add necessary objects to that list based on the selected list box. This list will populate the upper grid with the cells.

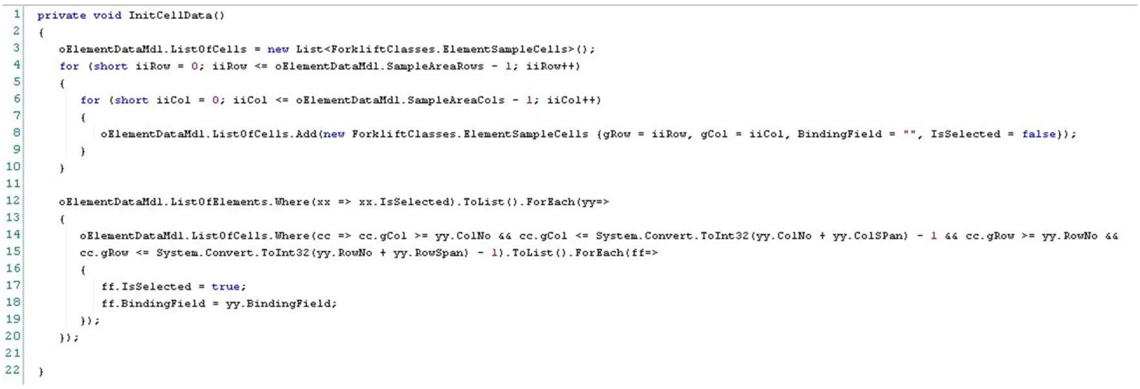


.	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
0	X	X	X	X	X	X	X	X	X	X	X	.	.	.	.	.	.	.	.	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
1	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
2	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

.	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66
0	X	X	X	X	X	.	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	.	.	.	.	.	.	.
1	X	X	X	X	X	.	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	.	.	.	.	.	.	.
2	X	X	X	X	X	.	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	.	.	.	.	.	.	.

FIGURE 16. Grid of cells

Here is an example function that demonstrates how the cells are filled in the upper grid:

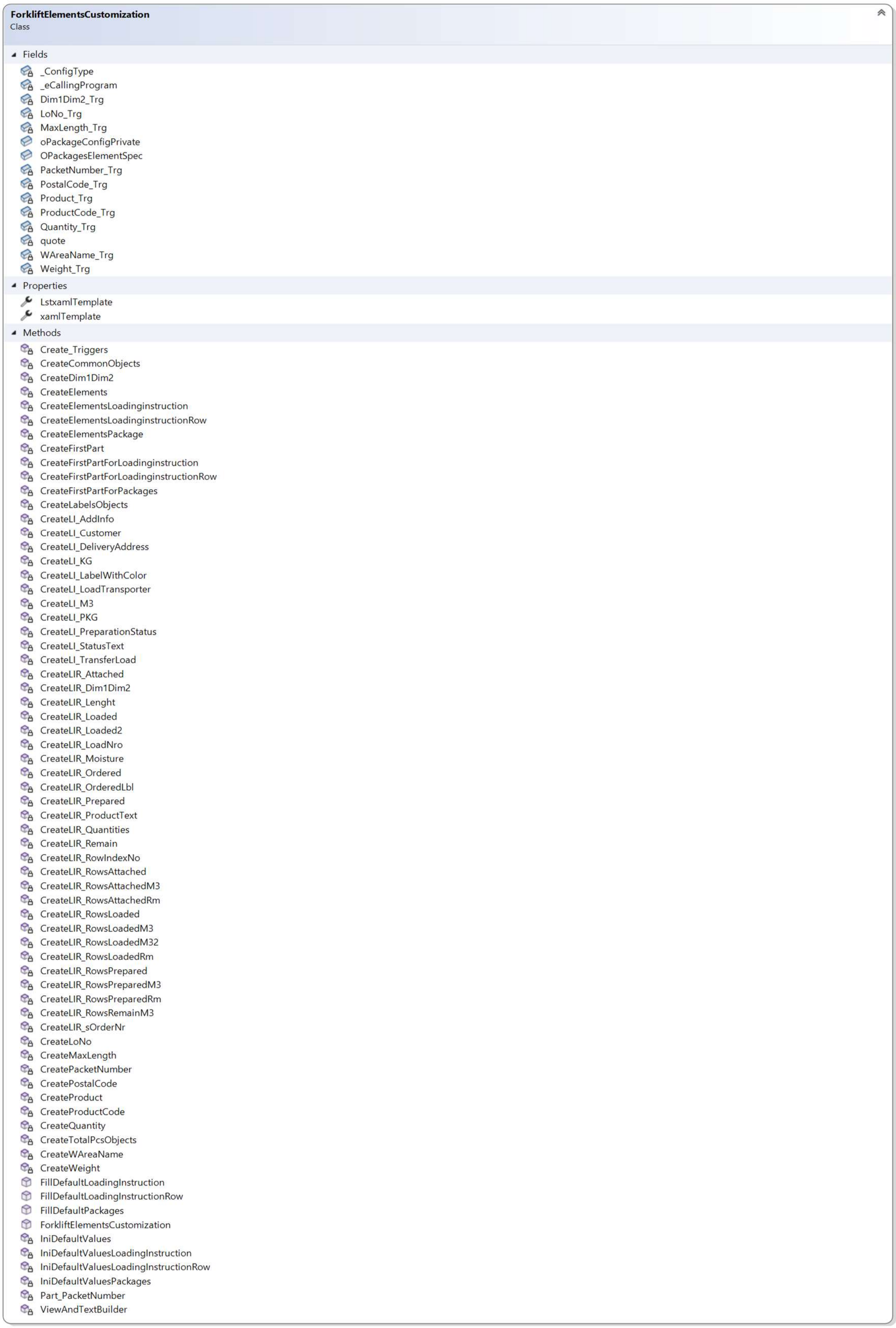


```
private void InitCellData()
{
   oElementDataMdl.ListOfCells = new List<ForkliftClasses.ElementSampleCells>();
   for (short iiRow = 0; iiRow <= oElementDataMdl.SampleAreaRows - 1; iiRow++)
   {
      for (short iiCol = 0; iiCol <= oElementDataMdl.SampleAreaCols - 1; iiCol++)
      {
         oElementDataMdl.ListOfCells.Add(new ForkliftClasses.ElementSampleCells {gRow = iiRow, gCol = iiCol, BindingField = "", IsSelected = false});
      }
   }

   oElementDataMdl.ListOfElements.Where(xx => xx.IsSelected).ToList().ForEach(yy=>
   {
      oElementDataMdl.ListOfCells.Where(cc => cc.gCol >= yy.ColNo && cc.gCol <= System.Convert.ToInt32(yy.ColNo + yy.ColSPan) - 1 && cc.gRow >= yy.RowNo &&
      cc.gRow <= System.Convert.ToInt32(yy.RowNo + yy.RowSpan) - 1).ToList().ForEach(ff=>
      {
         ff.IsSelected = true;
         ff.BindingField = yy.BindingField;
      });
   });

}
```

FIGURE 17. Filling cell function

This function takes a PackageElementSpecXml object, which contains information about the selected package element and its customization settings, as well as a list of PiiCell objects, which represent the cells in the upper grid. It then loops through each cell in the grid and sets its position and whether it is free or taken based on the customization settings. Finally, it adds the cell to the list.

After the creation of the PackageElementSpecXml object and the setting of its properties, a new object from the ForkliftElementCustomization class will be created. The constructor of this class receives the PackageElementSpecXml as an input parameter and, based on the listbox type and the list of elements, creates a data template for the given object.



ForkliftElementsCustomization
Class

Fields

_ConfigType
_eCallingProgram
Dim1Dim2_Trg
LoNo_Trg
MaxLength_Trg
oPackageConfigPrivate
OPackagesElementSpec
PacketNumber_Trg
PostalCode_Trg
Product_Trg
ProductCode_Trg
Quantity_Trg
quote
WAreaName_Trg
Weight_Trg

Properties

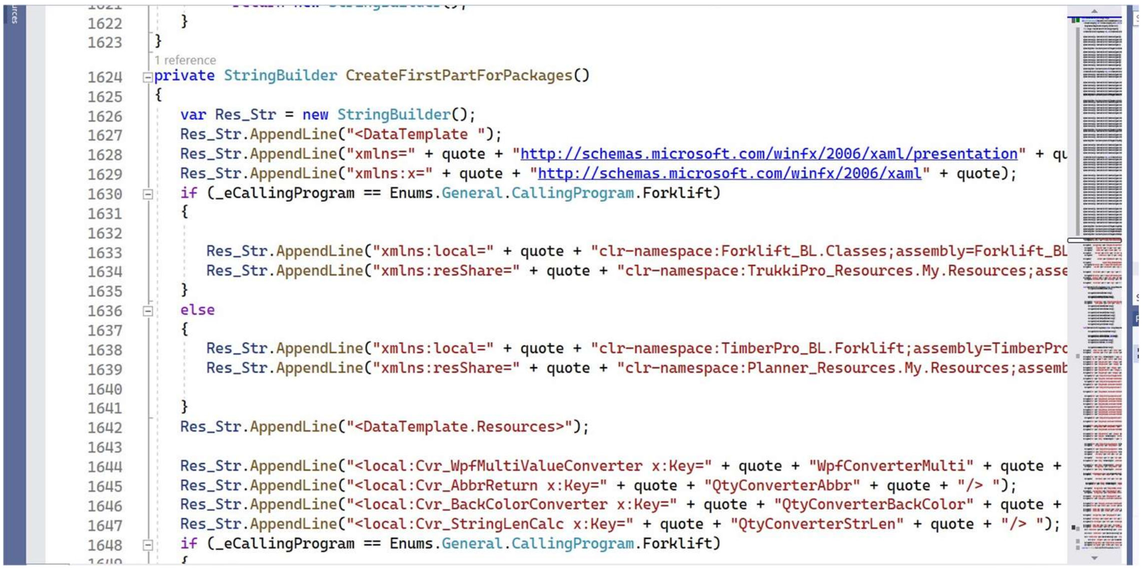
LstxamlTemplate
xamlTemplate

Methods

Create_Triggers
CreateCommonObjects
CreateDim1Dim2
CreateElements
CreateElementsLoadinginstruction
CreateElementsLoadinginstructionRow
CreateElementsPackage
CreateFirstPart
CreateFirstPartForLoadinginstruction
CreateFirstPartForLoadinginstructionRow
CreateFirstPartForPackages
CreateLabelsObjects
CreateLI_AddInfo
CreateLI_Customer
CreateLI_DeliveryAddress
CreateLI_KG
CreateLI_LabelWithColor
CreateLI_LoadTransporter
CreateLI_M3
CreateLI_PKG
CreateLI_PreparationStatus
CreateLI_StatusText
CreateLI_TransferLoad
CreateLIR_Attached
CreateLIR_Dim1Dim2
CreateLIR_Lenght
CreateLIR_Loaded
CreateLIR_Loaded2
CreateLIR_LoadNro
CreateLIR_Moisture
CreateLIR_Ordered
CreateLIR_OrderedLbl
CreateLIR_Prepared
CreateLIR_ProductText
CreateLIR_Quantities
CreateLIR_Remain
CreateLIR_RowIndexNo
CreateLIR_RowsAttached
CreateLIR_RowsAttachedM3
CreateLIR_RowsAttachedRm
CreateLIR_RowsLoaded
CreateLIR_RowsLoadedM3
CreateLIR_RowsLoadedM32
CreateLIR_RowsLoadedRm
CreateLIR_RowsPrepared
CreateLIR_RowsPreparedM3
CreateLIR_RowsPreparedRm
CreateLIR_RowsRemainM3
CreateLIR_sOrderNr
CreateLoNo
CreateMaxLength
CreatePacketNumber
CreatePostalCode
CreateProduct
CreateProductCode
CreateQuantity
CreateTotalPcsObjects
CreateWAreaName
CreateWeight
FillDefaultLoadingInstruction
FillDefaultLoadingInstructionRow
FillDefaultPackages
ForkliftElementsCustomization
IniDefaultValues
IniDefaultValuesLoadingInstruction
IniDefaultValuesLoadingInstructionRow
IniDefaultValuesPackages
Part_PacketNumber
ViewAndTextBuilder

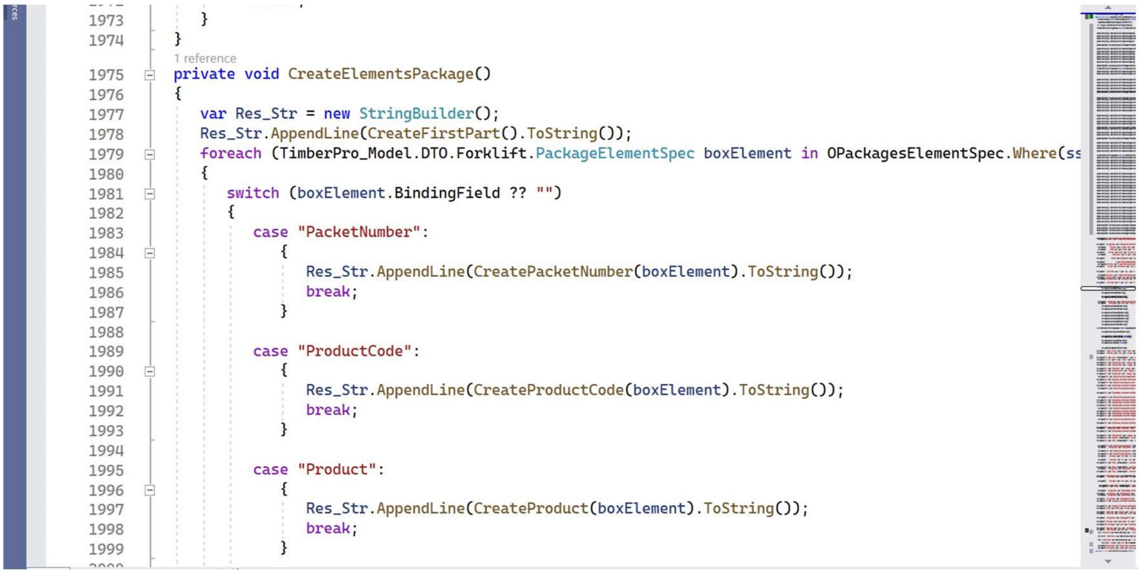
FIGURE 18. ForkliftElementsCustomization class

Here are some code snippets to illustrate how the data templates are created:



```
    }
}
1 reference
private StringBuilder CreateFirstPartForPackages()
{
    var Res_Str = new StringBuilder();
    Res_Str.AppendLine("<DataTemplate ");
    Res_Str.AppendLine("xmlns=" + quote + "http://schemas.microsoft.com/winfx/2006/xaml/presentation" + qu
    Res_Str.AppendLine("xmlns:x=" + quote + "http://schemas.microsoft.com/winfx/2006/xaml" + quote);
    if (_eCallingProgram == Enums.General.CallingProgram.Forklift)
    {

        Res_Str.AppendLine("xmlns:local=" + quote + "clr-namespace:Forklift_BL.Classes;assembly=Forklift_BL
        Res_Str.AppendLine("xmlns:resShare=" + quote + "clr-namespace:TrukkiPro_Resources.My.Resources;asse
    }
    else
    {
        Res_Str.AppendLine("xmlns:local=" + quote + "clr-namespace:TimberPro_BL.Forklift;assembly=TimberPro
        Res_Str.AppendLine("xmlns:resShare=" + quote + "clr-namespace:Planner_Resources.My.Resources;assemb

    }
    Res_Str.AppendLine("<DataTemplate.Resources>");

    Res_Str.AppendLine("<local:Cvr_WpfMultiValueConverter x:Key=" + quote + "WpfConverterMulti" + quote +
    Res_Str.AppendLine("<local:Cvr_AbbrReturn x:Key=" + quote + "QtyConverterAbbr" + quote + "/> ");
    Res_Str.AppendLine("<local:Cvr_BackColorConverter x:Key=" + quote + "QtyConverterBackColor" + quote +
    Res_Str.AppendLine("<local:Cvr_StringLenCalc x:Key=" + quote + "QtyConverterStrLen" + quote + "/> ");
    if (_eCallingProgram == Enums.General.CallingProgram.Forklift)
```

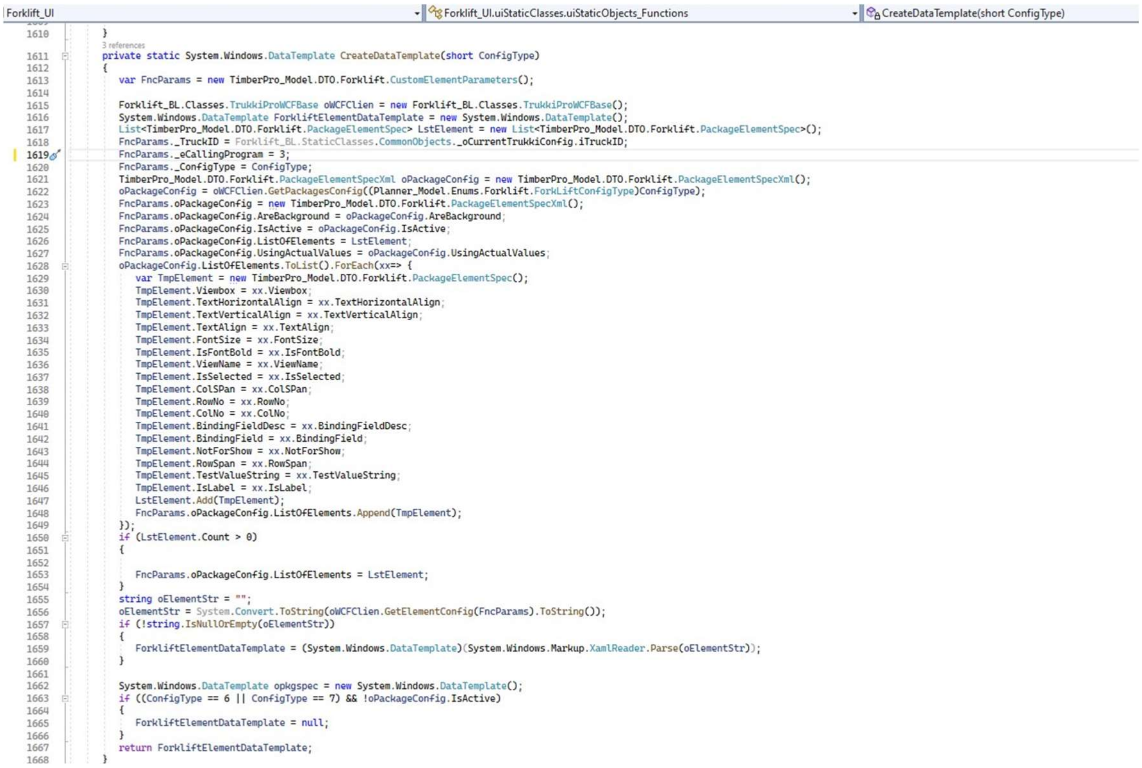
FIGURE 19. Code samples



```
        }
    }
    1 reference
    private void CreateElementsPackage()
    {
        var Res_Str = new StringBuilder();
        Res_Str.AppendLine(CreateFirstPart().ToString());
        foreach (TimberPro_Model.DTO.Forklift.PackageElementSpec boxElement in OPackagesElementSpec.Where(ss
        {
            switch (boxElement.BindingField ?? "")
            {
                case "PacketNumber":
                    {
                        Res_Str.AppendLine(CreatePacketNumber(boxElement).ToString());
                        break;
                    }

                case "ProductCode":
                    {
                        Res_Str.AppendLine(CreateProductCode(boxElement).ToString());
                        break;
                    }

                case "Product":
                    {
                        Res_Str.AppendLine(CreateProduct(boxElement).ToString());
                        break;
                    }
```

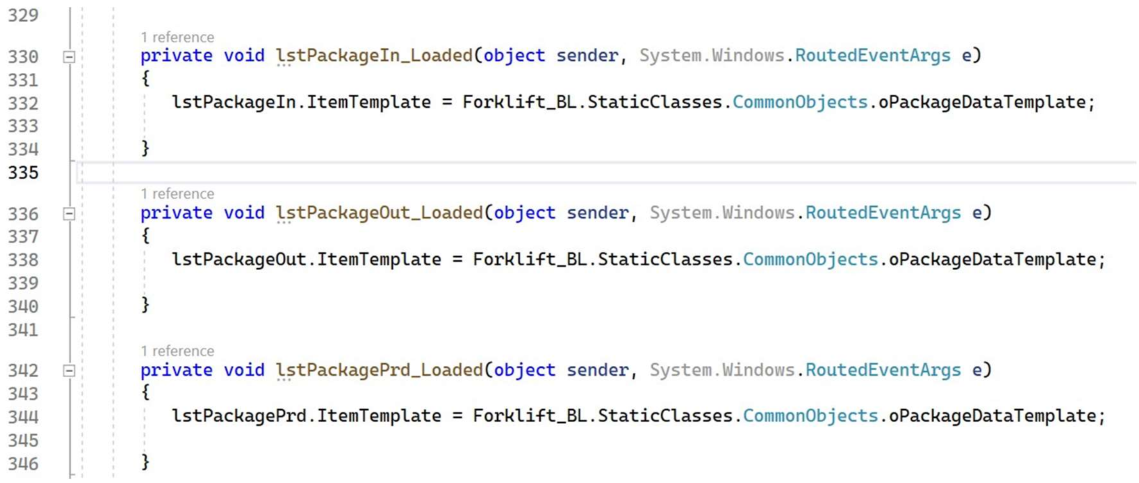
FIGURE 20. Code samples

On the forklift side, a data template must be created and assigned to the ListBox item template. This process begins when a user logs into the application. The Business layer attempts to retrieve saved data for the logged-in user from the database. If the data is not found, the list of elements is filled with default values for the properties. Subsequently, a new object from the ForkliftElementCustomization class is created (as shown in Figure 18). The constructor of this class receives the PackageElementSpecXml as an input parameter and, based on the ListBox type and the list of elements, it does not create the data template. This is because WCF has issues with serializing the data template datatype. Instead, a System.Text.StringBuilder object is created and string lines are added to it. WCF can serialize it and send it to the client. At the client-side, the data template is created and assigned to the ListBox item template.


Forklift_UI
Forklift_UI.uiStaticClasses.uiStaticObjects_Functions
CreateDataTemplate(short ConfigType)
3 references

```
    }
    private static System.Windows.DataTemplate CreateDataTemplate(short ConfigType)
    {
        var FncParams = new TimberPro_Model.DTO.Forklift.CustomElementParameters();

        Forklift_BL.Classes.TrukkiProWCFBase oWCFClien = new Forklift_BL.Classes.TrukkiProWCFBase();
        System.Windows.DataTemplate ForkliftElementDataTemplate = new System.Windows.DataTemplate();
        List<TimberPro_Model.DTO.Forklift.PackageElementSpec> LstElement = new List<TimberPro_Model.DTO.Forklift.PackageElementSpec>();
        FncParams._TruckID = Forklift_BL.StaticClasses.CommonObjects._oCurrentTrukkiConfig.iTruckID;
        FncParams._eCallingProgram = 3;
        FncParams._ConfigType = ConfigType;
        TimberPro_Model.DTO.Forklift.PackageElementSpecXml oPackageConfig = new TimberPro_Model.DTO.Forklift.PackageElementSpecXml();
        oPackageConfig = oWCFClien.GetPackagesConfig((Planner_Model.Enums.Forklift.ForkLiftConfigType)ConfigType);
        FncParams.oPackageConfig = new TimberPro_Model.DTO.Forklift.PackageElementSpecXml();
        FncParams.oPackageConfig.AreBackground = oPackageConfig.AreBackground;
        FncParams.oPackageConfig.IsActive = oPackageConfig.IsActive;
        FncParams.oPackageConfig.ListOfElements = LstElement;
        FncParams.oPackageConfig.UsingActualValues = oPackageConfig.UsingActualValues;
        oPackageConfig.ListOfElements.ToList().ForEach(xx=> {
            var TmpElement = new TimberPro_Model.DTO.Forklift.PackageElementSpec();
            TmpElement.Viewbox = xx.Viewbox;
            TmpElement.TextHorizontalAlign = xx.TextHorizontalAlign;
            TmpElement.TextVerticalAlign = xx.TextVerticalAlign;
            TmpElement.TextAlign = xx.TextAlign;
            TmpElement.FontSize = xx.FontSize;
            TmpElement.IsFontBold = xx.IsFontBold;
            TmpElement.ViewName = xx.ViewName;
            TmpElement.IsSelected = xx.IsSelected;
            TmpElement.ColSPan = xx.ColSPan;
            TmpElement.RowNo = xx.RowNo;
            TmpElement.ColNo = xx.ColNo;
            TmpElement.BindingFieldDesc = xx.BindingFieldDesc;
            TmpElement.BindingField = xx.BindingField;
            TmpElement.NotForShow = xx.NotForShow;
            TmpElement.RowSpan = xx.RowSpan;
            TmpElement.TestValueString = xx.TestValueString;
            TmpElement.IsLabel = xx.IsLabel;
            LstElement.Add(TmpElement);
            FncParams.oPackageConfig.ListOfElements.Append(TmpElement);
        });
        if (LstElement.Count > 0)
        {

            FncParams.oPackageConfig.ListOfElements = LstElement;
        }
        string oElementStr = "";
        oElementStr = System.Convert.ToString(oWCFClien.GetElementConfig(FncParams).ToString());
        if (!string.IsNullOrEmpty(oElementStr))
        {
            ForkliftElementDataTemplate = (System.Windows.DataTemplate)(System.Windows.Markup.XamlReader.Parse(oElementStr));
        }

        System.Windows.DataTemplate opkgspec = new System.Windows.DataTemplate();
        if ((ConfigType == 6 || ConfigType == 7) && !oPackageConfig.IsActive)
        {
            ForkliftElementDataTemplate = null;
        }
        return ForkliftElementDataTemplate;
    }
```

FIGURE 21. CreateDataTemplate function



```
private void lstPackageIn_Loaded(object sender, System.Windows.RoutedEventArgs e)
{
    lstPackageIn.ItemTemplate = Forklift_BL.StaticClasses.CommonObjects.oPackageDataTemplate;

}

private void lstPackageOut_Loaded(object sender, System.Windows.RoutedEventArgs e)
{
    lstPackageOut.ItemTemplate = Forklift_BL.StaticClasses.CommonObjects.oPackageDataTemplate;

}

private void lstPackagePrd_Loaded(object sender, System.Windows.RoutedEventArgs e)
{
    lstPackagePrd.ItemTemplate = Forklift_BL.StaticClasses.CommonObjects.oPackageDataTemplate;

}
```

FIGURE 22. Assigning data template to the listbox’s item template

8 TESTING

Testing the adding of a new feature to a software involves verifying and validating that the new feature functions correctly and meets the specified requirements and user needs. The goal of testing is to identify and fix any defects or issues before the feature is released to end-users.

The testing process typically involves the following steps:

8.1 Test Planning

This involves creating a test plan that outlines the testing approach, test cases, and expected results for the new feature.

8.2 Test Design

This involves creating test cases that are designed to exercise the new feature and verify that it meets the specified requirements and user needs.

8.3 Test Execution

This involves executing the test cases and observing the behaviour of the new feature, recording any defects or issues that are discovered.

8.4 Acceptance Testing

This involves performing acceptance testing with end-users to verify that the new feature meets their needs and expectations.

Overall, testing is a critical part of adding a new feature to a software system. By verifying and validating the new feature, testing helps ensure that it functions correctly, meets user needs, and does not introduce any new defects or issues into the system.

8.5 Practical approach

Once the model and process have been tested to ensure that the entire process works correctly and that the model covers all data and properties, a unit form test should be designed to test the form and all components and form actions. This test should include the following:

- Opening and closing the form
- Moving and resizing the form
- Testing component tab orders
- Changing component values
- Testing actions such as adding a new item to the list of items, copying data from another user, changing the configuration type, applying changes to the sample element, clearing values for a selected item, clearing values for all items, setting default values, saving data, and returning to the main page.

By passing this test, the main program can simulate the data template and save data in the database. The next test is an integration test, as the result of the data template creation will be used in other modules. One of the critical modules is Forklift, and the test team must ensure that all changes are visible in Forklift.

To perform this test, the test team creates a Forklift user and creates a data template for all list boxes. After running Forklift, the changes should be visible. Then they must change the data template and recreate it for the same Forklift user and run Forklift again to see the new changes. Finally, the tester must perform system testing to evaluate the entire system against the specified requirements.

9 CONCLUSION

This thesis provides a concise summary of a significant project undertaken by the company, aimed at showcasing the capabilities of WPF and .NET programming and sharing the experiences of a full-stack programmer. However, due to the need for permission from the company to display all documents, the thesis does not provide a comprehensive account of the project.

Throughout the project, the team encountered several challenges, including designing the system's performance, creating a visual structure for making changes to the display pattern, and determining the necessary models and processes. Additionally, the team faced difficulties with displaying sample elements, as data templates in Forklift may differ from those in the main program due to different environmental settings.

The use of converters, which are part of the business logic layer, presented further challenges, as they differ between Forklift and the main application. The serialization of the data template data type was also problematic, as it is not supported by WCF. To overcome this issue, the team changed the method of sending the display pattern information to the workstation, sending it as text and converting it to the data template on the workstation.

One of the most significant problems encountered during the project was restructuring the Forklift project to divide the main project into several sub-projects and convert it to the new net standard 2. The team concluded that it is best to complete the first sub-project before changing the structure of the main project or starting the sub-project entirely after changing the main project's structure.

10 REFERENCES

1. Microsoft. 2023a. What is .NET. Retrieved 3.3.2023 https://dotnet.microsoft.com/learn/dotnet/what-is-dotnet

2. Microsoft. 2023b. Windows Presentation Foundation (WPF) overview. . Retrieved 3.3.2023 https://docs.microsoft.com/en-us/dotnet/framework/wpf/overview/

3. Microsoft. 2023c. ListBox Class. Retrieved 3.3.2023 https://docs.microsoft.com/en-us/dotnet/api/system.windows.controls.listbox?view=net-6.0

4. Microsoft. 2023d. Data Templating Overview. Retrieved 3.3.2023 https://learn.microsoft.com/en-us/dotnet/desktop/wpf/data/data-templating-overview?view=netframeworkdesktop-4.8

5. Microsoft. 2023e. Windows Communication Foundation. Retrieved 3.3.2023 https://learn.microsoft.com/en-us/dotnet/framework/wcf/.

6. Consulting, Panorama. 2021. What is ERP (Enterprise Resource Planning)? Retrieved 3.3.2023 https://www.panorama-consulting.com/resource-center/erp-education-center/what-is-erp-enterprise-resource-planning/

7. Microsoft. 2023f. What is SQL? Retrieved 3.3.2023 https://docs.microsoft.com/en-us/sql/relational-databases/sql-server-what-is-sql

8. Microsoft. 2023g C# programming guide. Retrieved 3.3.2023 https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/

9. Accessibility Requirements Tool. 2021. GSA. Retrieved 3.3.2023 https://www.buyaccessible.gov/