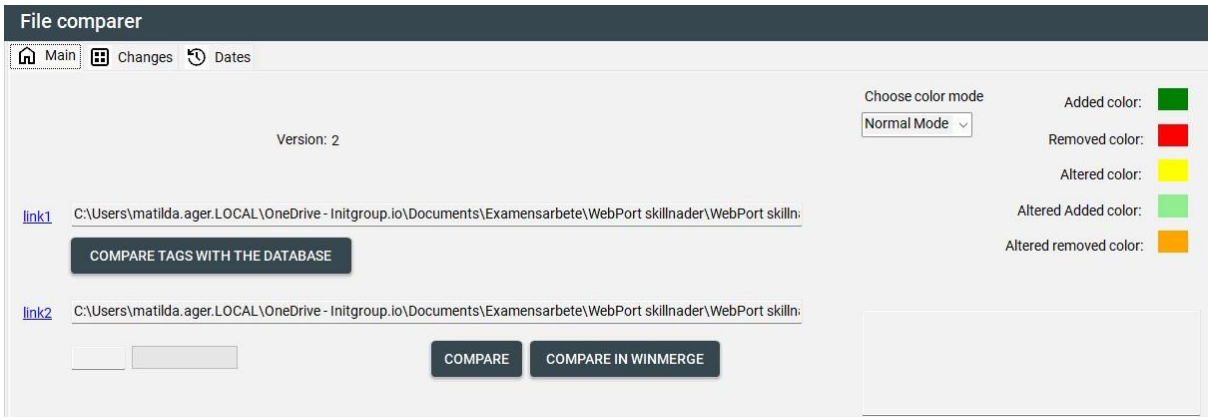


Development of a version management application for SCADA systems

Degree project report in Electrical Engineering

Matilda Ager
Marcus Bergman

Department of Computer Science and Engineering
Chalmers University of Technology
Gothenburg, Sweden 2024
www.chalmers.se

Development of a version management application for SCADA systems
MATILDA AGER
MARCUS BERGMAN

Supervisor:
Fareed Qararyah, *Chalmers University of Technology*
Andreas Karlsson, *Init Sweden AB*

Examiner:
Lars Svensson, *Chalmers University of Technology*

Department of Computer Science and Engineering
Chalmers University of Technology

ABSTRACT

This report presents a tool that compares different versions of files produced by the Web Port SCADA system and offers the ability to track, approve, or reject changes at fine granularity. This functionality is offered with a high degree of user-friendliness. The tool is developed with the purpose of addressing the limitations of pre-existing file comparison tools in functionality and user-friendliness. The developed tool offers the following features. First, an effective and memory-efficient method to compare .csv and .sqlite files. Second, a folder-wide comparison where sets of files are compared automatically and made available in the program interface and in the form of locally stored files. Third, a record of the comparisons done linking to the file locations and allowing annotation with comments. Fourth, an overview of the comparison result (additions, removals, changes) in the user interface. Fifth, the ability to validate, approve, or reject differences between the 2 versions of a file producing a third version that can be stored in a user-defined location. The finalized version of the tool has an easy-to-use, one-form interface. This interface displays all the necessary metadata including a commented history of comparisons, a table containing the done comparisons which also allows content validation, and links to the utilized storage locations. The proposed tool is implemented using .NET Framework and written in C# and uses the WinMerge application for external comparison of unhandled file types.

Keywords: WinMerge, SCADA

PREFACE

During the spring of 2024, Marcus Bergman and Matilda Ager, pursuing their degree in Electrical Engineering at Chalmers University of Technology, undertook the degree project.

We are delighted to present this thesis project, representing the culmination of our endeavors in developing a comprehensive application for version management in SCADA systems.

The primary objective of the project was to design and implement a user-friendly version management program that addresses common challenges encountered in organizing, accessing, and analyzing different file versions within large systems.

With a keen focus on usability, functionality, and accessibility, our goal was to create a solution that streamlined the process of comparing different files and enhanced user productivity.

We extend our gratitude to Andreas Karlsson, our supervisor at Init Sweden AB, for his invaluable assistance and feedback throughout the process. We are also grateful to Init Sweden AB for providing us with the opportunity to conduct our degree project within their organization. Additionally, we wish to express our appreciation to Fareed Qararyah, our supervisor at Chalmers, for his guidance and support on a weekly basis.

It is essential to recognize the scope and limitations of this project. While we endeavored to create a comprehensive solution, certain constraints such as time and resource limitations may have influenced the final product. Nevertheless, we are confident that the insights gained from this project provide a solid foundation for future enhancements and iterations.

Glossary

CSV file – Comma-separated value file
HMI – Human-machine interface
IDE – Integrated development environment
MSI – Microsoft Software Installer
PLC – Programmable logic controller
RTU – Remote terminal unit
SCADA – Supervisory control and data acquisition
SQL – Structured Query Language

Table of Contents

List of Figures

1 Introduction

This introductory chapter explores the project's background, establishing the foundation for its purpose and motivation. Additionally, it presents the initial formulation of the project, the outline of the planned goals, research question, and anticipated limitations.

1.1 Background

Init Sweden AB is a consulting company, that works with control and monitoring systems in the infrastructure and in the real estate sector [1]. Monitoring and control of machines in the industrial, real estate, and infrastructure sectors are an essential part of Init's work. SCADA systems are used to accomplish such monitoring and control tasks [2]. The automation of these systems generates a significant amount of data which is continuously updated by the system itself and by various users [3]. This large volume of generated files is modified across various versions of the system. When there is a need to compare and validate different versions of the generated files by the system, the task proves very difficult due to the sheer amount of data. Given the tedious nature and vast scale of the task, employing computer programs for comparison proves to be more effective than relying on human effort [4]. In addition to enhancing efficiency, this approach mitigates the risk of human errors while facilitating expedited data analysis. This enables individuals to concentrate on other significant aspects of a project.

The pre-existing file comparison tools, like WinMerge, often lack usability when it comes to comparing large, tabulated files [5]. The lack of effective file comparison tools is a crucial issue due to the widespread usage of control and monitoring systems across different fields and industries, specifically due to the significant amount of data they generate. However, pre-existing file comparison tools do not offer a user-friendly way to compare large volumes of tabulated data, which is exacerbated by this tabulated data often exceeding displayable sizes. Additionally, accessing overview data is neither straightforward nor efficient. These issues hinder data analysis tasks and lead to longer response times for identifying system modifications [5].

To enhance data safety and user-friendliness, and overcome the usability shortage in pre-existing tools, Init aims to introduce a comparison and search tool. This tool would enable displaying changes between different versions of files and validating the latest data version and thereby mitigating the risk of errors involving manual documentation and enhancing efficiency. This tool should facilitate intuitive identification of changes, additions, and removals - using color coding. Additionally, it should provide data filtering options to enhance navigation for the user.

1.2 Purpose

The purpose of the project is to introduce a file comparison tool that facilitates comparing the significant amount of data generated and continuously updated by the SCADA system itself and by various users.

The existing file comparison tools, like WinMerge, can already compare files, but it is hard for a human to read them (figure 1) and understand the changes of a *comma-separated value file* (.csv file). This task could be facilitated if there was a tool to compare multiple .csv files at once and display the result in a user-friendly interface. This project's purpose is therefore to create a tool that addresses the problem with usability found in pre-existing tools without sacrificing functionality. Allowing the user to compare different types of files in an intuitive environment and facilitating data analysis.

This project is therefore about how to write a program to compare .csv and .sqlite files.

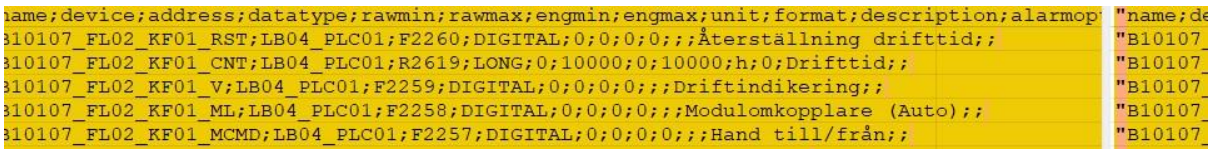


Figure 1. The result in WinMerge when comparing .csv files.

1.3 Goals

The goals of the project are the following:

- Compare data sources, including .csv files and .sqlite files (databases), marking encountered differences.
- Compare groups of files, which can consist of .csv or .sqlite files.
- Create a user-friendly interface (Either executable window or Excel interface) that displays the data by color coding it.
- Enable fine-grained modifications (marked by a specific color).
- Create a history of when something was changed containing relevant data.

1.4 Research questions

The questions that were handled under the project were the following:

1. What is the most reasonable and effective way to compare 2 .csv-, 2 .sqlite- and .csv with .sqlite files?
2. How can the comparison be expanded from single files to folder-wide comparisons and what is the most convenient way to keep track of the results in that case?
3. How can the visual interface present the information effectively while also being user-friendly?
4. What is a reasonable time for the comparison to occur and how resource-intensive should it be?
5. How should the data be validated and how should additions, removals and changes be chosen by the user to keep?
6. How can the program be designed to be general enough so that it can be used in other control systems as it further develops?

1.5 Limitations

The technical limitations of the project were the following:

- The project was coded in C#, which means certain features present in other programming languages that could simplify the project were not utilized. For example, using C++ could have resulted in a faster application due to its smaller memory footprint [6].
- The developed program was designed to function primarily with Web Port. This meant that some SCADA-systems that use other file types were not handled within the tool and would be redirected to WinMerge.
- There was a lack of direct integration with Web Port, which meant the program only analyzed the produced version files. Direct integration could have led to a better solution.
- The absence of a concrete sorting standard led to data misalignment, causing the program to sometimes produce false positives.

2 Method

This section details the preparatory stages undertaken prior to commencing the application development.

The project will be developed in C# as it is the primary language used at Init. The project will be developed with consideration for the SCADA system's storage format, which are .csv and .sqlite files. Consequently, these two file types will be the focus of the comparisons.

The following list outlines the initial planned steps for the program to achieve data extraction, comparison, formatting, storage, and output. Consequently, the workflow was structured as follows:

1. **Extraction phase:**
   In this phase the data is extracted from the .csv or .sqlite files produced by the SCADA system and stored as an intermediate C# representation arrays.
2. **Sorting phase:**
   In this phase the data is sorted in alphabetical order.
3. **Comparison phase:**
   In this phase the data of two files is compared in the following manner:
   a. It reads the space of the two files, if rows have been added or removed.
   b. The results of the comparison (changes) are temporarily stored in the memory keeping track of the old value, new value as well as the position in the data table.
4. **Output formatting phase:**
   In this phase colors are used to facilitate modification and data analysis. The user selected colors are applied to the data in the output table by using the stored comparison results.
5. **Storing data phase:**
   In this phase the resulting output file will be stored in a user-selected location.
6. **Output result phase:**
   In this phase after generating all the output files, a conclusive main file is created. This file includes links to each comparison result and offers a summary of the content, detailing the number of additions, removals, and changes.
7. **Make available in user interface phase:**
   Connects the primary file to the user interface to enable interaction within the interface.

To achieve this planned workflow, it would first be necessary to go through the following points:

- Look at similar programs [7] to get an idea of the user interface for functionality and note points of improvement in user friendliness.
- Obtain a surface level understanding of C# programming conventions and its memory allocation to ensure memory efficiency and reduce redundancy during development.
- Plan for the sort of user interface the data needed to be adapted towards.
- Identify the functional and nonfunctional objectives and give them a priority level, where nonfunctional objectives are ones that can increase usability but are not a requirement.
- Define how to handle additional sorts of data that are not in the form of the specified table-file types (.csv and .sqlite).

With the general workflow delineated and the program's structure outlined, the subsequent phase involves acquainting oneself with the various software components described in the technical background. Then after the familiarization phase the next step is the acquisition of test data generated by the real SCADA system and provided by Init in the form of version files is imperative. These files will be instrumental in testing the different data management functions, facilitating realistic assessments of runtime performance and memory usage.

3 Technical background

Init employs Web Port *supervisory control and data acquisition* (SCADA) in its control and monitoring systems. SCADA is a system for overseeing processes, with further details provided in section 3.1. Web Port SCADA, a web-based software, manages various project files, such as those related to production lines or property alarms. SCADA systems use *remote terminal unit* (RTU) systems that are suitable when the industrial process is small-scale and more appropriately used as an alternative to programmable logic controllers (PLC). However, RTU systems prove inadequate for larger systems due to limitations in capacity, size, and complexity [8]. Therefore, Init opts to utilize Web Port SCADA for such endeavors. Multiple users can modify data in Web Port SCADA without version management access, posing a risk of errors and complicating error tracking [9]. This task would be quite difficult and time-consuming for a human to do [4]. Considering the lack of usability, it was decided to address this issue in partnership with Init by developing a safe tool to check Web Port SCADA data. The developed application would allow users to control file changes and validate them based on user-defined parameters, thereby enhancing security and efficiency in the control process. The following section gives a technical background of the project and describes the programs, concepts and tools used in the project.

3.1 SCADA

SCADA systems are integrated software and hardware solutions designed to control numerous devices in Industrial and real estate settings, which would be too challenging for a human to manage by hand due to the complexity of the task. These devices controlled by SCADA fulfill various tasks such as energy consumption monitoring, HVAC management, and real-time production line supervision [10]. SCADA systems control large amounts of data which is sent from the overseeing SCADA system to either a local RTU or a PLC which controls the smaller sub systems. Since it is complicated for a human to read the data, it is interpreted in a readable manner and then transferred to a *human-machine interface* (HMI). SCADA systems are used in a variety of industries industry, such as the oil and gas industries, transportation industries, energy management industries, etc. [11].

Init works with various collaborators in Gothenburg and uses SCADA to have an overview of different devices in large systems [12]. In this project, the web-based SCADA system called Web Port will be used to collect data. The collected data will then be compared to identify any changes between the files.

3.2 Web Port

Web Port is a web-based SCADA system that collects and measures data and informs when something has been changed beyond certain user-defined parameters, but it does not show what has been changed within settings in a historical manner. It relates to an HMI so that the user can set their own parameter values as the standard. Meaning that when a measured value differs from this standard Web Port will send a message to the owner [9].

3.3 Programming languages

The programming languages that are used in this project are presented below.

3.3.1 SQLite

Structured Query Language (SQL) is a programming language designed to manage and manipulate relational databases. A relational database is a type of database that organizes data from different sources into tables [13]. SQLite is a free SQL database handler software library which is small and easy to integrate, due to not being server-reliant [14], and which allows programming in C#.

3.3.2 C#

C# is an object-oriented, and type-safe programming language that enables developers to build secure and robust applications within the .NET Framework described in section (3.5). C# has its roots within the C family of languages, so its syntax is quite like C, C++ and Java [15].

3.4 WinMerge

WinMerge is an open-source differencing and merging tool for Windows, widely used for code comparison, file synchronization, and text comparison. It allows users to check for differences not only within individual files but also across entire folders, facilitating the merging of different versions into a single file or folder. Additionally, WinMerge offers features such as syntax highlighting for various programming languages, customizable directory filters, and 3-way comparison [7].

3.5 Visual Studio Community

Visual Studio Community is a free *integrated development environment* (IDE) provided by Microsoft. Visual Studio Community is well suited for development within the .NET Framework and allows for a variety of languages including C#. It also supports a visual designer for interface building, templates, and a variety of extensions and add-ons to support development on the platform [16].

3.6 .NET Framework

.NET Framework is a software framework developed by Microsoft that provides a comprehensive platform for building various applications. It is designed to handle object-oriented programming concepts. It also makes sure that the code utilized within the .NET Framework can communicate with any other code within the framework regardless of language; this feature is known as language interoperability [17]. The .NET Framework also includes a pre-built class library as well as plugins such as NuGet that allow easy integration of ready-built functions and data types [18].

3.7 Installed NuGet packages

To handle all the code, it was necessary to install all of the packages that are described in this chapter.

- The CsvHelper is a NuGet package used for reading and writing .csv files, with configurable delimiters [19].

- EPPlus is a NuGet package used in .NET to read and write Excel files as well as format them with the help of a class called ExcelPackage. It holds all the variables and format for individual cells that are either read from or saved to a .xlsx file [20].

- The Microsoft-WindowsAPICodePack-Shell NuGet package is only used to interact with the Windows shell. It is the user interface of the Windows operating system and includes elements like the taskbar, desktop, etc [21].

- Spire.XLS is used to create and manage worksheets, workbooks, and writing to .xlsx files. In the program, it is used to apply content filters to the excel tables [22].

- The System.Data.SQLite NuGet package is used for integrating SQLite database functionality into the .NET application [23].

- The MaterialSkin.2 NuGet Package is designed to enhance the visual aesthetics and user experience of Windows Forms applications by drawing inspiration from Google's Material Design principles. This modern and clean design was chosen to improve readability and reduce visual strain while maintaining an informative interface [24].

3.8 Extensions

The extensions used in the development environment that differ from the pre-installed ones are the following.

3.8.1 Microsoft Visual Studio Installer Projects 2022

The Microsoft Visual Studio Installer Projects 2022 extension enables the creation of installer projects which allows for .NET Framework to be exported as an installer file with the termination .msi [25].

4 Implementation

This segment delves into the implementation phase of the project, detailing the timeline, re-evaluation process, and a comprehensive description of the program. This section provides a structured overview of the chronological steps taken, the iterative re-evaluation to ensure alignment with project goals, and an in-depth look at the program's functionality.

4.1 The timeline

The following segment details the steps taken during the project to achieve the development of the functional tool.

The project was done in the order that is shown below:

1. Read about SQLite, .NET Framework, and WinMerge to get an understanding of how to use them within the scope of the project.
2. Create a function that converts the .sqlite file to a .csv file.
3. Create a function that compares 2 .sqlite files, .sqlite with .csv file or 2 .csv files.
4. Create a function that compares groups of .csv or .sqlite files in a folder with another group.
5. Create a function that utilizes the comparison result to color code a output document.
6. Make the output document able to be validated by applying a color selection on the output document, facilitating modification in a visually readable manner.
7. Create a function that opens WinMerge to compare the files in two folders.
8. Finish the interface and make it user-friendly.
9. Make a historic list of every comparison.
10. Create a validation button.
11. Create an installer for the software.

Since there are two types of data-files the timeline was divided into two separate tasks, .csv table management and .sqlite table management. When the two parts were finished it was put together into a main program and implemented into the user interface.

4.2 The user interface

This section gives an overview of the finalized tool and shows what has been done during the development process.
The final version features three tabs: "Main" (Figure 2), "Changes" (Figure 3), and "History" (Figure 4). This interface design was inspired by analyzing similar applications and incorporating the best elements to suit the specific needs of this application.

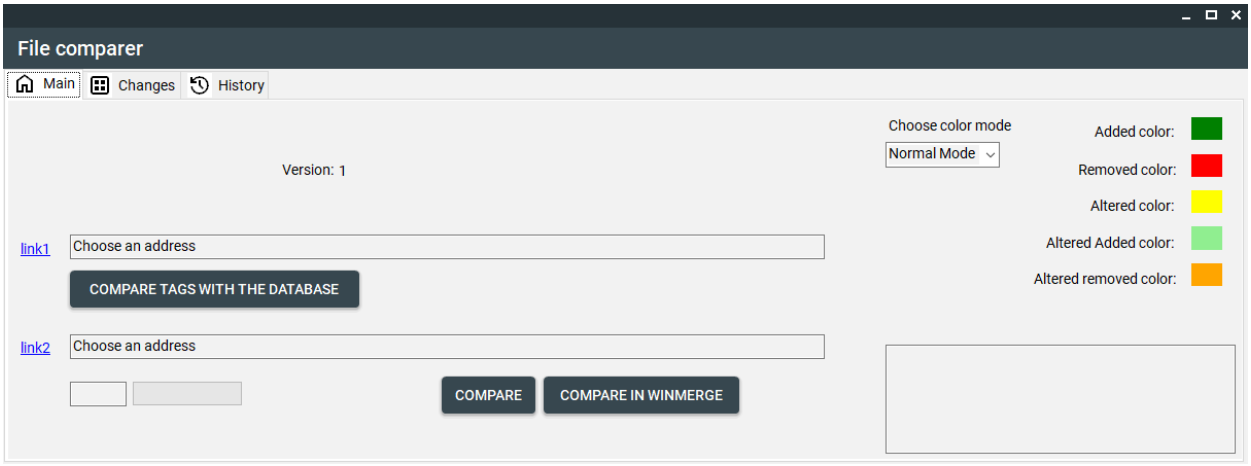


Figure 2. The "Main" tab, where the user has three ways to do the comparison.

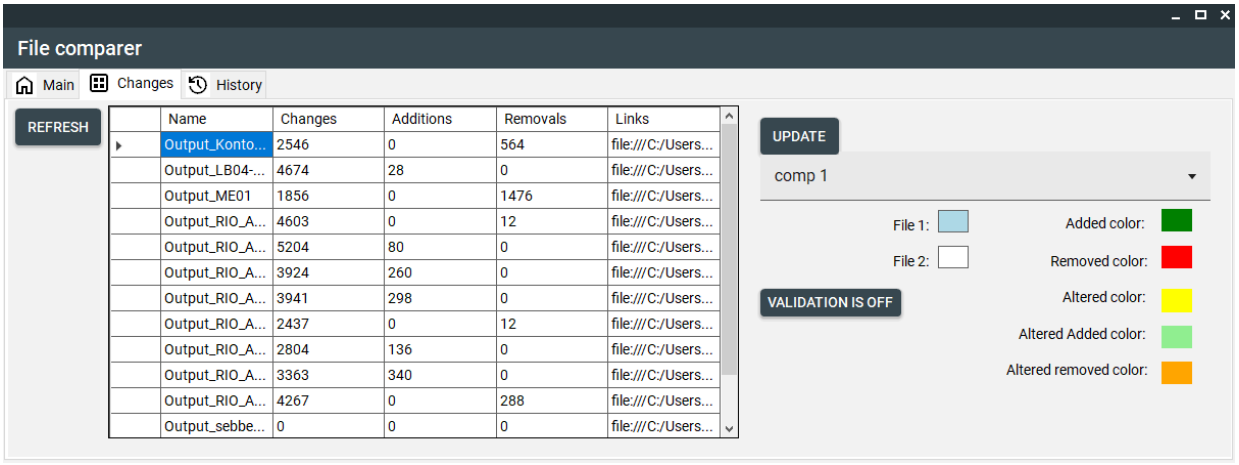


Figure 3. "Changes" tab, shows the file name, changes, additions, removals and links to the file which is clickable.

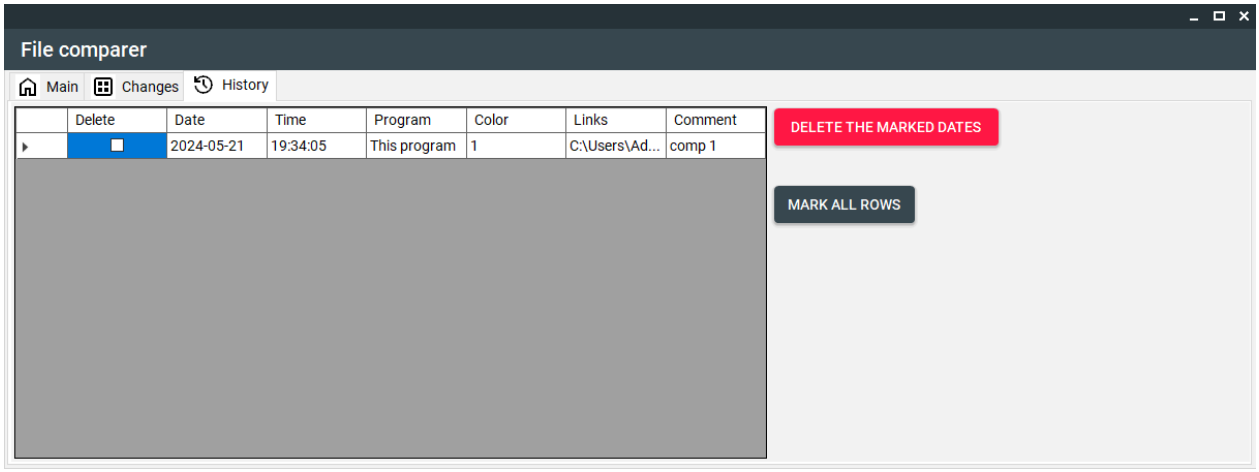


Figure 4." History" tab shows when the changes were made, where it is possible to delete and mark all the CheckBoxes.

With these three tabs, users can seamlessly navigate through various functionalities:

1. In the main tab, users can:
    - Select data sources presented in folder format.
    - Customize the user experience with color modes.
    - Choose different types of comparisons.
    - Obtain an overview of the entire process.
2. In the changes tab, users can:
    - Review past comparisons effortlessly.
    - Access comparisons directly at the folder location via the interface.
    - Validate changes within the data.
    - Gain an overview of each comparison's details.
3. In the History tab, users can:
    - View crucial data such as comparison date and time.
    - Review the type of comparison and color palette used.
    - Access the storage location of comparisons.
    - Add comments to comparisons for easier tracking.

With a foundational understanding of the basic functionality of each tab, the next step is to delve deeper into examining each one individually and explore how they relate during use.

4.2.1 Comparison types

It is in the main tab the user selects between three distinct comparison types with the buttons shown in Figure 5.

- Number 1 in Figure 5 marked as “COMPARE”, compares the .csv files contained in the folders from the addresses in the fields. It accomplishes this by comparing the data dimensions and examining the content of individual cells. A detailed description of this process is provided in the methodology chapter.

- Number 2 in Figure 5 marked as “COMPARE IN WINMERGE”, forwards the content of the address fields to the external program WinMerge that will then do the comparison, this makes it possible to check unhandled file types (not .csv or .sqlite) that may be found within the contents of the addresses.

- Number 3 in Figure 5 marked as “COMPARE TAGS WITH THE DATABASE”, does a comparison of the content found within the first address field by finding the .sqlite database and comparing it with the encountered .csv files in the same manner as the “COMPARE” version.

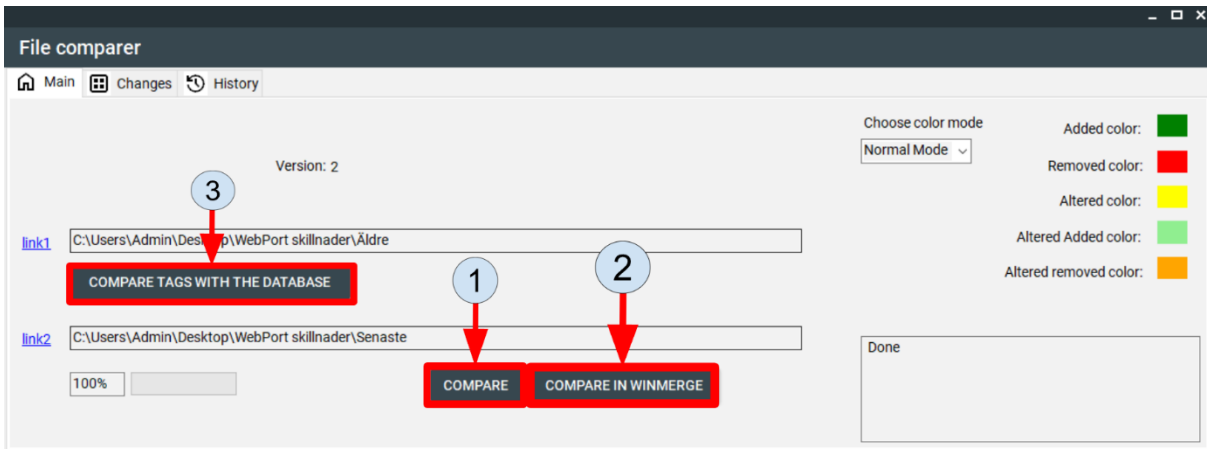
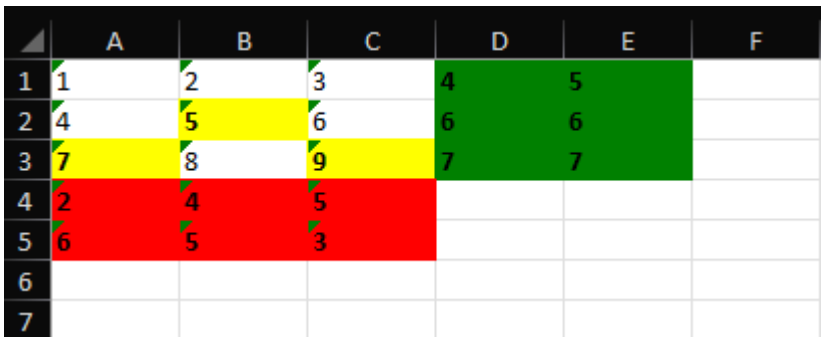


Figure 5. Buttons of the 3 different comparison types.

4.2.2 Color selection

The output of the comparisons described in the previous segment are formatted as an .xlsx files (figure 6), where colors are used to show what the differences between the two file versions are.



	A	B	C	D	E	F
1	1	2	3	4	5	
2	4	5	6	6	6	
3	7	8	9	7	7	
4	2	4	5			
5	6	5	3			
6						
7						

Figure 6. Color coded test data in which green=added, red=removed and yellow=changed data (Output).

Within the main tab still, the user can choose which color mode they want to use (figure 7 & figure 8). Which defaults to the first one if none is chosen and saved between uses if one is selected. This is done by the program using settings (figure 9) to remember the previous values, making the experience more customizable without the need to repeatedly adjust the color palette.

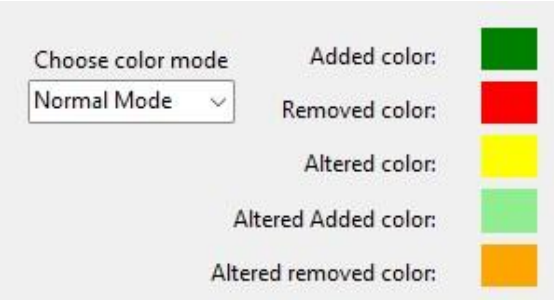


Figure 7. The normal colors that will be used.

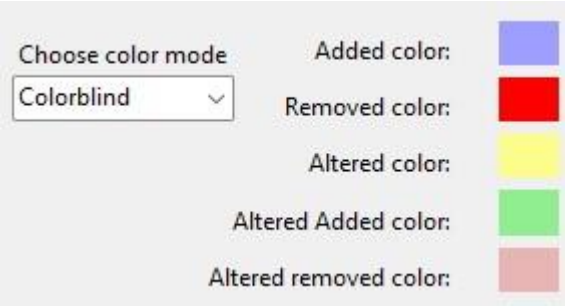


Figure 8. The colors that will be used if the user is colorblind.

Name	Type	Scope	Value
comboBoxValue	string	User	Not Set
version_x	int	User	1
address1	string	User	Choose an address
address2	string	User	Choose an address
daySettings	string	User	0
timeOfDay	string	User	0
programType	string	User	0
linkAddress	string	User	
specialID	string	User	

Figure 9. The settings to save values after every comparison.

4.2.3 Additional functions within the main tab

For every type of comparison, the user is required to write an ID (figure 10) every time, else the comparison will not be done. This is to ensure that the user can effectively track their comparisons and to enhance readability for other users who review the changes.

Figure 10. Where the user writes the ID.

When the user compares the files in two folders or the database with tags then the user must select a folder where they want to save the output every time. Then the program reads if it is empty or not. When the folder is not empty then the user will be prompted if they want to empty the selected folder (figure 11). If the user chooses “No” or cancels at Figure 11 then Figure 12 comes up to get a reminder for the next time.

Figure 11. Prompt to empty the folder.

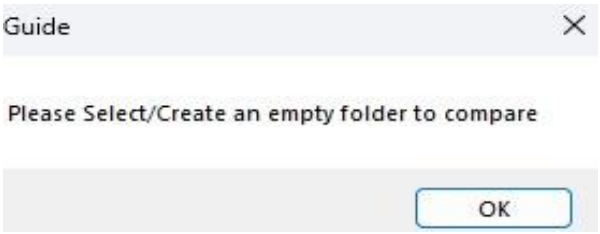


Figure 12. Reminder prompt.

Once the folder is cleared and an identifier is chosen, the comparison occurs, and the process can be tracked with help of the progress bar as shown in Figure 13
The progress bar is needed, due to the large number of files that may be compared, to make the interface more user-friendly and clearer when data is being processed.

Figure 13. Shows how long the process/comparison has come.

4.2.4 Functionality within the “Changes” tab

The second tab named “Changes” (figure 3) shows what has been added, removed, or changed in every tag. The comparison (stored as an .xlsx file) can be opened by clicking on the link. In the .xlsx file, the differences are marked with colors as shown in (figure 14). To make it easier for the user to understand the colors, there is also a color description in the “Changes” tab.

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	name	device	address	datatype	rawmin	rawmax	engmin	engmax	unit	format	description	alarmoptions	trendoptions
2	name	device	address	datatype	rawmin	rawmax	engmin	engmax	unit	format	description	alarmoptions	trendoptions
3	LB04_127665_ST71_01_PV	AS03-RC01	17@40057	UINT	0	4096	0	1000	Pa	0	Ärvärde tryck		
4	LB04_107768_ST71_01_AL	AS03-RC01	1@40033.6	INT	0	0	0	0		0	Larm fel ändläge		
5	LB04_127666_ST71_01_PV	AS03-RC01	19@40057	UINT	0	4096	0	1000	Pa	0	Ärvärde tryck		
6	LB04_107768_ST71_01_AL1	AS03-RC01	1@40033.3	INT	0	0	0	0		0	Spjäll fel vid motionering		
7	LB04_131012_ST71_01_PV	AS03-RC01	36@40057	UINT	0	4096	0	1000	Pa	0	Ärvärde tryck		
8	LB04_107768_ST71_01_V	AS03-RC01	1@40033.4	INT	0	0	0	0		0	Indikering Motionskörning		
9	LB04_137659_ST71_01_PV	AS03-RC01	25@40057	UINT	0	4096	0	1000	Pa	0	Ärvärde tryck		
10	LB04_107768_ST71_01_V0	AS03-RC01	1@40033.8	INT	0	0	0	0		0	Spjäll indikering stängt		
11	LB04_137867_ST71_01_PV	AS03-RC01	28@40057	UINT	0	4096	0	1000	Pa	0	Ärvärde tryck		
12	LB04_107768_ST71_01_V1	AS03-RC01	1@40033.9	INT	0	0	0	0		0	Spjäll indikering öppet		
13	LB04_137959_ST71_01_PV	AS03-RC01	27@40057	UINT	0	4096	0	1000	Pa	0	Ärvärde tryck		
14	LB04_107768_ST71_02_AL	AS03-RC01	2@40033.6	INT	0	0	0	0		0	Larm fel ändläge		
15	LB04_138071_ST71_01_PV	AS03-RC01	30@40057	UINT	0	4096	0	1000	Pa	0	Ärvärde tryck		
16	LB04_107768_ST71_02_AL1	AS03-RC01	2@40033.3	INT	0	0	0	0		0	Spjäll fel vid motionering		

Figure 14. Color marked data within the .xlsx file (Normal Mode).

To choose which comparison to review the user must these steps:

1. Click the button named "UPDATE" (1 in Figure 15).
2. Select the desired comparison from the drop-down menu below the "UPDATE" button.
3. Click the button "REFRESH" (2 in Figure 15) to see the result in the central part of the window.

There is also a button labeled “VALIDATION IS OFF”, 3 in Figure 15 that changes to “VALIDATION IS ON” 1 in Figure 16 when pressed. This button is used as a toggle for the validation function, and which replaces the click to open function of the “Links” column.

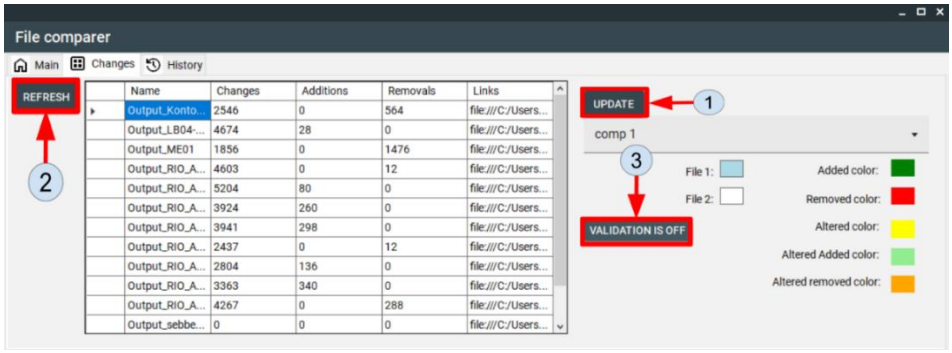


Figure 15. Buttons within the “Changes” tab.

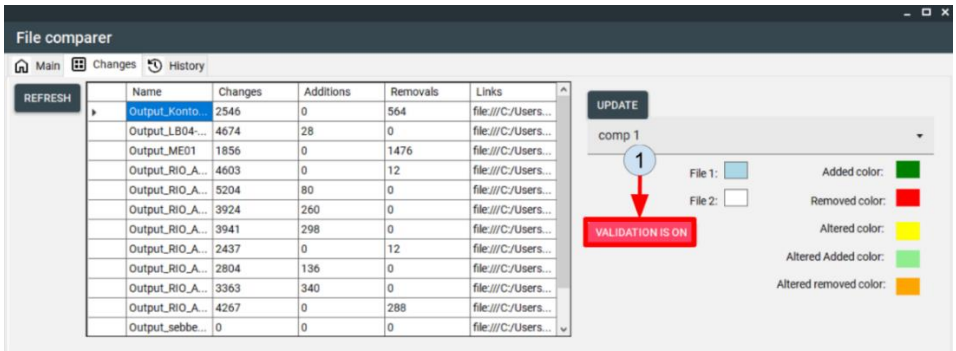


Figure 16. Appearance of button 3 while toggled (validation function enabled).

4.2.5 Functionality within the “History” tab

The third tab called “History” (figure 4) shows the date and time of the day when the comparison was conducted, along with a comment ID, a link to the folder where the comparison output is saved, and a column for users to choose which rows the user wants to delete.

It has two buttons, the button that is called “MARK ALL ROWS”, 1 in Figure 17 marks all rows since it can be user-friendly to mark all the CheckBoxes at one time. The second button called “DELETE THE MARKED DATES”, 2 in Figure 17, deletes those rows where the CheckBox has been marked and it also empties the folder where the output to that row was saved. The delete button is red to make it noteworthy and to indicate that once this deletion is done it cannot be reverted which is also assisted by a warning pop up if the button is pressed.

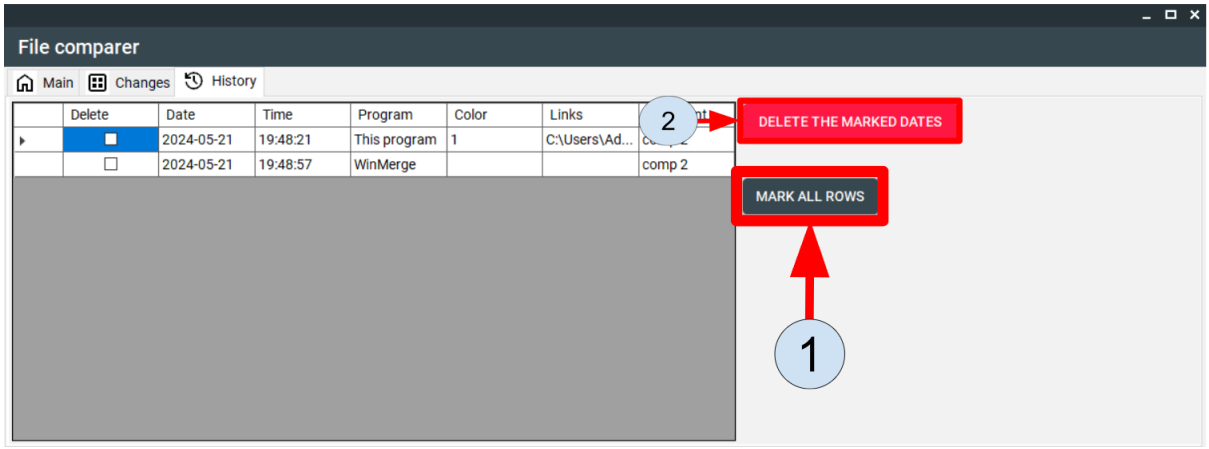


Figure 17. Buttons in the “History” tab.

The column called “Links” shows the file path to the folder where the output has been saved, but if WinMerge has been used it will be empty since this comparison is not saved within a folder.

The column called “Program” shows where/what kind of comparison that has been done. Either “WinMerge”, “DB vs tags” (database compared with the tags that are in the same folder) or “This program” (the files in two folders by using this program).

In the “Main” tab the user chooses a folder and if that address has been used earlier, it will be deleted from the history list since the old version is no longer of interest. There is also a column called “Color”; it shows 1 when the user has chosen normal mode and 2 when colorblind.

4.2.6 Development process

The first version had four forms, one main form (figure 18) where the user could determine what they would compare (the three buttons). Then once the desired comparison was chosen it would redirect and open a new form (figure 19, figure 20 & figure 21) depending on what the user wanted to compare. If a database was involved the user had to write the tag's name (figure 19 & figure 21), so it just searched for that tag in the database.

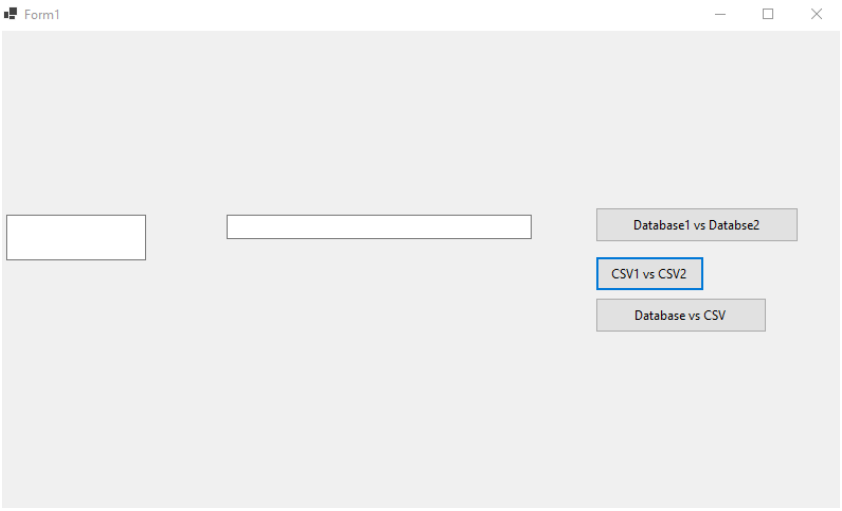


Figure 18. The first version of the main form (Form 1).

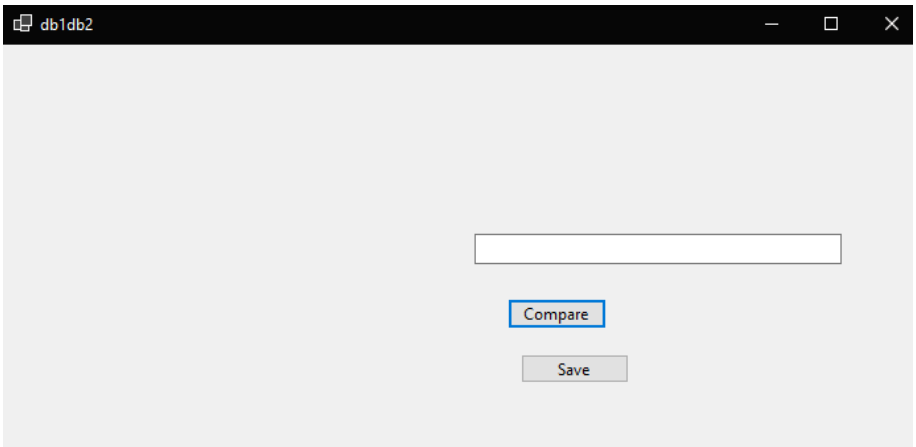


Figure 19. The first version of the form (db1db2) compares two database tables.

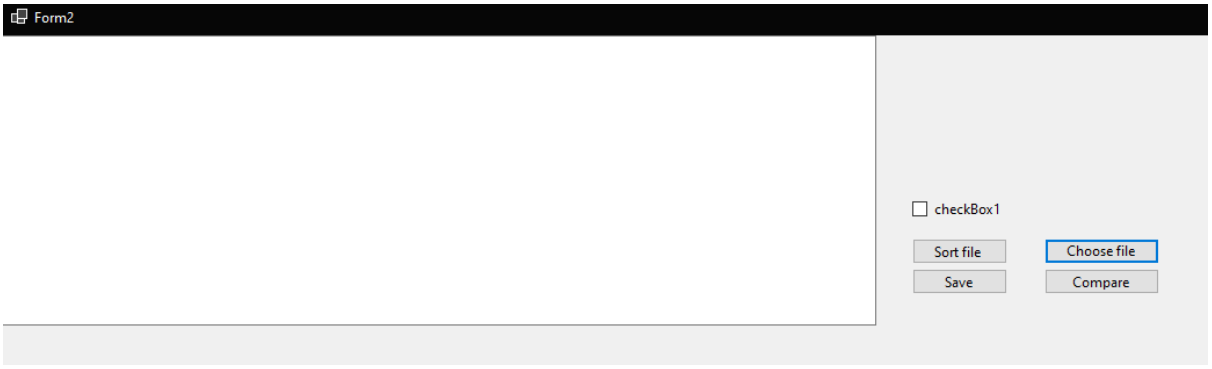


Figure 20. The first version of the form (Form2) compares two .csv files.

Figure 21. The first version of the form (Form 3) compares one database with a .csv file.

It became evident that there were too many buttons and forms in the first version which was good for testing but made the interface feel cluttered and unintuitive. Therefore, in the second version, the interface was simplified into two forms (figure 22 & figure 23). The user only needed to choose an address and then the program read which type of file it was (either .csv, .sqlite, or something else), if it was not a .csv or .sqlite file it opened WinMerge automatically.

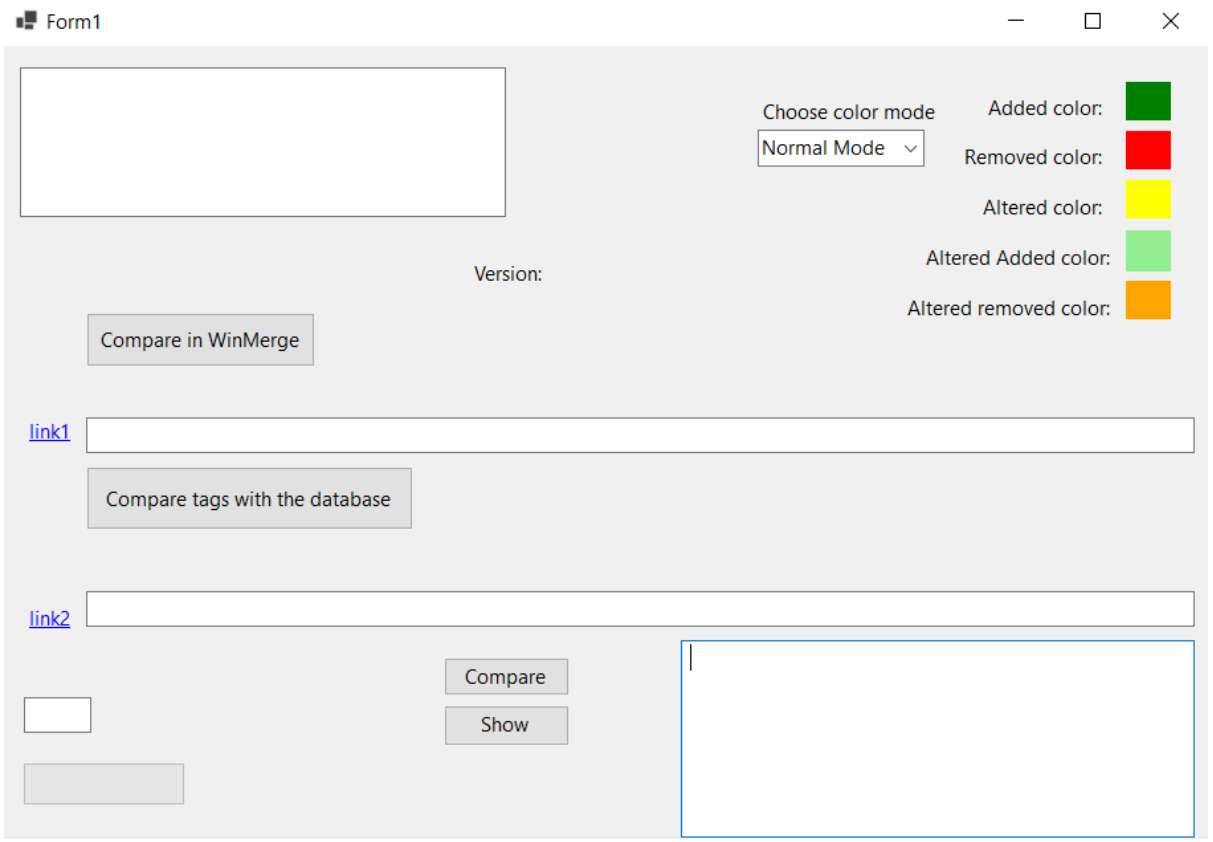


Figure 22. The main form (Form 1).

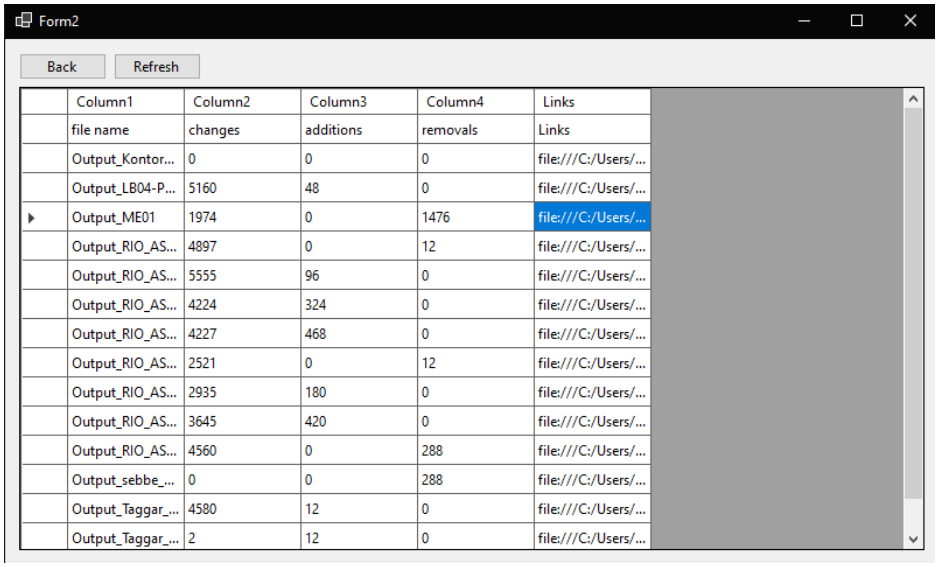


Figure 23. Form2 shows how much has been added, deleted, or changed.

Instead of only comparing one tag or file at a time it compared all files in the selected folder since it would be slow for the user to compare one tag or file at a time. The button called “Show” opened the second form (figure 23).

It was possible to choose two types of modes, either *Normal Mode* or *Colorblined*.

The second form (figure 23) showed all the tags and how much that had been changed, added, or deleted, and it opened the .xlsx file by clicking on the link.

During a meeting with the supervisor at Chalmers, the recommendation was made to use tabs instead of two forms, and this suggestion was accepted. Consequently, the screen was modified to feature one form with two tabs.

A third tab was added (figure 24) to show the history of changes. It showed the date, time, and where the comparison was made. Either “WinMerge” or “This program”, and the dataGridView is saved until “Delete the dates” is clicked, then all rows are deleted.

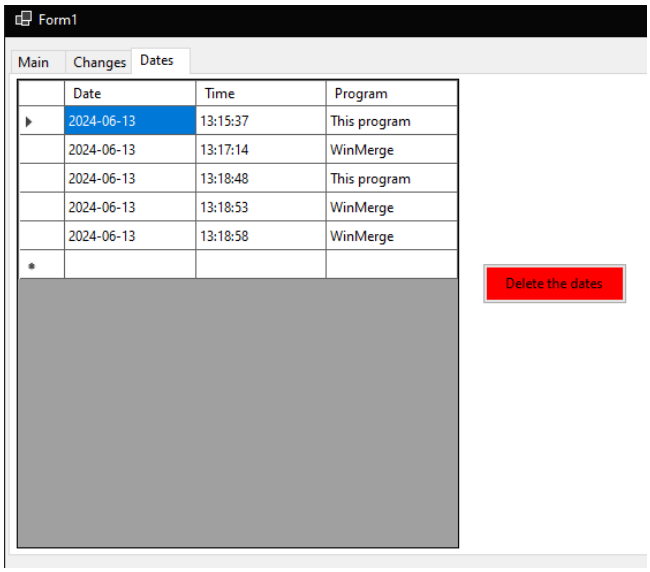
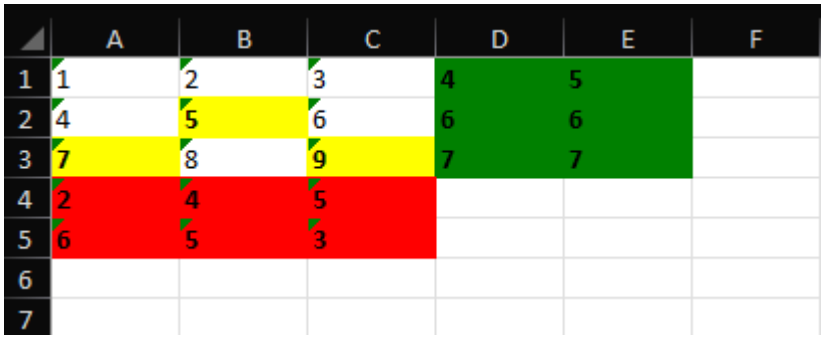


Figure 24. Tab called "History" shows when the changes were made and where.

To make the interface user-friendly color was added in the output to make it easier for the user to see the result of the comparison as shown in Figure 25 in which it was using test data.



	A	B	C	D	E	F
1	1	2	3	4	5	
2	4	5	6	6	6	
3	7	8	9	7	7	
4	2	4	5			
5	6	5	3			
6						
7						

Figure 25. Color coded test data in which green=added, red=removed and yellow=changed data (Output).

Following a discussion with the Chalmers advisor, a decision was made to add a CheckBox feature instead of deleting every row. This allows the user to select specific rows for deletion.

When the third version (figure 26, figure 27 & figure 28) was shown to the supervisor at Init. The feedback was that the technical requirements were met but there was still room for improvement to make the interface more user-friendly. It was decided to make it possible to drop the folders into the textBoxes.

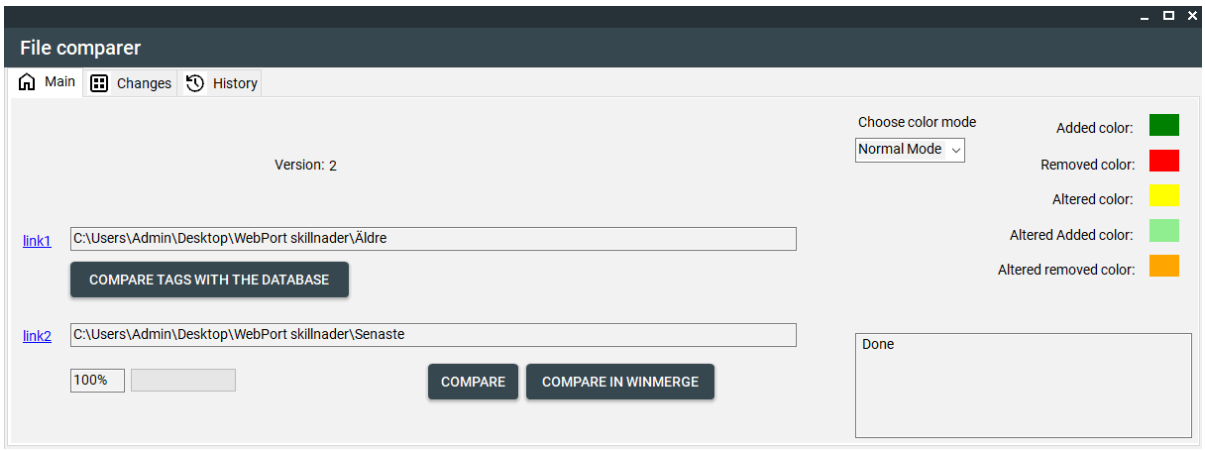


Figure 26. The main tab where the user can select the color mode, which folders to compare and have an overview of the process.

The used comparison tool is displayed in the “History” tab and can show “DB vs tags” indicating that the comparison was done with the file comparer but within the same folder with help of SQLite, “This program” indicating that the comparison was carried out exclusively with the file comparer for 2 folders and lastly “WinMerge” indicating that the comparison was done with the external tool WinMerge. The external WinMerge comparison can be done by clicking on “COMPARE IN WINMERGE” as shown in Figure 26.

The delete button has been made red since the button deletes historic records of done comparisons as well as the data in the save location of said record. it is therefore better to have a different color on it than the others to make it more noticeable.

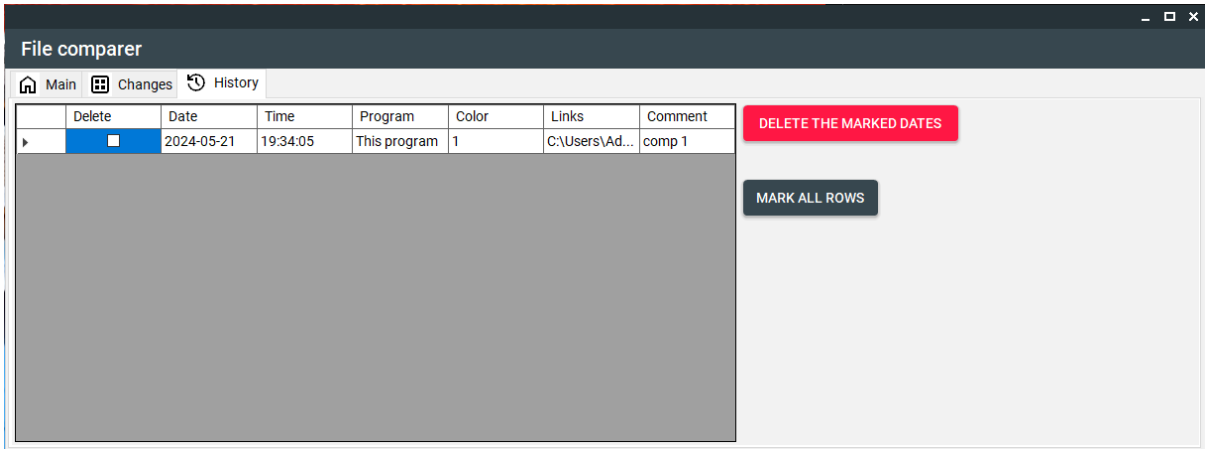


Figure 27. Shows when the changes were made, where and it is possible to delete.

To make it easier for the user to understand the colors in the .xlsx file, the color description of the comparison was added in the “Changed” tab (figure 28) and updates based on the selected identifier in the drop-down menu.

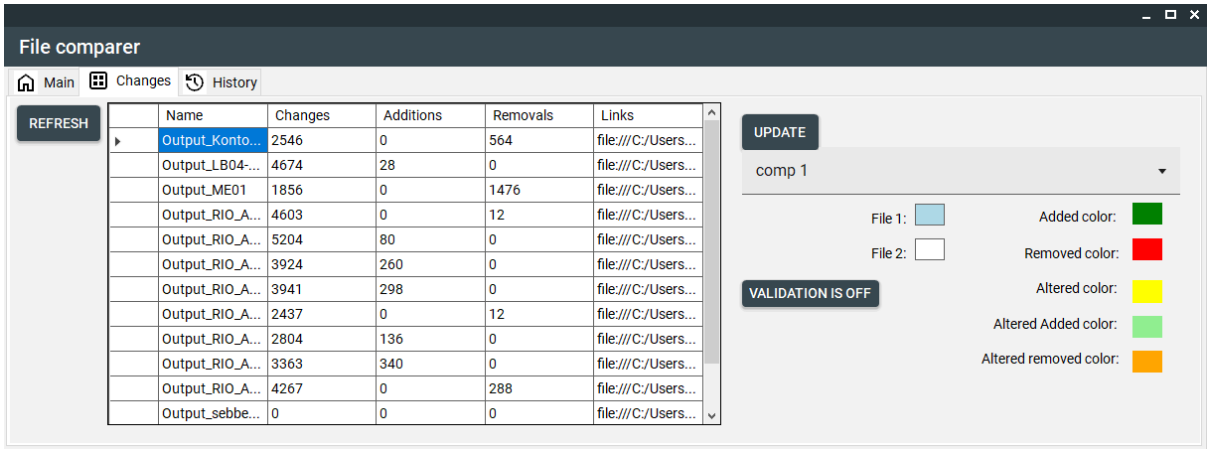


Figure 28. Shows the file name, changes, additions,removals and links to the file which is clickable.

During a meeting with the supervisor at Chalmers, the feedback indicated that incorporating a button to mark all CheckBoxes would enhance user-friendliness. Figure 28 was initially considered the final version; however, further review led to the addition of a button labeled “VALIDATION IS OFF” / “VALIDATION IS ON” to clarify the validation function. These two changes resulted in the final version presented at the beginning of this chapter.

4.3 Installer for the software

The installer, created using the Microsoft Visual Studio Installer Projects 2022 extension, enables easy program installation without the end user needing Visual Studio Community or running in a debugger. Users can now install the program at a custom location via a simple Microsoft Software Installer (.msi) wizard, which also facilitates uninstallation.

4.5 Problems during the process

There were two big problems during the process. The first problem, the process excessive memory consumption, reached 8 GB during the comparison of small folders, but in the end the process reduced memory usage to approximately 270 MB.

The second problem, the comparison took a long time, sometimes more than 30 minutes. In the final version the process takes approximately 2 to 7 minutes.

5 Results

In this segment, the inquiries outlined in the introductory chapter 1.3 are reiterated, and an assessment is conducted to determine whether the completed tool addresses them.

1. **What is the most reasonable and effective way to compare 2 .csv-, 2 .sqlite- and .csv with .sqlite files?**

The initial question is addressed by the tool through a broad comparison method tailored for .csv files, extended to encompass .sqlite files by initially converting them to .csv format. This simplifies the process, requiring only one comparison function where the first file is the old and the second is the new version.

2. **How can the comparison be expanded from single files to folder-wide comparisons and what is the most convenient way to keep track of the results in that case?**

The tool tackled the second point by enhancing the single file comparison function into a folder-wide comparison function. This function initially scans for file names across folders and subfolders, and the outcomes are monitored through a central file that records every comparison conducted within a group shown in Figure 28.

3. **How can the visual interface present the information effectively while also being user-friendly?**

The tool addresses the problem of ineffective and non-user-friendly presentation of information by incorporating color-coded output data and offering a summary of results within the primary file. Furthermore, it relies on standard .xlsx file formats, guaranteeing compatibility with widely used table management software such as Microsoft Excel, facilitating sorting and table scaling. Detailed information on this formatting is provided in section 4.3.4. However, the assessment of user-friendliness was somewhat limited due to the unavailability of a public testing version for the tool and constraints on gathering feedback within the allotted time.

4. **What is a reasonable time for the comparison to occur and how resource-intensive should it be?**

The fourth point is tackled by the tool employing the garbage collector function to empty the memory after each loop. This process prevents any accumulation in temporary memory, thus averting slowdowns and ensuring minimal memory usage. The runtime of the process was influenced by memory usage, with its duration ranging from 2 to 7 minutes depending on the quantity of files processed. This variability was considered reasonable.

5. **How should the data be validated and how should additions, removals and changes be chosen by the user to keep?**

The tool addresses this point by enabling users to select the specific data they wish to retain concerning additions, removals, and changes. This selected data is saved as a separate output file, retaining the original file name for straightforward overwriting, if necessary (figure 42), shown in section 5.2.4.

6. **How to make the program general enough so that it can be used in other control systems as it further develops?**

The tool handles the concern of being a general solution by only being restricted to the file type generated by the SCADA system it is intended for. However, it could potentially interface with systems using different file types if they are converted to .csv format.

5.1 The form

The final version features a single form comprising three tabs. The subsequent section provides a detailed examination of each tab, enhancing comprehension of the comparison process.

5.1.1 tab 1 “Main”

The interface of the “Main” tab (figure 29) consists of the following items formatted in material design:

- A version indicator.
- Color mode selector box.
- Two link boxes for folder addresses that can be used with a drag and drop or the file explorer.
- A progress bar and percentage indicator.
- Three buttons for the different types of comparisons.
- Text box to display if the file is found or not as well as individual comparison results.

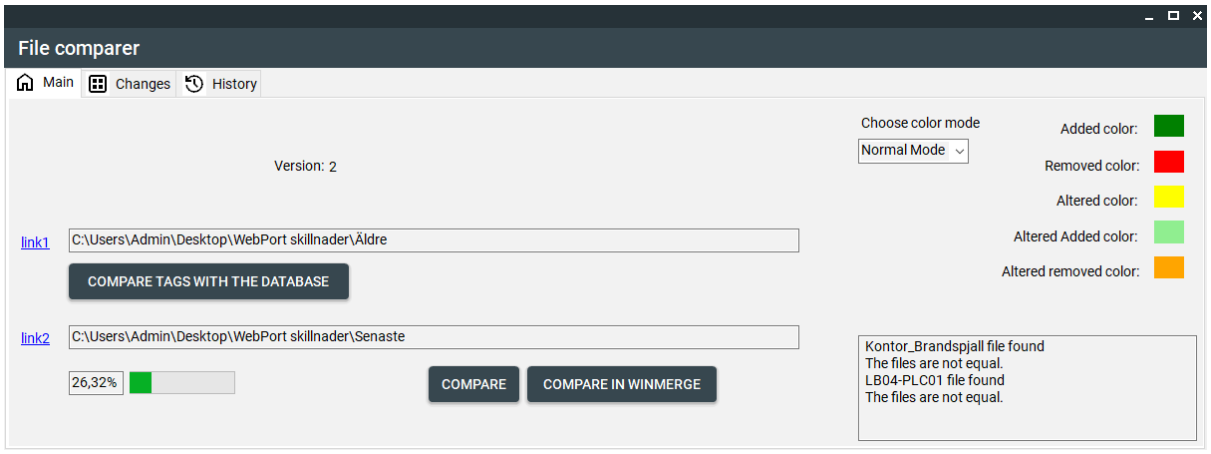


Figure 29.The main tab, where the user chooses what they want to compare.

5.1.2 tab 2 “Changes”

The second tab (figure 30) called “Changes” shows in detail how many entries have been added, removed, or changed for every file in a folder and consists of the following parts:

- Refresh button that updates the displayed comparison.
- Update button that updates the list of comparisons based on the contents in the "History" tab.
- A set of panels to indicate the different colors used in the sheets.
- A window that displays the content of the generated “Home” file that links to every comparison and offers a quick overview of the changes in every file.
- Validation button to mark rows that have been changed that the user wants to save.

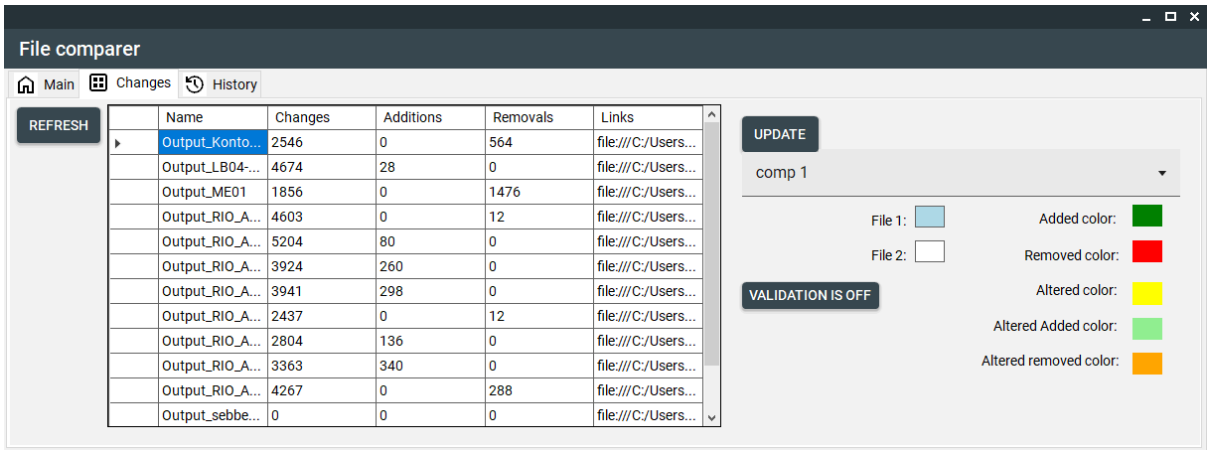


Figure 30. The tab with the changes.

5.1.3 tab 3 “History”

The third tab (figure 31) called “History” shows when someone did a comparison, where/type, every comparison has an ID “Comment”, it is possible to open the folder where the output was saved, and it can delete the selected rows. When WinMerge is used it is empty under the column called “Links” (figure 31) since the result is not saved in any folder. The column called “Color” has an ID of which color mode was used for that comparison, when WinMerge is used “Color” is empty since it does not use any color. It consists of the following parts:

- The data grid contains the date, time, program used, link if available, color, and the identifier comment.
- A checkBox on each row to indicate if one wants to delete the entry.
- A “DELETE THE MARKED DATES” button.
- A “MARK ALL ROWS”, marks all the checkBoxes.

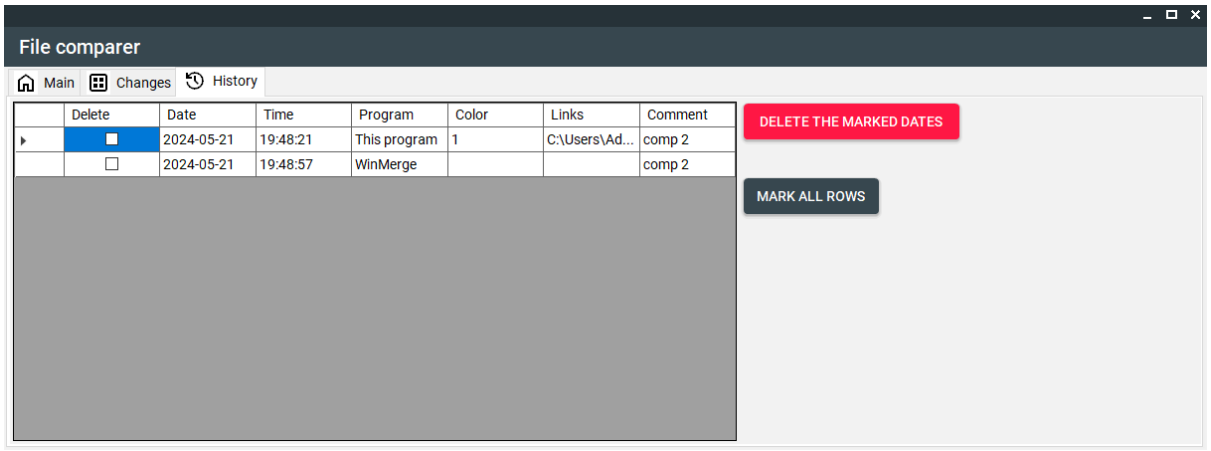


Figure 31. The historical tab.

5.2 The output

Within the "Main" tab, three buttons are available, each yielding distinct outputs that are recorded in tab 3 "History". This section provides individual descriptions of each button to facilitate a comprehensive understanding of their differences.

5.2.1 Compare in WinMerge

Pressing the button located within the “Main” tab initiates the program to launch WinMerge and transmit the two selected folder paths. WinMerge then proceeds to compare and display the discrepancies in its default format. Nonetheless, the utility of this feature diminishes when assessing .csv or .sqlite files. The primary purpose of this button is to enable users to compare files of types unsupported by the program itself, should they wish to do so.

5.2.2 Compare two folders

The program compares the two folders by comparing the files that have the same name in the folders. The result is shown in a main .xlsx file (figure 32) that shows how much has been added, deleted, or changed in every file and it gives the link to everyone. In every tag .xlsx file, there are four sheets, one for the new version and another for the old version. The sheet that is called “File1vsFile2vsComparisson” (figure 33) shows the old value in light blue, the new version in white and the yellow is the comparison part indicating that the value was changed. The sheet called “File1vsFile2” (figure 34) does only shows the new version, it marks the changes in yellow and it is the sheet where the user marks what they want to save with gray if a validation of the changes is desired.

file name	changes	additions	removals	Links
Output_default-specialdays	0	0	0	Link
Output_DrvExolineImportFilter	0	0	0	Link
Output_ImportFilter	0	0	0	Link
Output_LB04-PLC01_format0	4959	0	816	Link
Output_PIIGAB_Media	0	0	0	Link
Output_SAIAImportFilterExample	0	0	0	Link
Output_Taggar_UC-081002-AS01_PLC02_FS100_07	1407	0	0	Link
Output_tagsetup	0	0	0	Link
Output_Teknikrum-087662-AS03-2-Rumsreg-Kontor_v4	0	124	0	Link

Figure 32. The main .xlsx file.

	A	B	C	D	E	F	G	H	I	J	K
64	B10107_LB04B_GF41_PV	LB04_PLC01	R120	LONG	0	1000	0	1000	l/s	0	Flöde
65	B10107_AS01_GT31_PV	LB04_PLC01	R0	LONG	-300	400	-30	40	°C	000.0	Utetemperatur
66	B10107_LB04B_GP12_AD	LB04_PLC01	R2548	LONG	0	60	0	60	min	0	Tillslagsfördröjning tryckavvikels
67	B10107_AS01_GT31_SP1	LB04_PLC01	R1	LONG	0	500	0	50	°C	000.0	Börvärde uppstart
68	B10107_LB04A_GP12_ADL	LB04_PLC01	R58	LONG	0	1000	0	100	Pa	0.0	Larmgräns tryckavvikelse
69	B10107_OTV02_BGV02_V	LB04_PLC01	F2014	DIGITAL							I drift
70	B10107_LB04B_GP12_MAX	LB04_PLC01	R2584	LONG	0	3000	0	300	Pa	0	Max börvärde
71	B10107_LB04_NS_AL	LB04_PLC01	F9	DIGITAL							Nödstopp aktiverat
72	B10107_LB04B_GP12_MIN	LB04_PLC01	R2583	LONG	0	3000	0	300	Pa	0	Min börvärde
73	B10107_LB04_NS_AD	LB04_PLC01	R2016	LONG	0	120	0	120	s	0	Tillslagsfördröjning larm
74	B10107_LB04B_GP12_Y4	LB04_PLC01	R2581	LONG	0	3000	0	300	Pa	0	Kompenserat börvärde punkt 4
75	B10107_LB04_NS_V	LB04_PLC01	F2015	DIGITAL							Ind. aktiverat nödstopp
76	B10107_LB04B_GP12_Y3	LB04_PLC01	R2578	LONG	0	3000	0	300	Pa	0	Kompenserat börvärde punkt 3
77	B10107_AS_LB04A_AUT01_AL	LB04_PLC01	F10	DIGITAL							Utlöst huvudsäkring e fördröjn
78	B10107_LB04B_GP12_Y2	LB04_PLC01	R2575	LONG	0	3000	0	300	Pa	0	Kompenserat börvärde punkt 2
79	B10107_AS_LB04A_AUT01_AD	LB04_PLC01	R2017	LONG	0	120	0	120	s	0	Tillslagsfördröjning larm

File1vsFile2 | File1 | File2 | File1vsFile2vsComparisson

Figure 33. File1vsFile2vsComparisson, shows the new values, old values and the comparison's values.

	A	B	C	D	E	F	G	H	I	J	K	L	M
7	B10107_FL02_KF01_M	LB04_PLC01	F2256	DIGITAL	0	0	0	0			Auto/Hand		
8	B10107_FL02_KF01_AD	LB04_PLC01	R2618	LONG	0	1000	0	1000	sek	0	Tillslagsfördröjning larm		
9	B10107_FL02_KF01_CMD	LB04_PLC01	F2254	DIGITAL	0	0	0	0			Manöver		
10	B10107_FL02_KF01_AL	LB04_PLC01	F100	DIGITAL	0	0	0	0			Driftfel	p:B	
11	B10107_LB04X_IN_URSTEG_SP12	LB04_PLC01	R2599	LONG	0	1000	0	100	%	0	Vinterdrift, nedramp; frånluft		
12	B10107_LB04X_IN_URSTEG_SP11	LB04_PLC01	R2598	LONG	0	1000	0	100	%	0	Vinterdrift, låst varvtal; frånluft		
13	B10107_LB04X_IN_URSTEG_SP10	LB04_PLC01	R2597	LONG	0	1000	0	100	%	0	Vinterdrift, nedramp varvtal; tilluft		
14	B10107_LB04X_IN_URSTEG_SP9	LB04_PLC01	R2596	LONG	0	1000	0	100	%	0	Vinterdrift, låst varvtal; tilluft		
15	B10107_LB04X_IN_URSTEG_TD9	LB04_PLC01	R2595	LONG	0	120	0	120	sek	0	Vinterdrift, uppstartsfördröj; tilluft		
16	B10107_LB04X_IN_URSTEG_SP8	LB04_PLC01	R2594	LONG	0	1000	0	100	%	0	Sommardrift, uppdramp varvtal; frånluft		
17	B10107_LB04X_IN_URSTEG_SP7	LB04_PLC01	R2593	LONG	0	1000	0	100	%	0	Sommardrift, låst varvtal; frånluft		
18	B10107_LB04X_IN_URSTEG_SP6	LB04_PLC01	R2592	LONG	0	1000	0	100	%	0	Sommardrift, uppdramp varvtal; tilluft		

Figure 34. File1vsFile2 shows the new version.

5.2.3 Compare tags and the database

The program reads the name of every tag .csv file and searches for it in the database. Then it converts it to a .csv file to compare it with the tags. The output is all the tags that are compared from the database to separate .csv files and a main .csv file that has a link to every .csv file.

5.2.4 Procedure for Removing and Retaining Changes

The validation process involves selecting and preserving changes in the following steps:

1. Perform the desired comparison and save the result to a location specified by the user.

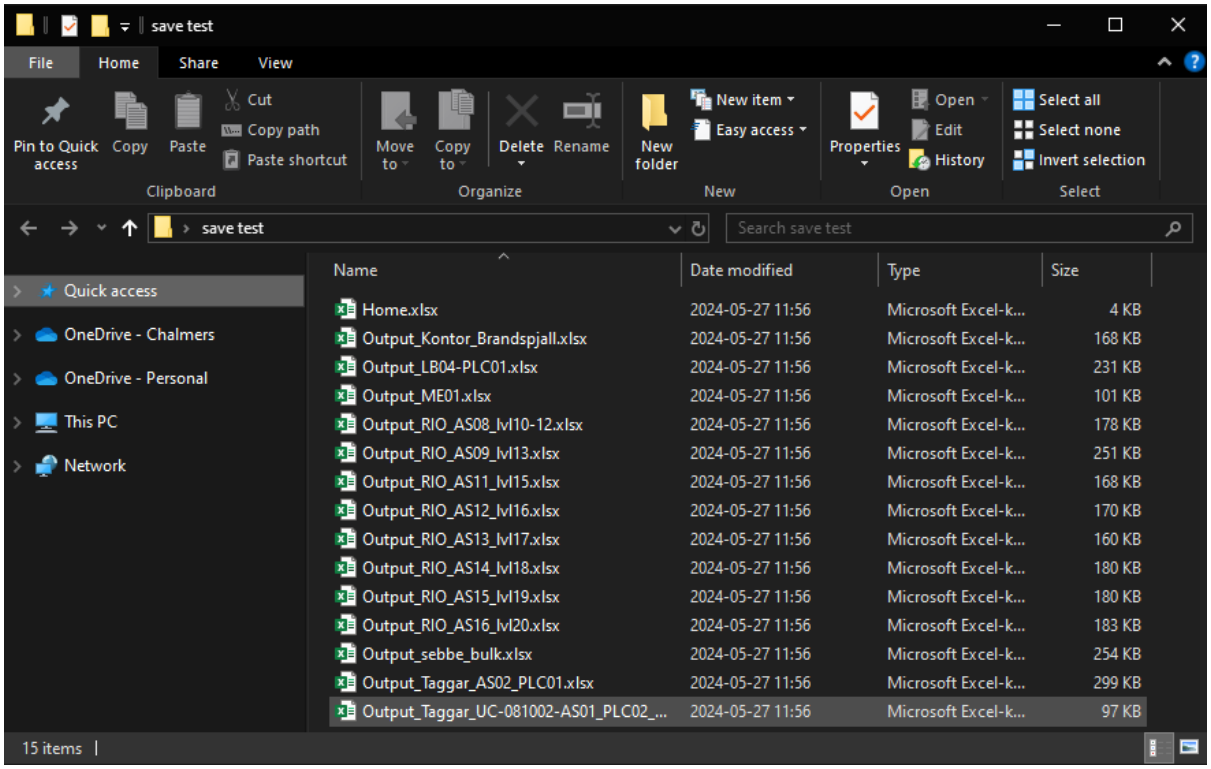


Figure 35. Comparison output consisting of the" Home" file and the output comparison files.

2. Update the comparison displayed in the changes tab to the latest one using the ID comment.

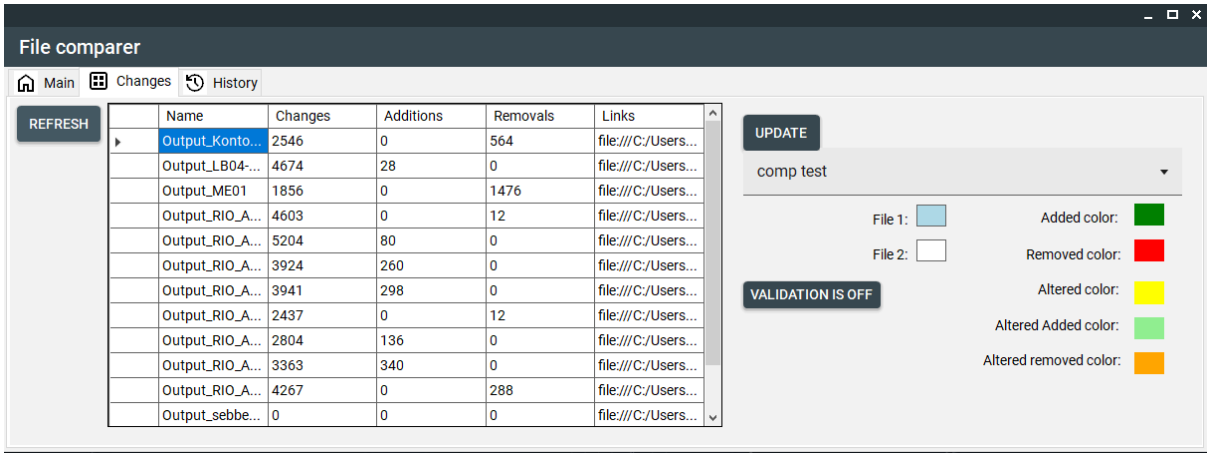


Figure 36. Content of the" Home" file displayed in the interface.

3. Toggle the "VALIDATION IS OFF" button to "VALIDATION IS ON".

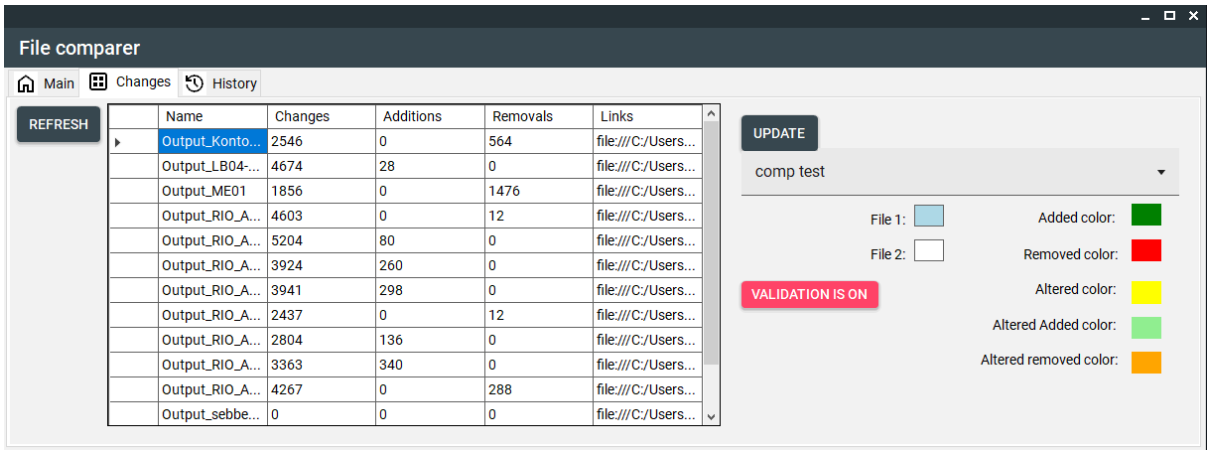


Figure 37. ” Changes” tab once validation is toggled on.

4. Access the desired compared file through the link column in the interface.

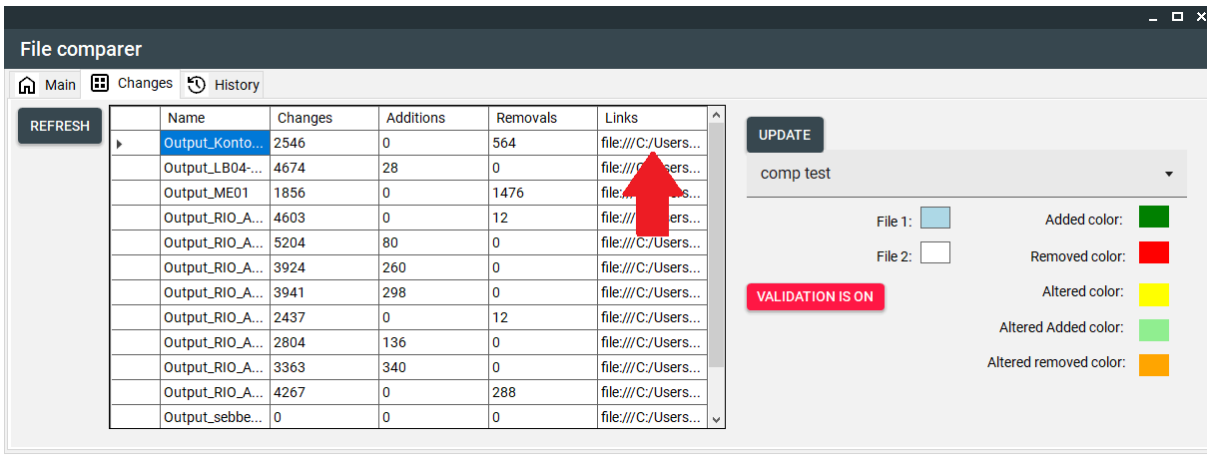


Figure 38. Pressed link displayed with a red arrow.

5. Identify and mark the maintained changes by shading a row in the first column with gray in the “File1vsFile2” sheet.

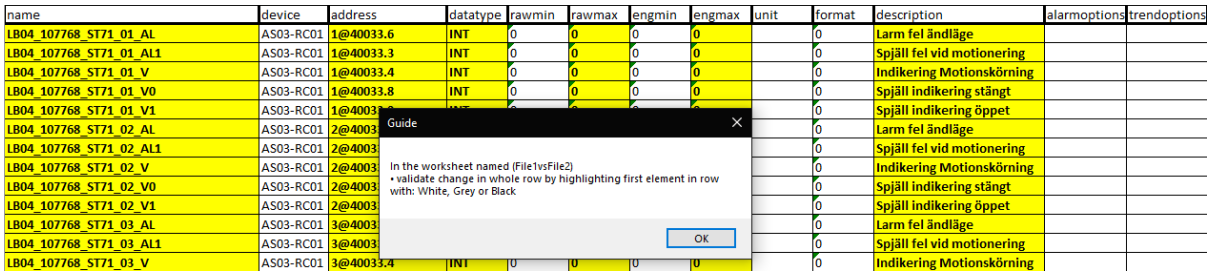


Figure 39. Interface of the comparison file and” Guide” notification.

6. Save the change and close the .xlsx sheet.

name	device	address	datatype	rawmin	rawmax	engmin	engmax	unit	format	description	alarmoptions	trendoptions
LB04_107768_ST71_01_AL	AS03-RC01	1@40033.6	INT	0	0	0	0		0	Larm fel ändläge		
LB04_107768_ST71_01_AL1	AS03-RC01	1@40033.3	INT	0	0	0	0		0	Spjäll fel vid motionering		
LB04_107768_ST71_01_V	AS03-RC01	1@40033.4	INT	0	0	0	0		0	Indikering Motionskörning		
LB04_107768_ST71_01_V0	AS03-RC01	1@40033.8	INT	0	0	0	0		0	Spjäll indikering stängt		
LB04_107768_ST71_01_V1	AS03-RC01	1@40033.9	INT	0	0	0	0		0	Spjäll indikering öppet		
LB04_107768_ST71_02_AL	AS03-RC01	2@40033.6	INT	0	0	0	0		0	Larm fel ändläge		
LB04_107768_ST71_02_AL1	AS03-RC01	2@40033.3	INT	0	0	0	0		0	Spjäll fel vid motionering		
LB04_107768_ST71_02_V	AS03-RC01	2@40033.4	INT	0	0	0	0		0	Indikering Motionskörning		
LB04_107768_ST71_02_V0	AS03-RC01	2@40033.8	INT	0	0	0	0		0	Spjäll indikering stängt		
LB04_107768_ST71_02_V1	AS03-RC01	2@40033.9	INT	0	0	0	0		0	Spjäll indikering öppet		
LB04_107768_ST71_03_AL	AS03-RC01	3@40033.6	INT	0	0	0	0		0	Larm fel ändläge		
LB04_107768_ST71_03_AL1	AS03-RC01	3@40033.3	INT	0	0	0	0		0	Spjäll fel vid motionering		
LB04_107768_ST71_03_V	AS03-RC01	3@40033.4	INT	0	0	0	0		0	Indikering Motionskörning		
LB04_107768_ST71_03_V0	AS03-RC01	3@40033.8	INT	0	0	0	0		0	Spjäll indikering stängt		
LB04_107768_ST71_03_V1	AS03-RC01	3@40033.9	INT	0	0	0	0		0	Spjäll indikering öppet		

Figure 40. Interface with desired changes marked in gray.

7. Choose a save location; if a file with the name of the compared file exists, it will be updated; otherwise, a new file will be created.

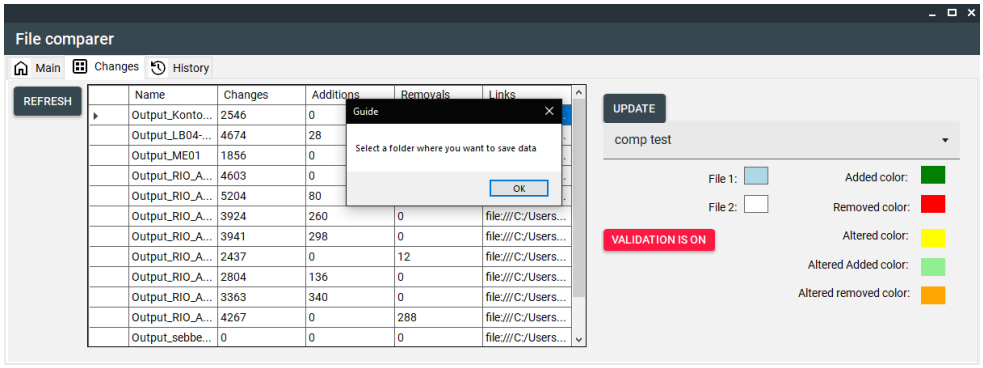


Figure 41. Notification once the desired changes have been marked.

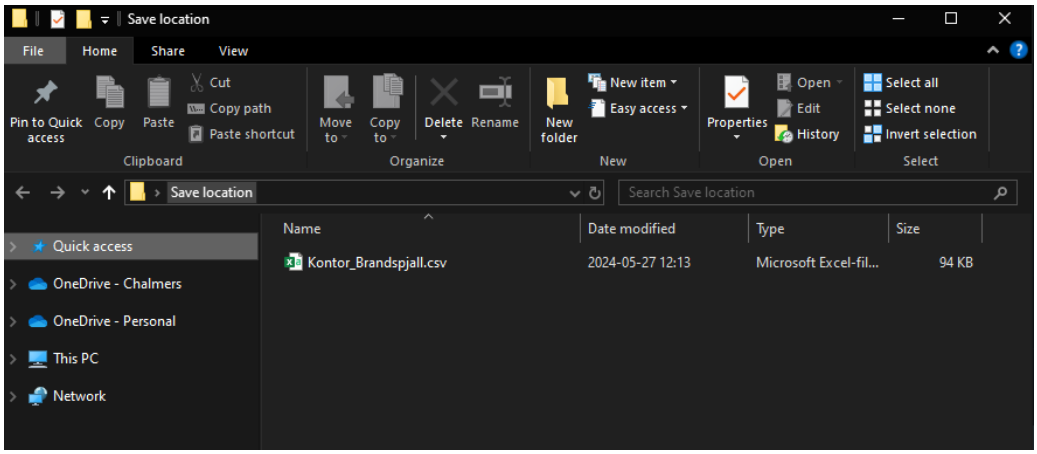


Figure 42. Resulting file from validation in .csv format.

6 Conclusion and discussion

The section comprises an introductory summary of the project, followed by a discussion on its results and a comparison with findings from other studies. It addresses ethical and ecological considerations, evaluates the timeline, and concludes with additional points for further development.

6.1 Summary

In summary, digital programs for comparison are more effective than relying on human effort due to them minimizing the risk of errors and being less time consuming. There already exist file comparison tools, like WinMerge, that compare files, but the resulting tabular data is hard for a human to read and understand. The purpose was therefore to develop a tool to compare both .csv and .sqlite files created by the Web Port SCADA system and focused on user-friendliness. The functionalities presented by the tool are the following: First, an effective and memory efficient method to compare .csv and .sqlite files. Second, a folder-wide comparison where sets of files are compared automatically and made available in the program interface and in the form of locally stored files. Third, a record of comparisons done linking to the file location and allowing annotation with comments. Fourth, an overview of the comparison result (additions, removals, changes) in the user interface. Fifth, the ability to validate, approve or reject differences between the 2 versions of a file producing a third version that can be stored in a user-defined location. The finalized version of the tool offers an easy-to-use, one-form interface. Tip for other projects: Convert all files to the same format whenever possible to make comparisons easier. But one disadvantage is that the tool can only handle two types of files, so it is very specific, but then it is useful to use tools like WinMerge.

6.2 Critical evaluation of the results

This section evaluates the results of the developed tool in relation to the defined goals and limitations as well as the decisions made during the development that may have led to a different result.

One critical point is that the program only handles .csv and .sqlite files, but to make it useful for everyone WinMerge have also been used, which is useful when there are other types of files. The reason is because Init was only interested in .csv and .sqlite files.

Opting out of a specific test protocol, the approach was to evaluate and tackle distinct use cases across the development process. However, this approach leaves room for unanticipated situations to emerge, potentially jeopardizing program stability. Furthermore, unexpected bugs or issues specific to certain Windows versions may arise, given that our testing primarily focused on Windows 10 and 11, and relatively robust hardware configurations. The issues that could arise range from performance differences, certain security measures interfering with file access or differences in registry locations. Employing a more rigorous testing regimen encompassing diverse use cases, operating systems, and hardware configurations would have resulted in a more stable program with well-defined limitations.

The tabulated comparison data was made available and formatted with the help of .xlsx files providing a straightforward solution, albeit not necessarily the optimal one. An alternative approach could have enhanced user experience by utilizing local .html and .css files for formatting, making the data interactive and navigation friendly. However, time constraints prevented this implementation, as it would have added complexity to the process.

When comparing the final version to previous studies, there were some challenges in pinpointing specific findings. This difficulty arises from the intricacies of data validation within expansive systems. Although not groundbreaking, this issue is commonly addressed through hardware verification, particularly concerning the concept of the "digital twin". The emphasis, however, is specifically on version management within the context of software, albeit within a broader framework. Nonetheless, insights from previous research can inform the evaluation.

One notable aspect discussed in earlier studies is the complexity posed by large volumes of generated data and its ramifications on system dynamics and decision-making processes in large-scale applications. These studies underscore the significance of big data applications and systems, emphasizing the critical need for ensuring the accuracy and interpretability of data.

The existence of prior studies addressing the necessity of data management for handling large volumes of data, with a predominant emphasis on ensuring accuracy and interpretability, suggests a significant interest in the overarching subject. However, the predominant focus of these studies on hardware solutions or general applications implies that either this aspect is not extensively addressed in academic circles or that existing solutions are typically perceived as sufficient. Moreover, it is plausible that bespoke SCADA systems could be developed with version management capabilities integrated, potentially obviating the necessity for standalone tools altogether.

As shown in [26] various tools are identified that cater to a range of file formats, including .csv and .sqlite. However, the study also highlights the limitations stemming from the generic treatment of file formats, as well as the absence of well-defined quality assurance standards and programs tailored to specific applications [26]. Creating powerful applications with limited usage capabilities without a large amount of competence.

This project partially addresses this lack of specialized applications while also striving to deliver a user-friendly experience, reducing the need for proficiency in a general program with a broader than needed usage. Nonetheless, there remains ample room for improvement and further development in the developed tool.

The hypothesis that pre-existing tools are often deemed usable enough, as seen in previous studies, is supported by [26]. Despite acknowledging certain limitations, the study's conclusion suggests that large, broad tools like Datameer and Informatica are sufficiently effective. This indicates that the result is a departure from previous approaches in addressing the problem of version management in systems where this is not a feature by default.

6.3 Development approach

This section delves deeper into the development process, highlighting insights and lessons learned from various hiccups and oversights encountered along the way. The challenges faced will be discussed, how they were addressed, and the corrections implemented to improve the overall development approach. It also contains a thorough evaluation of the programmed tool regarding the established inquiries and functionality requirements.

6.3.1 Development process

Starting with a more strategic coding approach could have led to better outcomes. Initially designing and testing functions individually before integrating them with buttons would have streamlined the development process. This method would have allowed for the consolidation of repetitive actions into reusable functions, significantly reducing the complexity of the code.

Moreover, employing classes to encapsulate functions would have been advantageous, enhancing modularity and facilitating collaboration among developers. The organizational structure provided by classes not only could have improved readability but also could have fostered a more efficient and scalable codebase.

In the beginning, it took less time to write code that reads from the database and converts it to .csv files than expected. And once all the necessary functions were created, building the forms-based user interface became quite straightforward. The process mainly involved determining where to call each function. Having known this beforehand would have led to a more accurate initial planning of the coding process.

The significance of user-defined addresses for the output folder was recognized during the addressing phase, facilitating universal program usability. Initially, the program was structured into multiple forms during development. However, this approach proved inefficient and resulted in unnecessary passing of variables between forms, introducing redundancy. Accordingly, the tab-based model was chosen, incorporating functional segments to enable variable reuse without compromising the design.

A notable oversight in the program's initial development was the handling of memory allocation. Initially relying on dynamic memory allocation due to the project's object-oriented nature, the encountered issues with memory overflow during processing, leading to unexpectedly prolonged tasks. After debugging the root cause was found to be that the garbage collector function failing to deallocate memory from the heap efficiently. This was resolved by implementing a solution that forced the garbage collector to deallocate memory after each loop iteration.

The finalized version, with its three functional tabs, embodies the culmination of continuous development from its initial iteration. Initially, the focus was primarily on functionality, resulting in a structure spread across four forms and laden with numerous buttons. However, this initial version proved cumbersome and lacked certain functionalities that were later realized through the streamlined three-tab approach. It is important to acknowledge how the initial approach influenced the eventual program and the pivotal decisions that guided its development journey.

6.3.2 Evaluation of the program tool

The developed program can handle large volumes of file comparisons and display the results in an interface window. It shows the comparison results within the interface through an .xlsx file, which provides an overview of the data and interactive links to the comparison files.

The evaluation of whether the program aligns with the previously described requirements for user-friendliness and functionality can now be conducted. The program efficiently compares files by minimizing the amount of allocated temporary data and clearing memory after each comparison. This approach reduces memory usage, with only a minor delay during the memory deallocation process. The minimal delay is outweighed by the significant memory savings achieved through garbage collection.

The user interface is designed with muted colors, simple terms, and icons, adhering to material design principles, making it both functional and visually appealing. This contrasts with other tools that use harsh highlights and outdated design principles.

For data management, users can quickly sort through the data thanks to the .xlsx format, which is navigated like a regular Excel sheet with color-coded content that can be easily filtered. This makes tracking modifications in the files straightforward, allowing for more efficient data management.

Regarding futureproofing, the program manages the output data, ensuring it will continue to function if the output files adhere to the specified formats, even if Web Port is updated. However, if the output file type changes, the program would need modifications. Such changes in output data types are rare in pre-established control systems due to the complexity and system reworking required.

6.4 Ethical and ecological aspects

Given the software nature of this project, assessing its ecological impact is challenging. The project aims to emphasize enhancements for user convenience, promoting time savings. Ethical implications, although not definitively determined, warrant consideration, especially regarding data sourcing. While not directly tied to the project, managing substantial data raises concerns like privacy, consent, transparency, data bias, and potential accountability in case of a data leak.

6.5 User Interface and User Friendliness

The program was developed with a focus on user-friendliness, accessibility, and a streamlined, readable interface. Initially, standard, unformatted Windows form objects were used for their convenience, forgoing additional libraries. However, this approach severely constrained the visual design, resulting in a dated and unattractive appearance.

To address this limitation, the MaterialSkin 2 library was selected for its contemporary aesthetic, open-source nature, and emphasis on bold, purposeful design elements.

Transitioning from the standard Windows objects to MaterialSkin objects proved straightforward, facilitated by the library's theme settings for easy formatting.

The icons for the project were sourced for the Google Fonts open-source Material Design icon repository, being slightly modified to align with the program's requirements. The combination of the Material Design objects, and the tab-based interface enhanced the program's modernity and structural organization.

However, the adoption of the MaterialSkin 2 library presented some challenges. Specifically, the library's focus on visual themes restricted the use of custom-colored buttons, as the button background color property was overridden by the theme. This obstacle was successfully resolved by utilizing the highlight color property within the theme settings, effectively mitigating the issue.

The application was originally developed in English to cater to the international scope of Init. However, incorporating language options could have enhanced accessibility further.

Initially was one of the goals to give the user flexibility to define color patterns for comparisons, aiming for increased customizability. Upon analysis, it was recognized that such a scenario might prompt users to choose analogous colors, thus reducing readability. To address this issue without sacrificing accessibility, the application instead offers a "normal" mode, featuring a standard color palette, and a "colorblind" palette, optimized for higher visibility for the most common types of colorblindness [27].

The program's final accessibility and user convenience feature involves saving previously used color and address settings. So that the users only need to define them once if they wish to do so.

6.6 Evaluation of the timeline

The following section comprises an assessment of the timeline (Appendix A), juxtaposing the intended development process and schedule against the actual development progress and timeframe. It also delves into the factors contributing to variations in time allocation for different stages.

Creating a function in C# to save the new version by marking the changes in colors took more time than expected while the function that converts the .sqlite file to a .csv file took less time. The first stable and functional version was finished on April 17th and in the schedule, it says that it would be finished on April 27th (Appendix A), it took less time than expected. This overestimation of the time required to achieve the functional goals can be attributed to the inexperience with the program as well as the parallel work distribution which accelerated the development. However, the tool would need some further revision and additional functionality was added making the finalized and installable version complete on May 6th.

When the first timeline was devised, it was serial, in which case the development would be more time consuming. However, during the development process the timeline was changed

to a parallel one to make the workflow more efficient. That might be the reason why the first stable and functional version was finished earlier than expected.

6.7 Further development

After the full development of the project a few questions arose that could be the point of study or further development within this project or a similar one.

- Could the program be developed using a custom class-oriented approach and which benefits would this have if any?
- Is there a better way to display and sort through the data within the excel interface?
- Could the program be expanded to allow for custom comparison parameters rather than just comparing the content of two files / folders.
- Could the task have been facilitated by utilizing another programming language / development environment.

Along with the points of interest there were also some notable points that could be useful when trying to create a file comparison program that are the following:

- Establish a standard for the file types the program should support and explore the possibility to convert between different types without data loss if the program needs to handle multiple formats.
- Sort the compared data according to a specified ruleset to make sure the right things are being compared.
- Utilize reusable functions to help with code readability which can be achieved through model view separation.
- Keep in mind memory allocation during the development process to ensure the program runs smoothly.

References

[1] Init Sweden AB, “Om Init Sweden,” 2024. [Online]. Available: https://www.initsweden.com/om-initsweden/ (accessed on: 2024-06-10).

[2] Init, “Lösningar,” 2024. [Online]. Available: Lösningar | Acobia (initsweden.com) (accessed on: 2024-04-12).

[3] eLynx Technologies, ”The Pain Points of SCADA Data Access & How to Solve Them”, elynxtech.com. [Online]. Available: https://www.elynxtech.com/post/the-pain-points-of-scada-data-access-how-to-solve-them (accessed on: 12-06-24)

[4] Z. Chou, ”Exploring user experience and performance of a tedious task through human–agent relationship”, 2023 [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9944929/ (accessed on: 12-06-24)

[5] Storied Data Inc, ”Understading the Challenges of Data Comparison in Tables”, 2024 [Online] Available: https://www.linkedin.com/pulse/understanding-challenges-data-comparison-tables-storied-data-80rne/ (accessed on: 12-06-24)

[6] T. Jakiša, “C# vs. C++: What’s at the Core?”, Toptal.2022. [Online]. Available: https://www.toptal.com/c-sharp/c-sharp-vs-c-plus-plus (accessed on:2024-06-10).

[7] winmerge, “winmerge”, [Online]. Available: https://winmerge.org/?lang=en. (accessed on: 2024-01-31).

[8] SCADA info, “SCADA vs. RTU”. [Online]. Available: https://www.scadainfo.com/scada-vs-rtu/ (accessed on: 2024-01-31).

[9] Kiona, “Create your building integration solution,” [Online]. Available: https://kiona.com/products/web-port (accessed on: 2024-01-31).

[10] Antaira, “What is SCADA and where is it used?”,2022. [Online]. Available: https://www.antaira.com/What-Is-SCADA-and-Where-Is-It-Used (accessed on:2024-06-10).

[11] Antaira, “WHAT IS SCADA AND WHERE IS IT USED?,” 2022. [Online]. Available: https://www.antaira.com/What-Is-SCADA-and-Where-Is-It-Used (accessed on: 2024-01-31).

[12] Acobia, “SCADA-utvecklare till Göteborg,” [Online]. Available: https://jobb-karriar.acobia.se/jobs/2485229-scada-utvecklare-till-goteborg (accessed on: 2024-01-31).

[13] EDUCBA, “SQL vs SQLite”,March, 2023. [Online]. Available:]https://www.educba.com/sql-vs-sqlite/ (accessed on: 2024-01-31).

[14] SQLite, “About SQLite,” 2023. [Online]. Available: https://www.sqlite.org/about.html (accessed on: 2024-06-11).

[15] Microsoft. “A tour of the C# language”,2023. [Online]. Available: https://learn.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/ (accessed on: 2024-01-31).

[16] ”Visual Studio Community”, Microsoft, [Online]. Available: https://visualstudio.microsoft.com/vs/community/ (accessed on: 2024-01-31).

[17] Microsoft. “Overview of .NET Framework,” 2023. [Online]. Available: https://learn.microsoft.com/en-us/dotnet/framework/get-started/overview (accessed on: 2024-01-31).

[18] Microsoft, “learn.microsoft.com”, 2021. [Online]. Available: https://learn.microsoft.com/en-us/nuget/what-is-nuget (accessed on: 2024-01-31).

[19] J. Close, “CsvHelper,” *CsvHelper*. [Online]. Available: A .NET library for reading and writing CSV files. Extremely fast, flexible, and easy to use. | CsvHelper (joshclose.github.io) (accessed on: 2024-04-02).

[20] Admin, “EPPlus-How to Read/Write Excel files in C# .NET Core,” 2024. [Online]. Available: EPPlus – How to Read/Write Excel files in C# .NET Core | TheCodeBuzz (accessed on: 2024-04-02).

[21] A,M, Steane, “Quick introduction to Windows API”,2009. [Online]. Available: https://users.physics.ox.ac.uk/~Steane/cpp_help/winapi_intro.htm (Accessed on: 2024-04-04).

[22] Microsoft, “Spire.XLS for .NET,” 2018. [Online]. Available: Spire.XLS for .NET - Visual Studio Marketplace (accessed on: 2024-04-02).

[23] Microsoft, “Microsoft.Data.Sqlite overview,” 2022. [Online]. Available: Overview - Microsoft.Data.Sqlite | Microsoft Learn (accessed on: 2024-04-02).

[24] M.Ignace, “MaterialSkin 2 for .NET WinForms”, 2021. [Online]. Available: MaterialSkin 2 for .NET WinForms (accessed on: 2024-04-16).

[25] Microsoft, “Microsoft Visual Studio Installer Projects 2022,” 2022. [Online]. Available: Microsoft Visual Studio Installer Projects 2022 (accessed on: 2024-05-07).

[26] J. Gao, C. Xie and C. Tao, "Big Data Validation and Quality Assurance -- Issuses, Challenges, and Needs," 2016 IEEE Symposium on Service-Oriented System Engineering (SOSE), Oxford, UK, 2016, pp. 433-441, doi: 10.1109/SOSE.2016.63. keywords: {Big data;Quality assurance;Organizations;Q-factor;Standards organizations;Quality assurance;big data quality assurance;big data validation;data validation} (accessed on: 2024-07-07).

[27] M. Yursa, “Building Colour-Blind Friendly Applications”,Dec. 2017. [Online]. Available: Building Colour-Blind Friendly Applications (accessed on:2024-05-16).

Appendix A - Timeline

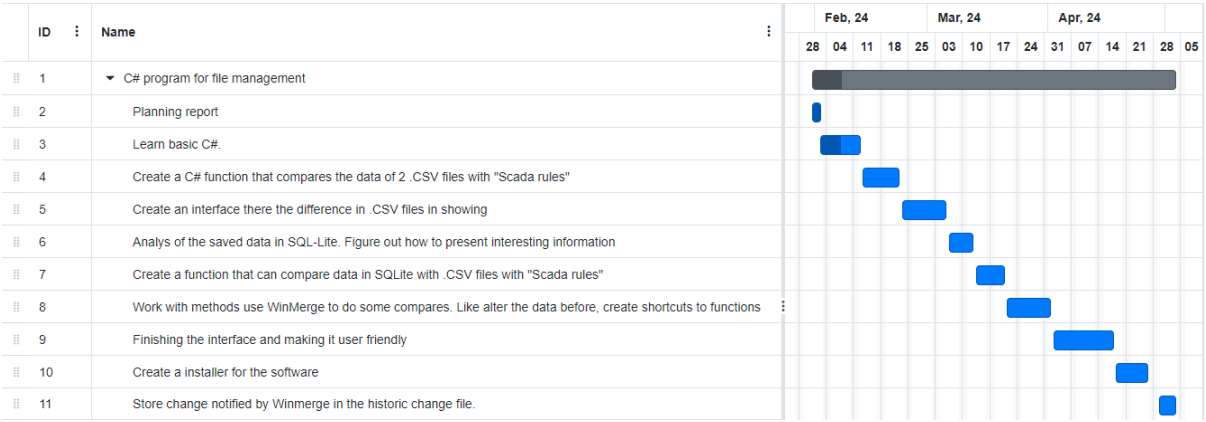


Figure 43 The first timeline.

Department of Computer Science and Engineering
Chalmers University of Technology
Gothenburg, Sweden 2024
www.chalmers.se

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY