FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Analytics and Usability metrics collection for Single Page Applications

Francisco José Paiva Gonçalves

WORKING VERSION

Master in Informatics and Computing Engineering

Supervisor: Rui Pedro Ferreira Pinto

July 1, 2023

Analytics and Usability metrics collection for Single Page Applications

Francisco José Paiva Gonçalves

Master in Informatics and Computing Engineering

July 1, 2023

Abstract

Achieving excellent usability levels within web applications has been a critical factor for product performance but has also been challenging for various reasons. On some occasions, assessing usability within an application is not easily depictable because there is no direct access to stakeholder feedback, making it difficult to evaluate a software product from a usability point of view.

It is standard for UI/UX teams to rely on user feedback and make improvements based on continuous feedback. However, to enable continuous improvement of a software product, it is beneficial to engineer a solution that automatically collects analytics and usability metrics since this approach outputs quantitative data that can shift the focus from the users and qualitative feedback. Based on this quantitative data, the UI/UX team can recommend user interface improvements that enable improved experiences for end-users. This technique is especially promising when the end-users cannot adequately give their feedback, which is the case with Critical Manufacturing's (CMF) Manufacturing Execution System (MES).

MES is a web application used by manufacturing operators on the shop floor, meaning that the feedback pipeline can be very inefficient, i.e., it takes work and time to get information from the operators to the UI/UX team. Finding key underlying bottlenecks in software usability can significantly improve the efficiency of the software on the shop-floor level as recurrent actions on the interface become easier to perform by end-users, facilitating operations and accelerating manufacturing. Consequently, this dissertation aims to assemble a general solution for metric collection within single-page applications and apply it to the specific interests of an existing single-page application, CMF's MES.

The goal is to develop an approach and a tool to gather metrics related to usability automatically and provide an overview with quantitative data of how the system interaction is occurring in a parallel and independent dashboard application.

This dissertation is done in the context of an internship with CMF and involves the company's product development team.

Keywords: Usability, metric collection, web analytics, user experience, usability insights, single-page application

Resumo

Atingir níveis ideais de usabilidade nas aplicações Web tornou-se num fator crítico para o desempenho de produtos de software, mas também releva ser um desafio por vários motivos. A avaliação da usabilidade numa aplicação nem sempre é facilmente concluída porque não há acesso direto a feedback proveniente dos utilizadores, o que torna difícil avaliar um produto de software do ponto de vista da usabilidade.

Tipicamente as equipas de UI/UX se baseiem no feedback dos utilizadores e fazem melhorias com base nesse feedback contínuo. No entanto, para permitir a melhoria contínua de um produto de software, é vantajoso conceber uma solução que recolha automaticamente analíticas e métricas de usabilidade, uma vez que esta abordagem produz dados quantitativos que podem desviar o foco dos utilizadores e do feedback qualitativo. Com base nestes dados quantitativos, as equipas de UI/UX podem recomendar melhorias no interface do utilizador que resultem numa utilização mais suave e eficaz para os utilizadores. Esta técnica é especialmente promissora quando os utilizadores não conseguem contribuir com feedback de forma adequada, como é o caso do *Manufacturing Execution System* (MES) da Critical Manufacturing (CMF).

O MES é uma aplicação Web utilizada pelos operadores do chão de fábrica, o que significa que o fluxo de feedback pode tornar-se muito ineficaz. Isto demonstra que é necessário trabalho e tempo para obter informações dos operadores para as equipas UI/UX. Encontrar os principais constrangimentos subjacentes à usabilidade do software pode melhorar significativamente a eficiência do software ao nível do chão de fábrica, uma vez que as ações recorrentes na interface se tornam mais fáceis de executar pelos utilizadores, facilitando as operações e acelerando a produção. Otimizar a usabilidade de software está diretamente ligado a aumentar a eficácia, que resulta numa vantagem competitiva muito aliciante. Consequentemente, esta dissertação visa reunir uma solução para a recolha de métricas em *Single Page Aplications* (SPAs) e aplicá-la ao contexto e necessidades concretas do MES da CMF.

O objetivo é desenvolver uma abordagem e uma ferramenta para automaticamente recolher métricas que se relacionem com a usabilidade da plataforma e fornecer uma visão geral com dados quantitativos de como a interação do sistema está a ocorrer numa aplicação de painel paralela e independente.

Esta dissertação é realizada no âmbito de um estágio na CMF e envolve a equipa de desenvolvimento de produto da empresa.

Keywords: Usability, metric collection, web analytics, user experience, usability insights, single-page application

Acknowledgemnts

First and foremost, I would like to thank Rui Pinto, my thesis supervisor, for the guidance and continuous feedback that helped shape this project into a valid scientific research and dissertation. I would also like to thank Luís Ponte and Critical Manufacturing for giving me this opportunity to work on an interesting project within the context of the company's activities and product development. Additionally, I must thank Carlos Samouco and Nélson Faria, who helped me feel integrated in a new environment for me and took a lot of their own time to help me shape this project into what it became.

Then I would like to thank my parents, who gave me all the conditions and support I could ask for. Without them, my entire journey up until now would not have been possible. Thank you for giving me such a privileged life and shaping me into the man I am today.

Finally, I would like to thank my friends that helped me pave my way up until this point. For those who have been there through thick and thin I thank you for every moment of laughter and growth together. I cannot wait to keep evolving alongside you. They know who they are.

Francisco Gonçalves

“In the wild, there is no healthcare. In the wild, healthcare is ’Ow, I hurt my leg. I can’t run. A lion eats me, and I’m dead.’ Well, I’m not dead. I’m the lion. You’re dead.”

Dwight K. Schrute

Contents

List of Figures

List of Tables

Abbreviations and Symbols

API	Application Programming Interface
CLI	Command-Line Interface
CMF	Critical Manufacturing
GDPR	General Data Protection Regulation
HTTP	Hypertext Transfer Protocol
IT	Information Technology
JSON	Javascript Object Notation
KPI	Key Performance Indicator
MES	Manufacturing Execution System
SPA	Single Page Application
UI	User Interface
UX	User Experience

Chapter 1

Introduction

Contents

The usability of web applications plays a critical role in determining the overall success and performance of software systems. It is essential to ensure that web applications are intuitive, user-friendly, efficient, and provide a seamless experience to users. The importance of usability stems from its direct impact on user satisfaction and productivity. A system with high usability can significantly enhance user engagement, reduce errors, and improve overall task completion efficiency.

However, measuring and enhancing usability can be a complex and demanding task due to many factors. The multifaceted nature of usability involves aspects such as the system's efficiency, effectiveness, learnability, memorability, and user satisfaction. Each of these aspects requires careful consideration and analysis. Furthermore, the dynamic nature of user behavior and the diversity of user needs and preferences add another layer of complexity to usability studies.

This chapter covers the problem we face and the motivation behind it. The primary focus is on the potential of using quantitative data for usability assessment in Single Page Applications (SPAs). Due to their unique architecture and interaction patterns, SPAs present specific challenges and opportunities for usability studies.

SPAs are a unique approach to web development that delivers a desktop-like application experience in the browser, providing a highly responsive user experience [20]. They are unique because they push UI rendering and business logic to the browser, communicating with the server only to synchronize data, which results in a smooth user experience akin to a native application. SPAs are often the best choice to provide the optimal user experience, driving their adoption and sophistication. They are easily updated and distributed, usually without requiring any action from the user, and can support different devices and operating systems.

We aim to explore the feasibility of using analytics and metrics to gain insights into user behavior and system usability and to propose a solution that addresses the unique requirements and constraints of SPAs.

1.1 Context

The usability of applications is often evaluated qualitatively through user testing and feedback. Qualitative methods can provide valuable insights into the user experience but are often subjective and open to interpretation. Furthermore, qualitative data is not always available or feasible to gather for all software platforms and environments. On the other hand, quantitative methods can provide plain data and objective measurements that can help determine the overall usability of an application, even if they may provide a different level of detail than qualitative methods. Both are important, but this dissertation work focuses mainly on the quantitative aspects, namely the collection of **analytics and usability metrics** within a specific single-page application that allows us to draw conclusions about how the software is being used and potentially allow developers to make recommendations for changes both in the interface and application layout.

1.2 Motivation

The impetus for this research stems from the difficulties encountered in obtaining user feedback for Critical Manufacturing's (CMF) Manufacturing Execution System (MES). MES emerged in the early 1980s as data collection systems for various corporate management disciplines. They were developed to address the need for real-time mapping of the value stream in production, which became increasingly complex. MES integrates originally separate data collection systems, providing real-time, detailed, and integrated views of production and services equipment and facilities. They offer transparency, responsiveness, and cost-efficiency, making them essential tools for effective production management. MES software has been standardized by some organizations, and they are used to significantly enhance performance in manufacturing industries [7].

The communication pipeline between the end users of CMF's MES (predominantly shop floor operators) and the development teams is not streamlined, making it difficult to connect these two parties. There are also multiple end-user groups since CMF's MES is packaged and shipped differently for different service clients, making it even more complex. These factors hinder the ability to gather reliable qualitative data on usability, as it is inconvenient for users to provide valuable feedback. Even if they do, their opinions may be subjective and overlook important software issues because CMF's MES is a complex platform, a SPA with dynamic and unique behavior, which exacerbates the challenge of obtaining a holistic view of the user experience within the application.

Considering the difficulties mentioned above, it would be advantageous to implement an automated process for analyzing user experience. By collecting relevant analytics and metrics related to usability, we can ensure a straightforward and impartial evaluation of the application's performance. This approach is particularly compelling for complex applications, as manual assessments become increasingly unfeasible as they scale. This implementation could provide the development teams with a clear and insightful view of the application's behavior, thereby facilitating the process of improving its overall user experience.

1.3 Requirements & Restrictions

Unlike any other study of a SPA, there are some restrictions or requirements that CMF has imposed that shape this investigation and help us narrow down our decisions. These restrictions make this investigation unique. Our two main restrictions are linked with security and confidentiality: CMF's MES is used by many clients, and each installation is carefully managed and deployed with high regard for security concerns. These environments generally do not have access to the internet for security reasons, which assures CMF's clients' total safety and confidentiality over their operations within the MES software. These requirements are pivotal in helping us choose a metrics collector tool and in how we design our architecture, seen as though we need a solution installed entirely on-premise to protect the collected data.

Furthermore, CMF's MES uses Angular 15, which is the most recent major release of the framework at the time of writing this dissertation, featuring "a series of developer experience improvements" [9]. This means we must find a metrics collector that is compatible with this framework and matches our capabilities and interests. Most technologies are able to show us "traditional" metrics like page views, actions per visit, and most used routes, among others, by default, but we require a customized approach to measure things like transactions on the UI.

1.4 Goals

The primary objective of this research is to create a validated solution that leverages a framework that bridges the gap from quantitative data to qualitative remarks when evaluating the usability of a SPA. This framework is adjusted to CMF's primary needs for this project. Regardless, we aim to gather a significant set of metrics that can provide comprehensive insights into the user experience and facilitate the generation of qualitative remarks. Additionally, we aim to assign scores to some of the collected data to assess the application's performance more objectively. However, it is essential to note that our focus is not on providing direct feedback for the application but rather on providing access points and insights into its behavior, allowing user interface/user experience (UI/UX) teams to make recommendations for changing the system later. In short, our goal is to develop a flexible solution that provides a developer's peek into the system.

Another core practical objective closely related to the theoretical objectives outlined above is to provide Critical Manufacturing with tooling that offers customized visual insights into their application. These insights will enable the teams to make more informed recommendations for UI improvements. The solution we aim to deliver is a starting point with the potential to improve and grow.

This practical approach offers a tangible advantage for CMF. If our tooling can detect user actions that would benefit from UI refactoring or actions that could be performed more efficiently, this translates to a significant competitive edge. We must remember that these actions, which might seem minute in isolation, are replicated countless times daily on the shop floor of factories utilizing CMF's MES. Therefore, if we succeed in identifying and addressing usability bottlenecks, our contribution could significantly impact manufacturing efficiency in the long term.

CMF stated that they would be interested in the following set of metrics:

- Devices used to access the application
- Most used routes (URLs)
- Pathways (sequence of views/page routes while navigating through your website to complete a certain action)
- Time taken to complete transactions on the UI
- Cancelled vs. completed transactions

These are the key metrics that CMF believes can offer significant value to their development teams. However, we plan to deliver additional insights and analytics in a similar domain to these, which should show how the interaction plays out and allow users to interact with analytics in depth.

1.5 Document Structure

Besides the introduction, this document comprises five additional main sections.

- Chapter 2, we take a look at the state of the art of metrics collection and analytics on web pages, specifically those that can provide information for usability and user insights and perform an analysis of related work and knowledge so that we can position our work in a well-defined scientific frame.
- Chapter 3 we explore the problem under study carefully, defining what we intend to do, formulate the hypothesis and research questions, describe why we have created them, how we attempted to answer them, describe our solution to the problem and how to validate it.
- Chapter 4 details how we structured and implemented our solution to the problem, carefully discussing design decisions and limitations and outlining details for our methodology.
- Chapter 5 talks about the evaluation and validation of our solution and our methods. This chapter aims to position our research to a point where we can start to draw conclusions from the study.

- Finally, chapter 6 wraps up the dissertation, and there we make general observations regarding what was and was not achieved, as well as describe what future work on this matter looks like and how we can achieve better results going forward.

Chapter 2

State of The Art

Contents

User insights and usability insights are integral to the development and refinement of SPAs. These insights, derived from both qualitative and quantitative data, provide a comprehensive understanding of user behavior, enabling developers to create more effective and user-friendly applications. There is no doubt that usability studies are relevant as this property is "increasingly recognized as an important quality factor for interactive software systems" [26].

User insights are fundamental to any User Experience (UX) process. They provide a deep understanding of who the users are, their behaviors, and their needs. This understanding is key to creating a good experience and making informed design decisions [11, p. 109-110].

Analytics data can enhance almost every aspect of UX work. It can be used to measure the impact of design changes, provide the basis for user research or usability testing recruitment, and show how people are currently navigating a website [11, p. 10].

Even though there are ISO standards for software characteristics like usability, there is a lack of a unified framework that developers can follow to measure the usability of a SPA. Studying and developing a model that consolidates a set of organized metrics to collect *could help developing usability measurement theory* as is stated in the article "Usability measurement and metrics: A consolidated model" [26]. Since the date of publishing of this article, we have seen technologies emerge to cover usability measures and studies. One of the most prominent technologies is Google Analytics. However, as is expected, there is no clear-cut framework outlined for measuring customized parts and components of a single-page application and providing scores associated to their usability. It would be interesting to observe scores for usability of a website measured on a custom scale (e.g., 0-100) based on heuristics that predict the quality and efficiency of the interactions.

This chapter goes into detail about state-of-the-art research on how user insights and usability studies can provide meaningful knowledge about how the application is performing on the level of user interaction and operability.

2.1 Background

To define the background of our research, we divided the background analysis into three well-defined lanes: **Single Page Applications** (SPA), **Manufacturing Execution Systems** (MES) and **Web Analytics and its linking to User Experience Insights**.

2.1.1 Single-Page Applications

Defining the unique aspects of SPAs is essential to distinguish our research from others. By highlighting the specific features that set SPAs apart from other applications, we can dive deeper into understanding their distinct characteristics and potential benefits.

Single Page Applications (SPAs) have been around for a long time, but their adoption and sophistication have been driven by the increasing demand for user-focused design in commercial and enterprise web applications [20]. SPAs are applications delivered to the browser that do not reload the page during use, providing a highly responsive and interactive user experience that traditional websites cannot match. They can be thought of as fat clients loaded from a web server [20].

The motivation behind creating SPAs was to provide a more fluid and interactive page, reducing the complexity of having to know many languages and data formats. JavaScript SPAs promised a number of enticing advantages over Flash and Java, such as no plugin requirement, less bloat, and the use of a single client language [20].

SPAs provide the best of both worlds: the immediacy of a desktop application and the portability and accessibility of a website. They can render like a desktop application, redrawing only the parts of the interface that need to change as needed. In contrast, a traditional website redraws the entire page on many user actions, resulting in a pause and a "flash" while the browser retrieves from the server and then redraws everything on the page [20].

SPAs can be instantly updated and distributed like a website, eliminating the hassle of maintaining multiple concurrent versions of software. They are cross-platform like a website, working on any operating system that provides a modern HTML5 browser. This is extremely useful for many users who have a combination of devices [20].

2.1.2 Manufacturing Execution Systems

To get a grasp of the nature of the SPA under study, we must define the historical context of MES, the motivation and purpose behind as well as their future. MES have evolved significantly since their inception in the early 1980s. Originally, they were data collection systems designed to support various corporate management disciplines such as production planning, personnel, and quality assurance. Each of these areas was equipped with dedicated data collection systems, creating a landscape of nearly independent task areas. However, with the rise of the Computer Integrated Manufacturing concept, efforts were made to reproduce the interdependencies of these task areas in Information Technology (IT) systems, leading to the development of MES [14].

MESs were primarily focused on improving machine utilization. However, as production complexity increased, the need for a holistic view of production and services equipment and facilities became apparent, which led to the development of the MES concept in the mid-1990s. The Manufacturing Execution System Association started standardizing these applications, defining the level of production, production management, and corporate management [14].

These applications have become increasingly important as manufacturing companies face changing market demands. Short delivery times, high cost pressures, smaller batch sizes, and stricter quality requirements necessitate a highly process-capable production organization that can react flexibly to customer needs and internal constraints. Integrated MES systems can reduce internal friction losses, produce less expensively, and provide better control over production processes. In the future, the MES functions will be an indispensable tool for all company departments in handling their daily tasks [14].

The future of MES systems is closely tied with the development of Industry 4.0. As digital products and systems proliferate in manufacturing, the need for managing and controlling production resources effectively is growing. MES systems, serving as the "backbone" of this digital transformation, facilitate communication and compatibility among these resources [14]. To fully leverage the potential of MES in the context of Industry 4.0, manufacturing firms must enhance their IT competencies. This includes extending their information systems to integrate with supply chain partners, promoting collaboration, and enabling real-time, product-centric information flow. Such advancements in MES are crucial for achieving traceability, planning, replenishment, and forecasting in smart factories [15].

In conclusion, MES systems have come a long way from their early days as data collection systems. They have evolved into integral parts of manufacturing operations, providing valuable insights and control over production processes. As the manufacturing industry continues to evolve and face new challenges, MES systems will undoubtedly continue to play a crucial role in ensuring efficiency and effectiveness in production [14].

2.1.3 Web Analytics & Usability Insights

The field of UX analytics has its roots in the desire to understand user behavior and improve the user experience. The evolution of the digital landscape, triggered the need for more sophisticated methods of understanding user behavior. The emergence of web analytics tools provided a new way to capture and analyze user behavior on larger scales. These tools could automatically record aspects of users' behavior, transform this behavior into data, and then analyze this data to gain insights into how users interact with websites and mobile apps [4].

Initially, web analytics tools were used to answer questions of a "what" nature, like "What are the most and least viewed pages on your website?" or "What did the people who ended up buying something on your website type in your search box?" [4]. However, not long after, UX professionals realized that these tools could also be used to complement traditional UX research methods, providing context and a way to answer more questions about user behavior [11].

The integration of web analytics into UX research marked a significant shift in the field. UX professionals could now access immediate data, allowing for open exploration and deep, iterative analysis. This direct access to data also facilitated better communication with business stakeholders, as UX professionals could now speak the same language of data [4]. However, the use of web analytics in UX research is not without its challenges. UX researchers need to be careful to ensure that the data collected is relevant and meaningful, while remaining able to interpret it correctly and in the context of the UX. This often involves collaborating with other teams and experts in different domains. Despite these challenges, the use of web analytics in UX research has proven to be a valuable tool for understanding user behavior. As the field continues to evolve, UX professionals will need to continue learning and adapting, applying their fundamental skills to new situations and technologies. The goal is not to replace traditional UX methods with web analytics, but to use these tools to complement and enhance existing methods, providing a more comprehensive understanding of user behavior [11].

As we have seen, collecting analytics and usability metrics has become increasingly prominent when evaluating and enhancing the user experience in software systems. SPAs are no exception to this. In this field of research, several tools have been created to help developers and designers gather relevant data about how users interact with applications. The goal of these platforms is to provide valuable insights into the application's usability, allowing teams to make informed decisions on improving the user experience [4].

One of the earliest approaches to collecting web analytics was log file analysis, which involved manually reviewing server logs to track the behavior of users. This method provided a basic understanding of user behavior but was rather time-consuming and lacked information for in-depth analysis [5]

In response, a new generation of web analytics tools was developed, providing developers with more sophisticated and accurate metrics about user behavior. These tools have evolved to accommodate the unique characteristics of SPAs, where the majority of user interactions occur on a single page, thus requiring a more event-driven approach to analytics [4]. SPAs have become a popular web development paradigm due to their ability to provide a smooth user experience similar to desktop applications. However, the dynamic nature of SPAs presents unique challenges in terms of analytics and usability metrics collection [4].

Analytics in the context of SPAs refers to the systematic computational analysis of data or statistics. It involves the collection, processing, and analysis of user interaction data to understand and optimize web usage. Usability metrics, on the other hand, are quantitative measures that provide insights into the user's experience with the application. These metrics can include task success rate, error rate, abandonment rate, time on task, and others [4]. This sort of data can be used to inform design choices and improve user experience. It provides facts that are hard to argue with, gives fast results, offers unique insight, combines well with other UX methods, and offers an effective way to present your findings [11].

In addition to tracking user interactions, analytics for SPAs also involves understanding and optimizing the performance of the application. This includes metrics such as load time, time to interactive, and other performance metrics that can impact the user experience. These metrics can be collected using various tools and APIs such as the Navigation Timing API, Resource Timing API, and others [4].

However, the use of analytics does not devalue other UX work. It is not about using quantitative data alone. Instead, it is about how you can use quantitative data to inform your qualitative work. Both types of data are valid types of measurement, and both should be used during the UX process [11].

There are several tools available for collecting and analyzing analytics data in SPAs. Google Analytics is one of the most widely used tools due to its comprehensive feature set and ease of use. However, it requires users to have JavaScript and cookies enabled, which may not always be the case. Other tools such as Adobe Analytics, Piwik, and Yandex Metrica also offer robust analytics capabilities for SPAs [11].

In conclusion, the collection and analysis of analytics and usability metrics in SPAs is a complex but crucial aspect of improving user experience. By understanding user interactions and performance metrics, developers and designers can make informed decisions to optimize the application and enhance user satisfaction [4, 11].

2.1.4 Summary

One great strength of web analytics and user insights in the context of SPAs lies in the ability to collect and analyze customized event data. This approach is not strictly defined and can be tailored to the specific needs of each application, providing a high degree of flexibility in understanding user behavior [11, 4]. Custom events, as described in Google Analytics, allow for the tracking of specific user actions that do not necessarily result in loading a new page. This is particularly relevant for SPAs, where most user interactions occur within a single page. These custom events can be set up to track a wide range of user activities, such as clicks on specific elements, video plays, form submissions, and more. The data collected from these events can provide valuable insights into how users interact with the application, informing design decisions and improving the user experience [11, 4]. Moreover, the ability to set up custom alerts based on specific data thresholds allows for proactive monitoring of key metrics. For example, an alert could be set up to notify when traffic from a certain region decreases by more than a certain percentage, or when daily revenue drops below a set figure. This enables timely responses to significant changes in user behavior or application performance [11].

To sum up, the flexibility of web analytics in SPAs, particularly through the use of custom events and alerts, provides a powerful tool for understanding user behavior and improving user experience. By tailoring the data collection to the specific needs of the application, developers and designers can gain unique insights that inform their design decisions and contribute to the overall success of the application [11, 4].

2.2 Methodology

This section describes the method we used to gather relevant literature, such as academic papers, books, and other scientific sources, which form the basis of our investigation. We also explain how we analyzed the content to understand the current advancements in this specific field of research.

Our literature research was conducted primarily via two digital databases, namely, **Google Scholar**[1] and **ACM Digital Library**[2]. We deemed these sources enough and reliable to yield pertinent results, which facilitates our task of defining our study's state of the art.

To retrieve relevant literature, we used a simple and lenient approach with our search queries. They included:

- "**Single Page Applications**": This straightforward query aimed to find the most relevant and frequently cited works that can elucidate the context, history and evolution of this software type.
- "**Manufacturing Execution System**": This query was intended to shed light on this emerging technology, thereby helping us comprehend the nature and specific system that we'll be examining (CMF's MES).
- "**Usability metrics**" AND "**Usability analytics**": We used this two-pronged search to find information on both measurable aspects of usability (metrics) and techniques for tracking and analyzing user behavior (analytics). The goal was to find important resources that discuss these areas in detail. This helps us understand and assess the user experience within Single Page Applications and Manufacturing Execution Systems more effectively.

Our strategy for filtering results relied on the eye-test screening process, using the following criteria:

- **Relevance**: Even without looking at the full article, we assessed the literature's relevance based on its publication date and citation frequency. Given our research domain, we strongly considered these parameters in the preliminary screening process.
- **Document Inspection**: Following the initial filtering, we explored the feasible results in depth by reading the **introduction, abstract, and conclusions**. These sections generally provided a clear indication of the document's relevance to our research or whether it diverged from our primary topics of interest. If the literature's applicability remained uncertain, we read additional sections to make the final decisions.

Apart from the articles we found through our database searches, CMF also suggested two books for us to read. These two books, due to their extensive content, were the prominent resources in terms of the usability, user insight and web analytics part of our investigation domain. These books go in-depth and overshadowed other sources, providing us with ample material for setting up our investigation into usability-related matters.

[1] `https://scholar.google.com`
[2] `https://dl.acm.org/`

While we recognize that our approach to sourcing articles may not be exhaustive, we find it suitable given the clear scope of our task and the specific practical requests from CMF. The specific requirements and initial goals provided by CMF, which guided this project, are elaborated in Chapter 1, particularly in section 1.4, and gave this project a lot direction from the start.

Along with assembling a collection of scientific literature to support our research, we also searched for existing technologies related to collection of metrics and analytics within SPAs. The search for these technologies was primarily done via google searches and following links in the results we found.

2.3 Existing Technologies

In this section, we take a look at existing technologies that perform similar tasks to the ones we are looking for. Ideally, we would combine the power of multiple tools to reach a point where we take advantage of different services and optimize them for our situation. The goal is to have a holistic view of the SPA metrics collection landscape and see how we can conduct our investigation after analyzing the existing options and comparing them to our requirements.

2.3.1 Google Analytics

One of the most widely used web analytics tools today is Google Analytics: "There are about 28.1 million websites currently using Google Analytics" [27].

Google Analytics is a powerful mainstream tool that provides insights into the performance of websites. In the context of SPAs, which is of our interest, Google Analytics offers capabilities that allow developers to grasp better how users interact with their applications. It "collects data from your websites and apps to create reports that provide insights into your business. You can use reports to monitor traffic, investigate data, and understand your users and their activity" [10]. One of the most prominent features of Google Analytics for SPAs is the ability to track user interactions with dynamic content since SPAs often use client-side rendering, making it difficult to track user activity. Google Analytics also lets the developer create and use custom events and virtual page views to track user interactions with dynamic content accurately.

Google Analytics also offers advanced segmentation and reporting capabilities, which allows developers to view performance data at a granular level, enabling them to identify trends and patterns in user behavior. For example, they can segment data based on user demographics, location, or device type to better understand how users interact with their applications. Just like the latter, Google Analytics provides other features that might not be so interesting for the particular case of MES, as our end-users are already well-defined. However, it is still a good feature to take into account. Another example of a feature that might fall out of the scope is tracking real-time data that allows the teams to make adjustments in real-time.

2.3.2 Matomo

Matomo is an open-source analytics package. It enables website owners to install the software on their servers and use it to interrogate web log data. This means that it can be run without JavaScript tracking, unlike other major analytics tools [11, p. 20]. It is also described as a "Google Analytics alternative" [19] with lots of interesting features. Some of them stand out for our research like real-time data tracking updates, a dashboard for our SPA and event tracking, offering more privacy and security than Google Analytics[19].

At its core, it is a powerful web analytics tool to measure the usage of websites and applications. It offers fewer limitations than Google Analytics with a free plan, and "what you can track and achieve is limitless" [19]. Matomo emphasizes complete data ownership, effectively ensuring user privacy and compliance with general data protection regulation (GDPR) regulations [19]. It also supports on-premise installation, affording more control over data and reducing reliance on third-party data storage [19]. As an open-source platform, Matomo's codebase is transparent and community-verified, providing an additional layer of security [19]. Consequently, for organizations with a high priority on security, confidentiality, and data ownership, Matomo emerges as a compelling analytics solution.

This is a promising platform for our situation, as it allows us to customize what to track extensively while not compromising our restrictions, leaving our work with no limitations in terms of research potential. Matomo provides developers complete control over their data and the ability to store it on their servers, which cannot be said for many other services. Matomo has enormous potential for our research, because our requirements are strict and Matomo fits them and is quite flexible in tracking.

2.3.3 Hotjar

Hotjar is another web analytics tool Hotjar that provides website owners with visual data and insights into their website users' behavior. Hotjar is meant to "help you understand how users behave on your site, what they need, and how they feel" [12]. Hotjar is "trusted by 1,148,546 websites in 180+ countries" [12] and provides a robust free plan, making it worth investigating further.

Hotjar differs from the previous tools because it focuses on user feedback. As we've seen, gathering user feedback isn't always easy, which is the case for MES. However, if we integrate the feedback input points seamlessly into the application, we can change the landscape of our problem. Hotjar guarantees that we can "gather a constant stream of unbiased user feedback" and "capture feedback in the moment with unintrusive widgets" [12].

2.3.4 Other Notable Web Analytics Tools

There are several other analytics tools available. Some tools work similarly to Google Analytics, while others collect data in different ways [11, p. 19-20]. As of the time of this project, there are lots of options available, but many of them were not be adopted in our solution for diverse reasons:

some require premium paid subscriptions, others are more focused on e-commerce platforms, and many do not conform to our security and confidentiality requirements. Anyhow, here are some other prevalent analytics technologies:

- **Adobe Analytics**: This is a comprehensive analytics platform that provides real-time analytics and detailed segmentation across all marketing channels. It's particularly useful for large e-commerce businesses that need to analyze large volumes of data [1].
- **Fathom**: Google Analytics alternative that doesn't compromise visitor privacy for data. With a one-line JavaScript code, you can collect key website metrics. It features a comprehensive dashboard for real-time data analysis, and offers periodic email reports and uptime monitoring for all account sites [3].
- **Kissmetrics**: This tool focuses on customer behavior, providing insights into how users interact with your site over time. It's particularly useful for tracking customer journeys and understanding conversion funnels [13].
- **Woopra**: Woopra provides real-time customer analytics and allows you to track anonymous website and mobile app users from their first touch until they identify themselves. It's useful for understanding the customer journey and improving customer engagement [30].
- **Mixpanel**: Mixpanel is an advanced analytics platform that allows businesses to analyze user behavior across platforms. It's particularly useful for tracking user engagement and conversion rates [21].
- **Crazy Egg**: This tool provides heatmaps, scroll maps, and other visual reports to show you where users are clicking and scrolling on your site. It's useful for understanding how users interact with your site and for optimizing page layouts [8].
- **Optimizely**: Optimizely is a platform that provides A/B testing tools and personalization features, which can help e-commerce businesses improve their conversion rates and personalize the shopping experience [22].

2.4 Summary

In this chapter, we looked at the origins of web analytics and how they tie into tracking usability metrics, with emphasis on doing so in an SPA that is also a MES. We also looked at potentially suitable tools for our research and from this point on we can focus on identifying the best one for our particular needs.

Chapter 3

Methodology

Contents

This chapter serves as a comprehensive overview of the primary problem under study in this dissertation, the proposed solution, and its evaluation and validation process. It begins with section 3.1, which defines the problem this research aims to address. Next, section 3.2 presents the hypothesis forming the basis of this research. The proposed solution to the problem, detailing the methods and approaches used, is described in section 3.3. The strategy for validating the proposed solution, explaining how we measure and evaluate the results, is outlined in section 3.4. The chapter concludes with section 3.5, which encapsulates the key points of the research statement.

3.1 Problem Statement

This investigation deviates from the traditional methods of conducting usability studies, which often heavily rely on qualitative approaches. Instead, we focus on exploring the suitability of using quantitative data to evaluate the usability of a particular SPA, namely CMF's MES. This study aims to demonstrate the relevance of using quantitative data to assess usability within CMF's MES and provide a framework for such evaluations in the future. The following sections present our formal hypothesis and questions alongside a proposed solution for achieving the goals of this dissertation and project.

3.2 Hypothesis & Research Questions

The work developed in this dissertation aims to validate the following main hypothesis:

Custom event tracking in SPAs can be used to generate meaningful insights regarding user interaction and usability. Furthermore, associating quantitative metrics with these events can enhance our understanding of a unique platform.

The hypothesis for this research is rooted in the need for a unified methodology for evaluating every component within a complex and unique SPA. A SPA, with MES's dynamic nature and rich interactivity, presents a challenge in terms of usability assessment. Traditional usability evaluation methods, which often rely on qualitative data and direct user feedback, may only partially capture the intricacies of user interactions within a SPA, especially the one we are studying, CMF's MES.

A complex SPA, like CMF's MES, contains many components and user interaction points, each contributing to the overall user experience. However, the lack of a standardized approach for assessing these individual components means that usability evaluations may need more comprehensive and consistent. This gap in the field of usability assessment for SPAs presents an opportunity to develop a new methodology that can provide a more detailed and nuanced understanding of user interactions and usability.

The hypothesis proposed in this research is that an automated system for collecting and processing custom event metrics could solve this problem. Such a system would track user interactions within the SPA, capturing quantitative data that can provide insights into user behavior and system usability. This data could later be processed and analyzed to generate meaningful insights about the usability of individual components within the SPA. Moreover, these insights could be visualized and presented through a custom dashboard, providing a clear and concise overview of CMF's MES usability. This would allow developers and designers to quickly identify areas of the application that may be causing usability issues and make informed decisions to improve the user experience.

When it comes to formulating our research questions, these are the three inquiries we have come up with:

RQ1 Can custom event tracking in SPAs be used to generate meaningful insights about user interactions, operability, and behavior?

RQ2 Can we create a scoring scale and formula that predicts usability for custom transaction components in SPAs?

RQ3 Can the insights derived from custom event tracking in SPAs be effectively visualized and presented in a custom dashboard to aid developers in making design decisions?

RQ4 How can we link quantitative results associated with events to qualitative remarks about user experience?

3.3 Proposed Solution

Our solution to tackle the problem at hand is integrating a metrics collector, specifically Matomo, into CMF's MES and pairing it with an additional layer of processing and visualization. Matomo is the technology of choice for our analytics research and collection of metrics, events, and data from CMF's MES. Section 2.3.2 goes into more detail about what Matomo is. Furthermore, section 4.2.1 explains in detail why we chose Matomo to the detriment of other existing technologies and 4.2.3 explains how we tailored the usage of Matomo to our particular needs.

Without going into too much detail, Matomo uses a database to store all its data, providing a robust foundation for our analytics needs. While Matomo offers a dashboard for visualizing traditional metrics and the raw content of collected events, we aim to go beyond these default capabilities. Our solution also features the development of a **customized dashboard** that queries the Matomo API service, processing and organizing the data to generate customized insights about unique components mainly related to transactions on the UI. This approach allows us to tailor our analytics to specific needs, providing an in-depth understanding of user interactions and usability within some of the core components of MES. By integrating Matomo's robust data collection with our custom dashboard, we aim to create a tailored analytics solution that effectively addresses the unique requirements of our project. Figure 3.1 shows a diagram that summarizes the solution.

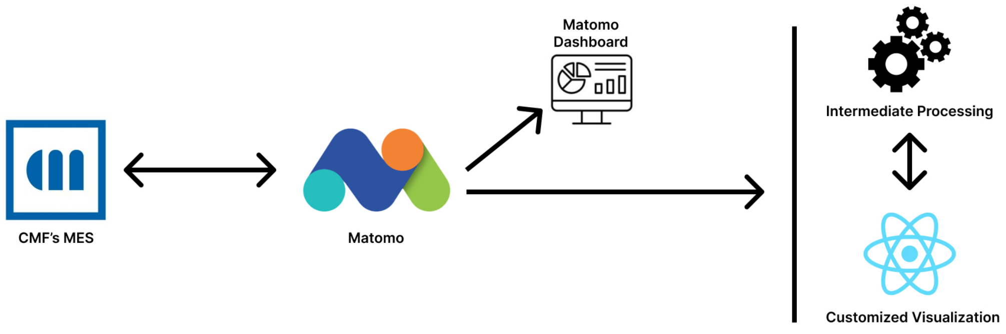


Figure 3.1: Proposed Solution Scheme

Chapter 4 goes into more detail about the design of our solution, especially section 4.2.2, which provides a view of the high-level architecture.

3.3.1 Scoring Methodology

Creating custom insights for unique components within MES is a crucial aspect of our proposed solution. This involves developing a scoring system, where each transactional component, specifically the **wizards** and **execution views**, is assigned a score within a controlled scale, for instance, 0 to 100. This score is determined based on the events within each component, such as activating a step, failing a step, encountering general wizard errors, canceling, or going back.

The scoring system is not rigid but relatively flexible, allowing adjustments and refinements as the project progresses. This flexibility is a significant advantage as it enables us to experiment with various formulas and adjust the values as necessary to achieve a more accurate representation of usability. This iterative process of refining the scoring system forms an integral part of our validation process.

The scoring system is designed to extract a wealth of metrics from each wizard or execution view. By evaluating these metrics, we can develop strategies to score these components based on the events that occur within them. The key to this approach is ensuring the validity of the measures used in the scoring system.

The ability to tweak the formula and reassess the same queried events directly from our dashboard presents a promising avenue for studying how we can evaluate components in terms of usability. This flexibility allows us to adapt our approach as we gain more insights into users' behavior and the components' performance.

3.3.2 Button Click Analysis

The analysis of button clicks forms another crucial aspect of our proposed solution. By tracking the frequency of button clicks, we can identify the most frequently used buttons and observe how their usage changes over time. Although this may seem less significant to the analysis, it provides valuable insights into user behavior and preferences.

The metrics we collect for button clicks allow us to rank buttons based on their absolute frequency of clicks and relative frequency compared to other buttons. This dual approach provides a comprehensive view of button usage, highlighting the most and least used buttons and revealing patterns in user interaction with the interface.

3.3.3 Meta Metrics Analysis

In addition to the custom insights we generate, our solution also provides comprehensive coverage of traditional or expected "meta" metrics. These include key statistics such as page views, time per visit, most used page routes, devices, browsers, screen sizes, and operating systems.

Our custom dashboard presents these metrics in a summarized and easily digestible format, providing critical insights at a glance. The interactive nature of the dashboard allows users to delve deeper into the data, exploring trends and patterns in more detail. This comprehensive and interactive approach to presenting meta-metrics provides a holistic view of user behavior and system usage, aiding in evaluating usability and informing design decisions.

3.3.4 Mapping Insights to Recommendations

The primary objective of identifying a robust set of metrics, analytics, and performance indicators for CMF's MES is to empower UX teams with data-driven insights to make informed decisions for future developments. In this section, we establish connections between metrics and potential findings on the user interface (UI) level:

- **Most viewed routes**: Understanding the most frequently accessed routes within a SPA like MES is crucial. By identifying these routes, we can consider designating a different page as the initial landing page upon platform login. Furthermore, if we observe relevant routes that are being overlooked, it is essential to enhance their discoverability in the UI. These are just two examples of how this feature can be leveraged.

- **Pathways**: This type of analytics provides a comprehensive perspective on the sequential flow of user routes. Suppose we identify a sequence of pages with significantly higher consecutive accesses than others. In that case, the UX team can explore options to make these pages more prominently accessible within the UI. Similarly, if we observe a recurring pattern of eight consecutive pages being frequently accessed, it indicates the potential for streamlining the interaction and reducing unnecessary clicks and actions.
- **Devices**: The devices, browsers, operating systems, and screen sizes used to access the application were another specific request by CMF, as this can show the development team what users use the most and shift the focus of development into optimizing the interaction for the most common conditions.
- **Wizard Stats and Scores**: Similar to the previous points, tracking faults and misusage within the system's wizards can help us identify potential performance bottlenecks that would otherwise remain unknown. By appropriately categorizing the data and assigning quantitative scores to wizards, we aim to make any underlying issues evident to the developer teams. This approach aims to improve the visibility and addressability of internal challenges.

It would be imprudent to assume that a complex and dynamic system like CMF's MES already possesses an inherently efficient layout and sequence of interactions. With the naked eye, it is challenging to visualize the flaws in the sequence of actions. The metrics we provide aim to address precisely this challenge by identifying areas of inefficiency in the UI layout. We aim to eliminate confusion and streamline the user experience by enhancing usability and user interaction.

3.4 Validating Solution

The validation of our solution is carried out in multiple stages, each designed to test different aspects of our solution and attempt to answer our research questions.

Firstly, we must verify whether the implementation of our solution is successful, ensuring that it indeed collects metrics and generates insights as intended. This process includes evaluating custom events within the system, which ties into RQ1 about generating meaningful insights about user interactions, operability, and behavior in SPAs.

Secondly, we conduct an experiment involving users interacting with MES. The data collected during these sessions is then compared with qualitative results obtained through a feedback form. The judgment of the collected quantitative data coupled with the comparison of the qualitative remarks allows us to validate the insights generated by our solution against direct user feedback. This stage aids the answering of RQ2, as the experiment automatically tests our scoring scales and considerations that predict usability for custom transaction components in SPAs.

Finally, the long-term metrics collection should provide CMF with a wealth of data over time. This data can be visualized and presented in a custom dashboard, allowing for the possibility of evaluating the effectiveness of the dashboard in aiding developers in making design decisions. This ongoing validation process should allow us to answer RQ3 about the effectiveness of visualizing insights derived from custom event tracking in SPAs in a custom dashboard, ensuring our solution remains relevant and effective in assessing the usability of CMF's MES.

3.5 Summary

In this chapter, we have defined the research problem and hypothesis, focusing on the potential of using quantitative data for usability assessment in SPAs. We have proposed a solution that involves the implementation of Matomo, a powerful analytics tool, within CMF's MES. We supplement our solution with a custom dashboard, which provides a more detailed and tailored view of user interactions and usability metrics.

We have also discussed our unique scoring methodology for transactional components and the generation of custom insights. These insights not only cover traditional metrics but also delve into the specifics of user interactions with buttons and the overall usage patterns of the application.

The research questions guide our solution's validation process and involve a combination of implementation success, experimental results, and long-term data collection. Despite certain limitations, the proposed solution offers a promising approach to usability assessment in SPAs, providing a comprehensive view of user behavior and interactions.

Chapter 4

Implementation

Contents

This chapter discusses the process of implementing our proposed solution in detail. It explains the critical decisions made during the process, the challenges encountered, and how we tackled them. We also provide a detailed overview of our solution's architecture, the technologies we used, and why we chose them. Additionally, we present a visual representation of our solution to give a clear understanding of its design and functionality. This chapter aims to provide a thorough understanding of the journey from the initial idea to the final solution, highlighting the complexities of its development. The implementation of our dashboard is public and available on GitHub[1] and also deployed as a standalone on Vercel[2]. Our dashboard is deployed publically for trials and demonstrative purposes, with the ability to use mock data for public use, thus not revealing sensitive information generated in CMF's environments. A demonstration video is also available on YouTube[3].

First, section 4.1 establishes the assumptions we must make and defines our domain of action carefully. Section 4.2 discusses our decisions and code design in detail, describing each component and providing an overall view. Following this, 4.3 describes the practical output of the custom dashboard, detailing its functionalities and the rationale behind its design. Section 4.4 is dedicated to discussing the flexibility of the scoring functions and how the dashboard behaves around them. Next, section 4.5 addresses the scalability of our solution as data grows and plans to evolve it grow

[1]`https://github.com/kiko-g/usability-dashboard-mes`
[2]`https://usability-dashboard-mes.vercel.app`
[3]`https://www.youtube.com/watch?v=5uJzntcn500`

as well, setting the stage for a later discussion on future work. Finally, section 4.6 summarizes the implementation chapter and draws some local conclusions.

4.1 Project scope

This research project centers on creating a methodology for assessing usability in SPAs through quantitative data. The tangible result is a dashboard that displays both traditional web analytics metrics and custom insights specific to our SPA of study.

Traditional insights encompass metrics such as page views, most visited routes, devices, operating systems, browsers, and user pathways. These metrics offer a general understanding of user behavior and interaction with the application.

Custom insights stem from tracking user interactions with transactional components within our specific SPA, CMF's MES. These components include wizards and execution views. We aim to quantify usability by evaluating and assigning these custom event scores. This approach offers a detailed perspective on user behavior and the usability of specific components. The custom dashboard facilitates an in-depth exploration of component statistics, including error rates, time spent within components, and success rates of interactions, among others.

However, this research has limitations. The methodology is tailored to CMF's MES and may not directly apply to other SPAs or web applications. Additionally, not all user interactions can be quantified, and some aspects of user experience may still require qualitative evaluation methods.

In conclusion, this research develops a usability assessment methodology for SPAs, resulting in a dashboard that presents a comprehensive view of user interactions and usability within a particular SPA.

4.1.1 MES Caveats

We must define CMF's MES in detail to position our research well and justify our decisions and reasoning. This MES is a complex system that is highly customizable, which complicates analysis in some ways. There is a learning curve with the usage of this system, and the end-users are trained to handle specific business data and rules. However, there is common ground to any installation of MES: there are core components that appear very recurringly in the UI, namely the transactional components: the **wizards** and **execution views**. Our primary focus is the wizards and the execution views tag along as they are similar in behavior but have slightly different natures.

It is vital to define exactly how these two components behave. Both wizards and execution views are UI components that lead the user through a sequence of steps or tasks to perform a specific action. In our case, the wizards have more detailed events and have the notion of steps - a sequence of actions where the user can go back and forth but must complete the steps in the prescribed order. Execution views, on the other hand, have tabs, and the user can complete them in any order. In the last step of a wizard, the user can submit. The amount of steps is variable inside the wizard. The lists with the events we register on these components can be seen in tables 4.1 and 4.2 for the wizards and execution views, respectively.

Wizard Events	
Start	The wizard is started
Close	The wizard is closed
Complete	The wizard is completed
Cancel	The wizard is canceled
Error	An error occurs during the wizard process
Activate Step	A step in the wizard is activated
Success Step	A step in the wizard is completed
Update Steps	Current steps or visible steps changed in the wizard
Fail Step	A step in the wizard fails
Next Step	Moves to the next step in the wizard

Table 4.1: CMF's MES Wizard Collected Events

Execution View Events	
Start	The execution view is started
Close	The execution view is closed
Complete	The execution view is completed
Cancel	The execution view is canceled
Error	An error occurs in the execution view
Fail Tab	A failure when attempting to submit a tab
Tab Change	The active tab in the execution view changes

Table 4.2: CMF's MES Execution View Collected Events

It is important to note that both components can be abandoned without closing, canceling, or completing. For example, if we reload the page when inside the wizard, we will not have a register for this action, so we assume that any wizard without complete, close, or cancel event was abandoned.

4.1.2 Assumptions & Considerations

When assessing the components with a usability score, we must make some discrimination in terms of how to grade the different events that happen within a wizard.

The previous section introduced all the events we track within wizards and execution views. Let us focus on the wizards, which are our primary concern overall. We have seen that there are several negative actions which are: **back clicks**, **errors**, and **failed steps**. Other adverse events involve not completing the wizard: **closing**, **canceling**, or simply **not completing**, where we assume the wizard is abandoned. These **negative events will be the basis of our assessment of wizards**, combined with some logic and some considerations. Section 4.4 talks about how we formulate our functions responsible for predicting a usability score, and subsection 4.4.1 points to the reasoning behind the coefficients of the formulas.

However, we must make several assumptions that still need to be clarified:

- **Step equality**: To simplify our research and to adapt to the time, we must assume that all steps require the **same effort** to complete, which is not true in practice. Measuring the effort to complete a step would be demanding and requires many considerations and touching up on intricacies in the MES code.
- **Wizard equality within group**: Another assumption we must make that does not necessarily mirror what happens in the application is considering that every wizard of the same kind (same name) is as challenging to complete as the others.
- **Wizard weight in average score**: CMF's MES contains wizards with just one step for simple actions. To simplify our study and adapt to time constraints, we will include any wizard collected regardless of the number of steps. However, this will likely skew the global average score for all wizards since a very difficult-to-complete and a simple wizard will be equally judged.

4.2 Architecture Overview

In this section, we present our reasoning for the technology choices that underpin our implementation.

4.2.1 Technology Choices

Our starting point is a SPA built with Angular and Typescript, which implies that we must find a **distributed node package** for metrics collection that seamlessly integrates with Angular. The chosen metrics collector must also align with our previously outlined requirements. Matomo emerged as the clear choice due to its free features that align with our needs while guaranteeing our analytics data to be our own, with no external parties looking in [17]. As we have described before, it is indispensable for CMF to have complete control over what happens to our analytics data and to protect user's privacy [17], so Matomo quickly became the front-runner to be our technology of choice for the metric collection aspect of the project.

We could choose Matomo because of the **ngx-matomo** package, available on **npmjs**. This package's latest version supports Angular 15 and newer [25], the current version installed in CMF's MES. The package is also in a public GitHub repository with helpful documentation, which was especially helpful for installing ngx-matomo without the angular CLI [24]. This is essentially a distribution of Matomo for Angular. It facilitates the default tracking procedures and allows us to inject tracking instructions into MES' components, enabling customization of our tracking parameters. Furthermore, Matomo offers a configurable dashboard presenting general metrics, analytics, and key performance indicators (KPIs) that are typically relevant to any SPA.

Matomo uses a database to store all its data and "only works on Mysql and MariaDB" [18]:

- **MySQL**: MySQL is a relational database (RDBMS) that was first released in 1995, during a time when Microsoft and Oracle's proprietary solutions dominated the market. As an open-source relational database management system rooted in SQL or Structured Query Language, MySQL stands among the most used databases worldwide [2].

- **MariaDB**: open-source fork of MySQL, created in 2009. It is a backward-compatible improved version of MySQL with various built-in features and security and execution improvements missing in MySQL. It supports the same features as MySQL and offers additional ones [2].

For simplicity and due to its reputation, we have chosen MySQL, as this decision does not significantly impact our project's overall direction.

While Matomo's dashboard provides a wealth of information, the custom events we aim to collect within transactional components require additional processing for meaningful interpretation. This need stimulates our next decision, incorporating an additional layer.

For this new layer, we chose to use Next.js, a powerful React framework by Vercel that is an industry standard trusted by many large companies due to its remarkable efficiency and powerful features. It is built with robust tooling and extends the latest React features to allow for the creation of full-stack web applications [28].

Here are some key strengths that made us opt for Next.js':

- Ability to build API endpoints, allowing secure connections with third-party services [28]
- Built-in optimization of images, fonts, and scripts, which significantly improves user experience and core web vitals [28].
- Flexible data fetching methods that run on the server and allow for versatile content rendering [28]
- Flexible rendering and caching options, including Incremental Static Regeneration (ISR) on a per-page level, contributing to its scalability [28].

The first feature in the list above is handy because, in our specific use case where the system needs to fetch and process data from the Matomo API, Next.js provides an **easy way to build a backend that acts as a middleware** in the grand scheme of things, enabling the creation of a frontend dashboard for data interaction. This seamless integration between the backend and frontend makes Next.js an ideal choice.

To maintain a tidy and readable codebase, our **Next.js installation uses TypeScript**. This decision aligns well with the existing MES' codebase that largely employs TypeScript, thus avoiding the introduction of superfluous technologies. Opting for JavaScript would be less advantageous for several reasons. TypeScript brings a variety of benefits that make it a superior choice over JavaScript. It introduces strong typing and type inference that enhance code readability and maintainability while preventing type-related errors and simplifying debugging [6]. Additionally, TypeScript provides robust tooling and integration options, reducing development time and ensuring higher code quality. It also introduces features like interfaces and classes, enabling a structured way of writing code. Overall, TypeScript paves the way for scalable, high-quality, and maintainable code that is easier to debug and test, making it an ideal choice for any scale of application development [6].

Although we referred to Next.js as a single layer in our solution, it essentially does more than that. One of Next.js' greatest strengths is how the code structure makes the barrier between a backend and a frontend very slim. However, the Next.js "backend" **acts as a middleware** in the greater scheme of things. This Next.js feature enables us to take care of two tasks simultaneously, thanks to its ease of webpage creation (custom dashboard frontend) and pre-configured API routes (middleware or custom dashboard backend). We can connect to our Matomo data via the Matomo API and use Next.js's API routes for these operations. This setup allows us to establish a middleware within Next.js that processes and sanitizes the data from Matomo, a crucial step for actions like grouping events by wizard and wizard type. We take a closer look into these details later in this chapter.

This is also very helpful since our Next.js frontend only needs to make requests upon page loading to its own built-in API, simplifying the maintenance and development of our pages. The HTTP responses from the middleware are carefully and adequately typed, ensuring our custom dashboard remains clean, readable, and type-safe.

For efficient and quick creation of flexible, concise, and easily maintainable styles and markup for our pages, we chose **TailwindCSS**, a technology that has earned a lot of praise in recent years, showing numerous advantages over traditional CSS, with minimal downsides. The main con with TailwindCSS is that it can lead to large HTML files [23]. Besides that, it provides excellent control and speed as it is a highly customizable utility-first framework, meaning we can dodge struggles with CSS classes with very similar styles for example [23]. It is also great performance-wise, as Tailwind automatically removes all unused CSS when building for production, which means the final CSS bundle is the smallest it could possibly be [29].

To bring our three services - **Matomo**, **MySQL**, and **Next.js** - together, we are going to use Docker[4], which allows us to package these services into containers, making sure they work together smoothly. These containers are easy to manage so that we can run our services locally without much setup. Plus, the containerization approach fits well with production environments, so it is easy to integrate our solution into CMF's existing setups. This approach simplifies the deployment process and improves our solution's scalability and reproducibility, vital for its long-term use and adaptability.

Having our decisions summarized and justified briefly, we are left with a simple the following technology stack:

- **Angular 15** using Typescript: the SPA under study, CMF's MES
- **Matomo**: ngx-matomo tailored for Angular
- **MySQL**: Matomo storage
- **Next.js** with Typescript+TailwindCSS: dashboard and middleware to process Matomo data
- **Docker**: compose configuration to put our services together

[4]`https://www.docker.com/`

4.2.2 High-Level Architecture

Considering the high-level architecture, we can visualize our technologies as a communication flow between components. The action sequence in our solution is straightforward: CMF's MES is where we want to track user interactions, so this is where Matomo generates data while the SPA is running. Matomo stores this data in MySQL and presents it in its standardized format on the Matomo dashboard. However, Matomo's capabilities extend beyond this, offering data access through its API. This API, accessible with an authentication token, allows us to retrieve data groups via HTTP requests. This data will be passed to our Next.js middleware, which understands the data format, parses it, and provides typed JSON responses. The middleware is then queried by the Next.js frontend, responsible for displaying the content understandably and interactively. This flow is represented below in figure 4.1.

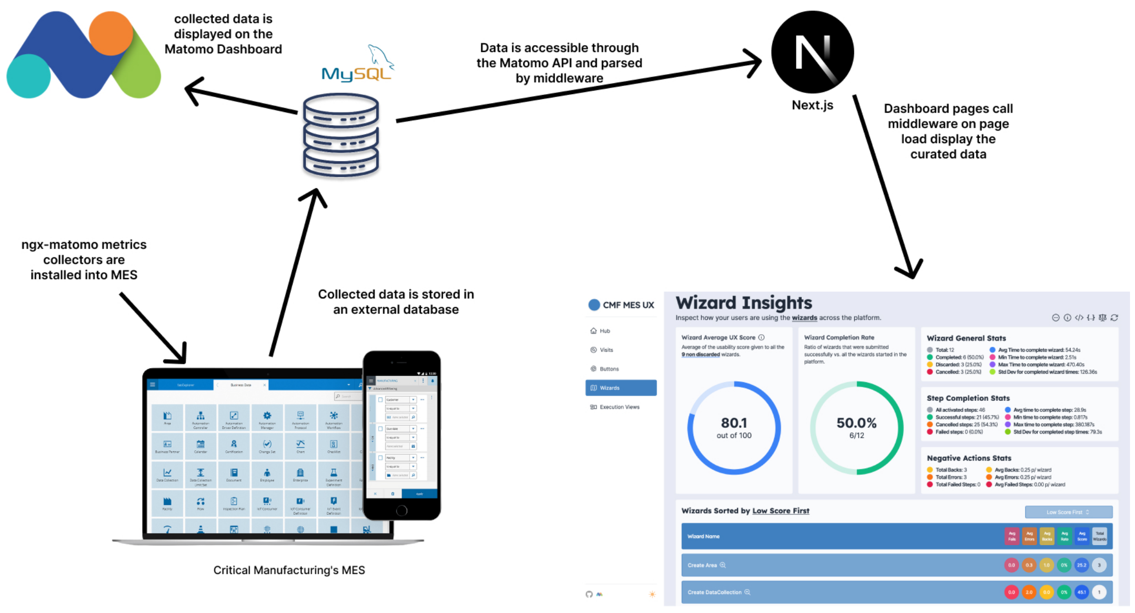
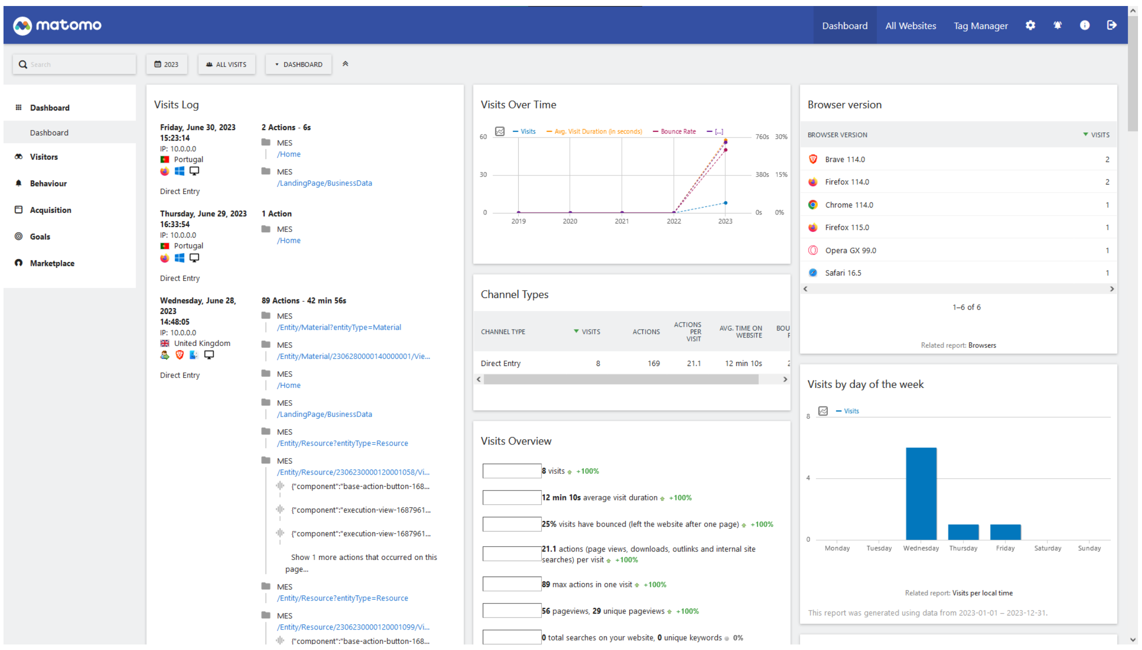

collected data is displayed on the Matomo Dashboard
MySQL
Data is accessible through the Matomo API and parsed by middleware
Next.js
Dashboard pages call middleware on page load display the curated data
ngx-matomo metrics collectors are installed into MES
Collected data is stored in an external database
Critical Manufacturing's MES
CMF MES UX
Wizard Insights
Inspect how your users are using the wizards across the platform.
Wizard Average UX Score
Wizard Completion Rate
Wizard General Stats
80.1
out of 100
50.0%
6/12
Step Completion Stats
Negative Actions Stats
Wizards Sorted by Low Score First
Wizard Name
Create Area
Create DataCollection

Figure 4.1: High-Level Architecture

4.2.3 Matomo within MES

Integrating Matomo within MES required careful consideration of how to utilize Matomo's features best. We added the package to the node project and configured Angular's **app module file** to include ngx-matomo, both the tracker and router packages. This modification already enables a significant amount of data tracking that comes with the package by default, allowing us to view traditional metrics in the Matomo dashboard, as seen in figure 4.2. This dashboard is customizable and allows us to select what metrics we wish to see, containing general information about the profile of the users.



Visits Log

Friday, June 30, 2023 15:23:14
IP: 10.0.0.0
Portugal
Direct Entry
2 Actions - 6s
MES /Home
MES /LandingPage/BusinessData

Thursday, June 29, 2023 16:33:54
IP: 10.0.0.0
Portugal
Direct Entry
1 Action
MES /Home

Wednesday, June 28, 2023 14:48:05
IP: 10.0.0.0
United Kingdom
Direct Entry
89 Actions - 42 min 56s
MES /Entity/Material?entityType=Material
MES /Entity/Material/2306280000140000001/Vie...
MES /Home
MES /LandingPage/BusinessData
MES /Entity/Resource?entityType=Resource
MES /Entity/Resource/2306230000120001058/Vi...
{"component":"base-action-button-168...
{"component":"execution-view-1687961...
{"component":"execution-view-1687961...
Show 1 more actions that occurred on this page...
MES /Entity/Resource?entityType=Resource
MES /Entity/Resource/2306230000120001099/Vi...
{"component":"base-action-button-168...

Visits Over Time

Channel Types

CHANNEL TYPE	VISITS	ACTIONS	ACTIONS PER VISIT	AVG. TIME ON WEBSITE	BOU
Direct Entry	8	169	21.1	12 min 10s	2

Visits Overview

8 visits +100%
12 min 10s average visit duration +100%
25% visits have bounced (left the website after one page) +100%
21.1 actions (page views, downloads, outlinks and internal site searches) per visit +100%
89 max actions in one visit +100%
56 pageviews, 29 unique pageviews +100%
0 total searches on your website, 0 unique keywords 0%

Browser version

BROWSER VERSION	VISITS
Brave 114.0	2
Firefox 114.0	2
Chrome 114.0	1
Firefox 115.0	1
Opera GX 99.0	1
Safari 16.5	1

1–6 of 6

Related report: Browsers

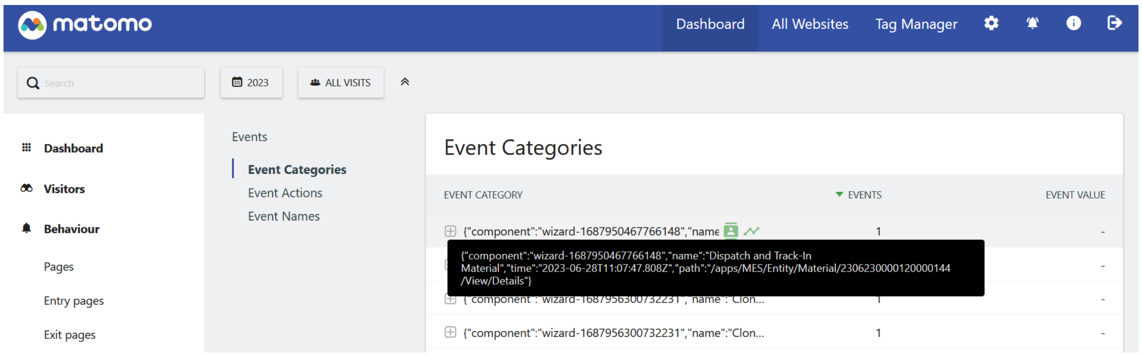
Visits by day of the week

Related report: Visits per local time

This report was generated using data from 2023-01-01 – 2023-12-31.

Figure 4.2: Matomo Default Customizable Dashboard

However, the real challenge lies in determining how to track events and serialize, group, and assess them later. As we have seen in section 4.2, the events we collect have an extra layer of processing after being collected, and figure 4.3 justifies this decision even further, as we can only see the events individually and require serious data manipulation to make sense of these events.



Event Categories

EVENT CATEGORY	EVENTS	EVENT VALUE
{"component":"wizard-1687950467766148","name	1	-
		-
{"component":"wizard-1687956300732231","name":"Clon...	1	-
{"component":"wizard-1687956300732231","name":"Clon...	1	-

{"component":"wizard-1687950467766148","name":"Dispatch and Track-In Material","time":"2023-06-28T11:07:47.808Z","path":"/apps/MES/Entity/Material/2306230000120000144/View/Details"}

Figure 4.3: Matomo Dashboard Events Categories Display

Matomo allows event tracking by injecting a line of code with four arguments: category, action, name, and value. If we were to define a type for this event, it would look like this:

```
type MatomoEvent = {
  category: string;
  action?: string;
  name?: string;
  value?: number;
};
```

This implies that only the category is mandatory. While the other fields could be helpful, they can limit how we store our event information. We will discard the name and value for our needs and use category and action. Despite only tracking these two fields for each event, we will use a workaround to carry more information: we can perform `JSON.stringify()` on the information in a consistent format, where we track a **unique ID for the component where the event occurs**, the **name of the component**, the **time of the event**, and the **current route where the event occurs**. We can create a **getter** method to retrieve the category every time we register an event:

```
public get trackerCategory() {
  return JSON.stringify({
    component: this.trackerUniqueId,
    name: this.title,
    time: new Date().toISOString(),
    path: typeof window !== 'undefined' ? window.location.pathname : ''
  });
}
```

Here is what tracking an event for a newly activated step looks like:

```
this.matomoTracker.trackEvent(this.bag.trackerCategory, `Activate Step:
 ↪  ${step.id}`);
```

This approach allows us to correctly parse the information later when we access the Matomo API. It is a matter of keeping track of the types of these variables and ensuring consistency on both sides of the code. However, this does mean that the events displayed in the Matomo dashboard may need to be more easily interpretable by a human reader. We are prepared to accept this trade-off, as our primary focus is on something other than consulting event information via the default dashboard.

4.2.4 Middleware with Next.js

As stated before, we used the Next.js API routes infrastructure to create a middleware that performs intermediate processing on the data from the Matomo API. Every file we create under the folder `pages/api` will create a route with the name `/api/path-to-route`.

The first step in creating this middleware is establishing a connection to the Matomo API, a routine used by all the Matomo routes. Our environment variables allow us to configure the Matomo interaction within Next.js.

```
export const config = {
  matomoToken: process.env.NEXT_PUBLIC_MATOMO_TOKEN,
  matomoSiteId: process.env.NEXT_PUBLIC_MATOMO_SITE_ID,
  matomoSiteUrl: process.env.NEXT_PUBLIC_MATOMO_SITE_URL,
};
```

The next step is to build a string with the correct format to make an HTTP Request to the Matomo service. Let us start with the most basic request, which is to query the Matomo version using the `API.getMatomoVersion` method [16]:

```
const apiUrl = `${config.matomoSiteUrl}/index.php?module=API
    &method=API.getMatomoVersion
    &format=json
    &token_auth=${config.matomoToken}`;
```

Every file we create creates a similar URL, varying only the parameters we pass in this string according to the functionality of each route. Alternatively, let us look at the parameters passed into the string for fetching all the events in the platform between two dates:

```
const period = 'range'; // day, week, month, year, range
const date = `2023-04-29,today`; // YYYY-MM-DD
const apiUrl = `${config.matomoSiteUrl}/index.php?module=API
    &method=Events.getCategory
    &secondaryDimension=eventAction
    &flat=1
    &format=json
    &idSite=${config.matomoSiteId}
    &period=${period}
    &date=${date}
    &token_auth=${config.matomoToken}
    &filter_limit=-1`;
```

Once we parse the response into JSON, we are interested in three parameters of our array entries, which are:

- `EventsEventTime`
- `EventsEventCategory`
- `EventsEventAction`

We use the event time to sort the events by ascending time of occurrence, the category for parsing the unique customized MES information, described in 4.2.3, and the action contains a string with the action description. We were careful with the JSON *stringified* content tracking category in section 4.2.3, so we can create a type on the Next.js side:

```
export interface ITrackerEventRawCategory {
  component: string;
  name: string;
  time: string;
  path: string;
}
```

With the usage of this type, we can safely parse the information in the category of the `event` and take advantage of the TypeScript linting:

```
const category = JSON.parse(event.Events_EventCategory) as
↪  ITrackerEventRawCategory;
const parsedEvent = { ...category, action: event.Events_EventAction } as
↪  ITrackerEventRawEvent;
```

The parsed event uses the same type with the simple addition of the action parameter:

```
export interface ITrackerEventRawEvent extends ITrackerEventRawCategory {
  action: string;
}
```

From this point, we can continue our parsing by grouping the events. The `component` variable inside the category allows us to join the events into arrays of the same component and identify its name. Ultimately, the parsing procedure returns an array with all the events grouped by component. Each group has a component identifier, a name, and an array of events. This array is of type `ITrackerEventGroup`, which is defined below:

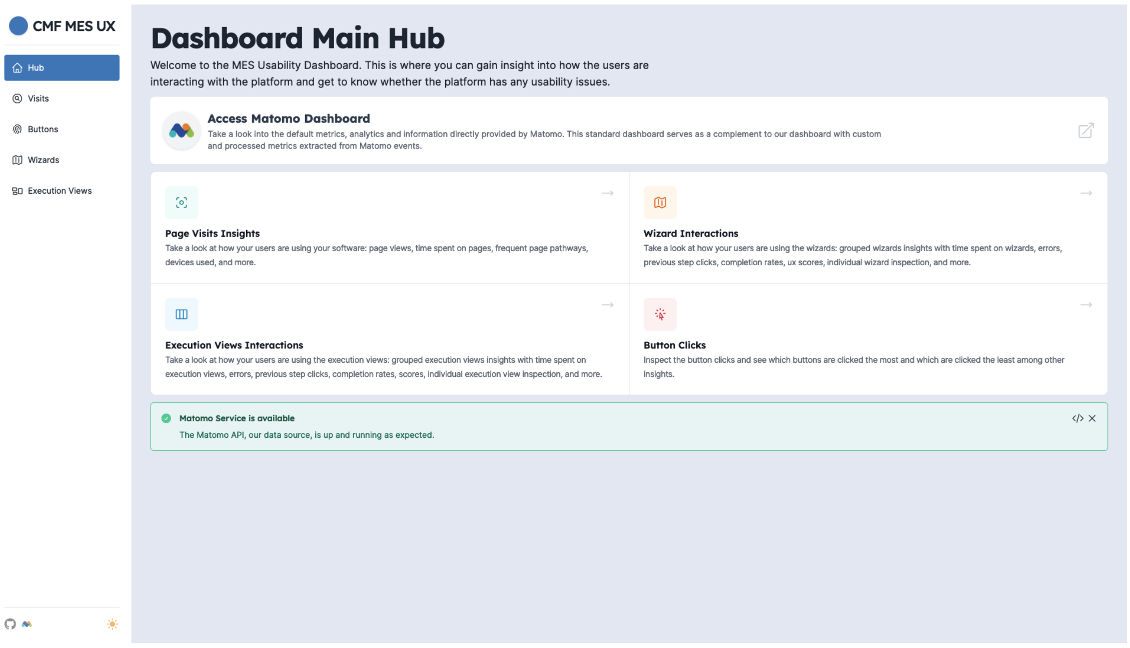
```
export interface ITrackerEventGroup {
  component: string;
  name: string;
  events: ITrackerEvent[];
}

export interface ITrackerEvent {
  time: string;
  path: string;
  action: string;
}
```

That said, we are now equipped to score the events based on assumptions and decisions. Information related to the evaluation is described based on assumptions made in 4.1.2, and the scoring process is carefully detailed in 4.4.

4.2.5 Dashboard with Next.js

Next.js allows us to create routes and pages automatically by adding a file to the `pages` folder and making it export a React component.

We shall start by presenting the home page, which can be seen in figure 4.4 below.



Dashboard Main Hub

Welcome to the MES Usability Dashboard. This is where you can gain insight into how the users are interacting with the platform and get to know whether the platform has any usability issues.

Access Matomo Dashboard
Take a look into the default metrics, analytics and information directly provided by Matomo. This standard dashboard serves as a complement to our dashboard with custom and processed metrics extracted from Matomo events.

Page Visits Insights
Take a look at how your users are using your software: page views, time spent on pages, frequent page pathways, devices used, and more.

Wizard Interactions
Take a look at how your users are using the wizards: grouped wizards insights with time spent on wizards, errors, previous step clicks, completion rates, ux scores, individual wizard inspection, and more.

Execution Views Interactions
Take a look at how your users are using the execution views: grouped execution views insights with time spent on execution views, errors, previous step clicks, completion rates, scores, individual execution view inspection, and more.

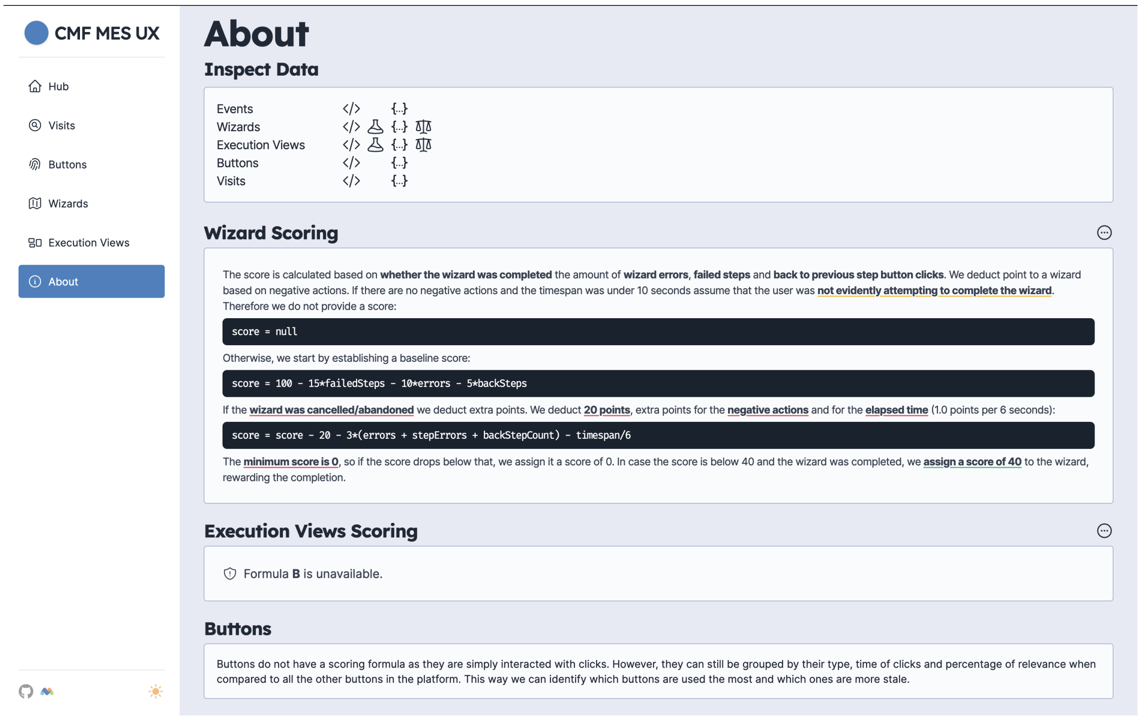
Button Clicks
Inspect the button clicks and see which buttons are clicked the most and which are clicked the least among other insights.

Matomo Service is available
The Matomo API, our data source, is up and running as expected.

Figure 4.4: Dashboard Home Page

As we can see, this is a landing page that directs the user to the core functionalities of our solution: Matomo dashboard outer link followed by a link to the inner pages for the user visits and general stats, wizards, execution views, and button insights. We are also notified about the status of the **Matomo Service**, which indicates whether our configuration to communicate with the Matomo API is successful.

Our **about page** can be seen in figure 4.5 below. Any information regarding the methodology and functionality of the custom insights pages belongs here. We created this page primarily to offer easy client-only access to information about the scoring approaches for the transactional components. However, we also made a section at the top for users to inspect raw data.



About

Inspect Data

Events	</> {..}
Wizards	</> {..}
Execution Views	</> {..}
Buttons	</> {..}
Visits	</> {..}

Wizard Scoring

The score is calculated based on **whether the wizard was completed** the amount of **wizard errors**, **failed steps** and **back to previous step button clicks**. We deduct point to a wizard based on negative actions. If there are no negative actions and the timespan was under 10 seconds assume that the user was **not evidently attempting to complete the wizard**. Therefore we do not provide a score:

```
score = null
```

Otherwise, we start by establishing a baseline score:

```
score = 100 - 15*failedSteps - 10*errors - 5*backSteps
```

If the **wizard was cancelled/abandoned** we deduct extra points. We deduct **20 points**, extra points for the **negative actions** and for the **elapsed time** (1.0 points per 6 seconds):

```
score = score - 20 - 3*(errors + stepErrors + backStepCount) - timespan/6
```

The **minimum score is 0**, so if the score drops below that, we assign it a score of 0. In case the score is below 40 and the wizard was completed, we **assign a score of 40** to the wizard, rewarding the completion.

Execution Views Scoring

Formula **B** is unavailable.

Buttons

Buttons do not have a scoring formula as they are simply interacted with clicks. However, they can still be grouped by their type, time of clicks and percentage of relevance when compared to all the other buttons in the platform. This way we can identify which buttons are used the most and which ones are more stale.

Figure 4.5: Dashboard About Page

The **Wizard and Execution View scoring** contains the scoring approach description. Users can select the desired formula for the components by clicking on the three-dot button on the right and making their selection. The top section, **Inspect Data**, allows the user to interact with the data on the website. Each row of buttons has functionality for its category:

- **API route**: Shows the source JSON for the data from the API in a new tab.
- **See evaluated data from API route**: This button is only available for the wizards and execution views and shows the evaluated source JSON for the data from the API in a new tab.
- **See mock data**: Shows the mock JSON data in a new tab.
- **See evaluated mock data**: This button is only available for the wizards and execution views. Shows the evaluated mock JSON data in a new tab.

Moving on to the technical side, we have **four other pages** that require using the Matomo API. These four pages are showcased in detail in the next section, 4.3. As mentioned in the previous subsection 4.2.4, the logic for interacting with the Matomo API is implemented in the middleware files in the `api` folder. However, the assessment of the events is done entirely on **client side**. Our "technical pages" fetch the information from the middleware API route; if needed, the additional processing is performed immediately. In appendix A, we can see client-side data fetching in the React `useEffect` procedure inside the wizards' page. This code is called upon page load and also if the user requests a re-fetch.

4.3 Dashboard Functionality & Custom Insights

This section showcases the details of our custom-built dashboard, created using Next.js, describing its creation process, the features it offers, and the reasons behind its inclusion. We also examine each significant dashboard component, highlighting the key features that hold potential value for CMF. This exploration presents the dashboard's functionality and role in our solution.

Before getting into the specific pages and their features, we must provide information about the general details and structure of the pages. On the top right of every technical page, there is a section with buttons like seen in figure 4.6 below.

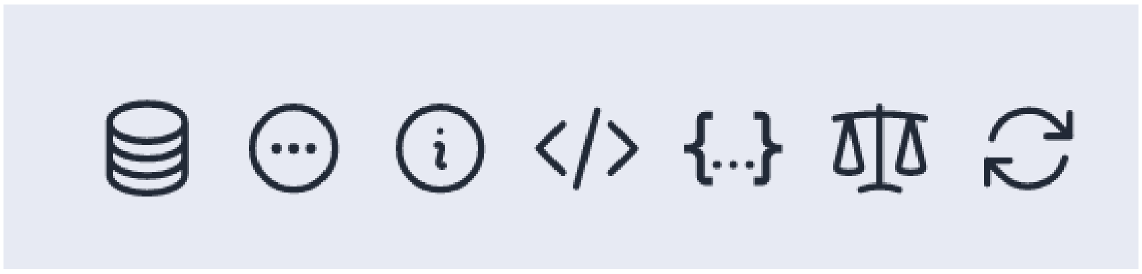


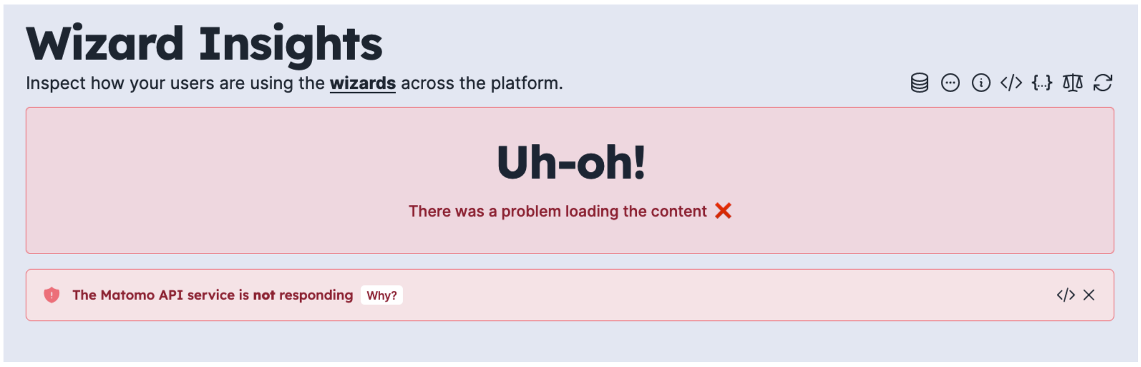
Figure 4.6: Control Buttons and Options

These buttons provide additional information and control over the data, allowing the user to have very transparent access to what is happening on the page. Here are the button functionality descriptions from left to right:

- **Use Mock Data**: Becomes available if the page fails to fetch the analytics data for the page. This button is handy for demonstration purposes because even if the fetching fails or is unavailable, it shows the dashboard features on hard-coded data.
- **Select Formula**: This button, with the horizontal ellipsis circle icon, allows the user to select a formula to change the scoring approach if the components in the page are assessed (e.g., wizards).
- **Formula Information**: This button opens a modal with information related to the currently selected formula describing the scoring approach.
- **API Source Response**: This button opens a new tab with the API route, attempting to provide the JSON response that the page needs to display analytics.

- **Raw Data**: This button allows the user to consult the full JSON to display the analytics if the loading has been completed.
- **Evaluated Data**: If the data has client-side assessment (e.g., wizards page), the user can inspect the processed JSON used to display the analytics.
- **Attempt Reload**: This button fetches data again.

As we have seen above, the information on the technical pages comes from the Next.js back-end, which means we need to make an HTTP request upon page load, and that fetching procedure can fail. If it does, the page will display an error message and make the use mock data button available, as seen in figure 4.7 below.


Wizard Insights

Inspect how your users are using the **wizards** across the platform.

Uh-oh!

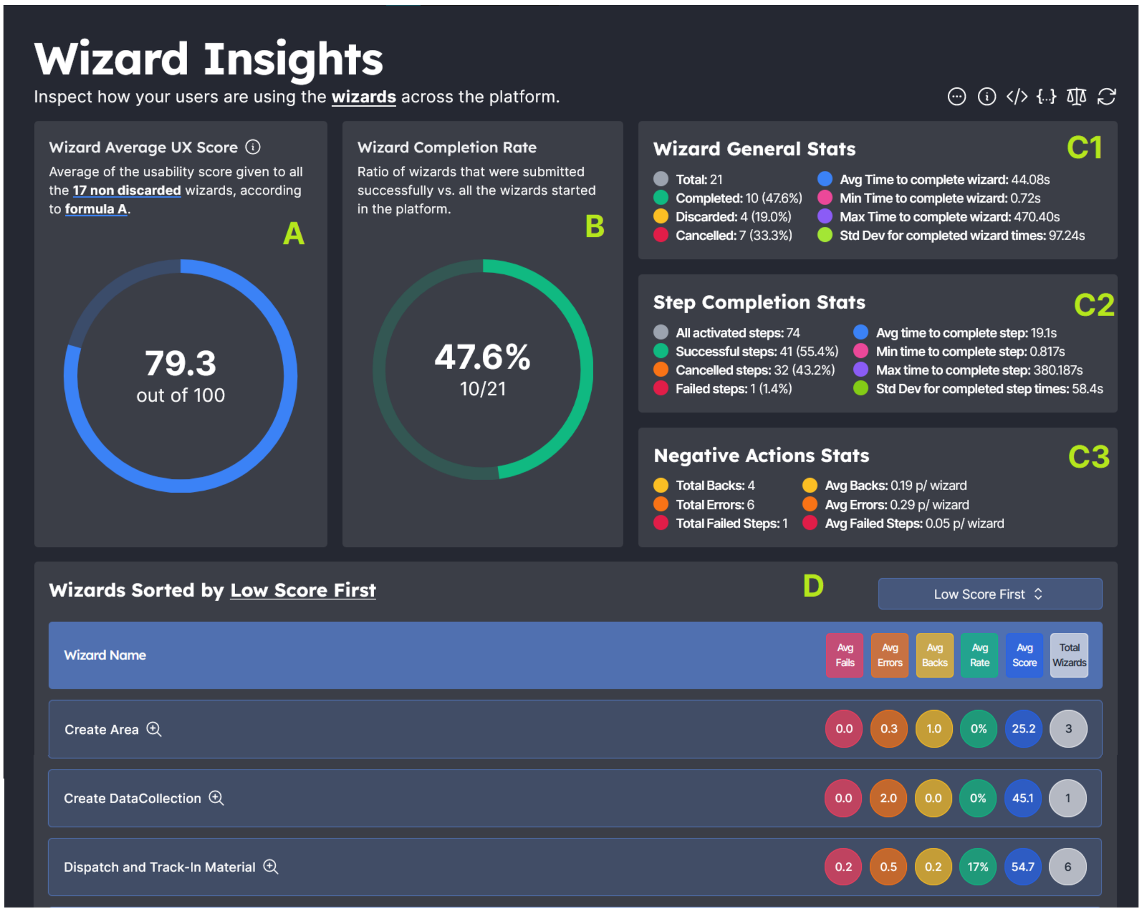
There was a problem loading the content ✕

The Matomo API service is not responding Why?

Figure 4.7: Data Fetching Error Feedback

4.3.1 Wizards

This subsection explains the most noteworthy part of the dashboard - the wizards' insights page. Wizards are likely to be the most frequently used components, and this section allows for interaction with data from previous MES interactions. A snapshot of the wizards' page, populated with some data generated locally in development, can be seen in figure 4.8.



Wizard Insights

Inspect how your users are using the wizards across the platform.

Wizard Average UX Score

Average of the usability score given to all the 17 non discarded wizards, according to formula A.

A

79.3
out of 100

Wizard Completion Rate

Ratio of wizards that were submitted successfully vs. all the wizards started in the platform.

B

47.6%
10/21

Wizard General Stats C1

- **Total:** 21
- **Completed:** 10 (47.6%)
- **Discarded:** 4 (19.0%)
- **Cancelled:** 7 (33.3%)
- **Avg Time to complete wizard:** 44.08s
- **Min Time to complete wizard:** 0.72s
- **Max Time to complete wizard:** 470.40s
- **Std Dev for completed wizard times:** 97.24s

Step Completion Stats C2

- **All activated steps:** 74
- **Successful steps:** 41 (55.4%)
- **Cancelled steps:** 32 (43.2%)
- **Failed steps:** 1 (1.4%)
- **Avg time to complete step:** 19.1s
- **Min time to complete step:** 0.817s
- **Max time to complete step:** 380.187s
- **Std Dev for completed step times:** 58.4s

Negative Actions Stats C3

- **Total Backs:** 4
- **Total Errors:** 6
- **Total Failed Steps:** 1
- **Avg Backs:** 0.19 p/ wizard
- **Avg Errors:** 0.29 p/ wizard
- **Avg Failed Steps:** 0.05 p/ wizard

Wizards Sorted by Low Score First D

Low Score First

Wizard Name	Avg Fails	Avg Errors	Avg Backs	Avg Rate	Avg Score	Total Wizards
Create Area	0.0	0.3	1.0	0%	25.2	3
Create DataCollection	0.0	2.0	0.0	0%	45.1	1
Dispatch and Track-In Material	0.2	0.5	0.2	17%	54.7	6

Figure 4.8: Wizard Insights Page Snapshot with Sections

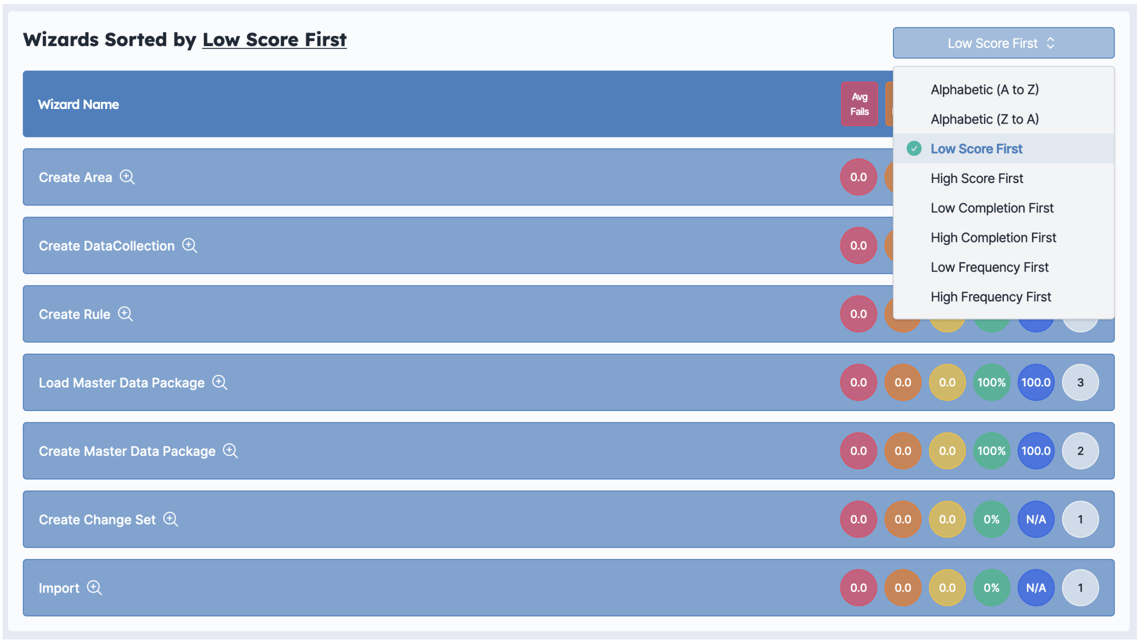
Let us dissect the various sections that make up this page. Each section is marked on the figure above with a letter to associate with this list:

- **Wizard UX Score Card (A)**: This card displays the average score for all non-discarded wizards across all types. The information circle icon button details the scoring formula currently in use.
- **Wizard Completion Rate (B)**: This section indicates the ratio of completed wizards, contrasting completed wizards and total wizards initiated.
- **Wizard General Stats (C1)**: This section presents core stats related to completing wizards of all types and the time spent on these components. It provides four entries for completion (total wizards, discarded, canceled, and completed) and four entries for time stats (average time to complete a wizard, the standard deviation of time to complete a wizard, and the minimum and maximum time to complete a wizard).
- **Step Completion Stats (C2)**: Same stats as those in C1 but applied to all the steps within the wizards.
- **Negative Action Stats (C3)**: This section highlights six stats related to what we consider to be negative actions on the wizards. We count errors, failed steps, and back-to-previous step clicks and show the frequency in which they occur per wizard.

- **Sorted Wizards (D)**: This section presents the wizards grouped by their type, allowing us to immediately see the six key stats related to each wizard group: average failed steps, average errors, average back-to-previous step clicks, completion rate, average UX score, and the total of wizards in the category. We can also sort the wizard groups by multiple criteria. Each row is a wizard group, and the rows are clickable, allowing us to inspect each and get a unique view for every group of wizards.

The stats in sections A, B, C1, C2, and C3 provide a general overview of the behavior within wizards. In contrast, section D allows the user to delve deeper into the interaction inside each type of wizard and inspect each initiated wizard of each type.

Section D requires further explanation, as it is undoubtedly the page's most complex and interactive section. It allows us to inspect each interaction by wizard type and is interactive and detailed. First, let us describe the implemented filters to sort the wizards grouped by type, which can be seen in figure 4.9 below.



Wizards Sorted by Low Score First

Wizard Name	Avg Fails					
Create Area	0.0					
Create DataCollection	0.0					
Create Rule	0.0					
Load Master Data Package	0.0	0.0	0.0	100%	100.0	3
Create Master Data Package	0.0	0.0	0.0	100%	100.0	2
Create Change Set	0.0	0.0	0.0	0%	N/A	1
Import	0.0	0.0	0.0	0%	N/A	1

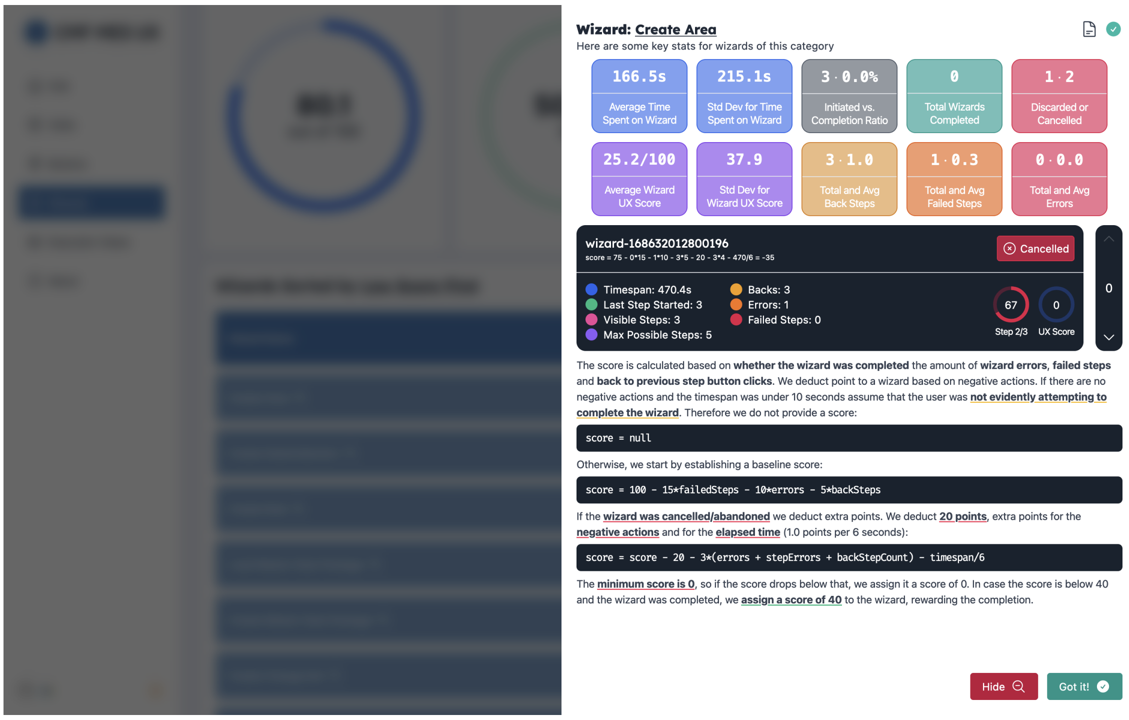
Figure 4.9: Wizard Group Insights Sort Filters

We created filters for our wizard types for clarity when consulting the data. This filter provides better means of consulting the information. According to our formulas for assessing usability, the most important filter is sorting by the low score, highlighting the underperforming wizards. This is crucial for making recommendations because, once the data grows, we can expect each wizard type to reveal its performance, thus enabling the developer teams to make recommendations on the UI.

Figure 4.9 shows eight filters which let us filter information based on four categories in descending and ascending order. These filters provide a manner of reading through the information based on different priorities. Here are the scenarios in which these filters can help:

- **Alphabetic**: General wizard group consultation is more human-friendly when sorting based on the alphabetic order.
- **Score**: Assuming we trust our formula, the score ranks our best and worst-performing wizards.
- **Completion**: We can see which wizards are being completed successfully versus those not.
- **Frequency**: We can be in touch with the wizards that have more and less relevance in the MES installation.

Moving on to another key feature of our Next.js dashboard, the user can inspect each wizard group by clicking on its row. As seen in figure 4.10 below, this view makes a summary of the performance of the wizards of this category and allows the user to examine every wizard in it.



Wizard: Create Area
Here are some key stats for wizards of this category

166.5s	215.1s	3 · 0.0%	0	1 · 2
Average Time Spent on Wizard	Std Dev for Time Spent on Wizard	Initiated vs. Completion Ratio	Total Wizards Completed	Discarded or Cancelled
25.2/100	37.9	3 · 1.0	1 · 0.3	0 · 0.0
Average Wizard UX Score	Std Dev for Wizard UX Score	Total and Avg Back Steps	Total and Avg Failed Steps	Total and Avg Errors

wizard-168632012800196
score = 75 - 0*15 - 1*10 - 3*5 - 20 - 3*4 - 470/6 = -35

Timespan: 470.4s
Last Step Started: 3
Visible Steps: 3
Max Possible Steps: 5
Backs: 3
Errors: 1
Failed Steps: 0

The score is calculated based on **whether the wizard was completed** the amount of **wizard errors**, **failed steps** and **back to previous step button clicks**. We deduct point to a wizard based on negative actions. If there are no negative actions and the timespan was under 10 seconds assume that the user was **not evidently attempting to complete the wizard**. Therefore we do not provide a score:

```
score = null
```

Otherwise, we start by establishing a baseline score:

```
score = 100 - 15*failedSteps - 10*errors - 5*backSteps
```

If the **wizard was cancelled/abandoned** we deduct extra points. We deduct **20 points**, extra points for the **negative actions** and for the **elapsed time** (1.0 points per 6 seconds):

```
score = score - 20 - 3*(errors + stepErrors + backStepCount) - timespan/6
```

The **minimum score is 0**, so if the score drops below that, we assign it a score of 0. In case the score is below 40 and the wizard was completed, we **assign a score of 40** to the wizard, rewarding the completion.

Figure 4.10: Wizard Group (Create Area) Focus View

In the top section of this view in figure 4.11, there are ten stat cards at the top, which expand the insights compared to the page view, providing the following stats (in order from left to right in figure 4.11):

- **Average time** to complete a wizard of this type, "Create Area."
- **time standard deviation** to complete a wizard of this type (needs more than two wizards).
- All **initiated wizards** coupled with the **conversion rate** (successfully submitted ratio).
- **Total wizards completed** (successfully submitted).
- **Total discarded wizards** coupled with **canceled wizards**.
- **Average UX Score** for this wizard group.

- **UX Score standard deviation** for this wizard group (needs more than two wizards).
- **Total back to previous step clicks** coupled with the **average back to previous step clicks** in this wizard group.
- **Total failed steps** coupled with the **average amount of failed steps** in this wizard group.
- **Total errors** coupled with the **average amount of errors** in this wizard group.

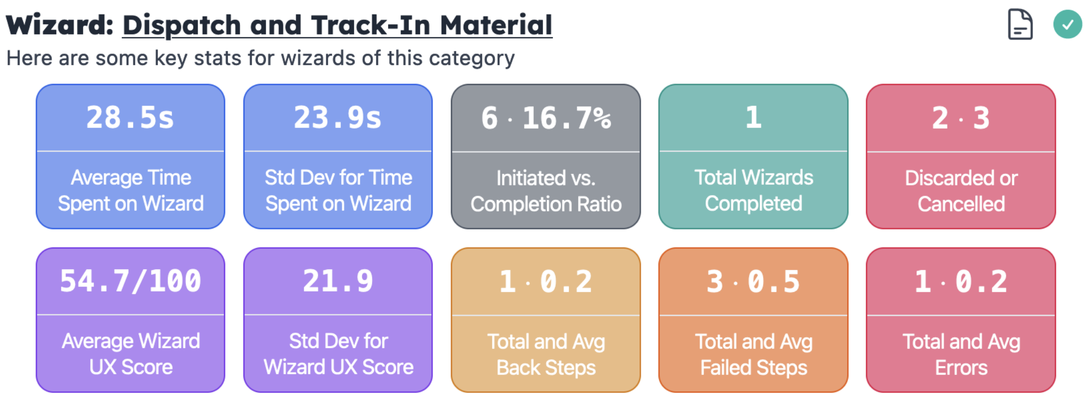

Wizard: Dispatch and Track-In Material
Here are some key stats for wizards of this category
28.5s
Average Time Spent on Wizard
23.9s
Std Dev for Time Spent on Wizard
6 · 16.7%
Initiated vs. Completion Ratio
1
Total Wizards Completed
2 · 3
Discarded or Cancelled
54.7/100
Average Wizard UX Score
21.9
Std Dev for Wizard UX Score
1 · 0.2
Total and Avg Back Steps
3 · 0.5
Total and Avg Failed Steps
1 · 0.2
Total and Avg Errors

Figure 4.11: Wizard Group Focus 10 Stats

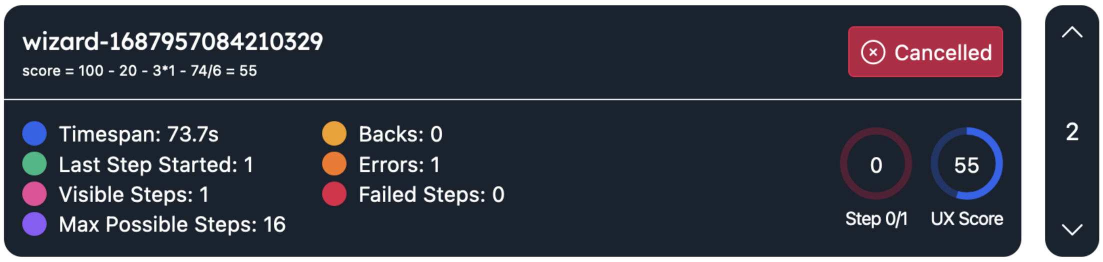
Below these cards, the user can find a larger card that showcases a selected wizard of the chosen group and inspect metrics about its interaction. As can be seen, there are seven listed stats on the left side of figure 4.12:

- **Timespan**: Time spent in wizard
- **Last Step Started**: Index of the last started step (minimum is 1).
- **Visible Steps**: Amount of steps visible in the wizard upon the last collected event.
- **Maximum Possible Steps**: The maximum steps this type of wizard can have.
- **Backs**: Amount of back to previous step clicks.
- **Fails**: Amount of failed steps.
- **Errors**: Amount of errors.

There are also two extra key analytics on the right side of figure 4.12:

- **Progress**: Progress made on the wizard, based on the last started step and visible steps ratio.
- **UX Score**: Associated usability score to this individual wizard.

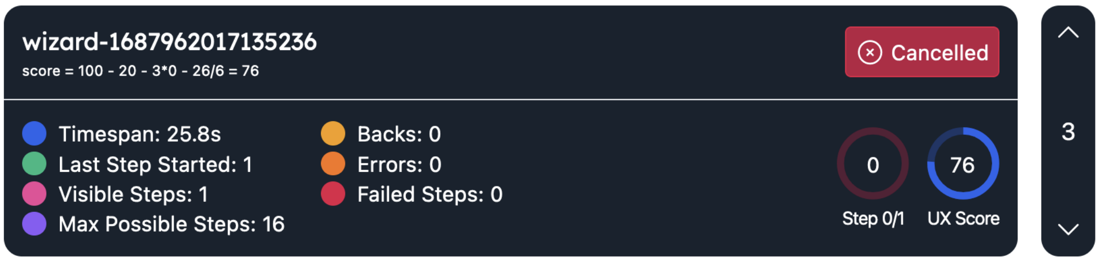
In addition to these stats, looking at the top section of the card in figure 4.12, we can also see the **unique identifier of the wizard**, whether it was **discarded, canceled or completed**, alongside the equation that **calculates the UX score before correction**. This inspection card also features a scroll bar-like button component on the rightmost of figure 4.12, which allows the user to navigate through the collected wizards of this group.


wizard-1687957084210329
score = 100 - 20 - 3*1 - 74/6 = 55

Timespan: 73.7s
Last Step Started: 1
Visible Steps: 1
Max Possible Steps: 16
Backs: 0
Errors: 1
Failed Steps: 0

Figure 4.12: Individual Wizard Stats Card

Figure 4.13 shows the state of the focused view after clicking the arrow down, which shows us the following wizard (with index 3) initiated in this group.


wizard-1687962017135236
score = 100 - 20 - 3*0 - 26/6 = 76

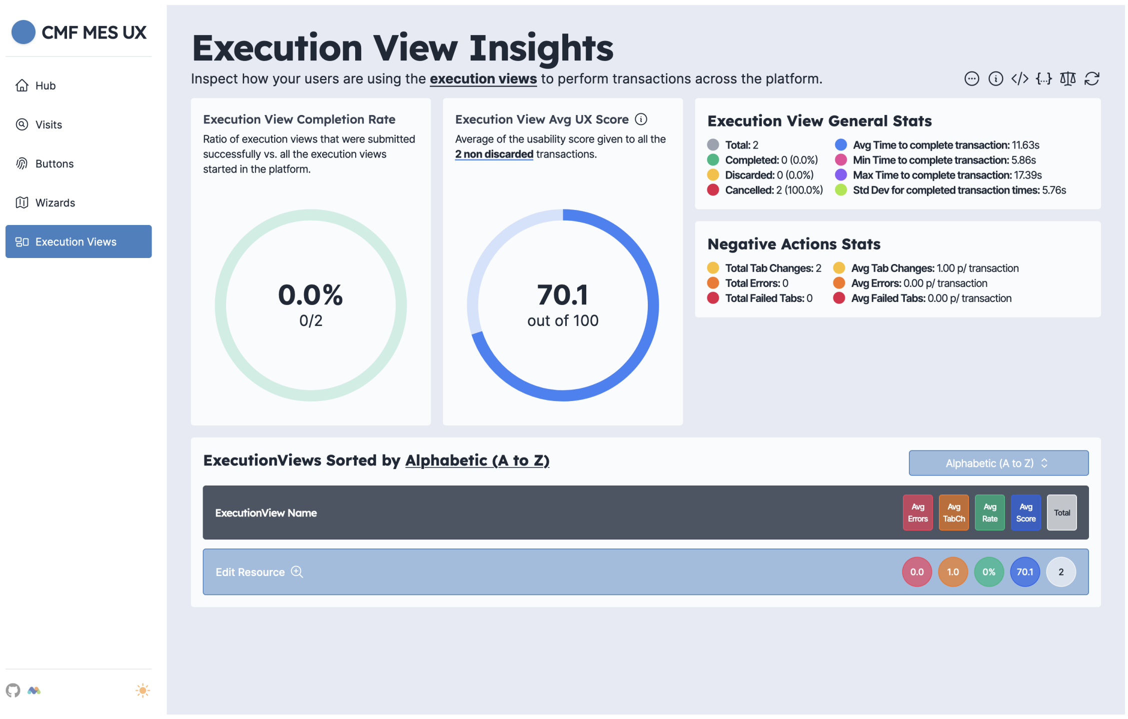
Timespan: 25.8s
Last Step Started: 1
Visible Steps: 1
Max Possible Steps: 16
Backs: 0
Errors: 0
Failed Steps: 0

Figure 4.13: Individual Wizard Stats Card Scrolled

Finally, we describe the formula to score the currently employed wizards in this wizard group inspection view. This section changes based on the formula that is selected in the UI.

4.3.2 Execution Views

The execution views page can be seen in figure 4.14 below. It replicates what the wizards' page does similarly since the components are transactional and similar in their events. However, this page requires some changes: execution views do not have back to the previous steps. Instead, they have tab changes. This event does not map identically as previously described because the tab change is not necessarily a negative action. For the sake of simplicity, we are treating it like that on this page.



CMF MES UX

Hub
Visits
Buttons
Wizards
Execution Views

Execution View Insights

Inspect how your users are using the **execution views** to perform transactions across the platform.

Execution View Completion Rate
Ratio of execution views that were submitted successfully vs. all the execution views started in the platform.
0.0%
0/2

Execution View Avg UX Score
Average of the usability score given to all the **2 non discarded** transactions.
70.1
out of 100

Execution View General Stats
- **Total:** 2
- **Completed:** 0 (0.0%)
- **Discarded:** 0 (0.0%)
- **Cancelled:** 2 (100.0%)
- **Avg Time to complete transaction:** 11.63s
- **Min Time to complete transaction:** 5.86s
- **Max Time to complete transaction:** 17.39s
- **Std Dev for completed transaction times:** 5.76s

Negative Actions Stats
- **Total Tab Changes:** 2
- **Total Errors:** 0
- **Total Failed Tabs:** 0
- **Avg Tab Changes:** 1.00 p/ transaction
- **Avg Errors:** 0.00 p/ transaction
- **Avg Failed Tabs:** 0.00 p/ transaction

ExecutionViews Sorted by Alphabetic (A to Z)

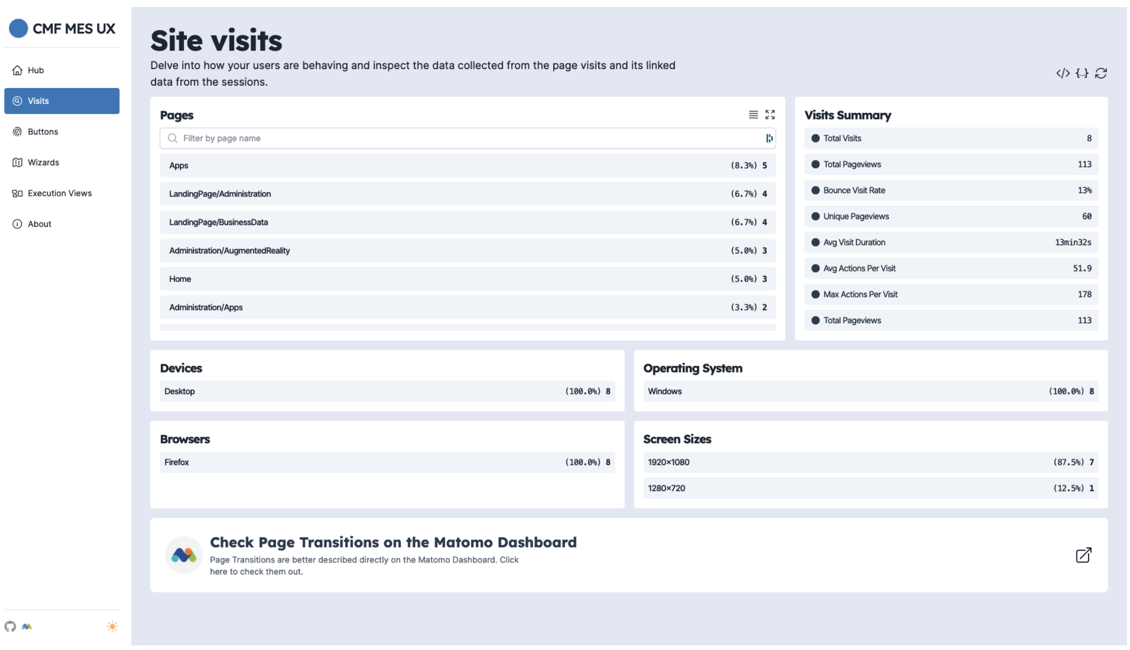
Alphabetic (A to Z)

ExecutionView Name	Avg Errors	Avg TabCh	Avg Rate	Avg Score	Total
Edit Resource	0.0	1.0	0%	70.1	2

Figure 4.14: Execution Views Insights Page

4.3.3 Visits

The visits page offers an additional, concise view of what Matomo mainly provides out of the box, as shown in figure 4.15 below. We have an interactable **pages** frequency list, a **visits summary** where some general statistics make their way into this page, a section for **device, operating system, browser, and screen size** usage frequency and we also outsource the **pathway visualization** (pathways are a sequence of visited user routes). Here is the full view of the page:



Site visits

Delve into how your users are behaving and inspect the data collected from the page visits and its linked data from the sessions.

Pages

Apps	(8.3%) 5
LandingPage/Administration	(6.7%) 4
LandingPage/BusinessData	(6.7%) 4
Administration/AugmentedReality	(5.0%) 3
Home	(5.0%) 3
Administration/Apps	(3.3%) 2

Visits Summary

Total Visits	8
Total Pageviews	113
Bounce Visit Rate	13%
Unique Pageviews	60
Avg Visit Duration	13min32s
Avg Actions Per Visit	51.9
Max Actions Per Visit	178
Total Pageviews	113

Devices

Desktop	(100.0%) 8

Operating System

Windows	(100.0%) 8

Browsers

Firefox	(100.0%) 8

Screen Sizes

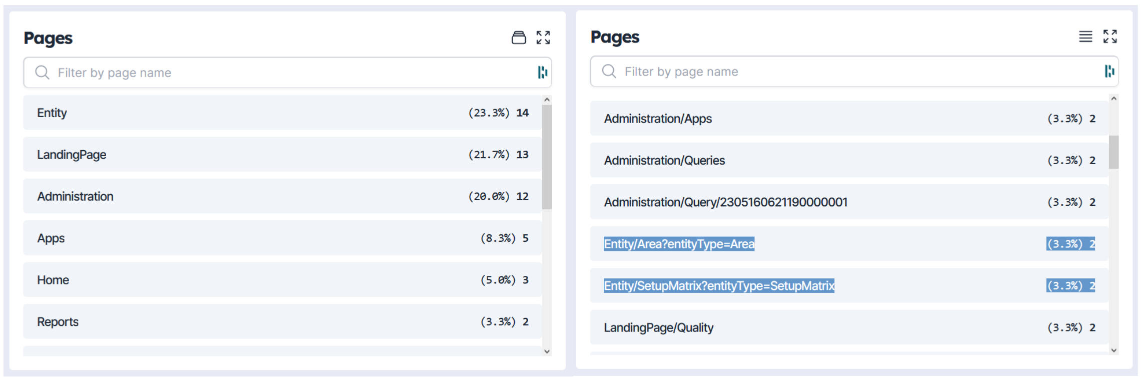
1920×1080	(87.5%) 7
1280×720	(12.5%) 1

Check Page Transitions on the Matomo Dashboard

Page Transitions are better described directly on the Matomo Dashboard. Click here to check them out.

Figure 4.15: User Visits and General Insights Page

The **pages** section shows how much the routes have been visited, both by absolute URL and grouping part of the URL, depending on the state of the leftmost top switch. This way, it is possible to detect the page traffic better. Figure 4.16 provides the two views: on the left, the routes are grouped, and on the right, they are ungrouped, showing the absolute path beyond the domain of the page. This is important to have an idea of the most interacted areas of the application without limiting the user to unique routes. On the right side of figure 4.16, we can see multiple pages starting with "Entity", but they appear grouped on the left side on top of the list with greater frequency.



Pages

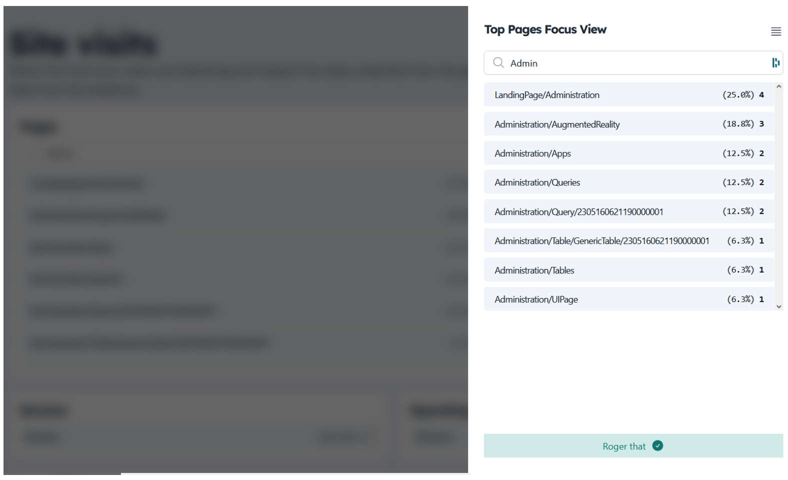
Entity	(23.3%) 14
LandingPage	(21.7%) 13
Administration	(20.0%) 12
Apps	(8.3%) 5
Home	(5.0%) 3
Reports	(3.3%) 2

Pages

Administration/Apps	(3.3%) 2
Administration/Queries	(3.3%) 2
Administration/Query/2305160621190000001	(3.3%) 2
Entity/Area?entityType=Area	(3.3%) 2
Entity/SetupMatrix?entityType=SetupMatrix	(3.3%) 2
LandingPage/Quality	(3.3%) 2

Figure 4.16: Page Routes Grouping

We also allow the user to focus on the pages section in case the UI gets excessive entries. Additionally, the user can filter the routes by substring. Both of these features can be seen in figure 4.17 below.



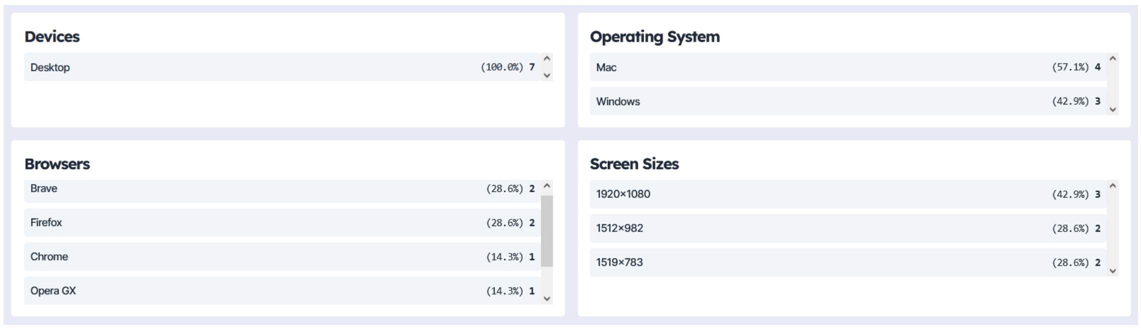
Top Pages Focus View

Admin

LandingPage/Administration	(25.0%) 4
Administration/AugmentedReality	(18.8%) 3
Administration/Apps	(12.5%) 2
Administration/Queries	(12.5%) 2
Administration/Query/2305160621190000001	(12.5%) 2
Administration/Table/GenericTable/2305160621190000001	(6.3%) 1
Administration/Tables	(6.3%) 1
Administration/UIPage	(6.3%) 1

Figure 4.17: Page Routes Grouping

The **Devices, Browsers, Operating Systems, and Screen Sizes** usage section can be seen in figure 4.18 below. This section contains four grouped metrics cards related to characteristics of the hardware used to access the platform and provides percentages to add more perspective to these analytics, which was a specific request by CMF.



Devices

Desktop	(100.0%) 7

Operating System

Mac	(57.1%) 4
Windows	(42.9%) 3

Browsers

Brave	(28.6%) 2
Firefox	(28.6%) 2
Chrome	(14.3%) 1
Opera GX	(14.3%) 1

Screen Sizes

1920×1080	(42.9%) 3
1512×982	(28.6%) 2
1519×783	(28.6%) 2

Figure 4.18: Devices, Browsers, Operating Systems, and Screen Sizes

Figure 4.19 below shows the **pathways**, otherwise known as **page transitions** over on the Matomo Dashboard, as this is a self-explanatory and helpful diagram for understanding page route sequential flows. The dropdown in the top right of figure 4.19 above is a selection of a page to focus on, and we can see the incoming and outgoing routes from the focused page. Figure 4.20 showcases the selection process of a new page route to focus on.

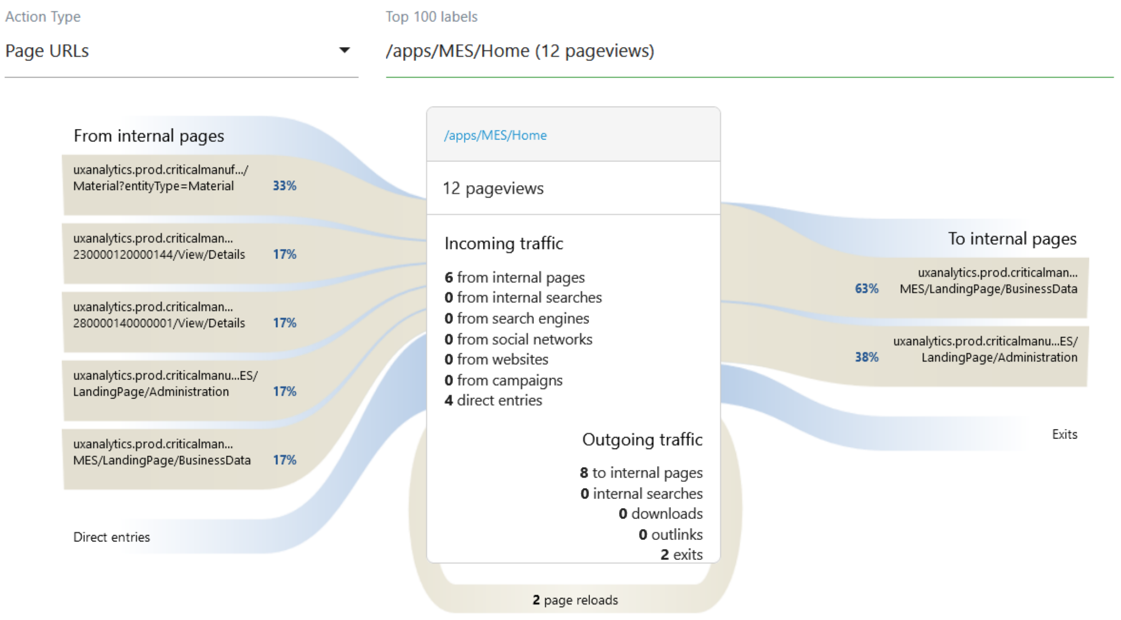

Action Type
Page URLs
Top 100 labels
/apps/MES/Home (12 pageviews)
From internal pages
uxanalytics.prod.criticalmanuf.../ Material?entityType=Material 33%
uxanalytics.prod.criticalman... 230000120000144/View/Details 17%
uxanalytics.prod.criticalman... 280000140000001/View/Details 17%
uxanalytics.prod.criticalmanu...ES/ LandingPage/Administration 17%
uxanalytics.prod.criticalman... MES/LandingPage/BusinessData 17%
Direct entries
/apps/MES/Home
12 pageviews
Incoming traffic
6 from internal pages
0 from internal searches
0 from search engines
0 from social networks
0 from websites
0 from campaigns
4 direct entries
Outgoing traffic
8 to internal pages
0 internal searches
0 downloads
0 outlinks
2 exits
2 page reloads
To internal pages
63% uxanalytics.prod.criticalman... MES/LandingPage/BusinessData
38% uxanalytics.prod.criticalmanu...ES/ LandingPage/Administration
Exits

Figure 4.19: Pathways: Page Route Transitions

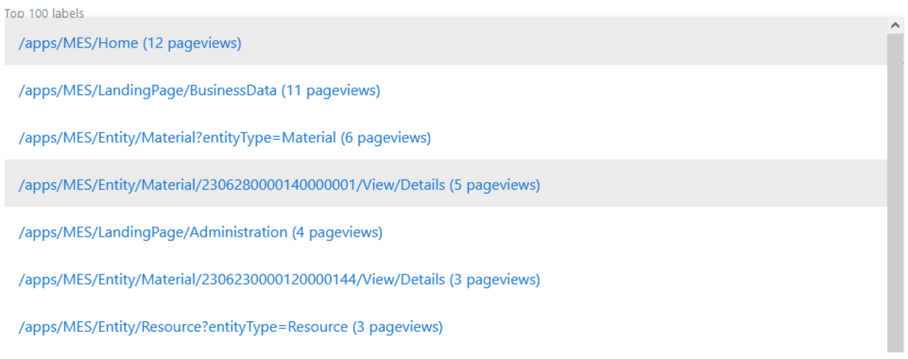

Top 100 labels
/apps/MES/Home (12 pageviews)
/apps/MES/LandingPage/BusinessData (11 pageviews)
/apps/MES/Entity/Material?entityType=Material (6 pageviews)
/apps/MES/Entity/Material/2306280000140000001/View/Details (5 pageviews)
/apps/MES/LandingPage/Administration (4 pageviews)
/apps/MES/Entity/Material/2306230000120000144/View/Details (3 pageviews)
/apps/MES/Entity/Resource?entityType=Resource (3 pageviews)

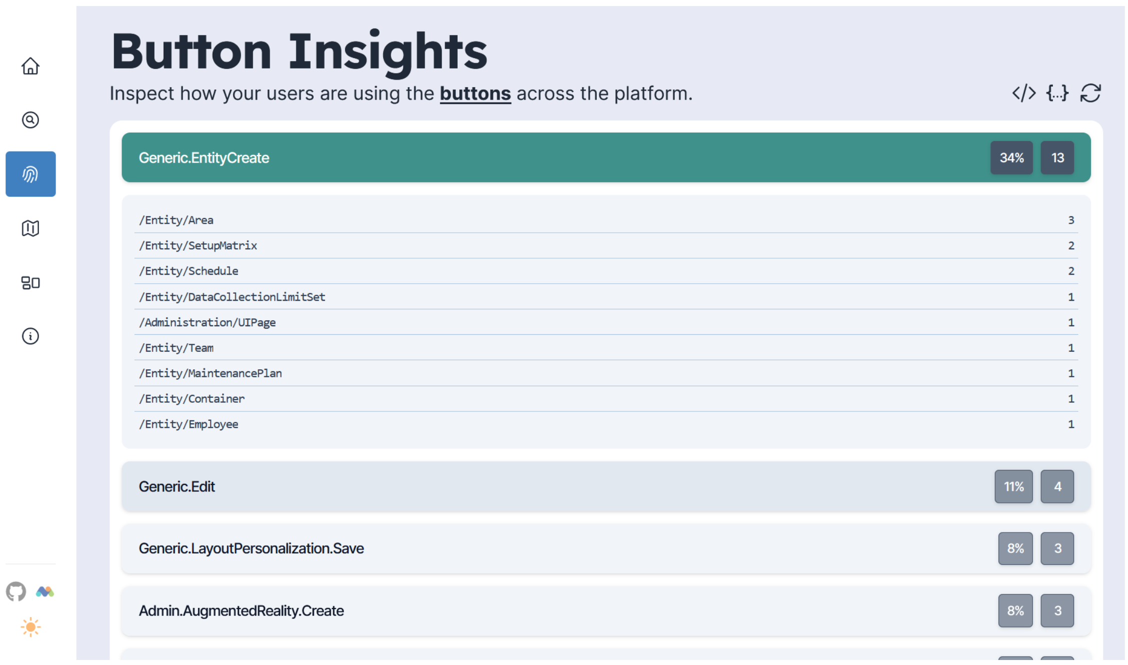
Figure 4.20: Pathways: Page Route Transitions 2

4.3.4 Buttons

While not as critical as others, the **buttons** page still offers valuable insights. It may not directly inform UI recommendations, but it sheds light on the platform's most frequently used buttons and how they compare to others.

Figure 4.21 below shows that button clicks are grouped by type. Each button type that has been clicked more than once generates a row. These rows display the button's name on the left and the number of clicks received on the right. This includes the total number of clicks and the relative proportion of clicks across the entire platform.

Furthermore, each row can be expanded or collapsed. The expandable sections reveal a list of routes where the button was clicked and the number of clicks recorded for each specific route.


Button Insights

Inspect how your users are using the **buttons** across the platform.

Generic.EntityCreate 34% 13

/Entity/Area	3
/Entity/SetupMatrix	2
/Entity/Schedule	2
/Entity/DataCollectionLimitSet	1
/Administration/UIPage	1
/Entity/Team	1
/Entity/MaintenancePlan	1
/Entity/Container	1
/Entity/Employee	1

Generic.Edit 11% 4

Generic.LayoutPersonalization.Save 8% 3

Admin.AugmentedReality.Create 8% 3

Figure 4.21: Button Insights Page

4.4 Scoring Functions

As we have seen in previous sections like, 4.3.1 and 3.3.1, we have a working hypothesis that is possible and useful to make use of heuristic functions that predict a usability score to our transactional components: the **wizards** and **execution views**. This section will detail how these heuristic functions work, how we determined the expressions, their advantages, and how they provide value in our solution. We will only focus on wizards for the sake of simplicity.

4.4.1 Scoring Coefficient Determination

In subsection 4.1.1, we described the internal functionalities of CMF's MES and the events we were tracking on wizards. We also stated that the negative actions of these wizards would be used to determine an assessment for each wizard. This section describes the three scoring approaches we established and their reasoning. This first formula has the most prolonged description, and the last two are adaptations of it, which is why we present them as variations of the base formula. We created multiple formulas to find how our assumptions apply and how they reflect on the results and make comparisons.

Formula A: The score is calculated based on whether the wizard was completed, the number of wizard errors, failed steps, and back-to-previous-step button clicks. We deduct points from a wizard based on negative actions. If there are **no negative actions** and the **timespan was under 10 seconds**, we mark this wizard as **discarded** because we assume that the user was not evidently attempting to complete the wizard. Therefore, we **do not provide a score**. Otherwise, we start by establishing a baseline score:

$$\text{score} = 100 - 15 \times \text{failedSteps} - 10 \times \text{errors} - 5 \times \text{backSteps}$$

If the wizard is canceled or abandoned, we deduct extra points. We deduct 20 points, extra points for the negative actions, and for the elapsed time (1.0 points per 6 seconds):

$$\text{score} = \text{score} - 20 - 3 \times (\text{errors} + \text{stepErrors} + \text{backStepCount}) - \frac{\text{timespan}}{6}$$

The **minimum score is 0**, so if the score drops below that, we assign it a score of 0. If the score is below 40, and the wizard was completed, we **assign a score of 40 to the wizard, rewarding the completion**.

This is our number one option for assessing usability within a wizard, but we have yet to **justify the literal values** found in the equation. Those assumptions and explanations are presented below in table 4.3.

Criterion	**Points**	**Description**
Errors	10	Usually associated with another task to complete, requiring business rules knowledge of MES. We suggest deducting points if the next actions are unclear to the user. However, errors have many origins.
Failed Steps	15	Errors exclusive to the current step. Generally depicts bad user interaction. Example: introducing a value too low in an input box.
Back to Previous Step	5	Shows some hesitation and lackluster interaction if we need to go back any steps.
Cancellation Penalty	20	In case the wizard is canceled, we deduct 20 points directly to punish not completing.
Time Spent Penalty	6	Every 6 seconds on the wizard costs 1 point if the wizard was canceled, to penalize staying on the wizard long enough, assuming the user is attempting to submit it but struggling.
Negative Actions	3	An additional multiplier for punishing the user for keeping on trying to complete but not being able to, hence the other struggles with negative actions.
Minimum Points	0	Our scale goes from 0 to 100, so we apply a correction here and assign 0 if the score drops to negative numbers.
Completion Reward	40	In case points drop below 40, and the wizard was completed, we reward the wizard completion.
Discarded Status	N/A	A wizard is discarded if the time spent is below 10 seconds and there are no negative actions while not being completed/submitted successfully.

Table 4.3: Default Wizard Scoring Approach (Formula A)

Formula B: Similar to the default formula, as it uses the same members in the equation, but using some different assumptions and coefficients, which are explained below in table 4.4.

Criterion	Points	Description
Negative Action Penalty	10	This formula takes an approach where the sum of negative actions greatly impact the final score. If the wizard is canceled, the negative actions sum will be multiplied by ten and subtracted from the score.
Cancellation Penalty	0	Following the negative action extra penalty, formula B does not contemplate a static cancellation penalty, assuming that the negative actions drive the shifts in score variations.
Time Spent Penalty	60	Formula B takes a step back from judging the additional time spent in the wizard as harshly, assuming that the time spent is not as strong of an indicator of struggle and poor usability.

Table 4.4: Variation of Wizard Scoring Approach (Formula B)

Formula C: introduces different variations but still keeps the exact evaluation structure. The alterations are described below in table 4.5.

Criterion	Points	Description
Errors, Failed Steps, Back Steps	10	Formula C considers any negative action equally relevant. This approach does not make speculations about the nature of the negative actions, meaning that we simply penalize actions that do not contribute to completing the wizard equally.
Negative Action Penalty	0	Formula C does not add extra penalties for negative actions if the wizard is canceled. Instead, this formula **only considers a constant cancellation penalty**.
Cancellation Penalty	60	In line with the alteration in the line above, formula C subtracts many points when users do not complete wizards, assuming that any non-discarded wizard that is not completed is due to usability complications.
Discarded Status	N/A	A wizard is discarded if the time spent is below 5 seconds while not being completed/submitted successfully. Formula C ignores negative actions for discarding wizards. It simply ignores non-completed wizards under 5 seconds of usage.

Table 4.5: Variation of Wizard Scoring Approach (Formula C)

4.4.2 Scoring Flexibility

The scoring functions utilized in our system have a noteworthy characteristic: they are **executed on the client side**. Consequently, when fetching events from the Matomo API within the middleware, our operations are limited to grouping actions, devoid of any subjective manipulations at this level. Instead, the responsibility for these judgments is delegated to the respective pages for implementation.

For example, on the wizard insights page, an interface element in a button allows users to **select specific formulas**, altering the scoring approach for the wizards. Although our current offering for wizard evaluation comprises only three distinct formulas, our solution boasts a substantial advantage by providing effortless scalability. The inclusion of additional scoring functions can be very easily achieved with minor code refactoring. Notably, within our `matomo.ts` file, we have established an "evaluating wizard" function defining the requisite variables for conducting the assessment.

The code block below showcases the particular piece of code that contributes to the versatility of our system and simplifies the integration of new formulas.

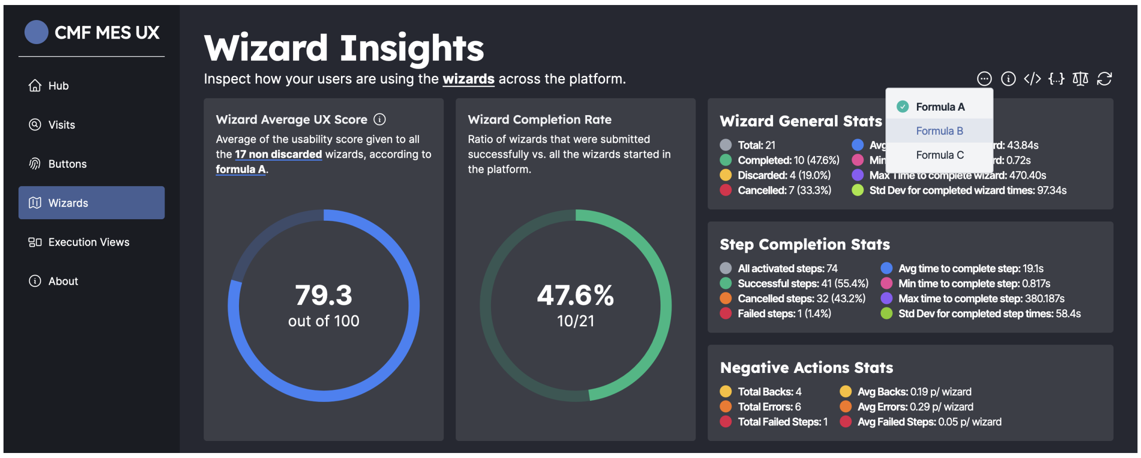
```
// scoring constants for default formula
let timeThreshold = 10; // 10 seconds
let failedStepPenalty = 15; // 15 points
let errorPenalty = 10; // 10 points
let backStepPenalty = 5; // 5 points
let negativeActionPenalty = 3; // 3 points
let cancelStaticPenalty = 20; // 20 points
let secondsToPenalty = 6.0; // 6.0 seconds per point
let negativeActions = errorCount + failedStepCount + backStepCount;
let minimumScoreIfCompleted = 40;
let discarded = negativeActions === 0 && timespan < timeThreshold;
let score: number | null = 100;
let formulaStr = `${score}`;

if (formula === 'B') {
  negativeActionPenalty = 10;
  cancelStaticPenalty = 0;
  secondsToPenalty = 60.0; // 60.0 seconds per point
} else if (formula === 'C') {
  timeThreshold = 5; // 5 seconds
  discarded = timespan < timeThreshold;
  errorPenalty = 10;
  backStepPenalty = 10;
  failedStepPenalty = 10;
  negativeActionPenalty = 0;
  cancelStaticPenalty = 60;
  secondsToPenalty = 600.0; // 600.0 seconds per point
}
```

Adding extra options to the formulas and tweaking values allows us to create extra hypotheses for judging the wizards and see how they perform on the client side. The example above shows simple alterations, but the function where this code block lies is easily refactorable, allowing for entirely different approaches, like incorporating the visible step count into the formula and discarding low total step wizards, among others.

Meanwhile, as we saw in section 4.2.5, our frontend pages perform the fetching of the data from the Middleware (Next.js backend) and only later evaluate the data based on the selected formula. Here is an adapted excerpt of that key part in the `useEffect` on the wizards' page:

```
const evaluatedWizards = evaluateAndGroupWizards(data, formula);
setProcessedData(evaluatedWizards); // update state variable
```

Figure 4.22 below shows how the user can select a different formula on the wizards' page UI and obtain **different results immediately**, as the data is not refetched. The state of the variable containing the selected formula controls the results that the UI renders, updating all of the KPIs upon changing the value.



Wizard Insights

Inspect how your users are using the **wizards** across the platform.

Wizard Average UX Score

Average of the usability score given to all the **17 non discarded** wizards, according to **formula A**.

79.3
out of 100

Wizard Completion Rate

Ratio of wizards that were submitted successfully vs. all the wizards started in the platform.

47.6%
10/21

Wizard General Stats

- **Total:** 21
- **Completed:** 10 (47.6%)
- **Discarded:** 4 (19.0%)
- **Cancelled:** 7 (33.3%)
- **Avg** ... **rd:** 43.84s
- **Min** ... **rd:** 0.72s
- **Max Time to complete wizard:** 470.40s
- **Std Dev for completed wizard times:** 97.34s

Step Completion Stats

- **All activated steps:** 74
- **Successful steps:** 41 (55.4%)
- **Cancelled steps:** 32 (43.2%)
- **Failed steps:** 1 (1.4%)
- **Avg time to complete step:** 19.1s
- **Min time to complete step:** 0.817s
- **Max time to complete step:** 380.187s
- **Std Dev for completed step times:** 58.4s

Negative Actions Stats

- **Total Backs:** 4
- **Total Errors:** 6
- **Total Failed Steps:** 1
- **Avg Backs:** 0.19 p/ wizard
- **Avg Errors:** 0.29 p/ wizard
- **Avg Failed Steps:** 0.05 p/ wizard

Figure 4.22: Select Formula on Wizards UI

4.5 Scalability

In this section, we cover the concept of scalability regarding our dashboard, focusing on its improvement potential and performance as the volume of fetched data increases. A fundamental aspect of utilizing this new customized dashboard is its ability to provide valuable insights over an extended period as the data accumulates. Despite the fact that CMF's MES contains many interaction points, we can confidently state that as the dataset grows, the dashboard will become increasingly prominent in uncovering truths about either CMF's MES itself, as originally intended or our methodology for evaluating events.

To elaborate, imagine that we amass a substantial number of interactions over time, such as thousands of wizards collected and displayed on our Next.js Dashboard. Assuming our formulas are reasonably accurate, we will have the means to identify wizard categories that encounter difficulties and determine specific wizards within a given group that exhibit particularly poor performance. This enables us to comprehensively understand the underlying factors contributing to these observations and formulate hypotheses to explain them.

In essence, this line of reasoning empowers us to either improve our scoring system or achieve our initial objective of revealing previously unseen aspects of MES, which have remained obscure due to the sheer magnitude of the application and the difficulties in obtaining wizard feedback.

4.6 Summary

Our implementation takes the default capabilities of Matomo and adds an extra layer on top to provide additional meaningful customized insights in the form of an interactive dashboard that is automatically fed data from CMF's MES. This dashboard enables users to closely examine the intrinsic interaction data of CMF's MES and gain valuable insights into user behavior and patterns. It offers an engaging and visually appealing interface, making it an exciting tool for developers to explore vast amounts of data accumulated over time. Through our solution, developers can uncover subtle behavioral patterns that may not be apparent previously. This capability is precious since it helps identify otherwise imperceptible issues or strengths related to the interactions with custom components of the application.

Furthermore, this dashboard serves as a valuable starting point for conducting UI/UX research in CMF's MES, and as time progresses, developers can leverage both of the dashboards' functionality to make informed recommendations for further improving UI/UX aspects.

Chapter 5

Validation & Evaluation

Contents

In this chapter, the focus shifts toward assessing the fitness and effectiveness of our solution. To start the validation process, section 5.1 describes the methods we will employ to evaluate the solution, followed by an explanation of the experimental setup in section 5.2. In sections 5.3 and 5.4, we observe and scrutinize the collected data on our experiments, aiming to understand the strengths and weaknesses of our approach by looking at hard evidence. Section 5.5 describes the limitations of the validation process, and the chapter is closed with a summary and key takeaways section (5.6), as we answer our research questions (elaborated in 3.2).

This evaluation process sheds light on the practical benefits our solution brings to the table while also highlighting the limitations that require attention. This analysis balances our solution's performance, setting the stage for future improvements and adaptations.

5.1 Methodology

Our testing and validation procedures will revolve around performing or waiting for interactions in CMF's MES in different scenarios, followed by observing and discussing the generated results on our Next.js dashboard and the Matomo dashboard. To achieve this, we must prepare the environments for the collected data to be generated appropriately.

For each environment, observing results is tied to these tasks:

- **Dashboards' fitness**: The first step of validating the solutions is ensuring that the dashboards pick up the expected data without looking at biased assessments. Next, we must evaluate the layout and even usability of our new platforms.
- **Wizard insights**: This is the trickiest stage of our validation since we have multiple working hypotheses that we can evaluate wizards based on formulas. We can test results for every formula we have provided and explain the differences in the results concerning the differences of the formulas. Results on this stage are long as we have general insights for all wizards, but also narrowed down stats for each wizard group and even each wizard.

5.2 Experimental Setup

This section outlines the methods for configuring the environments where our proposed solution is tested. For each environment, we must have our solution's components (Matomo, MySQL, Next.js) running alongside CMF's MES, and the setups vary between scenarios.

5.2.1 Local Setup

In the local environment, we run CMF's MES using the Angular Command-Line Interface (CLI) in a terminal window. Parallel to this execution, we use a docker-compose configuration that joins our three additional components: Matomo, MySQL, and Next.js. The configuration can be seen below.

```
version: '3.9'
services:
  matomo:
    build:
      context: .
      dockerfile: Dockerfile.matomo
    image: matomo
    ports:
      - '31089:80'
    depends_on:
      - mysql
    env_file:
      - .env
    volumes:
      - matomo_data:/var/www/html
  mysql:
    image: mysql:latest
    ports:
      - '5550:3306'
    env_file:
      - .env
    volumes:
      - mysql_data:/var/lib/mysql
  nextjs:
    build:
      context: .
      dockerfile: Dockerfile.nextjs
    volumes:
      - .:/usr/src/app
    ports:
      - '31087:3000'
    depends_on:
      - mysql
      - matomo
volumes:
  matomo_data:
  mysql_data:
```

Once we execute `docker compose up -d --build`, our images will be pulled (if they are not available already), and make the three services run. As we can see from the compose configuration, our Next.js and Matomo services become available in ports **31087** and **31089** if they are exposed in their docker files, which they are.

The next step is to access the Matomo Dashboard page on **localhost:31089** and begin the Matomo setup. Matomo requires us to follow some steps to initiate the service and provide information about the applications we want to track (timezone, URL, among others). Once we have completed this setup, which is relatively straightforward, we head over to **localhost:4200** where CMF's MES is running. By simply logging into the platform, Matomo registers a page view on the predefined default page.

5.2.2 Staging Environment Setup

We contacted CMF's deployment services for the staging environment setup, as this operation was tricky and required higher-up information about the day-to-day operations. Thorough details of these procedures shall remain undisclosed in order not to reveal any sensitive information, but, to put it simply, we take our **three containers** seen in the previous section, 5.2.1, and we join them to another docker-compose configuration in a staging MES environment that contains the changes on MES for the Matomo tracking on the SPA.

With this staging environment enabled, any CMF employee with the credentials we defined in this configuration can now interact with our version of CMF's MES. Doing so will result in populating the staging database, which, over time, can grow and display the potential of our solution. These developments also mean that UI/UX teams can request access (credentials) to our solution and play around to test it.

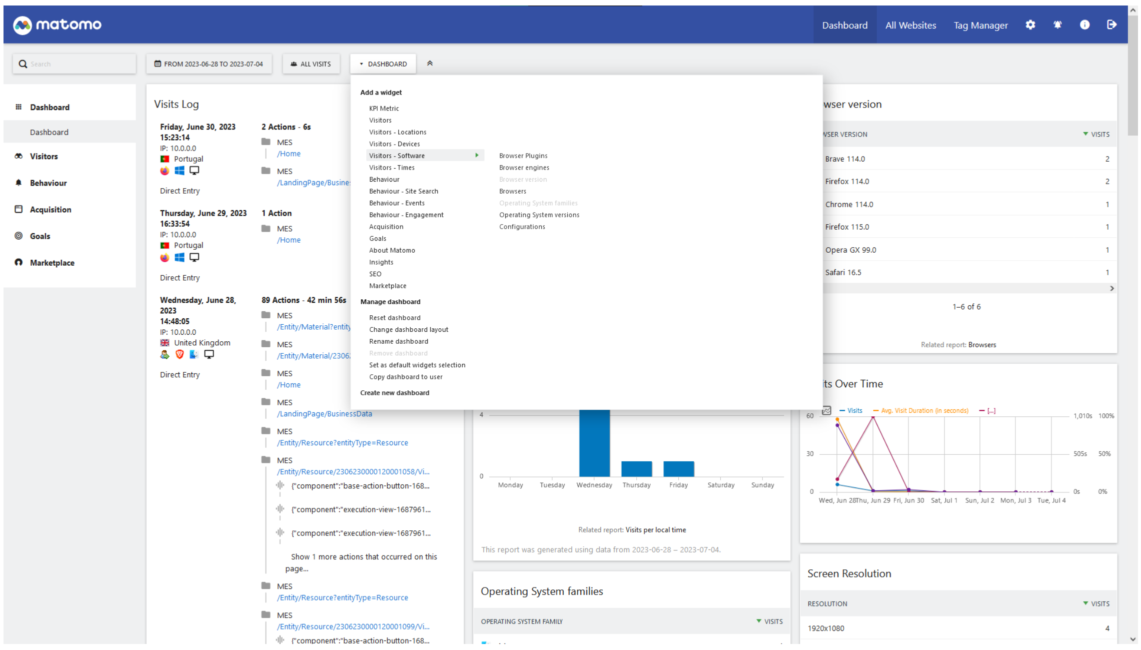
5.3 Experiments

This section showcases data segments from our solutions, providing a foundation for subsequent analysis and discussion. We have divided the presentation into subsections, each corresponding to a specific environment where we have conducted our data-gathering activities.

5.3.1 Local Data Collection

We labeled this stage local data collection to symbolize the simplest form of testing and evaluating our solution. In this environment, the interaction on CMF's MES to populate the Matomo database and obtain data to make our dashboards come to life was individually performed in development. It is especially relevant to look at the wizard insights on the customized Next.js dashboard, where we have analytics independent of the biased formulas and results that vary depending on said formulas, which we can scrutinize easily. However, this section also showcases Matomo results and the other areas of the Next.js dashboard, despite the wizards having a lot more relevance for the validation process and the fact that we have shown revealing snapshots in 4.

Figure 5.1 shows the potential of the Matomo dashboard when it already contains data. The dropdown in the figure allows the users to view additional panels on the page and drag them between positions to fit a unified view. Figures 5.2 and 5.3 showcase the visits and buttons page.



Visits Log

Friday, June 30, 2023
15:23:14
IP: 10.0.0.0
Portugal
Direct Entry

2 Actions - 6s
MES
/Home
MES
/LandingPage/Busine

Thursday, June 29, 2023
16:33:54
IP: 10.0.0.0
Portugal
Direct Entry

1 Action
MES
/Home

Wednesday, June 28, 2023
14:48:05
IP: 10.0.0.0
United Kingdom
Direct Entry

89 Actions - 42 min 56s
MES
/Entity/Material?entity
MES
/Entity/Material/2306
MES
/Home
MES
/LandingPage/BusinessData
MES
/Entity/Resource?entityType=Resource
MES
/Entity/Resource/2306230000120001058/Vi...
{"component":"base-action-button-168...
{"component":"execution-view-1687961...
{"component":"execution-view-1687961...
Show 1 more actions that occurred on this page...
MES
/Entity/Resource?entityType=Resource
MES
/Entity/Resource/2306230000120001099/Vi...
{"component":"base-action-button-168...

wser version

WSER VERSION	VISITS
Brave 114.0	2
Firefox 114.0	2
Chrome 114.0	1
Firefox 115.0	1
Opera GX 99.0	1
Safari 16.5	1

1–6 of 6

This report was generated using data from 2023-06-28 – 2023-07-04.

its Over Time

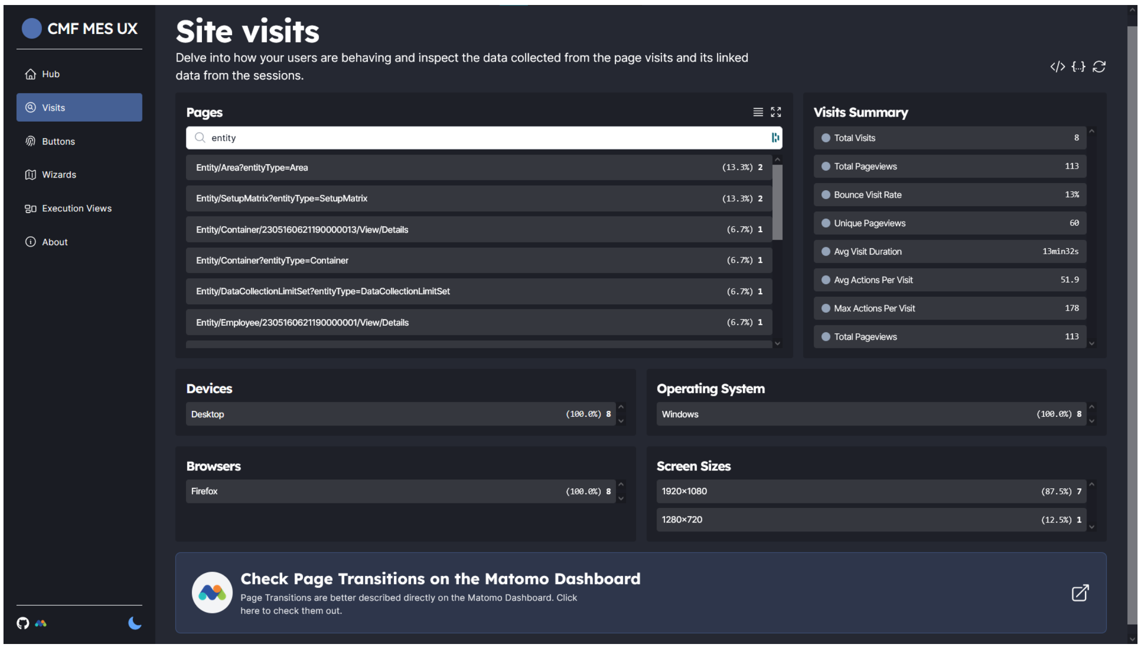
Operating System families

OPERATING SYSTEM FAMILY	VISITS

Screen Resolution

RESOLUTION	VISITS
1920x1080	4

Figure 5.1: Matomo Dashboard with Local Data



Site visits

Delve into how your users are behaving and inspect the data collected from the page visits and its linked data from the sessions.

Pages

entity

Entity/Area?entityType=Area	(13.3%) 2
Entity/SetupMatrix?entityType=SetupMatrix	(13.3%) 2
Entity/Container/2305160621190000013/View/Details	(6.7%) 1
Entity/Container?entityType=Container	(6.7%) 1
Entity/DataCollectionLimitSet?entityType=DataCollectionLimitSet	(6.7%) 1
Entity/Employee/2305160621190000001/View/Details	(6.7%) 1

Visits Summary

Total Visits	8
Total Pageviews	113
Bounce Visit Rate	13%
Unique Pageviews	60
Avg Visit Duration	13min32s
Avg Actions Per Visit	51.9
Max Actions Per Visit	178
Total Pageviews	113

Devices

Desktop	(100.0%) 8

Operating System

Windows	(100.0%) 8

Browsers

Firefox	(100.0%) 8

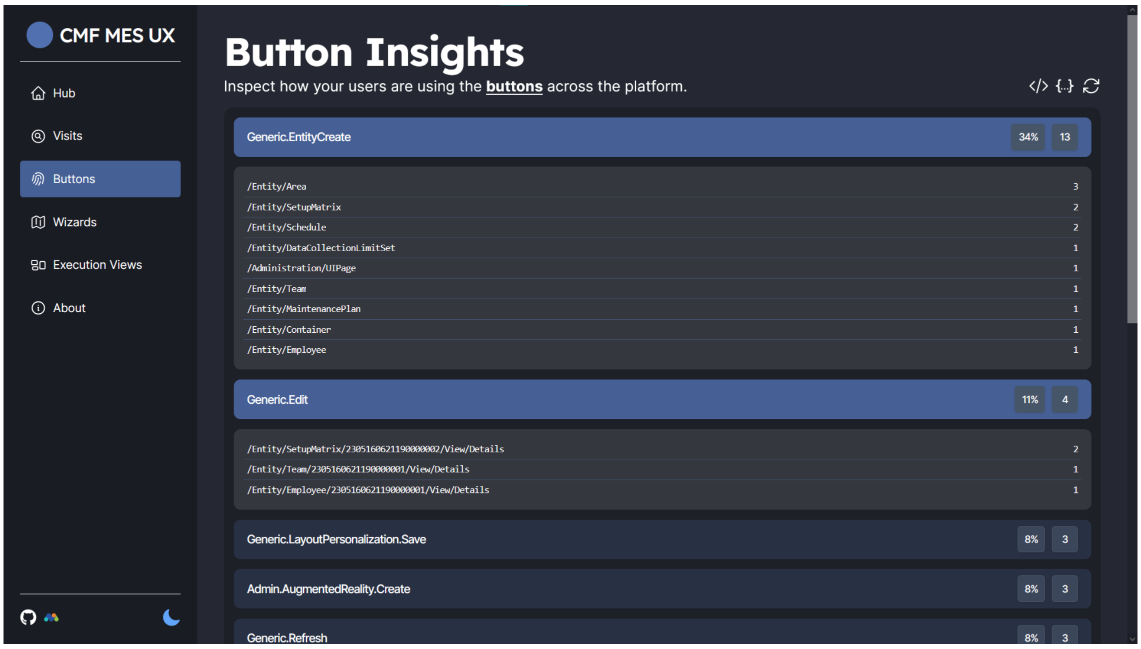
Screen Sizes

1920×1080	(87.5%) 7
1280×720	(12.5%) 1

Check Page Transitions on the Matomo Dashboard

Page Transitions are better described directly on the Matomo Dashboard. Click here to check them out.

Figure 5.2: Next.js Visits Page with Local Data



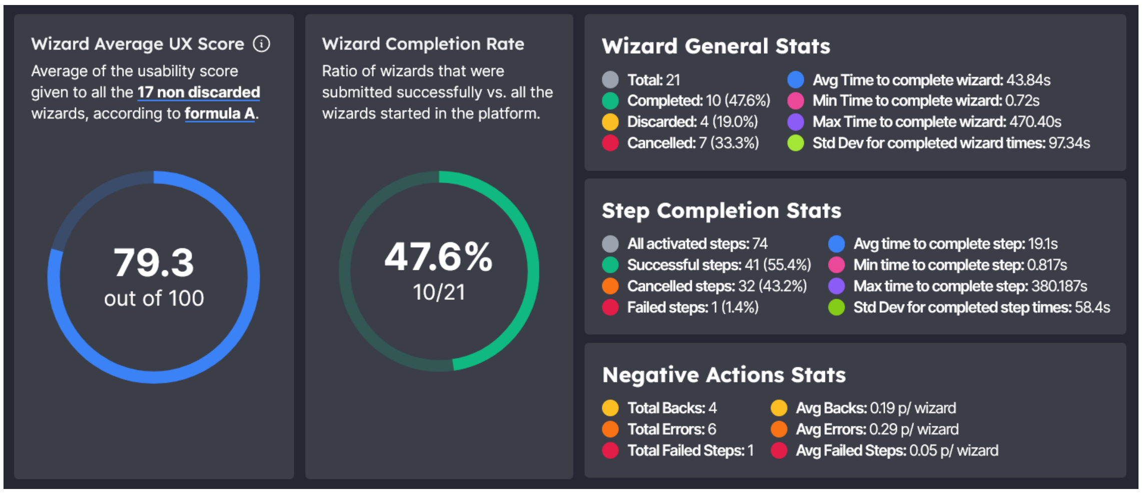
Button Insights

Inspect how your users are using the **buttons** across the platform.

Generic.EntityCreate	34%	13
/Entity/Area		3
/Entity/SetupMatrix		2
/Entity/Schedule		2
/Entity/DataCollectionLimitSet		1
/Administration/UIPage		1
/Entity/Team		1
/Entity/MaintenancePlan		1
/Entity/Container		1
/Entity/Employee		1
Generic.Edit	11%	4
/Entity/SetupMatrix/2305160621190000002/View/Details		2
/Entity/Team/2305160621190000001/View/Details		1
/Entity/Employee/2305160621190000001/View/Details		1
Generic.LayoutPersonalization.Save	8%	3
Admin.AugmentedReality.Create	8%	3
Generic.Refresh	8%	3

Figure 5.3: Next.js Buttons Page with Local Data

Figure 5.4 below showcases the section with general insights for all the assessed wizards, which contains the global average of scores (using formula A, the default), the completion rate, stats related to time, negative action stats, and completion metrics.



Wizard Average UX Score

Average of the usability score given to all the **17 non discarded** wizards, according to **formula A**.

79.3
out of 100

Wizard Completion Rate

Ratio of wizards that were submitted successfully vs. all the wizards started in the platform.

47.6%
10/21

Wizard General Stats

- **Total:** 21
- **Completed:** 10 (47.6%)
- **Discarded:** 4 (19.0%)
- **Cancelled:** 7 (33.3%)
- **Avg Time to complete wizard:** 43.84s
- **Min Time to complete wizard:** 0.72s
- **Max Time to complete wizard:** 470.40s
- **Std Dev for completed wizard times:** 97.34s

Step Completion Stats

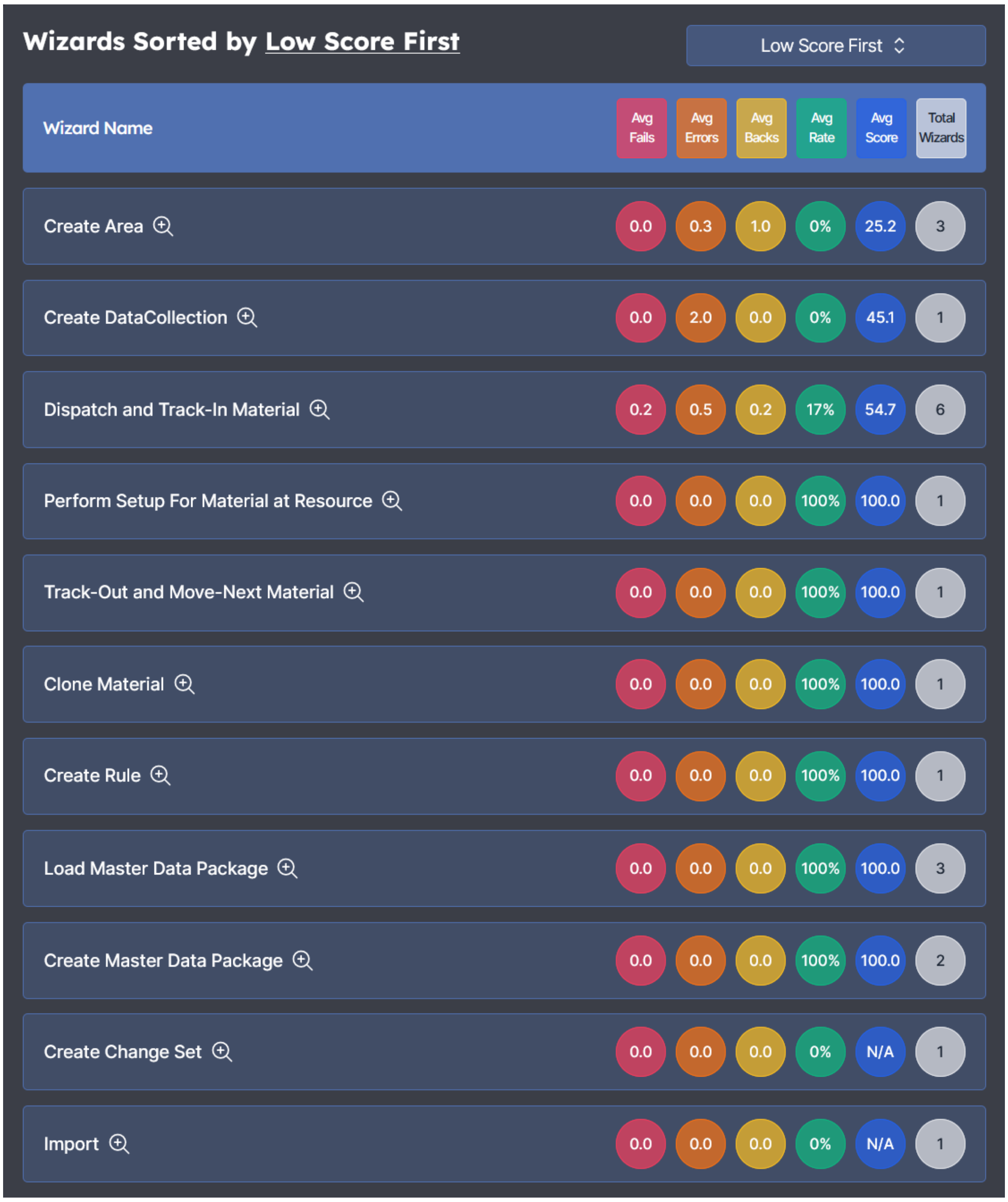
- **All activated steps:** 74
- **Successful steps:** 41 (55.4%)
- **Cancelled steps:** 32 (43.2%)
- **Failed steps:** 1 (1.4%)
- **Avg time to complete step:** 19.1s
- **Min time to complete step:** 0.817s
- **Max time to complete step:** 380.187s
- **Std Dev for completed step times:** 58.4s

Negative Actions Stats

- **Total Backs:** 4
- **Total Errors:** 6
- **Total Failed Steps:** 1
- **Avg Backs:** 0.19 p/ wizard
- **Avg Errors:** 0.29 p/ wizard
- **Avg Failed Steps:** 0.05 p/ wizard

Figure 5.4: Wizard Insights Headline Section

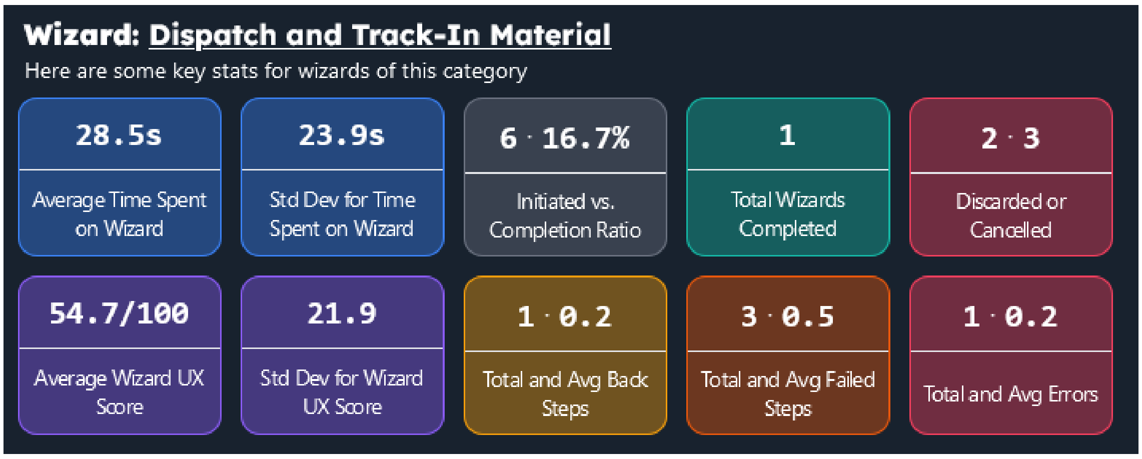
Next, in figure 5.5, we can see the wizards' groups initiated in CMF's MES. Each row symbolizes a type of wizard containing six surface-level stats about the group: **average errors**, **average failed steps**, **average back clicks**, **average completion rate**, **average UX score**, and **wizard count**. The default formula, A, calculates the UX score.


Wizards Sorted by Low Score First

Wizard Name	Avg Fails	Avg Errors	Avg Backs	Avg Rate	Avg Score	Total Wizards
Create Area	0.0	0.3	1.0	0%	25.2	3
Create DataCollection	0.0	2.0	0.0	0%	45.1	1
Dispatch and Track-In Material	0.2	0.5	0.2	17%	54.7	6
Perform Setup For Material at Resource	0.0	0.0	0.0	100%	100.0	1
Track-Out and Move-Next Material	0.0	0.0	0.0	100%	100.0	1
Clone Material	0.0	0.0	0.0	100%	100.0	1
Create Rule	0.0	0.0	0.0	100%	100.0	1
Load Master Data Package	0.0	0.0	0.0	100%	100.0	3
Create Master Data Package	0.0	0.0	0.0	100%	100.0	2
Create Change Set	0.0	0.0	0.0	0%	N/A	1
Import	0.0	0.0	0.0	0%	N/A	1

Figure 5.5: Wizard Groups Summary for formula A

Figure 5.6 presents the top section of a focused wizard group, namely **Dispatch and Track-In Material**, which provides ten stats for the wizard group enumerated previously in section 4.3.1 on list 4.3.1.


Wizard: Dispatch and Track-In Material
Here are some key stats for wizards of this category

28.5s
Average Time Spent on Wizard

23.9s
Std Dev for Time Spent on Wizard

6 · 16.7%
Initiated vs. Completion Ratio

1
Total Wizards Completed

2 · 3
Discarded or Cancelled

54.7/100
Average Wizard UX Score

21.9
Std Dev for Wizard UX Score

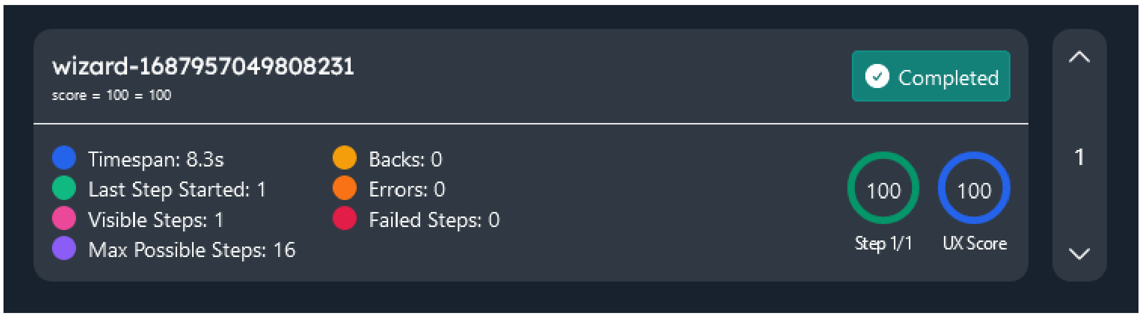
1 · 0.2
Total and Avg Back Steps

3 · 0.5
Total and Avg Failed Steps

1 · 0.2
Total and Avg Errors

Figure 5.6: Dispatch and Track-In Material Wizard Group: Ten Stats

Moving on to the next section of the focused wizard group view, figures 5.7 and 5.8 showcase two distinct individual wizards of the same group, evaluated by formula A. Figure 5.9 below compares global score averages using each formula (A, B, and C). We must note that formula C has different parameters for discarding wizards, which resulted in discarding an additional one.

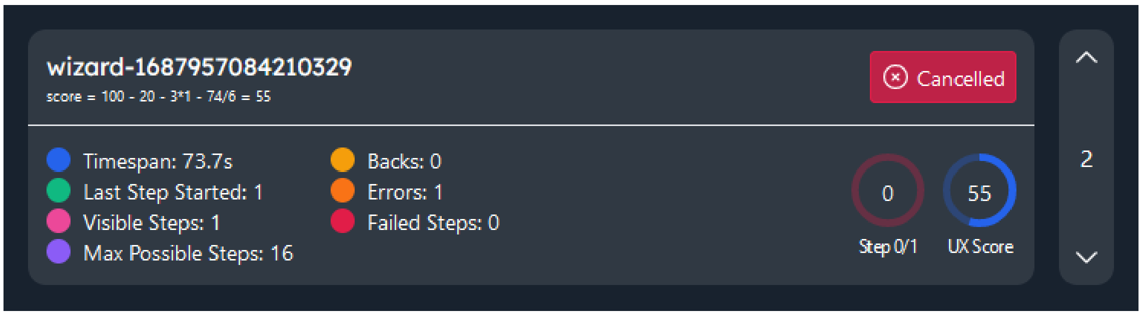

wizard-1687957049808231
score = 100 = 100
Completed

● Timespan: 8.3s
● Last Step Started: 1
● Visible Steps: 1
● Max Possible Steps: 16
● Backs: 0
● Errors: 0
● Failed Steps: 0

100 Step 1/1
100 UX Score

1

Figure 5.7: Dispatch and Track-In Material Wizard Group: Individual Wizard 2 (formula A)


wizard-1687957084210329
score = 100 - 20 - 3*1 - 74/6 = 55
Cancelled

● Timespan: 73.7s
● Last Step Started: 1
● Visible Steps: 1
● Max Possible Steps: 16
● Backs: 0
● Errors: 1
● Failed Steps: 0

0 Step 0/1
55 UX Score

2

Figure 5.8: Dispatch and Track-In Material Wizard Group: Individual Wizard 3 (formula A)

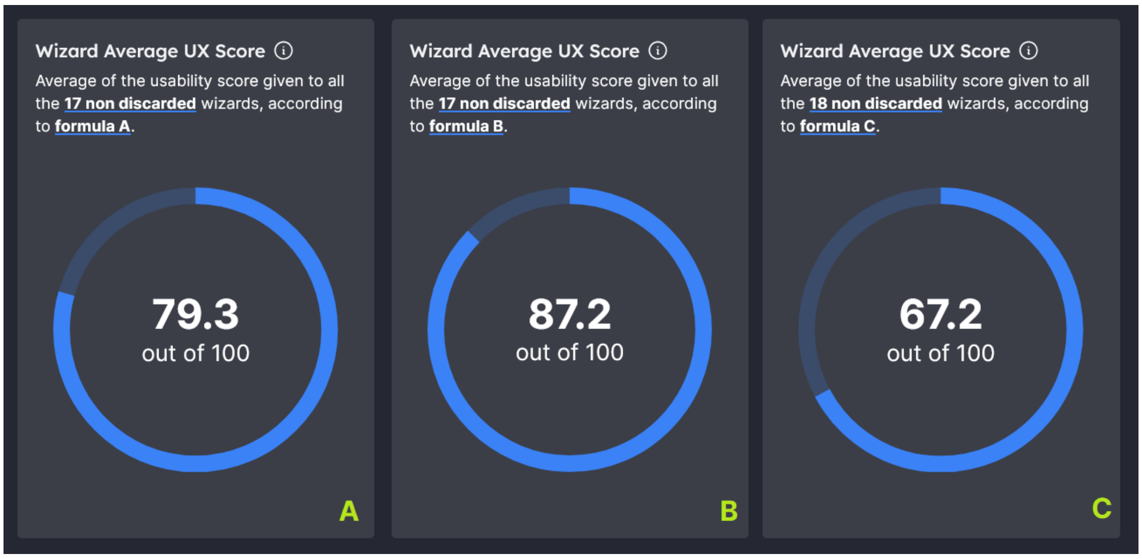

Wizard Average UX Score
Average of the usability score given to all the 17 non discarded wizards, according to formula A.
79.3
out of 100
A
Wizard Average UX Score
Average of the usability score given to all the 17 non discarded wizards, according to formula B.
87.2
out of 100
B
Wizard Average UX Score
Average of the usability score given to all the 18 non discarded wizards, according to formula C.
67.2
out of 100
C

Figure 5.9: Global Wizard Averages for all Formulas

Next, figure 5.10 compares three identical rows representing the same wizard group using each formula (A, B, and C), which only affects the score for the group (fifth and blue badge).

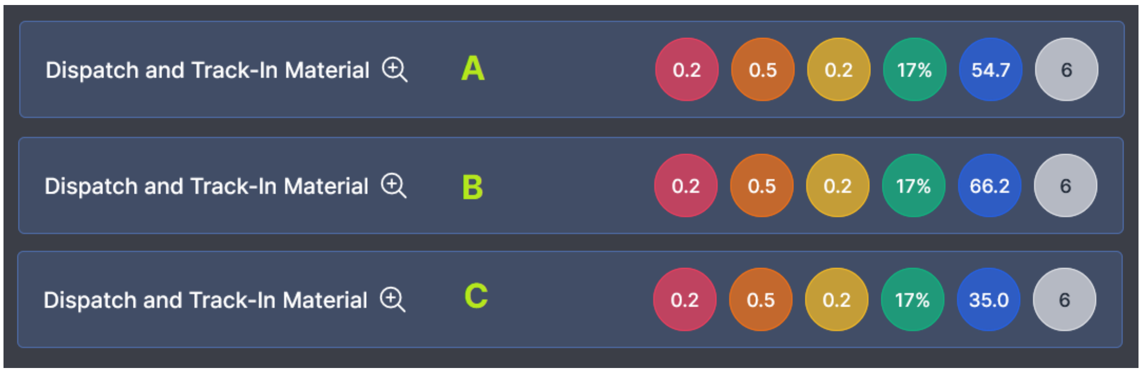
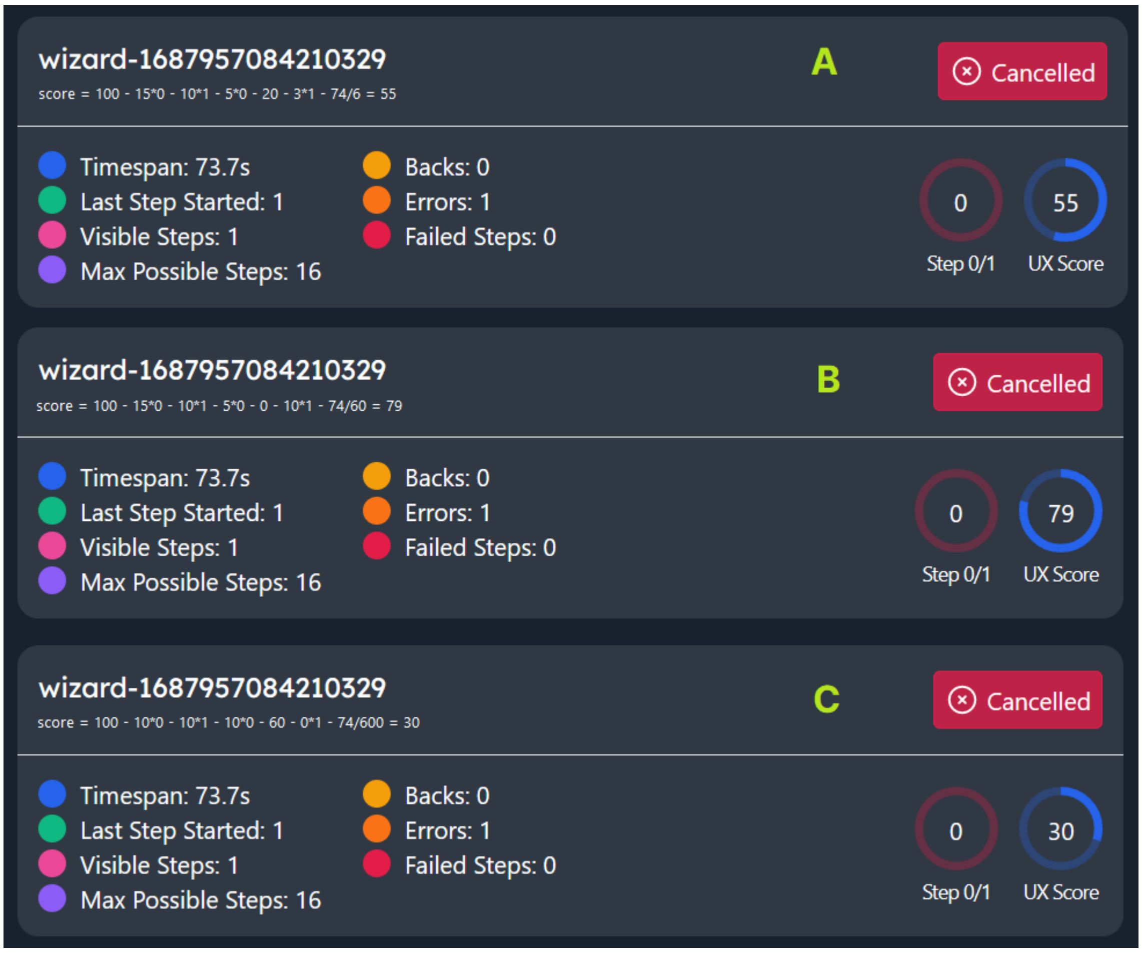

Dispatch and Track-In Material A 0.2 0.5 0.2 17% 54.7 6
Dispatch and Track-In Material B 0.2 0.5 0.2 17% 66.2 6
Dispatch and Track-In Material C 0.2 0.5 0.2 17% 35.0 6

Figure 5.10: Wizard Group for all Formulas

Finally, figure 5.11 takes the same exact wizard and compares the assessment from all the formulas (A, B, and C), which only affect the blue circular score badge on the right.



wizard-1687957084210329
score = 100 - 15*0 - 10*1 - 5*0 - 20 - 3*1 - 74/6 = 55

- Timespan: 73.7s
- Last Step Started: 1
- Visible Steps: 1
- Max Possible Steps: 16
- Backs: 0
- Errors: 1
- Failed Steps: 0

0 Step 0/1
55 UX Score

wizard-1687957084210329
score = 100 - 15*0 - 10*1 - 5*0 - 0 - 10*1 - 74/60 = 79

- Timespan: 73.7s
- Last Step Started: 1
- Visible Steps: 1
- Max Possible Steps: 16
- Backs: 0
- Errors: 1
- Failed Steps: 0

0 Step 0/1
79 UX Score

wizard-1687957084210329
score = 100 - 10*0 - 10*1 - 10*0 - 60 - 0*1 - 74/600 = 30

- Timespan: 73.7s
- Last Step Started: 1
- Visible Steps: 1
- Max Possible Steps: 16
- Backs: 0
- Errors: 1
- Failed Steps: 0

0 Step 0/1
30 UX Score

Figure 5.11: Wizard Comparison depending on Formula

5.3.2 Staging Data Collection

For this stage, we take our solution to the next level, asking people with decent knowledge of CMF's MES to complete and set of actions within CMF's MES to simulate recurring scenarios in production. We also ask these test subjects to complete a form asking questions about the user's profile and opinions on the application's usability during the experience. We also ask an extra question about the scoring approach for wizards, aiming to integrate qualitative feedback into our quantitative solution. This form is available in appendix B.

As of writing this dissertation, the results of these experiments conducted alongside CMF staff are not available yet but may be incorporated and discussed at a later date.

5.4 Results Discussion

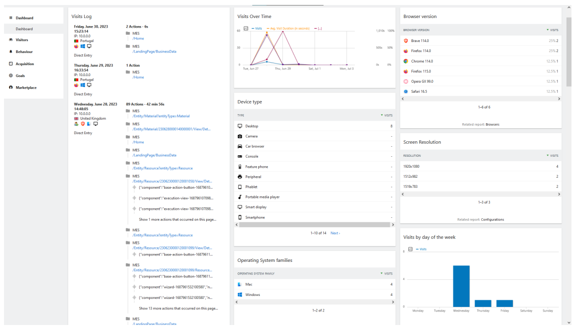
In the previous section 5.3, we saw the pieces of evidence from the dashboards, setting up a discussion of results. Discussing our particular results includes **discussing the fitness and usability** of our solution, taking high regard to the layout and content of the dashboards. It also includes

assessing quantitative results that can help us understand whether our scoring approaches for the wizards are reasonable.

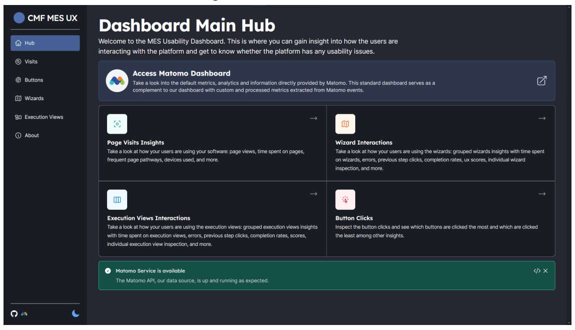
Matomo's analytics, tracked and provided out-of-the-box, are available for CMF's teams' benefit. Metrics like page routes views rankings, device usage, and browser usage are working correctly both in the local and staging environments, with real-time updates. Many of these supplied metrics are straightforward and were directly proposed for this dissertation, meaning that these are useful for CMF to be equipped to both understand the platform better and have the ability to make design recommendations.

Only in later stages, when this solution is applied to production environments in a long-term scenario, can we know the proper faults in our solution. However, we complied with the instructions and provided further insights than the requested ones. However, we have two **working dashboards** already mounted on a staging environment for inspecting the inner workings of CMF's MES, which is this project's practical output.

(a) Matomo Dashboard



(b) Next.js Custom Dashboard



Dashboard Main Hub

Welcome to the MES Usability Dashboard. This is where you can gain insight into how the users are interacting with the platform and get to know whether the platform has any usability issues.

Access Matomo Dashboard

Take a look into the default metrics, analytics and information directly provided by Matomo. This standard dashboard serves as a complement to our dashboard with custom and processed metrics extracted from Matomo events.

Page Visits Insights

Take a look at how your users are using your software: page views, time spent on pages, frequent page pathways, devices used, and more.

Wizard Interactions

Take a look at how your users are using the wizards: grouped wizards insights with time spent on wizards, errors, previous step clicks, completion rates, ux scores, individual wizard inspection, and more.

Execution Views Interactions

Take a look at how your users are using the execution views: grouped execution views insights with time spent on execution views, errors, previous step clicks, completion rates, scores, individual execution view inspection, and more.

Button Clicks

Inspect the button clicks and see which buttons are clicked the most and which are clicked the least among other insights.

Matomo Service is available

The Matomo API, our data source, is up and running as expected.

Figure 5.12: **Hub Pages of the Dashboards** in Staging Environment

However, when it comes to **validating our scoring approaches**, we can do so by analyzing the results and referring back to the decisions we made. Below are the locally collected wizard facts and scores assessed using different formulas. These are a decent basis to validate our decisions in creating the formulas for assessment if we compare identical wizards.

Wizard Group Name	i	t(s)	Errors	Fails	Backs	A	B	C
Create Area	1	470.4	1	0	3	0	27	0
Create Area	2	25.8	0	0	0	76	100	40
Create Area	3	3.2	0	0	0	N/A	N/A	N/A
Create Data Collection	1	53.7	2	0	0	45	59	20
Dispatch and Track-In Material	1	0.7	0	0	0	N/A	N/A	N/A
Dispatch and Track-In Material	2	8.3	0	0	0	100	100	100
Dispatch and Track-In Material	3	73.7	1	0	0	55	79	30
Dispatch and Track-In Material	4	25.8	0	0	0	76	100	40
Dispatch and Track-In Material	5	40.7	1	1	1	34	39	10
Dispatch and Track-In Material	6	22.0	1	0	0	63	80	30
Perform Setup For Material at Resource	1	25.8	0	0	0	100	100	100
Track-Out and Move-Next Material	1	16.4	0	0	0	100	100	100
Clone Material	1	56.5	0	0	0	100	100	100
Create Rule	1	17.6	0	0	0	100	100	100
Load Master Data Package	1	2.6	0	0	0	100	100	100
Load Master Data Package	2	18.5	0	0	0	100	100	100
Load Master Data Package	3	4.3	0	0	0	100	100	100
Create Master Data Package	1	13.8	0	0	0	100	100	100
Create Master Data Package	2	34.6	0	0	0	100	100	100
Create Change Set	1	3.8	0	0	0	N/A	N/A	N/A
Import	1	2.5	0	0	0	N/A	N/A	N/A

Table 5.1: All Local Wizard Scores with Different Formulas

At first glance at this results table, we can tell that many of the wizards got a perfect score. This is due to the simplicity of some wizards (e.g., one step with trivial actions) in the platform and the fact that we do not consider the difficulty of completing a certain wizard. Having so many perfect scores affects the global score averages, as the values will likely always be high due to the volume of easy wizards often found. We can confirm this observation in figure 5.9, as regardless of the employed formulas, the lowest average was 68.8.

Nevertheless, we can overlook that and compare the work of our formulas on wizards with score disparities by focusing on an excerpt of table 5.1, with more meaningful entries. Table 5.2 contains only some of the rows we hand-picked to showcase the behavioral differences of our formulas.

Wizard Group Name	i	t(s)	Completed	Err	Fail	Back	A	B	C
Create Area	1	470.4	N	1	0	3	0	27	0
Create Area	2	25.8	N	0	0	0	76	100	40
Create Data Collection	1	53.7	N	2	0	0	45	59	20
Dispatch and Track-In Material	3	73.7	N	1	0	0	55	79	30
Dispatch and Track-In Material	5	40.7	N	1	1	1	34	39	10
Dispatch and Track-In Material	6	22.0	N	1	0	0	63	80	30

Table 5.2: Excerpt of Local Wizard Scores with Different Formulas

Create Area 1 was not completed, and the user spent a lot of time on it. According to formula A, it will get heavily punished for that on top of the negative actions. However, formulas B and C do not regard time spent as much. C is still 0 due to the high static cancellation, and B is low due to the extra negative action multiplier.

- $a = 100 - 15 \times 0 - 10 \times 1 - 5 \times 3 - 20 - 3 \times 4 - \frac{470.4}{6} \approx -35$ (corrected to 0)
- $b = 100 - 15 \times 0 - 10 \times 1 - 5 \times 3 - 0 - 10 \times 4 - \frac{470.4}{60} \approx 27$
- $c = 100 - 10 \times 0 - 10 \times 1 - 10 \times 3 - 60 - 0 \times 4 - \frac{470.4}{600} \approx -1$ (corrected to 0)

For wizard **Create Area 2**, we can see that formula B believes there is nothing wrong with the interaction, while the others make some judgments. Formula C drops any score severely if the wizard was not completed, while formula A is more balanced

- $a = 100 - 15 \times 0 - 10 \times 0 - 5 \times 0 - 20 - 3 \times 0 - \frac{25.8}{6} \approx 76$
- $b = 100 - 15 \times 0 - 10 \times 0 - 5 \times 0 - 0 - 10 \times 0 - \frac{25.8}{600} \approx 100$
- $c = 100 - 10 \times 0 - 10 \times 0 - 10 \times 3 - 60 - 0 \times 0 - \frac{25.8}{600} \approx 40$

A good way of visualizing the scores' differences is through a bar chart, shown in figure 5.13 below. We labeled Create Area wizards as **CA**, Create Data Collection as **CDC**, and Dispatch and Track-In Material as **DTIM**.

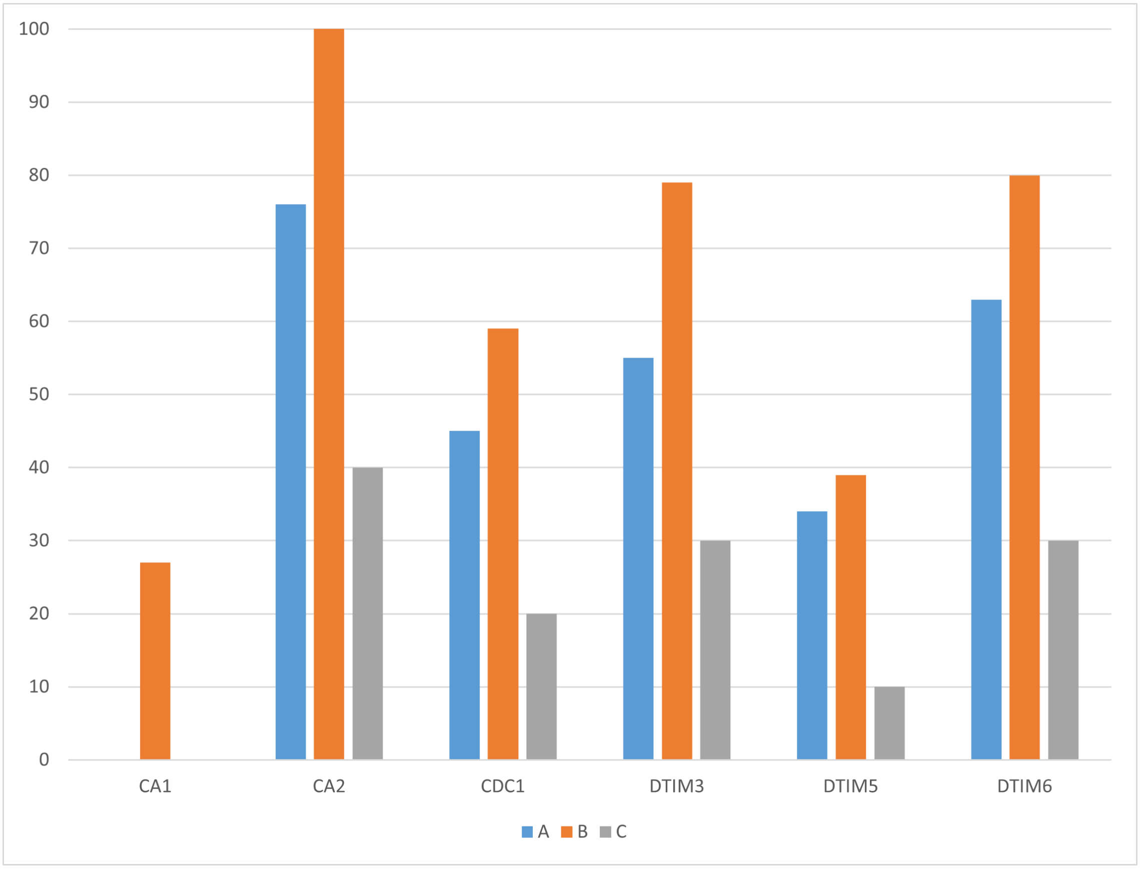


Figure 5.13: Bar Chart for Wizard Scores depending on Formula

As shown in the bar chart in figure 5.13, formula B is the most forgiving assessor for the selected sample of wizards. The reason for this is a combination of **low regard for time spent**

in wizard, **absence of a static cancellation penalty** and, despite the **high extra penalty for negative actions**, the **absence of many negative actions on the wizards we tested**. Formula B is mostly about punishing what we consider to be negative actions.

On the other side of the spectrum is formula C, which uses a philosophy of **heavily punishing any wizard that is not completed**, but also not discarded. This approach almost entirely **disregards the influence of time**, **does not apply any additional negative action penalty**, and judges **all negative action equally**. Because our sample only consists of only canceled wizards, every wizard only gets at most 40 points. Pairing this with a few negative actions results in very low scores.

Finally, formula A is the most balanced of the three formulas for the sample we selected. This approach focuses on **punishing time significantly** (1 point per 6 seconds), gives a **moderate additional negative action penalty** and also a **moderate static cancellation penalty**. This results in very middle-of-the-pack scores.

However, we do not have to choose the "best" formula for multiple reasons. Firstly, the three formulas we provided are based on hypotheses, and we can come up with more appropriate formulas depending on the collected events we have and how we scrutinize them. Secondly, the formulas heavily depend on the demography of the sampled wizards, and only by collecting lots of data in a real scenario can we perform a fair analysis. Our research attempts to **lay a foundation of wizard assessment**, which should be perfected over time. This is how we hand over the project to CMF: a work in progress with valuable biased and unbiased insights and a sound basis for more research. From this point on, CMF teams can use our tooling to discuss what should be improved in terms of features and layout in our solution and investigate how we can continue to assess application components based on judgments on events.

5.5 Limitations

The most significant limitation in validation comes from the nature of CMF's MES deployment and its security-related constraints. The best way to validate our solution would be to apply it in real shop floor scenarios where CMF's MES is used to manage manufacturing. In a scenario of this nature, our solution would be rapidly populated by interactions within MES rapidly and vastly, according to the use cases of the concrete client using our MES. However, we are unable to take advantage of this setup for a couple of main reasons:

- **Sensitive client information**: Including our solution in a real scenario would require contacting management personnel from the client, as our system would collect data, study, and publicly scrutinize it. Getting this approval is complicated.
- **Pipeline and code approval**: Our Matomo tracking instructions which we implemented on a development branch of one of CMF's MES repositories, would need to be approved and shipped within a production release to reach a real deployment stage.

These reasons were enough to rule out this possibility in the time frame of this study and internship. It is essential to remember that the usability of CMF's MES also depends on the user's knowledge of CMF's MES and its business rules, meaning that end-users generated data is more meaningful than development or staging-generated data.

With that said, we still can perform meaningful validation of our solution, and this limitation can be pinned as a future work task.

One example that can showcase the magnitude of this limitation is the fact that we cannot predict the average behavior of the end-users in a real scenario. For example, in section 4.4.1, we assumed one of the formulas that time spent on wizards would strongly impact the points in case the wizard was not completed since we assume that the longer the user spent without completing the wizard, the more struggles were being faced. The rate at which we deduct points and the relevance of the time spent on the wizard were based on assumptions that can only truly be validated upon sending our solution into complete production scenarios. Only then will we have clarity on our scoring methodology and in defining the next steps of our project.

5.6 Takeaways & Summary

Following the previous two sections, we can formulate answers to the research questions:

- **RQ1**: "*Can custom event tracking in SPAs be used to generate meaningful insights about user interactions, operability, and behavior?*". Yes. Even discarding our biased usability assessments with a score, our event tracking and processing allow CMF to identify wizards with the most negative actions (e.g., errors and failed steps). Directly observing these facts alone can help identify the likely poor-performing wizards (or other transactional components) in the application.
- **RQ2**: "*Can we create a scoring scale and formula that predicts usability for custom transaction components in SPAs?*" Yes, but this scoring scale remains a work in progress. The scoring scale can be a massive advantage in the observation of metrics. Suppose there are thousands of wizards for each wizard group. In that case, we can automate determining the worst-performing wizards by attributing weighted values to the events inside a wizard. Perfecting the parameters and coefficients for this formula will help bring to light the wizards (or other transactional components) that need attention and is certainly a point that requires future work.
- **RQ3**: "*Can the insights derived from custom event tracking in SPAs be effectively visualized and presented in a custom dashboard to aid developers in making design decisions?*" Yes, the customized dashboard can very arguably be favorable, as it provides a lens into the specialized data collected, allowing developers to have a detailed and refined view of events that happened in the application. However, the dashboard's layout, usability, and features can only be validated by those who will use it in later stages where a real CMF's MES installation accompanies it.

- **RQ4**: "*How can we link quantitative results associated with events to qualitative remarks about user experience?*". Our quantitative approach can be tweaked via feedback from the UI/UX team using our solution. To validate the biased quantitative methods, like the wizard scoring, we can run controlled experiments in CMF's MES, ask questions about the interaction, and use the feedback to compare to what we see on the dashboard. Furthermore, we could introduce components in CMF's MES to provide a qualitative assessment of the interaction in a random wizard and, upon collecting a significant volume of data, compare the results to the automated wizard scoring.

As we saw in this chapter, our solution automatically collects many data that will be reported both on the customized Next.js dashboard and on the Matomo dashboard, regardless of the qualitative judgments we made. Our customized insights provide transparent and interactive potential with the collected data, equipping CMF with organized, innovative insights into their primary platform.

Chapter 6

Conclusions & Future Work

Contents

In this section, we look to wrap up this research project with some interesting takeaways, some things to look back on and reflect on the decisions and developments made. We start by looking at the advantages and impact of our work in section 6.1. We follow that by looking at the difficulties and limitations we faced in section 6.2. Last but certainly not least, we wrap up the dissertation in section 6.3 by stating key directions and next steps for this project and research.

6.1 Advantages & Impact

Our scheduled research and development phase has culminated in a robust, scalable, and valuable solution that addresses the proposed challenge and serves as a foundation for future research. The impact of this research is clear, as it provides a framework for making informed decisions related to the application layout, thereby enhancing the efficiency and effectiveness of CMF's operations. Even if we consider that our scoring approaches are not feasible and the validation needs further proof, the insights and metrics we already collect are objectively valuable regardless of the evaluations that we made.

We can uncover and address the bottlenecks within CMF's MES by continuously expanding and refining our solution. This proactive approach will streamline the application's operations, making it easier for users to perform their desired tasks.

Based on our solution's observation and future versions, the UI optimizations can place both the company and its software in a privileged position within the industry. In conclusion, our research has delivered a promising solution and paved the way for future investigations that can further enhance CMF's competitive advantage.

6.2 Difficulties

The security and confidentiality restrictions we faced from the beginning did not hinder this investigation, as we managed to deliver a solution that complies with the requirements. However, the validation of our solution was cut short due to the overhead and impracticability of joining our solution components to a real-life scenario. As we saw before, we moved our project components to docker containers, which facilitates creating a staging environment. However, CMF's MES has complex customized installations for different enterprises and clients and uses cases of the application. Alongside that difficulty, approving the changes we made within CMF's MES would take a long time and require clients' permission to use their data. Deploying new versions of CMF's MES is a complicated procedure, and it was not feasible in the context of our project.

Although there were complications along the way, the only significant severe difficulty that was impossible to overcome or circumvent was not making our solution reach a real scenario. If we overcame this issue, with just one week of actual production activity within CMF's MES, we could gain a strong sense of what we missed, what we did right, and how to improve our solution. Alongside that, the UI/UX teams could immediately make recommendations for real-life scenarios.

6.3 Future Work

Future work for this research encompasses resolving the difficulties we faced but also investing in some other changes to the solution that should have been considered in the scope of this dissertation. From this point forward, we can suggest the following future work points that can help improve our solution and further validate it:

- **Streamline solution validation process**: One major difficulty we faced was the inability to take our solution to a real production scenario. Instead of settling for local data collection or doing so in a controlled staging environment, we must find a way of testing and validating the application in a real production scenario to make informed adjustments to our insights platform. The feedback from a production scenario could even provide additional ideas of considerations we should make or invalidate others we already are. A real scenario would be a great stress test for our solution and a great eye-opener to define the next steps of development.
- **Drop unrealistic assumptions**: In section 4.1.2, we ignored some characteristics of the wizards because it was unfeasible to do otherwise in the time frame we had. Suppose we had an unlimited amount of time. In that case, we should **determine step difficulty** in a wizard, **determine wizard difficulty** after its submission (sum of difficulty of all steps), and also base the average scores and progress metrics **depending on the difficulty of steps and wizards**. These comparisons would be fairer and provide more trustworthy insights.

- **Collect events with extra information**: In section 4.1.1 we showcased the collected events within transactional components. However, for example, we considered that **all failed steps and all errors are of the same nature and severity**. Evaluating the type and nature of errors requires much work on CMF's MES side, which made it unfeasible for this research but is another key future work point that can take this solution to the next level.
- **Manually configurable formulas**: One feature that could accelerate the testing, validation, and improvement of the scoring formulas would be the possibility of manually defining the scoring constants (also known as weights) for each formula. Section 4.4.2 talks about the flexibility of adding and using new formulas on the client, but we could go a step further. For example, the formula of type A considers that failed steps should deduct 15 points. However, suppose the UI allowed the user to tweak these values. In that case, we could be looking at a more customizable and interactable solution, which makes way for finding the best constant values for our formula. Perhaps, in our research, we may realize that the scoring approach is harshly judging a particular event like an error, and if we had this capability in our UI, we would not have to touch the code. This feature would involve **allowing the user to add new formulas to the UI** or manually **editing them there**.
- **User feedback toasts**: As previously discussed, one of the primary motivations for this project is the challenge of obtaining user feedback due to various factors such as shop floor communication complexities, dynamic and complex behavior, and varying custom MES installations. An approach not considered within our project scope is incorporating UI components in MES to gather user feedback periodically. For instance, MES could ask its users to rate the ease and smoothness of a sequence of actions on a scale of 1 to 5. Assuming we collect this data over time, we could correlate these ratings with other metrics we have extracted to validate our solution further and develop better-automated usability assessment methods. These sporadic qualitative methods could enhance our quantitative measures by allowing for comparative analysis, provided we have sufficient data.

References

[1] Adobe. Analytics that give you actionable insights. not just canned reports. https://www.adobe.com/analytics/adobe-analytics.html. Accessed: 2023-06-26.

[2] Shahzeb Ahmed. Mariadb vs mysql. https://www.cloudways.com/blog/mariadb-vs-mysql/, 2022. Accessed: 2023-06-28.

[3] Fathom Analytics. How to use fathom analytics. https://usefathom.com/, 2023. Accessed: 2023-06-28.

[4] M. Beasley. *Practical Web Analytics for User Experience: How Analytics Can Help You Understand Your Users*. ITPro collection. Elsevier Science, 2013.

[5] Boxes and Arrows. The limitations of server log files for usability analysis. https://boxesandarrows.com/the-limitations-of-server-log-files-for-usability-analysis, 2023. Accessed: 2023-06-24.

[6] Deniz Colak. What are the advantages of typescript over javascript? https://www.altogic.com/blog/what-are-the-advantages-of-typescript-over-javascript, 2023. Accessed: 2023-06-28.

[7] B. Saenz de Ugarte, A. Artiba, and R. Pellerin. Manufacturing execution system – a literature review. *Production Planning & Control*, 20(6):525–539, 2009.

[8] Crazy Egg. Make your website better. instantly. https://www.crazyegg.com/. Accessed: 2023-06-26.

[9] Minko Gechev. Angular v15 is now available, 2022. Accessed: 2023-06-28.

[10] Google. Google analytics capabilities for spas. https://support.google.com/analytics/#topic=9143232. Accessed: 2023-02-05.

[11] L. Hay. *Researching UX: Analytics*. Aspects of UX. SitePoint Pty. Limited, 2017.

[12] Hotjar. Website heatmaps & behavior analytics tools. https://www.hotjar.com/. Accessed: 2023-02-05.

[13] Kissmetrics. Simple analytics that track human behavior to increase revenue. https://www.kissmetrics.io/. Accessed: 2023-06-26.

[14] Jürgen Kletti. *Manufacturing Execution Systems—MES*. Springer, 2007.

[15] Soujanya Mantravadi et al. An overview of next-generation manufacturing execution systems: How important is mes for industry 4.0? *Procedia Manufacturing*, 30:588–595, 2019.

[16] Matomo. Matomo api reference. https://developer.matomo.org/api-reference/reporting-api. Accessed: 2023-06-29.

[17] Matomo. Matomo data ownership and privacy. https://matomo.org/100-data-ownership/. Accessed: 2023-06-29.

[18] Matomo. Matomo database setup. https://matomo.org/faq/how-to-install/faq_55/. Accessed: 2023-06-28.

[19] Matomo. The google analytics alternative that protects your data. https://matomo.org/, Feb 2023. Accessed: 2023-02-05.

[20] Michael Mikowski and Josh Powell. *Single Page Web Applications: JavaScript End-to-End*. Manning Publications Co., USA, 1st edition, 2013.

[21] Mixpanel. Progress is possible: Simple and powerful analytics that helps everyone make better decisions. https://mixpanel.com/. Accessed: 2023-06-26.

[22] Optimizely. Welcome to the home of exceptional digital experiences. https://www.optimizely.com/. Accessed: 2023-06-26.

[23] Sujeet Pillai. Incentius. https://www.incentius.com/blog-posts/pros-and-cons-of-using-tailwind-css/, 2022. Accessed: 2023-06-28.

[24] Emmanuel Roux. @ngx-matomo public repository on github.com. https://github.com/EmmanuelRoux/ngx-matomo, 2023. Accessed: 2023-06-28.

[25] Emmanuel Roux. @ngx-matomo published on npmjs.com. https://www.npmjs.com/package/@ngx-matomo/tracker#advanced-use-cases, 2023. Accessed: 2023-06-28.

[26] Ahmed Seffah, Mohammad Donyaee, Rex B. Kline, and Harkirat K. Padda. Usability measurement and metrics: A consolidated model. *Software Quality Journal*, 14(2):159–178, 2006.

[27] Social Shepherd. 14 essential google analytics statistics you need to know in 2023. https://thesocialshepherd.com/blog/google-analytics-statistics, 2023. Accessed: 2023-06-24.

[28] Vercel. Next.js by vercel - the react framework. https://nextjs.org, 2023. Accessed: 2023-06-28.

[29] Adam Wathan. Tailwind css presentation. https://tailwindcss.com, 2022. Accessed: 2023-06-28.

[30] Woopra. End-to-end customer journey analytics. https://www.woopra.com/. Accessed: 2023-06-26.

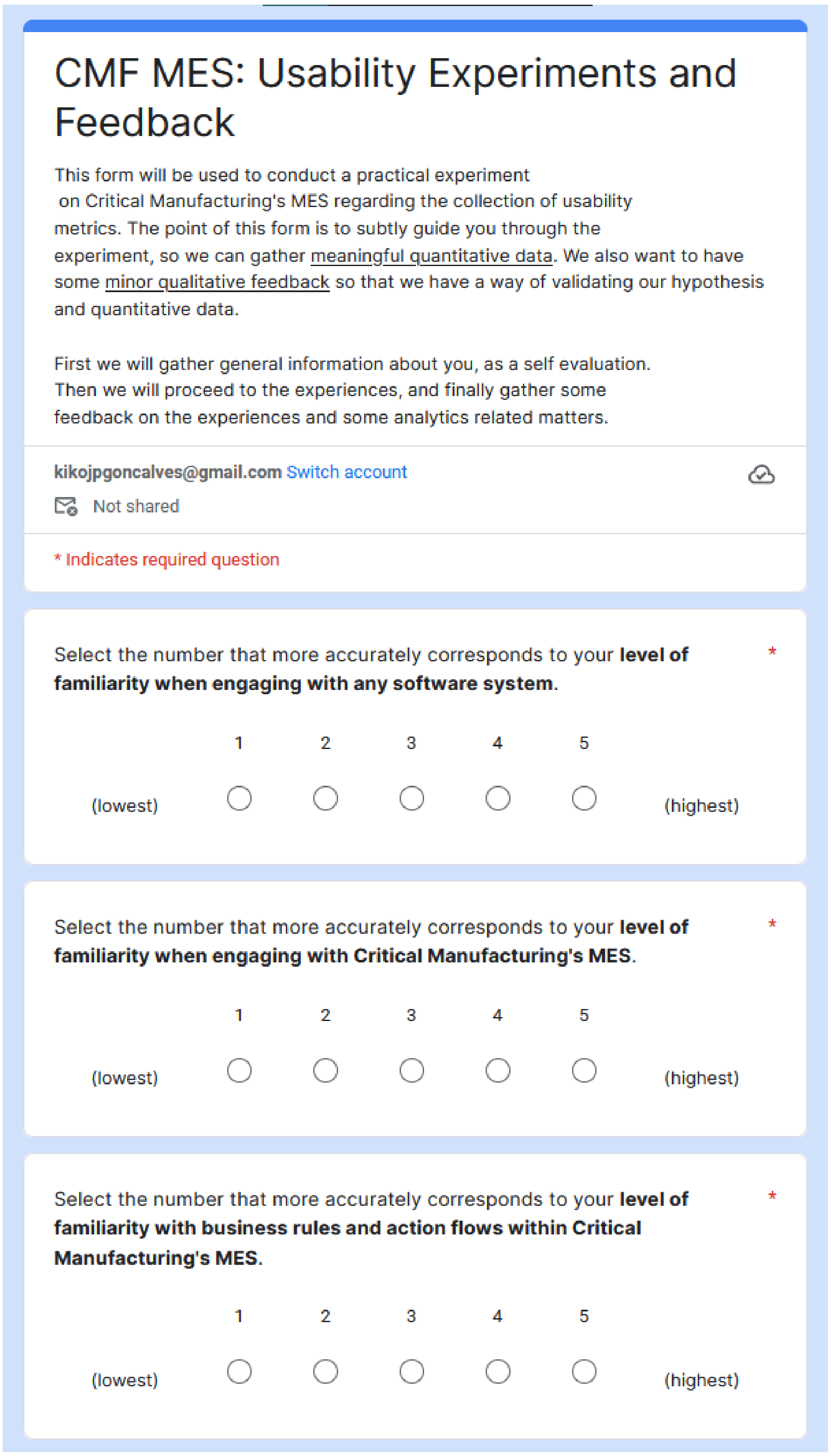
Appendix A

Next.js Client-side Fetching and Processing Example

```
React.useEffect(() => {
  if (!willFetch) return;
  setError(false);
  setLoading(true);

  fetch('/api/matomo/events/wizard')
    .then((res) => {
      if (!res.ok) throw new Error(res.statusText);
      return res.json();
    })
    .then((data: ITrackerEventGroup[]) => {
      setLoading(false);
      setWillFetch(false);
      setRawData(data);
      const evaluatedWizards = evaluateAndGroupWizards(data);
      setProcessedData(evaluatedWizards);
    })
    .catch((error) => {
      setError(true);
      setLoading(false);
      setWillFetch(false);
      console.error(error);
    });
}, [willFetch]);
```

Appendix B

Controlled Experiments Form



CMF MES: Usability Experiments and Feedback

This form will be used to conduct a practical experiment on Critical Manufacturing's MES regarding the collection of usability metrics. The point of this form is to subtly guide you through the experiment, so we can gather meaningful quantitative data. We also want to have some minor qualitative feedback so that we have a way of validating our hypothesis and quantitative data.

First we will gather general information about you, as a self evaluation. Then we will proceed to the experiences, and finally gather some feedback on the experiences and some analytics related matters.

kikojpgoncalves@gmail.com Switch account

Not shared

* Indicates required question

Select the number that more accurately corresponds to your **level of familiarity when engaging with any software system**. *

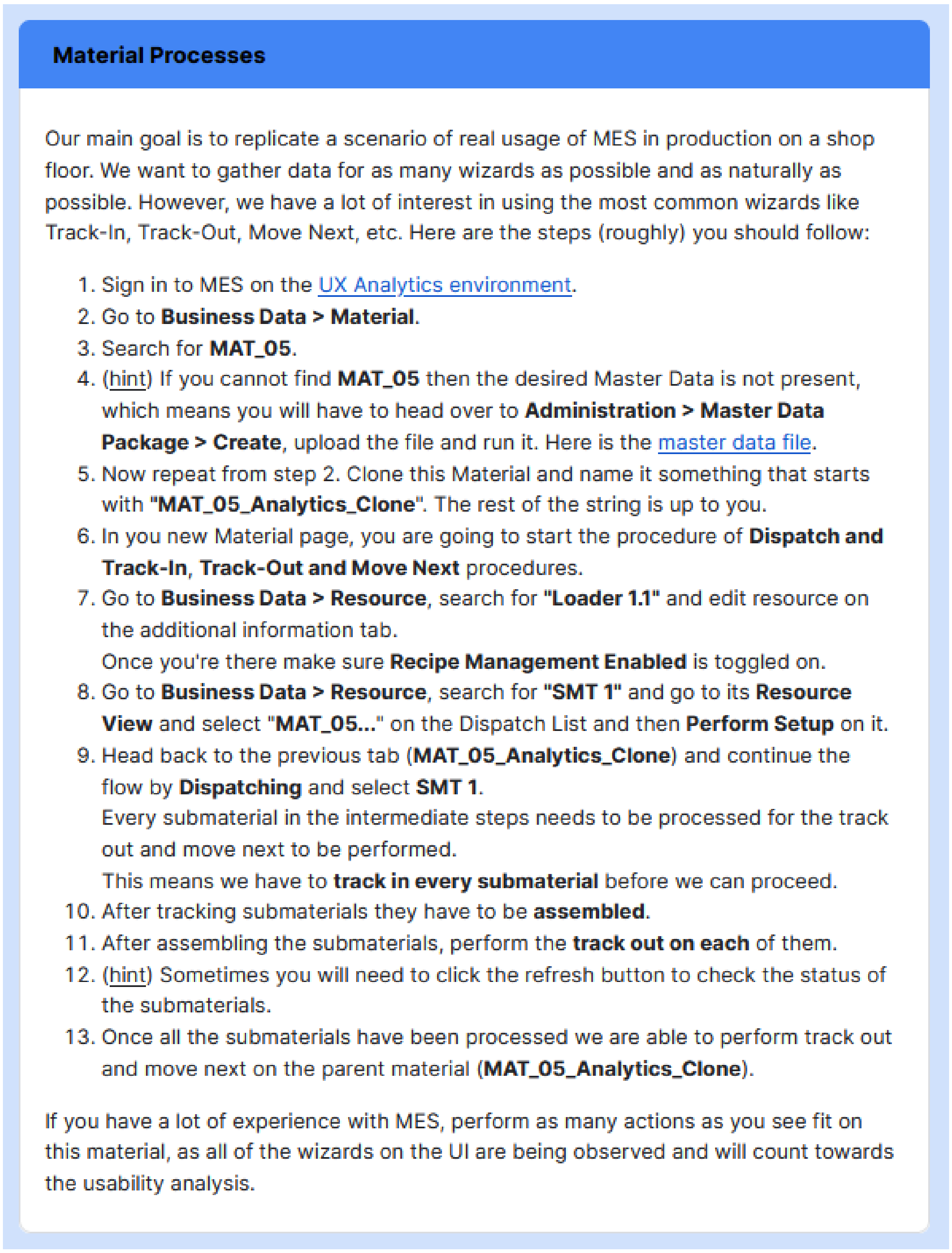
	1	2	3	4	5	
(lowest)	○	○	○	○	○	(highest)

Select the number that more accurately corresponds to your **level of familiarity when engaging with Critical Manufacturing's MES**. *

	1	2	3	4	5	
(lowest)	○	○	○	○	○	(highest)

Select the number that more accurately corresponds to your **level of familiarity with business rules and action flows within Critical Manufacturing's MES**. *

	1	2	3	4	5	
(lowest)	○	○	○	○	○	(highest)

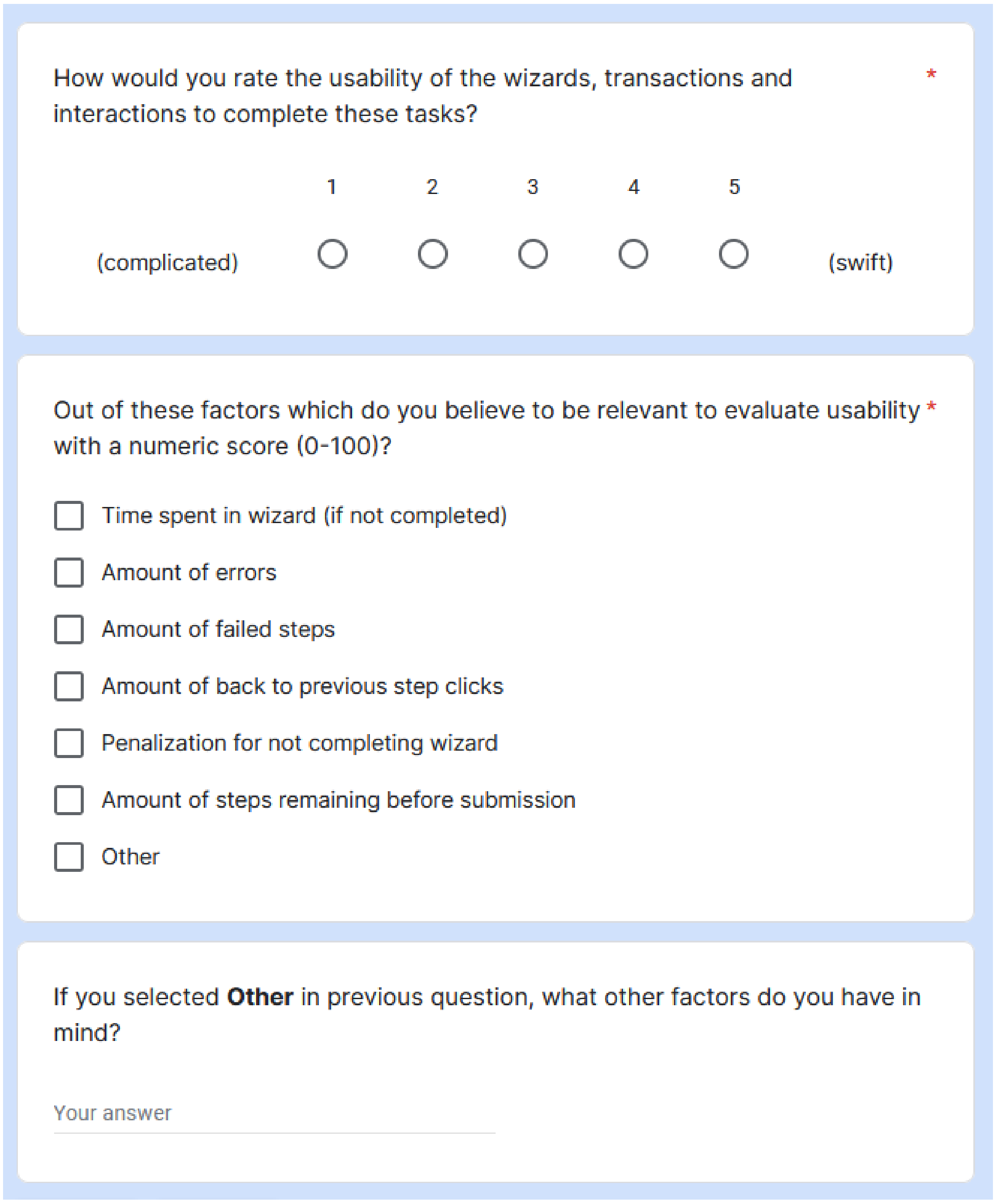


Material Processes

Our main goal is to replicate a scenario of real usage of MES in production on a shop floor. We want to gather data for as many wizards as possible and as naturally as possible. However, we have a lot of interest in using the most common wizards like Track-In, Track-Out, Move Next, etc. Here are the steps (roughly) you should follow:

1. Sign in to MES on the UX Analytics environment.
2. Go to **Business Data > Material**.
3. Search for **MAT_05**.
4. (hint) If you cannot find **MAT_05** then the desired Master Data is not present, which means you will have to head over to **Administration > Master Data Package > Create**, upload the file and run it. Here is the master data file.
5. Now repeat from step 2. Clone this Material and name it something that starts with **"MAT_05_Analytics_Clone**". The rest of the string is up to you.
6. In you new Material page, you are going to start the procedure of **Dispatch and Track-In**, **Track-Out and Move Next** procedures.
7. Go to **Business Data > Resource**, search for **"Loader 1.1"** and edit resource on the additional information tab.
   Once you're there make sure **Recipe Management Enabled** is toggled on.
8. Go to **Business Data > Resource**, search for **"SMT 1"** and go to its **Resource View** and select "**MAT_05...**" on the Dispatch List and then **Perform Setup** on it.
9. Head back to the previous tab (**MAT_05_Analytics_Clone**) and continue the flow by **Dispatching** and select **SMT 1**.
   Every submaterial in the intermediate steps needs to be processed for the track out and move next to be performed.
   This means we have to **track in every submaterial** before we can proceed.
10. After tracking submaterials they have to be **assembled**.
11. After assembling the submaterials, perform the **track out on each** of them.
12. (hint) Sometimes you will need to click the refresh button to check the status of the submaterials.
13. Once all the submaterials have been processed we are able to perform track out and move next on the parent material (**MAT_05_Analytics_Clone**).

If you have a lot of experience with MES, perform as many actions as you see fit on this material, as all of the wizards on the UI are being observed and will count towards the usability analysis.



How would you rate the usability of the wizards, transactions and interactions to complete these tasks? *

	1	2	3	4	5	
(complicated)	○	○	○	○	○	(swift)

Out of these factors which do you believe to be relevant to evaluate usability with a numeric score (0-100)? *

- ☐ Time spent in wizard (if not completed)
- ☐ Amount of errors
- ☐ Amount of failed steps
- ☐ Amount of back to previous step clicks
- ☐ Penalization for not completing wizard
- ☐ Amount of steps remaining before submission
- ☐ Other

If you selected **Other** in previous question, what other factors do you have in mind?

Your answer