communicated by:
Lehr- und Forschungsgebiet Informatik 9

Prof Dr. Ulrik Schroeder

RWTH AACHEN UNIVERSITY

Diese Arbeit wurde vorgelegt am Lehr- und Forschungsgebiet Informatik 9

The present work was submitted to Learning Technologies Research Group

Entwicklung eines Qualitätssicherungsmodells für das OER Conversion Tool (convOERter)

Developing a Quality Assurance Model for OER Conversion Tool (convOERter)

Masterarbeit

Master-Thesis

von / presented by

Khalid, Muhammad Waseem

403545

Prof. Dr. Ulrik Schroeder

Prof. Dr. Horst Lichter

Betreut von /Supervised by
Dr. rer. nat. Lubna Ali

Aachen, 12.11.2024

Table of Contents

List of Figures

List of Tables

Abstract

Open Educational Resources (OER) has vast impact in enhancing educational openness, but creation of OER documents are associated with number of challenges. The convOERter, a web-based tool is designed to address these challenges and simplifies the process by helping in the conversion of non-OER resources through the extraction and replacement of images, providing educators with an automated solution. This thesis focuses on applying a Software Quality Assurance (SQA) model and developing software quality metrics for the convOERter tool. The research investigates the use of user testing techniques to assess the functionality, reliability, and availability of the convOERter tool. Furthermore, it evaluates quality metrics using various techniques, such as Black-box testing and the Goal-Question-Metric approach, offering a comprehensive assessment of the tool's effectiveness and identifying potential areas for improvement. The overall objective of this thesis is to provide valuable insights to enhance the performance and adoption of the convOERter tool in the domain of Open Educational Resources conversion.

Zusammenfassung

Offene Bildungsressourcen (Open Educational Resources, OER) haben einen großen Einfluss auf die Verbesserung der Offenheit des Bildungswesens, aber die Erstellung von OER-Dokumenten ist mit einer Reihe von Herausforderungen verbunden. Der convOERter, ein webbasiertes Tool, wurde entwickelt, um diese Herausforderungen anzugehen und vereinfacht den Prozess, indem es bei der Konvertierung von Nicht-OER-Ressourcen durch die Extraktion und Ersetzung von Bildern hilft und Lehrenden eine automatisierte Lösung bietet. Diese Arbeit konzentriert sich auf die Anwendung eines Softwarequalitätssicherungsmodells (SQA) und die Entwicklung von Softwarequalitätsmetriken für das convOERter-Tool. Die Forschung untersucht den Einsatz von Benutzertestverfahren, um die Funktionalität, Zuverlässigkeit und Verfügbarkeit des convOERter-Tools zu bewerten. Darüber hinaus werden die Qualitätsmetriken mit Hilfe verschiedener Techniken wie Black-Box-Tests und dem Goal-Question-Metric-Ansatz bewertet, um eine umfassende Bewertung der Effektivität des Tools zu ermöglichen und potenzielle Verbesserungsbereiche zu identifizieren. Das übergeordnete Ziel dieser Arbeit ist es, wertvolle Erkenntnisse zu liefern, um die Leistung und die Akzeptanz des convOERter-Tools im Bereich der Konvertierung offener Bildungsressourcen zu verbessern.

Acknowledgement

In the name of Allah, the most beneficent and the most merciful.

I extend my deepest gratitude to my wife for her continuous support, patience, and encouragement throughout my studies. A special mention to my beloved daughter, Anaya, whose joy and warmth strengthened me during the journey of completing this thesis.

I would like to thank my supervisor Lubna Ali for her consistent support and guidance. Her ideas, feedback and insights were always precious and available throughout my thesis. Without her guidance, I would not be able to complete this research.

I appreciate the assistance of my examiners, Prof. Dr Ulrik Schroeder and Prof. Horst Lichter, for my thesis study.

Lastly, I wish to honor the memory of my late father, whose dream was for me to pursue higher studies abroad. Though he is not here to share this moment, I hope this achievement fulfills a part of his vision for me.

Chapter 1 Introduction

1.1 Motivation

The educational sector has undergone a significant change due to the emergence of Open Educational Resources (OER). OER provides freely accessible educational resources that promote educational openness and facilitate the sharing of knowledge worldwide. However, while OER has the potential to improve educational access, the conversion of non-OER materials into OER-compliant resources remains a major challenge. Educators frequently encounter challenges like managing open licenses, choosing suitable images, and substituting resources with open-source alternatives.

To handle these issues, the Learning Technologies Research Group at RWTH has developed the convOERter tool, an online platform aimed at facilitating the OER conversion process. convOERter streamlines the extraction and substitution of non-OER images with OER images, offering educators an effective tool for developing OER-compliant resources. However, as the need for OER keeps increasing, guaranteeing the reliability, functionality, and usability of the convOERter tool is important for its acceptance and sustained success.

In this context, the application of a Software Quality Assurance (SQA) model becomes essential. SQA includes various methodologies and testing techniques designed to ensure that the software meets quality standards. For the convOERter tool, applying SQA practices will ensure that it meets educators' needs while having good performance, reliability, and user satisfaction.

1.2 Thesis Goal

The goal of this thesis is to apply a Software Quality Assurance (SQA) model to the convOERter tool, focusing on enhancing its functionality, reliability, and availability (Khalid et al., 2024). This research uses black-box testing methods to assess the tool's external functionality and performance without examining its internal code structure. This approach allows for a comprehensive evaluation of how the tool functions from the viewpoint of the end user, including testing its reliability in real-world scenarios. To implement a software quality assurance model to the tool, the following research question needs to be addressed.

Research Questions:

How does the application of software quality assurance techniques resolve the issues of user experience and functionality of convOERter?

This research question is further divided into sub-questions to utilize a modular approach for solving the problem, which are as follows:

I. Which software quality assurance model could be applied to convOERter?
II. How do software quality metrics help identify and resolve user experience and adoption issues in convOERter?
III. How can functionality assessment enhance the OER conversion process?

Additionally, this thesis aims to develop software quality metrics that will serve as benchmarks for assessing the convOERter tool's effectiveness. These metrics will guide the evaluation process, providing quantitative data that can identify potential areas for improvement. By combining user testing techniques and software quality metrics, the research will offer a detailed examination of the tool's strengths and limitations.

The findings of this study will contribute valuable insights toward enhancing the convOERter tool's performance and overall utility in the OER conversion process. By applying established SQA methodologies, this research aims to provide a more reliable and user-friendly tool, supporting the broader adoption of OER and promote the goal of educational openness.

1.3 Structure of the Thesis

The rest of the thesis is structured as follows:

Chapter 2 presents an overview of Open Educational Resources (OER), discussing definitions, the different types of Creative Commons licenses, and the benefits and challenges associated with OER adoption and structure of convOERter tool.

Chapter 3 reviews the scientific background and related work in the field of Software Quality Assurance (SQA). It introduces various testing techniques such as black-box, white-box, and grey-box testing and security vulnerability testing.

Chapter 4 introduces web application testing on the basis of quality metrics, user-centric testing, existing techniques available for web applications and research gaps which build the basis of designing research methodology.

Chapter 5 describes the techniques applied to convOERter i.e, blackbox testing technique, performance and accessibility testing, security vulnerability testing using ZAP and deriving software quality metrics using the GQM model technique..

Chapter 6 presents the results of the evaluation, analyzing the tool's functionality, reliability, usability, and areas for improvement. The chapter also describes the method to evaluate the applied methodology using user feedback and develop feedback analysis.

Chapter 7 concludes the thesis, offering insights into the findings, discussing limitations, and proposing future work aimed at further enhancing the convOERter tool's performance and expanding its impact within the OER community.

Chapter 2 Open Educational Resources (OER)

Open Educational Resources (OER) represent a significant advancement in the global education landscape. These resources comprise of learning material in the form of text, multimedia, images etc and are freely accessible, publicly licensed, that can be utilized for teaching, learning, and research purposes. The increasing integration of technology into everyday life has fostered a demand for accessible educational content that enables learners to get knowledge at their own pace. OER facilitates this by promoting the unrestricted publication of knowledge, allowing educators and learners to freely share, adapt, and build upon existing resources. By removing barriers to access, OER supports the continuous generation of new knowledge, fostering collaboration and innovation across disciplines. Consequently, OER has become a pivotal tool in enhancing educational opportunities and improving the accessibility of high-quality instructional materials globally.

2.1 OER Definitions

There are a number of definitions available for Open Educational Resources. Some of these can be found below:

1. The German Commission for UNESCO defined OER as "Open Educational Resources (OER) are teaching, learning and research materials in any medium – digital or otherwise – that reside in the public domain or have been released under an open license that permits no-cost access, use, adaptation and redistribution by others with no or limited restrictions." (*Education | German Commission for UNESCO*, n.d.)

2. Creative Commons defined OER as "teaching, learning and research materials that are either in public domain or licensed in a manner that provides everyone with free and perpetual permission to engage in the 5R activities – retaining, remixing, revising, reusing and redistributing the resources". ("O*pen Education," n.d.)*

The 5R principles mentioned in Creative Commons definition explain the purpose and basis of the resources categorized as an OER resource.

The principle of reusing allows the educational resource can be reused in its original form. This content can be reused in a number of ways like preparing lectures, examinations, presentations, websites etc (*OER Principles*, n.d.). The copy of the resource can be downloaded, duplicated or retained for personal reference. The resource can be changed or modified to meet specific needs like

an image can be used in a presentation by adding marks to it. The resource or content can be remixed with other content to create something new. The original or the altered resource can be redistributed to others to take advantage of it. If any of the educational resource comply with the above-mentioned principles then it can be considered as an Open Educational Resource.

2.2 Creative Commons License

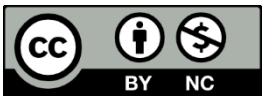
Creative Commons (CC) licenses(*About CC Licenses*, n.d.) are legal tools that allow creators to grant the public permission to use, share, and modify their work under specific conditions. These licenses were developed to facilitate the sharing of creative and educational resources, ensuring that authors can retain certain rights while promoting open access. For OER, CC licenses are crucial as they allow educators and institutions to legally distribute and adapt educational materials without violating copyright laws. By offering different levels of permission (such as attribution, non-commercial use, or no derivatives), CC licenses help maintain a balance between openness and intellectual property protection.

There are different types of Creative Commons licenses available which retain different forms of permissions to the creator and the user which are as follows:

CC BY: This license gives permission to the users to adapt, remix, redistribute or generate further resources based on original material in any format but the attribution or credit must be given to the creator.

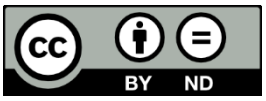
CC BY-SA: This license gives permission to the users to adapt, remix, redistribute or generate further resources based on of original material in any format but the attribution or credit must be given to the creator. This license allows commercial usage as well. If the resources are adapted or remixed then they should be licensed under the same terms.

CC BY-NC: This license gives permission to the users to adapt, remix, redistribute or generate further resources based on original material in any format but the attribution or credit must be given to the creator. The resources should be used for non-commercial purposes only.



CC BY-NC-SA: This license gives permission to the users to adapt, remix, redistribute or generate further resources based on original material in any format but the attribution or credit must be given to the creator. The resources should be used for non-commercial purposes only. . If the resources is adapted or remixed then it should be licensed under the same terms.

CC BY-ND: This license gives permission to copy and redistribute the resource in any format but it should be in original form and the user should give credit to the creator of the resource. The resource can be used for commercial purposes.



CC BY-NC-ND: This license gives permission to copy and redistribute the resource in any format but it should be in original form and the user should give credit to the creator of the resource. The resource can only be used for non-commercial purposes.

2.3 OER Benefits

Open Educational Resources contain different types of material ranging from digital textbooks, images, video lectures, and PowerPoint slides as well as some evaluation materials with automated answers. The convOERter tool currently supports following file formats for the extraction of images and replacement with OER images:

1. .pptx
2. .docx
3. .odp
4. .odt

OER resources can be found on a number of online platforms but one thing we need to consider before using an OER resource that we should check it is accompanied by creative commons license and our intended use should comply with the licensing requirements(Bates & Bates, 2015). If the license allows the use of resources for commercial purposes such as in the case of CC BY-SA then we are allowed to use it commercially otherwise not. Some of the major benefits of OER resources (*OER Benefits*, n.d.) are listed below:

Affordability: OER significantly reduces the cost of educational materials, making education more affordable and accessible for students.

Customization: Educators can easily modify OER to fit specific needs, enhancing both the teaching and learning experience.

Diverse Learning Formats: OER offers various content formats such as text, audio, and video allowing students to use material for multiple learning approaches such for a presenting research pptx format is available which can be generated using OER images and text.

Remote Access: As most OER are available digitally, students can access resources without expiration dates or the need for access permissions, ensuring ongoing availability.

Financial Relief and Success: By eliminating the cost of traditional textbooks, OER lessen students' financial overheads, which can positively impact their academic success and course completion rates.

Every educational resource that is freely available cannot be categorized as OER as there should be certain criteria that needs to be fulfilled. The Table. 1 below clearly explains which criteria make an OER resource:

Material Type	**Openly Licensed**	**Freely Available**	**Modifiable**
Open educational resources	Yes	Yes	Yes
Free online resources under all rights reserved copyright	No	Yes	No
Materials available through the university library	No	Yes	No
Open access articles and monographs	Yes	Yes	Maybe

Table 1 Components of an OER(What an OER Is Not, n.d.)

2.4 OER Challenges

The use of Open Educational Resources (OER) in higher education is accompanied by various challenges.

Quality Assurance: Developing reliable quality assurance mechanisms for OER is a considerable challenge. Unlike traditional educational materials, OER often lacks formal peer review, leading to concerns about the quality and credibility of these resources. This uncertainty may impact user trust and acceptance within educational institutions (*2010_Hodgkinson-Williams_Benefits-Challenges-of-OER-for-HE-Institutions.Pdf*, n.d.).

Financial Sustainability: Long-term funding for OER initiatives is a recurring issue. Many projects rely initially on grants and external funding sources, but sustaining these resources over time without a secure financial model proves difficult for most institutions. Continuous funding is essential for maintaining and updating OER content, so continuous support often remains uncertain (*Open Educational Resources Reviewing Initiatives and Issues.Pdf*, n.d.).

Technical Limitations and Interoperability: Many institutions face technical barriers, including limited broadband, outdated hardware, and incompatible software, which restrict the ability to create, share, and use OER effectively. This challenge is especially pronounced in areas with less robust technological infrastructure, making it difficult for the global accessibility of these educational resources (OECD, 2007).

Incentives and Institutional Support: Within academic institutions, faculty members may lack motivation to develop and share OER due to non-availability of incentives. Academic reward structures typically prioritize research output over resource development, leaving educators with little institutional support for engaging in OER initiatives. This lack of formal recognition limits educator participation in OER projects *(Hylen, 2006).*

Legal and Copyright Concerns: Copyright issues represent another significant challenge in OER development. Many educators are not familiar with copyright policies, especially regarding third-party content within OER. This lack of understanding can create hesitancy in using or adapting OER, as educators may fear unintentional legal implications (*PDF*, n.d.).

2.5 Structure of ConvOERter Tool

The convOERter tool comprises of three main components which are as follows:

Frontend: The frontend, designed as a Single Page Application using Angular Material Design, employs TypeScript classes for image extraction and singleton services for REST API calls and session management.

Backend: The backend handles user feedback, and authentication through GitLab, ensuring a secure and efficient architecture for OER conversion.

Database: It utilizes MongoDB database for data storage related to user feedback.

In addition to these components, the convOERter tool uses Docker for deployment of application, facilitating scalable deployment process. The architectural overview of the convOERter tool is shown below:

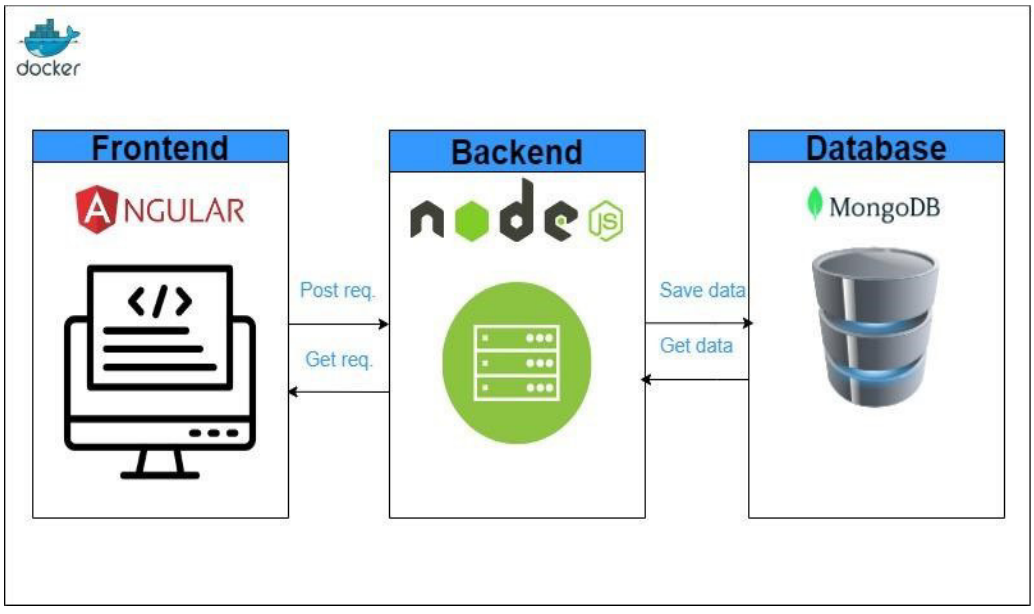


Figure 1. convOERter development structure

The Angular version 13.0.1 is used for the development of convOERter whereas Node version 18.13.0 and Mongoose version 5.11.13 are used for the development purpose.

2.6 Research Questions

To address the challenges of ensuring that the convOERter tool is both functional and user-friendly, this research focuses on applying software quality assurance (SQA) techniques. As more educators are using Open Educational Resources (OER), tools like convOERter must be reliable, easy to use, and meet educators' expectations. This research aims to explore how applying SQA methods can help to solve issues like user experience and tool functionality in convOERter, making it a more effective tool.

Main Question

1. How does the application of software quality assurance techniques resolve the issues of user experience and functionality of convOERter?

   This question focuses to understand how using quality assurance methods can help improve the tool's usability and performance, ensuring it meets the needs of educators. By focusing on both functional and user experience aspects, this research will offer insight into how SQA can enhance the tool's overall effectiveness in converting educational resources. To make this question easier to address and to organize the research effectively, the main question is divided into several sub-questions. These sub-questions break down specific parts, allowing for a focused study of each part and making the research process more manageable.

Sub-Questions

IV. Which software quality assurance model could be applied to convOERter?
V. How do software quality metrics help identify and resolve user experience and adoption issues in convOERter?
VI. How can functionality assessment enhance the OER conversion process?

In exploring these sub-questions, this study seeks to identify a suitable software quality assurance model that aligns with the specific needs of convOERter as an OER tool. Selecting the right technique will provide a structured way to evaluate and enhance the tool. By using software quality metrics, this research will gather measurable insights into convOERter's performance and user-experience, helping to identify the areas of improvement in the tool so that it be more easily adopted and effectively used by educators. Additionally, regularly assessing the functionality of convOERter will ensure it consistently works as expected across various user requirements, enhancing its reliability and value for those engaged in creating open educational resources. These approaches aim to build a more robust, efficient, and user-friendly tool that meets the needs of its users.

Chapter 3 State of the art

3.1 Overview of Software Quality Assurance in Web Applications

In web application development, ensuring software quality is the basis of creating reliable, efficient, and user-friendly applications. Software Quality Assurance (SQA) involves a set of systematic approach to ensure that the software processes and product confirms to the goals of SQA which are to improve software quality by appropriately monitoring both software and the development process to ensure compliance with the established software development standards and procedures *(El-Rayyes & Abu-Zaid, 2012)*. Software Quality Assurance is particularly important in web applications as these serve large and diverse user groups, requiring high performance, security, and usability. The user interface and user experience as key parameters in getting a higher level user satisfaction. Robust software quality assurance practices are key to avoid issues such as functionality problems, security vulnerabilities, and low user experiences, all of which can lead to significant financial and time losses for the companies. Quality improvements affect operations performance in different ways like reducing development or maintaining cost, improving productivity and increasing revenue. Quality is considered as one of the major drivers of competitive strategy in every industry *(Li & Meissner, 2009)*.

A study has presented*(Nidhra & Dondeti, 2012)* that achieving comprehensive quality assurance often involves combining black-box and white-box testing methodologies. Black-box testing focuses on evaluating the application's external behavior without looking into the internal code, which helps testers verify if the application meets user requirements and works as expected. On the other hand, white-box testing takes a closer look at the internal code structure, allowing testers to identify the root causes of issues within the system. These methodologies are beneficial for large-scale web applications that involve multiple functionalities and interfaces, as it enhances the detection and resolution of potential errors across various components. The term validation is often referred to as black-box testing whereas verification is referred to as verification in software development *(Nidhra & Dondeti, 2012)*.

The expansion of web applications in recent years has increased the need for robust Software Quality Assurance (SQA) practices to ensure reliability, and security. As web applications expand, they often integrate various technologies like JavaScript, PHP, and HTML and need to perform seamlessly across multiple platforms and browsers. This complexity makes effective SQA techniques

essential, enabling early detection and resolution of issues within the development process, which significantly enhances application performance and user experience. SQA for web applications typically involves multiple stages of testing, including unit testing, integration testing, and system testing. These stages provide a structured approach to analyze both the software's internal functionality and external behavior, supporting a comprehensive validation process.

In a research *(Suryaningrat, 2019)*, the importance of identifying redundant and not frequent components within software, as these can contribute to inefficiencies and increase the chances of bugs within the application. Their findings suggest that ongoing code-level analysis, combined with user-focused testing methods, plays an important role in maintaining the integrity and performance of web applications. By applying this technique, developers can improve the stability and functionality of web applications, significantly improving the user experience which contributes to software's sustainability. This approach not only minimizes the potential errors but also contributes to creating more robust and reliable software *(Suryaningrat, 2019)*.

In a further study *(Di Lucca & Fasolino, 2006)*, a key challenge in Software Quality Assurance (SQA) for web applications i.e, the limited availability of standardized testing tools that comprehensively address essential quality issues, such as security, interoperability, and usability. While some tools primarily focus on cross-platform compatibility, others target functional issues like identifying broken links or performing load-testing. However, these tools often lack in providing comprehensive structural testing methods required to verify that the software's development aligns with its intended design and functionality. This limitation underscores the need for more advanced SQA models built specifically for web applications, ensuring they meet both functional requirements and high standards for performance and security. Developing such models could significantly enhance the overall quality and reliability of web applications *(Di Lucca & Fasolino, 2006)*.

3.2 Software Testing Techniques for Web Applications

Software testing techniques are vital for evaluating the reliability, security, and functionality of web applications. Effective testing techniques ensure that the software application works properly, meet its goals it and assess the quality of software. By applying testing methodologies, developers can identify and resolve issues early, which significantly enhances the application's stability and enhances the user experience (Jain, 2024). Key testing approaches for web

applications include manual and automated testing (using tools), provide a comprehensive view of software quality. This dual approach allows for a closer understanding of the application's strengths and weaknesses at early stages and ensure that the software works as expected (Jain, 2024). In next sub-sections, further details about well know testing techniques including blackbox, whitebox, greybox and their areas of application is explained.

3.2.1 Blackbox Testing Techniques

Blackbox testing is a technique of software quality assurance that is done by testing the external structure of the software without knowledge of internal code. This is usually named as functional testing as it is based on requirement specifications. The blackbox is used for it because in this technique the software acts as a blackbox in which don't know what is inside it but we can see only the external behavior and it delivers the desired output *(Nidhra & Dondeti, 2012)*. The system can only be interacted by user-interface blackbox testing.

There different types of testing involved in blackbox testing which are as follows:

Functional Testing: Software features are testing based on functional test cases focusing on requirements and design specifications of the system under testing *(Nidhra & Dondeti, 2012)*.

Non-functional Testing: The aspects of the software which are not related to functions of the software come under non-functional testing. It includes performance, usability, security etc *(Jain, 2024)*.

Regression Testing: This testing checks whether the new changes or updates do not affect the existing behavior or functionality of the software and the changes do not introduce new errors to the previously tested system *(Duggal & Suri, 2008)*.

Compatibility Testing: This testing approach determines how well the software performs in different environments and versions *(Yoon et al., 2008)*.

The pictorial representation of blackbox testing is shown in Figure 2 below which depicts software as a blackbox, with only the input and output exposed to the user.

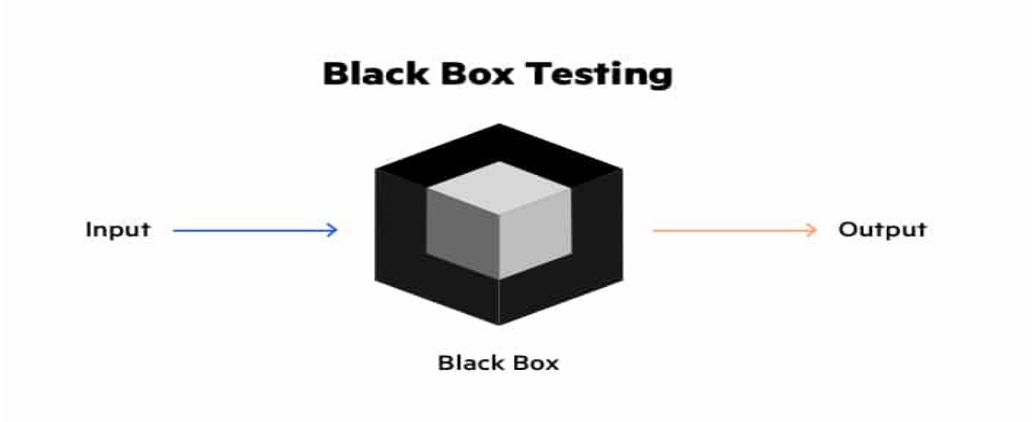


Figure 2. Blackbox Testing

3.2.2 Whitebox Testing Techniques

Whitebox testing in software development is a technique in which the internal structure of the software i.e, code, design architecture etc is tested. This technique is also known as glassbox or clearbox testing technique. The flow of data and logic behind any function is taken into account for performing whitebox testing. This is also known as structural testing. *(Nidhra & Dondeti, 2012)*

Whitebox testing checks whether each path in the code is correctly functioning and all edge cases are handled. The output produced by the software is thoroughly validated. Security issues are usually considered in whitebox testing, such as penetration testing, which requires knowledge of internal structure of the code. Whitebox testing can be considered as more time taking due to complexity of the code and usually this is performed by the developer. Unit and integration testing are main types of whitebox testing performed during the development phase of software (*White Box Testing - Software Engineering*, 2018).

In the Figure 3 below, whitebox testing represents that the internal code including logical and conditional statements are exposed to the tester. Whitebox testing is performed by developers and testers.

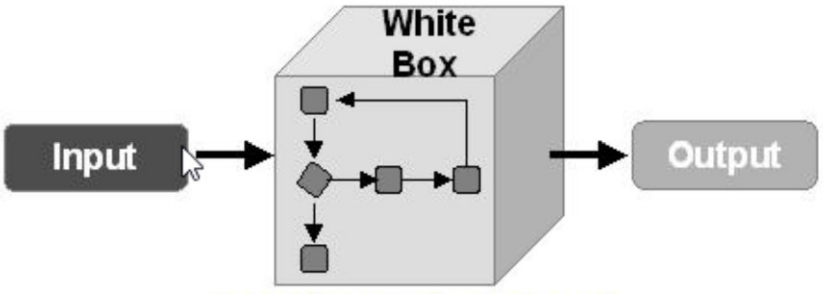


Figure 3. Whitebox Testing

As shown above, whitebox is basically done by the developers or quality assurance tester who have complete knowledge of the tech-stack used and write the automated test cases to check the internal logic, loops, conditions and integration of other services with the web application.

3.2.3 Greybox Testing Techniques

Greybox testing is a technique is software quality assurance which is a combination of whitebox and blackbox testing. The internal data structures and algorithms are accessed for designing test cases but testing is done at the user or blackbox level *(Hussain & Singh, 2015)*. This is good approach for testing web application as test cases can be more detailed considering external behavior as well as internal structure of code (*Difference between Black Box and White and Grey Box Testing*, 2021). In the Figure 4., it is shown that greybox testing takes attributes from whitebox and blackbox which results into hybrid testing approach.

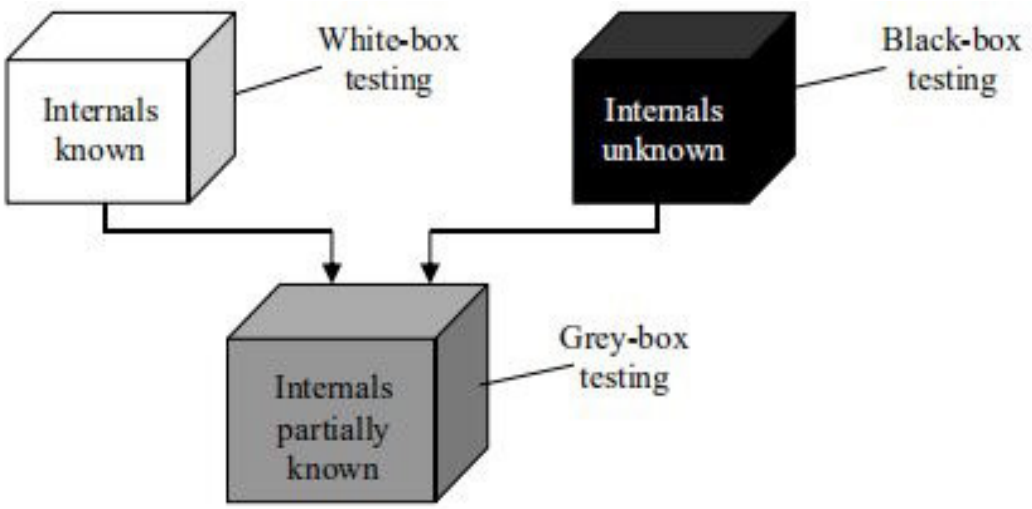


Figure 4. Greybox Testing

Greybox testing increases the test coverage as it allows us to focus on all layers of software application by combination of whitebox and blackbox testing techniques. In greybox, testing is done by user's point of view rather than designer's points of view as show in the figure 5. *(Khan, Mohd. E. & Khan, F. , 2012).*

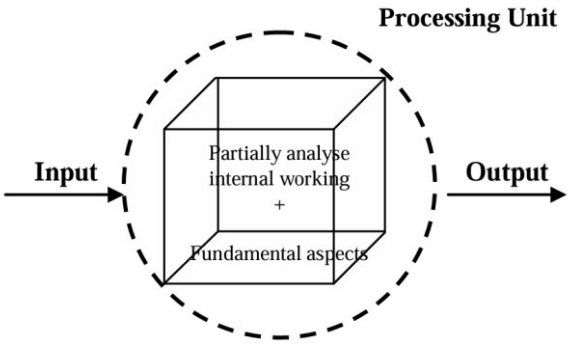


Figure 5. Represents Greybox Testing (IJACSA, Vol.3, 2012)

This approach helps identify issues that may go unnoticed in traditional testing methods, such as security vulnerabilities and integration errors. By using Greybox testing, developers can ensure that web applications are both secure and functionally robust, enhancing overall user experience and reliability. Table 2 below presents a comparative study of whitebox, greybox, and blackbox testing, providing brief information about selecting a testing technique based on software requirements.

Comparison of Testing Techniques

No.	Whitebox	Greybox	Blackbox
1.	Complete internal information needed	Partial internal information needed	No internal information needed
2.	Suitable for testing code path and internal logic	Suitable for comprehensive functional testing	Suitable for functional or business testing
3.	Performed by developers and testers	Conducted by end users (user acceptance testing), developer and tester	Performed by end users, testers, developers

Table 2 Difference between Whitebox, Greybox and Blackbox Testing

The comparative study of black-box, grey-box, and white-box testing techniques highlights the advantages and applications of these techniques in software quality assurance. Black-box testing focuses solely on the software's external

behaviors, making it useful for functional testing without needing access to source code, though it may lack depth in coverage. White-box testing, in contrast, requires full access to the code and allows for thorough validation of the internal logic, ideal for detecting code-level issues and achieving higher test coverage, but it can be time-consuming and costly. Grey-box testing strikes a balance, combining limited knowledge of internal structures with an external testing approach. This method is particularly effective for complex applications, as it can identify both functional issues and structural flaws without needing full code access, enhancing overall test coverage efficiently. The study demonstrates that using a combination of these techniques can maximize detection of vulnerabilities and provide a more robust quality assurance process for diverse software systems. (Khan & Khan, 2012)

3.3 Existing Approaches in Software Testing for Web Applications

A further study on Software Fault Prediction Models for Web Applications *(L. T. Giang et al., n.d.)* introduces an approach to Software Quality Assurance (SQA) specifically tailored for web applications. This approach utilizes a set of unique metrics designed to find the different features of web applications, such as complex user interfaces and extensive hyperlink structures. The selected metrics include JavaScript(LOJ) and CSS-related metrics(LOCSS), hyperlink analysis(NOH), and code size(LOC). These metrics are important for identifying common fault types in web applications, such as broken links and user interface inconsistencies*(L. T. Giang et al., n.d.)*.

Table 3 represents the web metrics selected by for prediction model.

Metric	Type	Description
LOJ	User Interface	The number of lines of JavaScript defined in the source code of a module
NOFJ	User Interface	The number of functions of JavaScript defined in the source code of a module
LOCSS	User Interface	The number of lines of CSS defined in the source code of a module
NOIC	User Interface	The number of IDs and classes of CSS defined in the source code of a module
NOH	Hyperlink	The number of hyperlinks defined in the source code of a module
LOC	Size Metric	The total number of lines of code in the source code files of a module

Table 3 List of software metrics used as fault predictors in web applications

The study applies various classification techniques to find faulty modules, including logistic regression and machine learning models like neural networks, random forests, and Naïve Bayes(Sunghun Kim, 2012). They found these techniques to be very effective for fault prediction specific to web applications. For instance, metrics related to hyperlinks and user interface elements (e.g., JavaScript functions and CSS classes) were useful for indicating a significant correlation with the likelihood of faults in web applications. Moreover, machine learning models, particularly neural networks and random forests, further improve prediction accuracy, demonstrating the value of machine learning in identifying and categorizing areas within web applications that have high chances of bugs.

This approach allows developers to allocate testing resources efficiently, focusing on areas most likely to contain errors. This proactive fault detection enhances software stability, reduces costs, and improves user experience which are main factors for development of complex and user-focused web applications. This research shows that a specialized quality assurance approach is more beneficial as compared to general fault prediction models but this depends on the components of web application.

Lakshmi and Mallika conducted a review of web application testing, considering the importance of rigorous testing due to the increasing number of web applications. The authors discuss the unique challenges of web application testing, such as the need for high compatibility across various platforms, frequent release cycles, and the various production environments for web applications. They found that the major issues in software quality assurance include achieving cost-efficient and comprehensive testing to ensure the reliability, usability, and security of web applications.*(D. Rajya Lakshmi & S. Suguna Mallika, 2017)*

To address these challenges, the paper evaluates a range of testing methodologies and tools, including automated, coverage-based, and structural testing, as well as techniques like mutation testing and combinatorial interaction testing. It also reviews various frameworks and models that support web-specific testing requirements. They highlight the lack of comprehensive tools for non-functional testing such as usability and compatibility testing—and suggest a need for further research to develop metrics and frameworks that enhance the overall quality of web applications, particularly due to development of cloud-based and service-oriented architectures. This review offers valuable insights into the current trends and gaps in web application testing, suggesting development of robust and adaptable testing frameworks.

3.4 Web Application Vulnerability Testing for Security Assurance

Bau et al. examine the capabilities of automated black-box web application vulnerability scanners in detecting security flaws, focusing on commonly exploited vulnerabilities like Cross-Site Scripting[1] (XSS) and SQL Injection [2](SQLI). They assessed eight commercial scanners by testing them against known vulnerabilities in older versions of popular web applications (Drupal, PHP, and WordPress), as well as a custom application designed to mimic real-world security issues. (Jason Bau et al., 2009)

The findings show that while these black-box scanners are generally effective at identifying common vulnerabilities, such as basic XSS and SQLI, they struggle with more complex vulnerabilities, like second-order SQL injection, and active content vulnerabilities embedded in technologies such as Flash and Java applets. Additionally, the tools demonstrated low detection rates for vulnerabilities like Cross-Site Request Forgery (CSRF) and advanced XSS, revealing gaps in their coverage.

This research highlights the need for developing more sophisticated detection capabilities within these tools. Enhancing their ability to manage stored data states and improving the monitoring of complex interactions could strengthen detection rates for stored and advanced vulnerabilities. The need for continuous improvement in automated vulnerability scanning technologies is a must to keep pace with the growing security needs of modern web applications.

3.5 Software Quality Metrics in Web Applications

Lee researched how software quality metrics can improve Software Quality Assurance (SQA) in different stages of the software lifecycle. By analyzing major quality models i.e, McCall's, Boehm's, and the ISO/IEC 25010 standard, he identifies essential software quality attributes, including reliability, usability, efficiency, and maintainability. It proposes a method for linking these quality attributes to specific metrics, allowing for systematic measurement and

[1] OWASP. (n.d.). Cross-site scripting (XSS). Available at: https://owasp.org/www-community/attacks/xss/ (Accessed: 6 September 2024)

[2] OWASP. (n.d.). SQL injection. Available at: https://owasp.org/www-community/attacks/SQL_Injection (Accessed: 6 September 2024).

assessment during development. This method enables the numeric measurement of software quality by aligning quality attributes with measurable criteria, making evaluations more objective. (Ming-Chang Lee, 2014)

The McCall[3] quality model focuses on three areas i.e, product transition, revision, and operations. It provides quality factors like maintainability and portability with criteria to assess adaptability and operational characteristics.

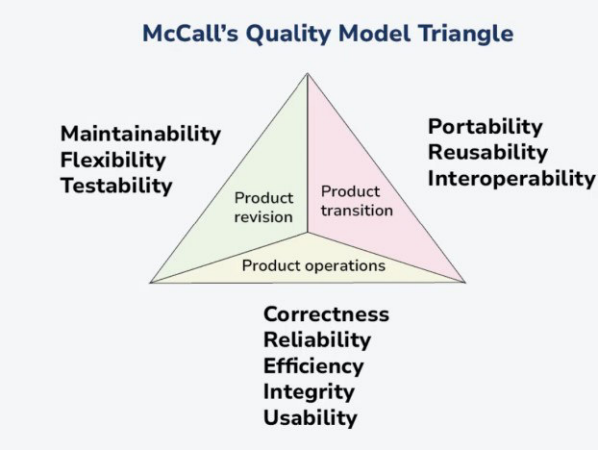


Figure 6. McCall's Software Quality Model Triangle

Boehm's[4] model evaluates quality qualitatively, classifying utility, maintainability, and portability, with seven main factors including reliability and flexibility whereas Dromey's[5] model is structured to evaluate quality across software lifecycle stages (requirements, design, and implementation) and emphasizes correctness and contextual qualities.

Each model prioritizes different software quality aspects, with McCall and Boehm emphasizing operational characteristics, and Dromey covering

[3] GeeksforGeeks. (n.d.). McCall's quality model. Available at: https://www.geeksforgeeks.org/mccalls-quality-model/ (Accessed: 6 November 2024).

[4] ProfessionalQA. (n.d.). Boehm software quality model. Available at: https://www.professionalqa.com/boehm-software-quality-model (Accessed: 6 November 2024).

[5] Airccse. (2013). *A Study of Software Quality Models*. Available at: https://airccse.org/journal/ijsea/papers/4113ijsea01.pdf (Accessed: 6 November 2024).

development phases comprehensively. Table 4 illustrates the software quality factors and their derived metrics.

Quality Factors	**Metrics**
Reliability	Accuracy, Error Tolerance
Usability	Operability, Training
Efficiency	Execution Efficiency
Maintainability	Modularity, Simplicity
Flexibility	Expandability, Modularity
Testability	Simplicity
Portability	Software Independence
Reusability	Modularity (Component approach)
Integrity	Access Control, Auditability
Interoperability	Common Data

Table 4 Software Quality factors and derived metrics

These findings suggest that comprehensive quality models, when used with targeted metrics, significantly enhance the SQA process. This approach provides measurable insights into product quality, offering a structured way to identify areas for improvement and monitor quality assurance activities. It emphasizes that well-defined metrics not only enhance product stability and maintainability but also result in more consistent software quality assurance outcomes in web applications.

Chapter 4 Research Methodologies and Design Approach

This chapter explains the research methodologies that will be used to develop a quality assurance model for OER conversion tool (convOERter). By having good knowledge from the literature review and quality factors of web applications, research gaps are identified which will serve as a basis for developing a software quality assurance model for convOERter. The main focus of research methodologies will be to maintain high standard of quality, availability and functionality of web applications.

4.1 Metrics-Based Evaluation

Metrics-based evaluation plays an important role in measuring software quality, particularly in web applications such as convOERter, where performance, usability, and reliability etc are quantifiable indicators. Software quality assurance metrics contains different quantitative and qualitative measures which are used to assess the software quality. Early detection of bugs and taking corrective measures are only possible by continuous monitoring and measurement of QA metrics (Kasowaki & Eymen, 2023).

The Goal-Question-Metric (GQM)model, used in software quality assurance, provides a hierarchal structured approach, starting with goal which is redefined into several questions which result into measurable metrics (Caldiera et al., n.d.).

For example, to achieve high usability, metrics like task completion rate and user satisfaction can be used to address specific questions about user interaction and satisfaction. Similarly, reliability metrics such as mean time to failure (MTTF) can help analyze the software stability in production environment. Taking into consideration these types of goals will ensure convOERter quality and meet educators' requirements.

4.2 User-Centric Testing

These testing techniques focus on user experience metrics, such as ease of use of a software, task efficiency, and accessibility, to determine how interactive and functional the tool is from the user's perspective. For educators who have different technical skills and accessibility needs, ensuring that convOERter is

easy to navigate and effective in its functionality is very important for adoption at a large scale and their satisfaction.

A research study on design and implementation of an evaluation system for OER conversion tool (convOERter) (Shetty, 2022) has explained usability data is often collected by running usability tests on a website, where actual user interactions and feedback are collected to see how well a software works. Typically, a usability expert selects people from the target audience, takes them to a usability lab, and asks them to complete a series of realistic tasks. The usability analyzer watches for any issues or difficulties the users face and then conducts surveys or interviews to gain more insights into their experience. Common usability testing methods include cognitive walkthroughs, interviews, and think-aloud sessions, where users describe their thought process as they interact with the product. Observing users in their usual environment while they perform everyday tasks also adds valuable context to the usability testing process.

However, usability testing has some drawbacks. It can be time-consuming, especially with larger groups, as scheduling participants, observing them, and analyzing data requires significant effort. This approach often results in mostly qualitative data, representing only a small number of users, which means certain usability problems may go undetected. Additionally, the costs associated with getting participants, arranging test space, conducting the tests, and analyzing the results make usability testing a relatively expensive process (Shetty, 2022).

Usability testing is one of the primary methods used to measure how effectively users can complete tasks within the application, identify any challenges they face, and gather feedback on overall satisfaction. Metrics such as task completion time, error rate, and user satisfaction scores provide valuable insights into how educators interact with the tool and whether it aligns with their needs. As this tool is mostly used for converting non-OER images into OER images, so most of the usability factors for users are related to image to be converted and replacement image suggestions.

Accessibility testing is also important to ensure that the user can easily navigate through the web application and most common functions of the software are easily accessible to the user. Moreover, the explanations for the steps should be easy to understand and this will impact user retention. For the accessibility testing, feedback form as well as automated tools are used in this research.

4.3 Analysis of Existing Tools and Techniques

A number of tools and techniques are available for testing the quality and security of web applications, each with its pros and cons. Whitebox testing is a commonly used approach that requires analyzing the internal code and structure of the application. This method allows developers to identify specific issues at the code level, giving detailed insights into functionality and security. However, whitebox testing can be time-consuming, as it needs to look into the code. For web applications like convOERter, where the goal is to attract educators who may not have technical expertise, relying completely on whitebox testing may not be practical. However, Blackbox and Greybox testing provide more accessible options by evaluating the tool's functionality from the user's perspective without understanding of the code. This approach allows more users to participate in testing, gathering diverse feedback on the tool usability and functionality.

For vulnerability testing, several tools exist that work primarily with whitebox testing methods. Tools such as SonarQube[6], and Fortify Static Code Analyzer[7] are popular for identifying security vulnerabilities, code flaws, and adherence to coding standards. However, these tools typically operate on a whitebox basis by having full access to the code. This dependency on internal code knowledge can be a barrier for convOERter's broader testing and user adoption since it restricts testing participation to developers or individuals with technical skills. Additionally, whitebox vulnerability tools cannot be used to get user experience issues, which are vital for a tool aiming at educators. By adopting Blackbox or Greybox techniques alongside usability and functionality testing, convOERter can gather more comprehensive insights that reflect real-time usage, ensuring that the tool is both secure and accessible to its targeted users.

4.4 Research Gaps and Objectives

After reviewing existing software quality assurance (SQA) tools and techniques, various research gaps have been identified, highlighting the need for a tailored approach to evaluate and enhance the usability, functionality, and security of convOERter. While traditional whitebox testing techniques and vulnerability analysis tools provide deep insights into code structure and security, they require

[6] SonarSource. (n.d.). SonarQube. Available at: https://www.sonarsource.com/products/sonarqube/ (Accessed: 7 November 2024).

[7] OpenText. (n.d.). Fortify Static Code Analyzer. Available at: https://www.opentext.com/en-gb/products/fortify-static-code-analyzer (Accessed: 6 November 2024).

access to internal code and technical expertise, making them less suitable for tools like convOERter. Additionally, while some testing techniques focus particularly on functionality or security, a mixed approach that combines usability, functionality, and security metrics is required to meet the diverse needs of convOERter users.

To address these challenges, this research proposes a multi-type approach combining both automated testing tools and blackbox testing techniques. Automated tools such as Lighthouse[8], AXE[9], and WebPageTest[10] will be used to provide objective metrics on convOERter performance, accessibility, and usability. These tools do not require the access to code and are ideal for evaluating how the tool performs from an end-user perspective, focusing on user experience and accessibility issues.

Additionally, ZAP[11] (Zed Attack Proxy) will be used as a blackbox vulnerability scanner to assess convOERter security. This scanner will help identify potential vulnerabilities without requiring internal code access, thus enhancing security.

For functionality testing, blackbox testing techniques such as use case testing, cause-effect analysis, and boundary value analysis will be applied. Although blackbox testing includes techniques such as equivalence class testing, these are not applicable to our web application but are useful for other software with a wide variety of user inputs. These techniques will allow analyzing convOERter behavior under different scenarios, helping to identify functional limitations and ensuring the tool's robustness across a variety of use cases. By using these methods, we can evaluate how well convOERter performs in scenarios that educators are likely to encounter, thus aligning the testing approach closely with user needs.

Finally, the Goal-Question-Metric[12] (GQM) model will be used to establish specific quality metrics tailored to convOERter objectives. GQM will help

[8] Google Developers. (n.d.). Lighthouse overview. Available at: https://developer.chrome.com/docs/lighthouse/overview (Accessed: 6 November 2024).

[9] Deque Systems. (n.d.). Axe accessibility testing. Available at: https://www.deque.com/axe/ (Accessed: 6 November 2024).

[10] WebPageTest. (n.d.). WebPageTest. Available at: https://www.webpagetest.org/ (Accessed: 6 November 2024).

[11] OWASP. (n.d.). ZAP (Zed Attack Proxy). Available at: https://www.zaproxy.org/ (Accessed: 6 November 2024).

[12] GeeksforGeeks. (n.d.). Goal Question Metric (GQM) approach in software quality. Available at: https://www.geeksforgeeks.org/goal-question-metric-approach-in-software-quality

define clear goals, develop questions that align with these goals, and identify metrics to measure progress. Metrics derived from the GQM model will focus on usability, functionality, reliability, and security, providing a structured framework for assessing and improving the tool.

This combination of automated tools, blackbox testing techniques, and GQM-based metrics forms the basis for the implementation strategies in the following chapter. This approach will allow for a comprehensive assessment of convOERter, enabling targeted improvements that make the tool more accessible, reliable, and user-friendly for educators.

The pictorial representation of the tailored software quality model for convOERter is shown in Figure 7 below. It comprises four base areas: blackbox testing, software quality metrics, performance and accessibility, and security testing.

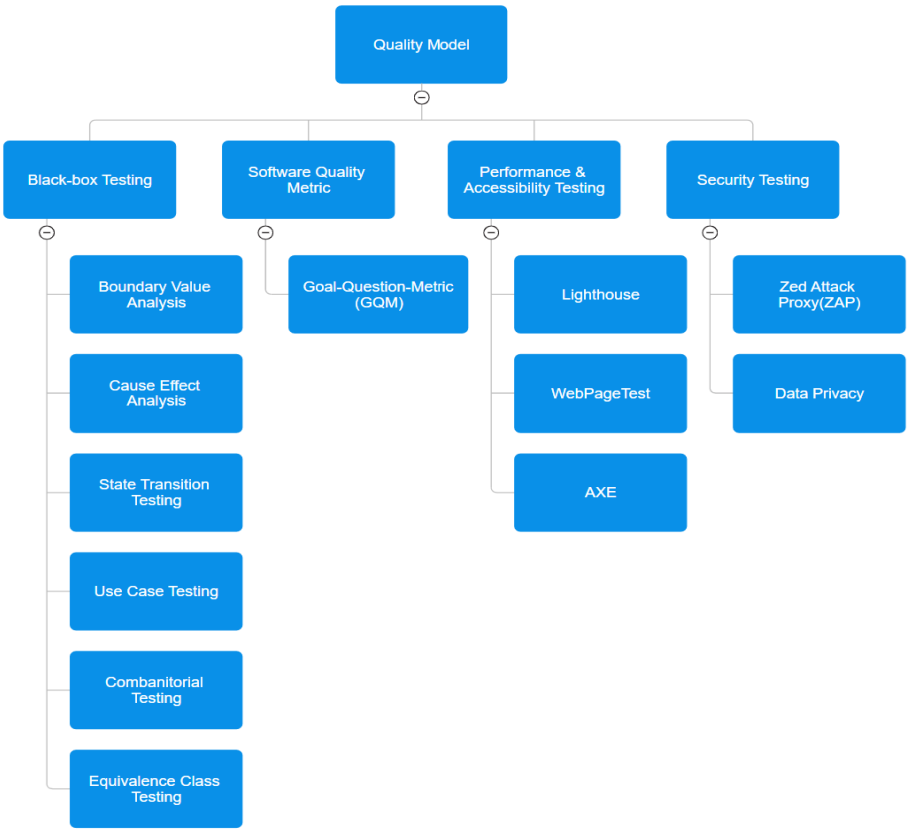


Figure 7. Quality Assurance Model

(Accessed: 7 November 2024)

Chapter 5 Implementation of Testing Techniques

This chapter focuses on the implementation of testing techniques to convOERter for the evaluation and enhancement of the quality. Further, Goal-Question Metric technique and automated tools to test performance and accessibility of web application will be applied. Afterwards the vulnerability test scanner named ZAP will be applied to security testing.

5.1 Blackbox Testing Techniques

In this section, the application of blackbox testing techniques will be explained. By applying blackbox testing techniques, we aim to assess the application's functionality, usability and reliability from end user perspective. This section explains the techniques including Boundary Value Analysis, Cause-Effect Analysis, Usecase Testing, Equivalence class testing, State Transition Testing and Combinatorial Test Technique which will provide valuable insights about the tool and how it performs under different conditions and user inputs.

5.1.1 Boundary Value Analysis

A software has an input domain which is further divided into sub-domains which result in similar output. Testing the software on the boundaries of the sub-domains is known as Boundary Value Analysis *(Dobslaw et al., 2023)*. Boundary Value Analysis[13] (BVA) is a software testing technique that evaluates the application's behavior at the extreme input boundaries. It focuses on testing values at the edges or just beyond the limits to identify potential errors, vulnerabilities, or unexpected behaviors in the system. BVA[14] is one of the forms of functional testing, It can be applied to input and output. Moreover, BVA can be used to test robustness. Test cases are designed in BVA without considering the functionality of unit under test.

Boundary Value Exploration (BVE) is an important technique in software testing aimed at enhancing the effectiveness of boundary value analysis (BVA) and boundary value testing (BVT). Traditional BVA and BVT focus on examining

[13] GeeksforGeeks. (n.d.). Software testing – Boundary value analysis. Available at: https://www.geeksforgeeks.org/software-testing-boundary-value-analysis/ (Accessed: 6 November 2024).

[14] Lichter, H. (n.d.). Boundary Value Analysis. Published at RWTH Aachen University for the course Software Construction. Available at: Google Drive file (Accessed: 6 November 2024).

boundaries within input spaces to identify areas where software behavior changes significantly, but they often lack clear, formal methods to define these boundaries, especially for complex input spaces. BVE addresses these challenges by introducing techniques that automate the detection of boundary values, thereby supporting both BVA and BVT processes. By applying BVE, testers can automate parts of boundary analysis, identify unexpected boundary behavior, and generate tests that strengthen software reliability.(Dobslaw et al., 2020)

The application of BVA to convOERter is analyzed as follows:

Test case 1: Selection of file for conversion

BVA ensures testing with supported file formats, such as .pptx, .docx, .odp, and .odt. It verifies that the application does not process invalid file formats and works only with valid ones.

Different file formats are tested as an input by altering file names and concatenating the correct file format in name. The application responded as expected which means that this test case is passed.

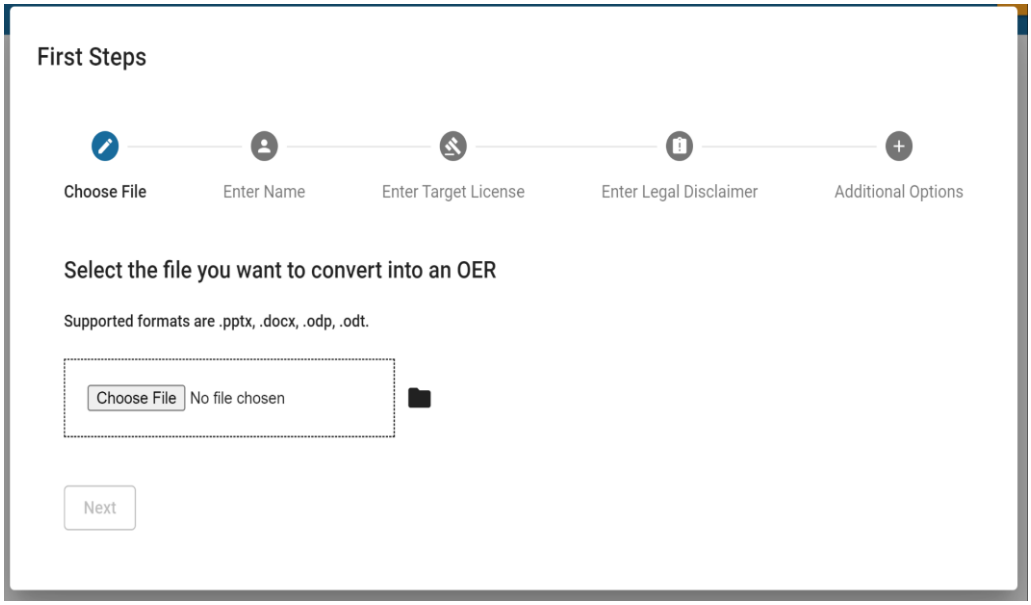


Figure 8. File input step convOERter

As the supported file formats are clearly mentioned above the 'Choose File' button, processing files other than those specified is not possible with

convOERter, ensuring the expected response. This ensures that convOERter handles file selection reliably, preventing users from intentionally or unintentionally uploading incompatible formats. By confirming this behavior, BVA demonstrates that the application effectively guides users in choosing appropriate file types, reinforcing a smooth and error-free user experience.

Test case 2: Enter name field

This test case applies Boundary Value Analysis (BVA) to verify that the tool behaves correctly when users enter various formats of names, including those with numeric characters. The main objective here is to ensure that the name field is not left blank and that it only accepts properly formatted names that meet the tool's requirements.

This test case also checks that users cannot proceed to the next step without filling in this mandatory field. Since the entered name will be displayed in the legal disclaimer and included in the final OER file, it is important for accuracy and consistency. Testing involved trying to leave the name field empty and using only digits to see if the tool restricts access to the next step. The tool responded as expected, blocking access when the field was left blank and accepting names with digits where necessary. This behavior indicates that convOERter enforces proper input for the name field, ensuring that important details appear correctly in both the disclaimer and the final OER document.

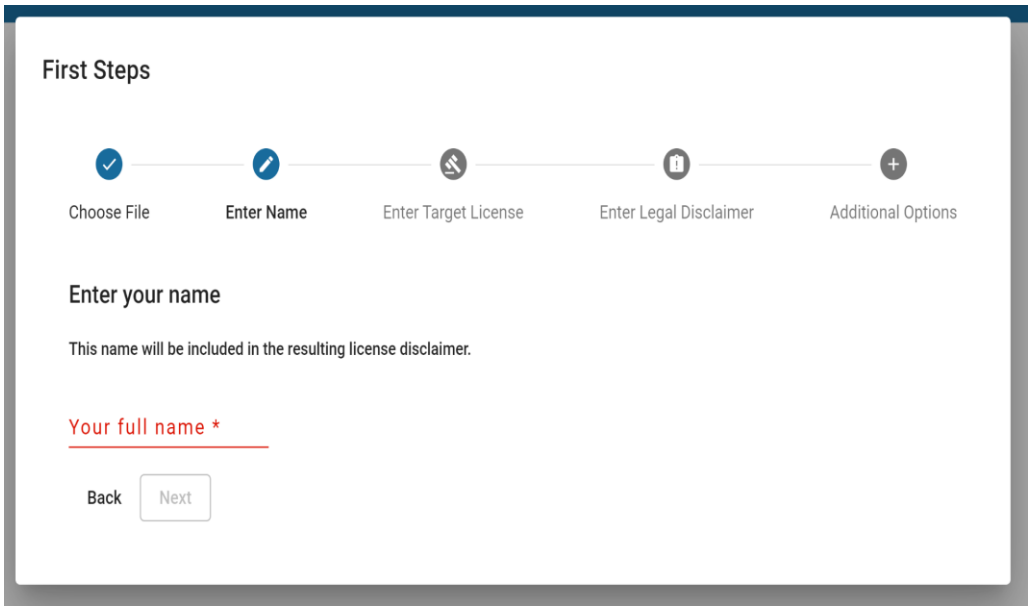


Figure 9. Enter name step convOERter

In Figure 9, for the second step of convOERter, it is seen that the 'Next' button is disabled. This means the user cannot proceed to the next step without entering the name of the file creator, which is a requirement of OER to include user details. In convOERter, this information is added at the end of the document as 'Cropped by' or 'Updated by.' Therefore, it is necessary to input the name during this step.

Test case 3: Select of License from dropdown

This test case validates the "Select License" dropdown menu in convOERter, using BVA to ensure that users can only proceed after selecting a valid license from the available options. The dropdown provides a range of Creative Commons licenses (CC-0 1.0, CC-BY 4.0, CC-BY-SA 4.0), which are necessary for defining the permissions and terms associated with the generated OER file.

To test this functionality, the tool was evaluated by attempting to bypass the license selection, as well as by selecting each of the available license options. BVA checks that all edge cases within the dropdown are properly managed, confirming that each option is selectable and that no unsupported values can be entered and this can be seen in Figure 10 below. The tool responded correctly, only allowing moving to the next step once a valid license was selected and preventing any attempts to proceed without making a selection.

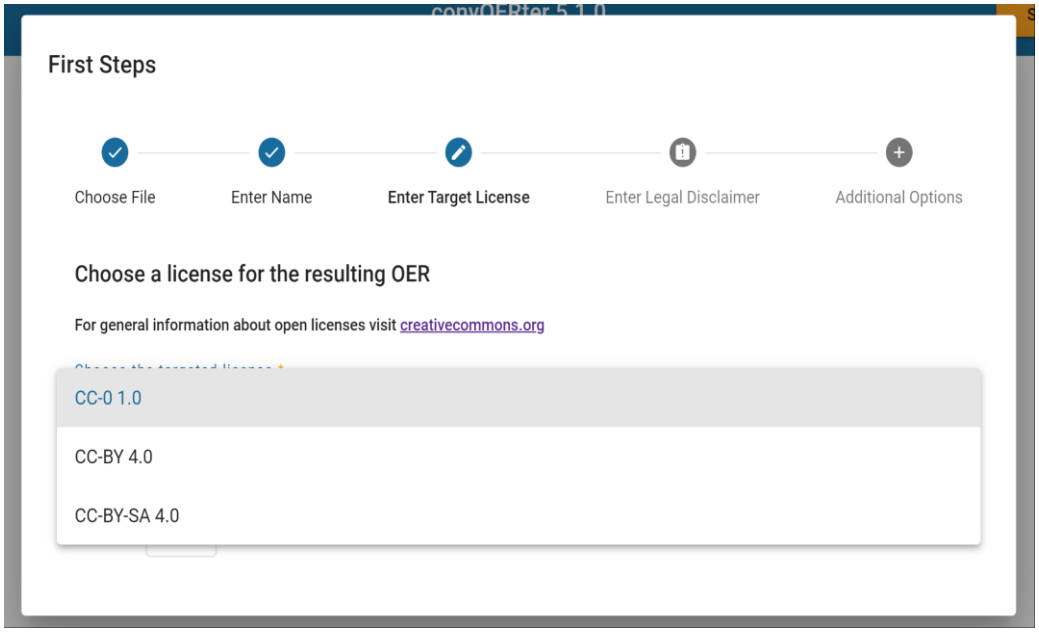


Figure 10. Enter name step convOERter

This ensures that users cannot overlook this step, reinforcing the integrity of the licensing process in the OER creation workflow. This validation confirms that convOERter is developed to handle different licensing scenarios accurately, helping educators adhere to OER standards by clearly defining the terms under which their content can be shared.

Test case 4: Editing the Generated Disclaimer

This test case focuses on verifying convOERter's functionality in allowing users to edit the generated disclaimer, ensuring that the tool correctly handles disclaimers of varying lengths and formats. By BVA, this test evaluates how the tool responds when users input disclaimers of minimum and maximum lengths, as well as various formats, to confirm that the tool can accommodate a range of user needs.

During testing, disclaimers of different lengths were entered, including extremely short and long text, to observe if any content was truncated or displayed incorrectly. Additionally, the disclaimer was checked to ensure that all edits were accurately saved and displayed in the final processed OER file. The test confirmed that the tool correctly handled the full disclaimer text into the final output, regardless of length or format, ensuring that the legal information remains intact and accurately reflected in the completed document. Figure 11 shows the editable disclaimer field below.

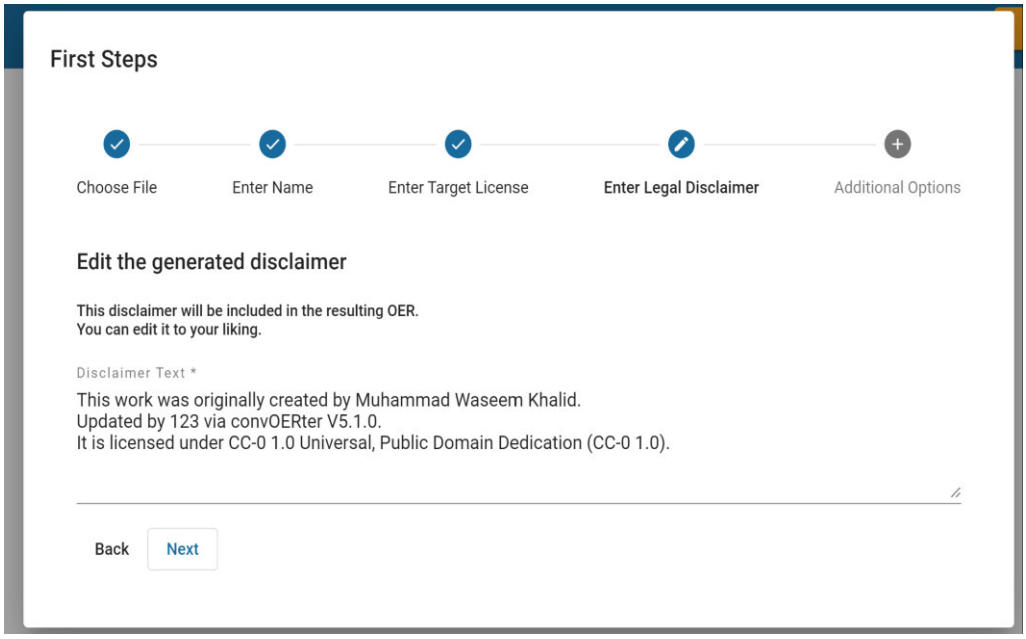


Figure 11. Enter Legal Disclaimer step convOERter

This testing process confirms that convOERter is designed to support flexible editing of disclaimers, allowing educators to tailor legal statements according to their needs while maintaining the integrity of the content. By handling extreme cases, the tool demonstrates its reliability in managing disclaimer content effectively, providing a user-centered approach to customization without compromising on accuracy.

Test case 5: Selection of Embedded Images for processing

This test case checks the functionality of the image selection checkboxes within convOERter, ensuring that users can easily select or deselect images for replacement or editing. Using Boundary Value Analysis (BVA), this test examines how the tool responds when users interact with the checkboxes under various conditions, such as selecting all images, selecting none, and making partial selections. Figure 12 illustrates all the extracted images from the input document and the selections of images.

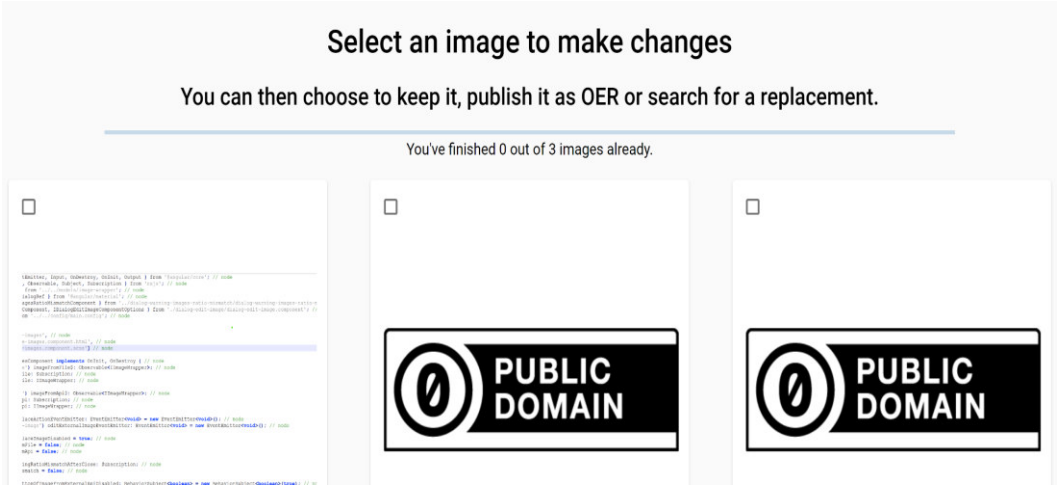


Figure 12. Image Selection for change

This test confirms that the tool processes each selection and deselection responds as expected to all possible interactions. By validating checkbox functionality, this test ensures that users have full control over image selection, enhancing the tool's usability and allowing educators to efficiently customize the OER content according to their preferences.

Test case 6: Manual uploading of replacement image

This test case focuses on verifying convOERter's handling of metadata input fields required when a user manually uploads a replacement image. Using BVA, we assess

how the tool manages varying inputs in each metadata field, ensuring that the system correctly processes essential details for Open Educational Resources (OER) compliance. These fields include the author's name, author's site, work title, author's affiliation, selection of license, origin URL, and origin name. Each of these fields plays a critical role in documenting the source and license of the replacement image, maintaining the integrity of the OER. The attributes for the manual replacement of the image is shown in Figure 13.

Figure 13. Manual Image Replacement

To validate functionality, tests were conducted with both minimum and maximum boundary inputs. For example, very short and excessively long names were entered in the "Author's Name" field, and URLs of varying lengths were tested in the "Origin URL" field to confirm that convOERter accepts only valid data within expected limits. BVA also ensures that all fields are correctly populated before proceeding, as the user cannot bypass this step without completing the required fields, particularly the "Select License" dropdown.

This functionality ensures that educators can confidently use replacement images with proper documentation, fulfilling the legal and ethical requirements of OER creation.

5.1.2 Cause Effect Analysis

Cause-Effect Analysis is one of the important blackbox software quality techniques. It is a systematic approach to identify and understand the relationships between different factors and their outcomes in a system. It helps uncover root causes, predict problems, and formulate targeted solutions for improving software reliability and performance. It uses logical relations to describe relationships between events labeled as causes and events labeled as effects. It does not require the knowledge of source code *(Krupalija et al., 2022).*

Cause-Effect Analysis[15] determines systematic combinations of inputs which needs requirement specifications. Cause-Effect Graphing (CEG) is used to determine incompleteness and contradictions in applications' behavior. CEG is graphical representation of inputs(causes) with their associated output(effects) which is used to design test cases .

Cause Effect analysis can be applied to convOERter by designing the test cases and then applying these test causes to the application which helps in finding the cause of error and effect that will incur in result of such error.

Test Case 1: Addressing Inconsistent Image Recognition

Cause: Inconsistent image recognition within convOERter.

Inconsistent image recognition can occur due to different reason like variations in image quality, file formats, and the presence of visual elements in the input file. When the system fails to accurately identify images, it may incorrectly process or skip certain images, leading to incorrect output.

Effect: Incorrect replacements and potential licensing issues.

The primary effect of inconsistent image recognition is that convOERter may replace images incorrectly or fail to replace with OER-compliant image. This can lead to mismatches between the required and actual output and impacting the quality of the final OER document. Additionally, if non-OER-compliant images are not properly identified and replaced, it can result in licensing violations, putting the user at risk of legal issues or copyright issues.

[15] Lichter, H. (n.d.). Cause Effect Analysis. Published at RWTH Aachen University for the course Software Construction. Available at: Google Drive file (Accessed: 6 November 2024).

Test Case 2: Improving User Interface for Image Selection

Cause: Lack of a user-friendly interface for image selection.

When the image selection interface is not interactive or easy to navigate, users may face difficulty to understand how to choose or replace images effectively within the tool.

Effect: Potential errors in decision-making.

This confusion can lead to mistakes, such as selecting incorrect images for replacement or overlooking essential images, ultimately impacting the quality and accuracy of the final OER document.

Test Case 3: Difficulty in Image Recognition and Replacement

Cause: Inadequate search algorithms for replacement images.

If the search algorithms are not optimized, users may struggle to find appropriate OER-compliant images, leading to inefficient workflows and difficulty in convOERter adoption.

Effect: Difficulty in finding suitable OER-compliant images.

This can result in users facing problem with replacements or spending excessive time searching, which lowers the overall user experience and may affect the educational quality of the final resource. Figure 14 shows the alternative images suggested by the tool.

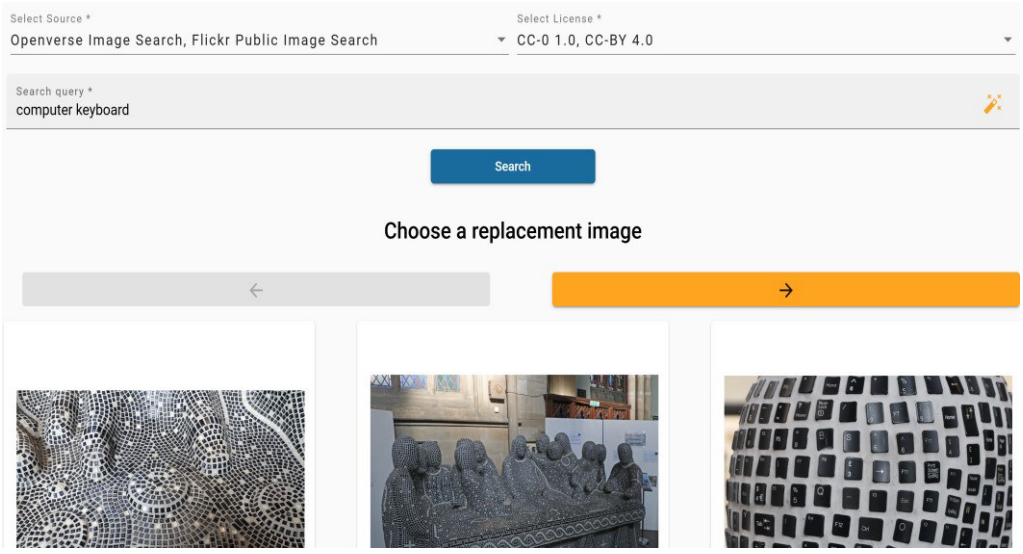


Figure 14. Replacement Image Selection

In the figure above, it can be seen that the tool has correctly detected the embedded image and the search query (i.e., computer keyboard) is accurate. However, the suggested replacement is incorrect, which affects user satisfaction and requires manual intervention to change the query or navigate to the next pages to find a relevant replacement image.

Resolution: Enhance search algorithms or platforms (e.g., Openverse[16], Flickr[17]), potentially by integrating machine learning for better image matching.

Improving these algorithms can help users locate relevant, high-quality OER-compliant images more quickly and accurately, thereby streamlining the replacement process and supporting better outcomes for OER content.

Test Case 4: Improving User Guidance on License Selection

Cause: Insufficient user guidance on License[18] Selection.

In license selection, users may find it challenging to understand the differences between licenses and how each option affects the usage rights of their content.

Effect: Users may incorrectly select inappropriate licenses.

This can lead to potential licensing conflicts, where users unintentionally assign rights that either restrict or do not protect their content, impacting legal compliance.

Resolution: Implement clear instructions and tooltips to guide users in selecting the correct license.

Adding concise instructions and contextual tooltips near the license selection dropdown will help users make informed choices. This guidance ensures that users select the license that best suits their content's intended use, minimizing the risk of errors.

Although, the link for selected license is showed after selection but navigating the user to another page can be time consuming so it will be more convenient if short description can be displayed on selection dropdown.

---

[16] Openverse. (n.d.). Openverse. Available at: https://openverse.org/ (Accessed: 6 November 2024).

[17] Flickr. (n.d.). Flickr. Available at: https://www.flickr.com/ (Accessed: 6 November 2024)

[18] Creative Commons. (n.d.). Share your work: CC licenses. Available at: https://creativecommons.org/share-your-work/cclicenses/ (Accessed: 6 November 2024)

Test Case 5: Mitigating Ambiguity in Disclaimer Editing

Cause: Ambiguity in disclaimer editing.

Without clear guidance or structure, users may be uncertain about how to edit the disclaimer effectively, leading to ambiguity about what information is required or how it should be presented.

Effect: Users may unintentionally generate unclear disclaimers.

This can result in disclaimers that are vague or incomplete, potentially causing misunderstandings about the legal terms of the OER content and creating risks for both creators and users.

Resolution: Provide predefined disclaimer templates and a user-friendly editor to ensure clarity and compliance.

Offering built-in templates and an editor will help users create clear, comprehensive disclaimers. The built-in template should have some fixed text that will comply with OER disclaimer guidelines and rest can be changed by the user. This will make it easier for users to include all necessary information, supporting legal compliance and minimizing usage unclear terms.

Test Case 6: Enhancing Transparency in Local Processing

Cause: Lack of transparency in local processing.

When users are unsure about how their data is processed, they may cause low adoption of tool and they think about where their documents are stored and whether their privacy is protected.

Effect: User concerns regarding data security and privacy.

This uncertainty can lead to privacy issues, with users hesitating to fully engage with the tool due to fears that their content may be exposed or shared without consent.

Resolution: Clearly communicate the tool's local processing approach, assuring users that their documents are not sent to external servers.

By providing clear, visible information about the local processing of files, the tool can assure users about data security. This transparency builds trust, ensuring users feel comfortable that their documents remain private and secure.

The cause-effect analysis provides a structured framework for identifying specific issues within convOERter, examining their root causes, and

understanding their impacts. Each test case addresses a particular challenge, clarifying how targeted resolution can improve user experience, functionality, and security. This approach highlights the value of systematically analyzing cause-effect testing to make improvements. Such an analysis can be similarly applied to other web applications, offering a practical method to enhance software quality by addressing core issues directly.

5.1.3 State Transition Testing

State Transition Testing is one of the useful techniques in blackbox testing which will make sure that software behaves as expected while going through different states of a software. It is useful when testing state machines and navigation of graphical user interface. It consists of states, transition, events and actions *(Ehmer Khan, 2011)*. There are few steps involved in the state transaction testing which can be defined as followed:

I. Identifies finite set of states that a software possess.
II. Software is tested by passing it from one state to another using different input combinations.

This technique can be used in number of software whenever some user actions or system input cause the transition next state of the software. A simple example of this is a traffic signal system in which the signal turns green after passing certain time set like 15 seconds or it operates on some other input like sensors whenever it detects some traffic waiting on the signal it turns on depending on certain set criteria. Similarly, state transition testing can be applied to web applications, as they typically process user interactions, making it easier to track and test. It is usually represented by state transition diagram which is graphical representation of software states and the events that cause transition. (*State Transition Testing Techniques in Software Testing*, n.d.)

According to Lichter, State transition testing[19] "The system under test should offer state reporters i.e, services to access state information" The effort needed for this testing depends upon the size of the application or state machine.

The state transition diagram of convoerter in Figure 15 shows that the flow of states that the application undergoes from start to end during the conversion process. The process begins with the File upload state where the user can select a file (Choose File) and review the images (ImageOverview). The ConversionProcessInit state leads to important decision point, ImageReplacementDecision, where users choose to replace images. After processing, the tool transitions to LocalProcessingCompleted and ProcessingImagesDone, leading to ConversionSuccess. In the UserDecisionState, users can decide to save the document or upload again, which ultimately brings the system to the ConversionCompleted state, marking

[19] Lichter, H. (n.d.). State transition testing. Published at RWTH Aachen University for the course Software Construction. Available at: Google Drive file (Accessed: 6 November 2024).

the end of the process. This structured flow ensures that each required action is executed in a sequence for successful completion of task.

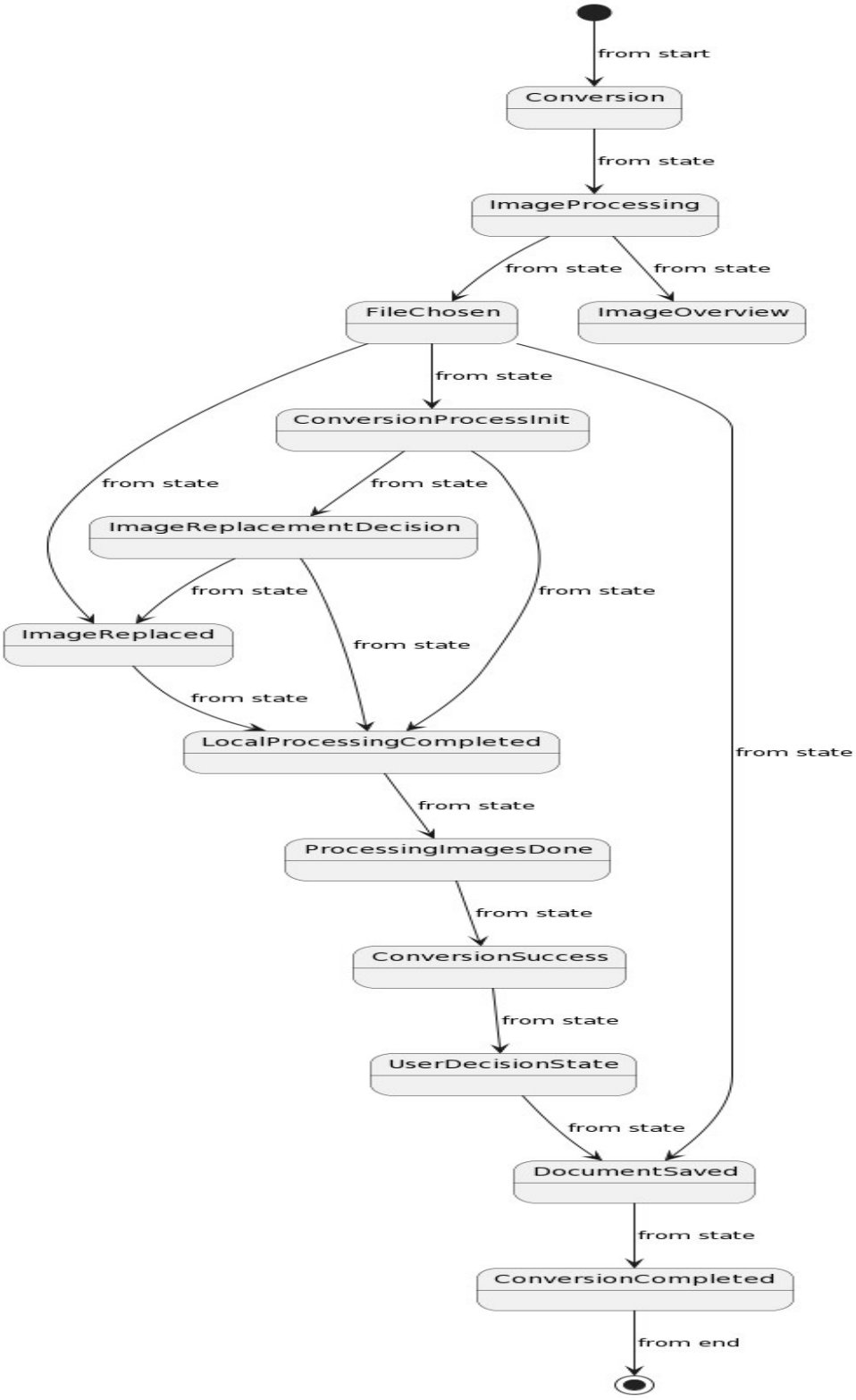


Figure 15. State Transition Diagram convOERter

5.1.3.1 State Transition Testing for Conversion Process

Initiating Conversion Process

Initial State: This is the starting state for conversion process.

Transition: Triggered when the user selects a file to convert.

Next State: The system enters the conversion process initiation state, preparing for further actions.

Image Overview

State: The user views images within the selected document to decide which ones require processing.

Transition: Triggered when the user selects an image to modify.

Next State: The user can choose to replace, mark, or exclude the selected image from further processing.

Decision-Making for Image Replacement

State: The user evaluates options for replacing the selected image.

Transition: Activated when the user decides to replace an image.

Next State: The system facilitates a search for OER-compliant images, enabling the user to find suitable replacements.

Local Processing

State: The document is modified locally based on user-defined selections and image replacements.

Transition: Triggered when all selected images have been processed.

Next State: The system saves the modified document, including license.

Auto Recognition or Manual Selection

State: The tool can either perform automatic image recognition or allow for manual selection, based on user preference.

Transition: Initiated when the user chooses to enable or disable auto recognition.

Next State: If manual selection is chosen, the user can upload images to be used as replacement as well metadata for the image will be save as well.

User Decision After Image Recognition

State: The user reviews recognized images and decides whether to keep, publish, or search for alternative replacements.

Transition: Triggered by the user's choice regarding each image.

Next State: The process continues based on the user's choice, either moving to the next image or completing the process.

These states and transitions are developed to align with convOERter's core functionalities and user workflow. Using the State Transition Testing Technique comprehensive testing can be performed for convOERter's behavior as it progresses through each state, ensuring reliability and accuracy throughout the OER conversion process. This technique helps validate that each possible user action leads to the expected system response, maintaining smooth user experience

5.1.4 Use Case Testing

Use Case(UC) Testing is a black-box testing technique that focuses on validating the system's functionality from an end-user perspective(*Software Testing - Use Case Testing*, 2019). There are some important features of use case testing which are given below:

1. UC testing depends on the user actions and system's response to those actions
2. UC testing is user oriented rather than system oriented.
3. UC testing describes the functional requirements of the system.
4. This helps in identification of errors during the design process.

In the context of the convOERter tool, use case testing focuses on identifying and validating different scenarios that represent typical interactions between the

user and the system. By testing these scenarios, we can ensure that convOERter's functionalities meet user expectations and work as intended under various conditions. Based on the tool's primary functions, several critical use cases can be defined to ensure convOERter's effectiveness in creating Open Educational Resources (OER).

Below in Figure 16, an overview of the user interactions with system is shown in use case diagram which can be further explored by diving deeper into each use case and documenting all the steps involved to pass that use case.

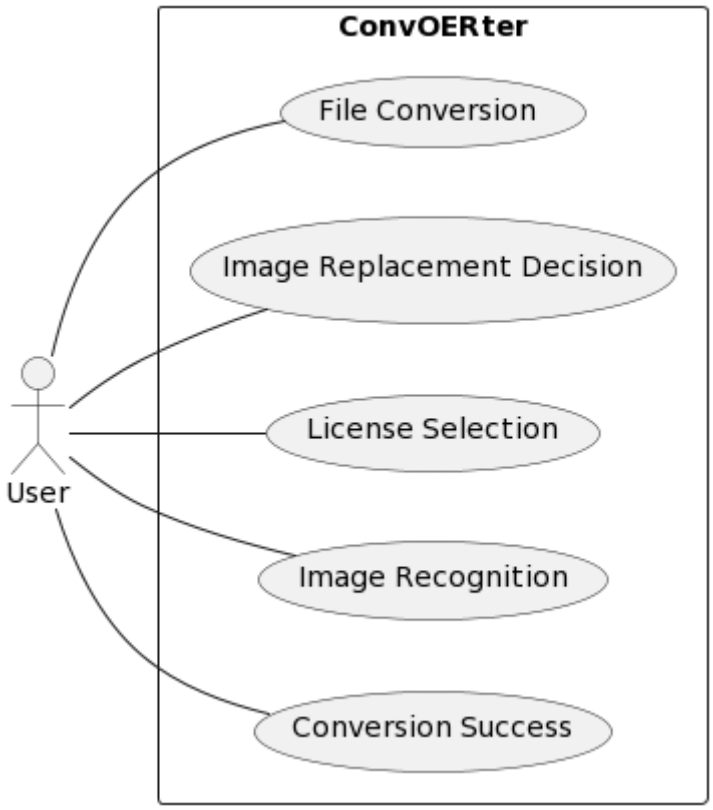


Figure 16. Use Case Diagram convOERter

Use Case 1: File Conversion

Description: Verify that the tool successfully converts files into OER-compliant documents as shown in the Figure 17 below.

1. User selects a file and initiates the conversion process.
2. Ensure the tool detects and lists all embedded images.
3. User decides on image replacements and other license-related options.
4. Confirm the local processing of the document and save file.

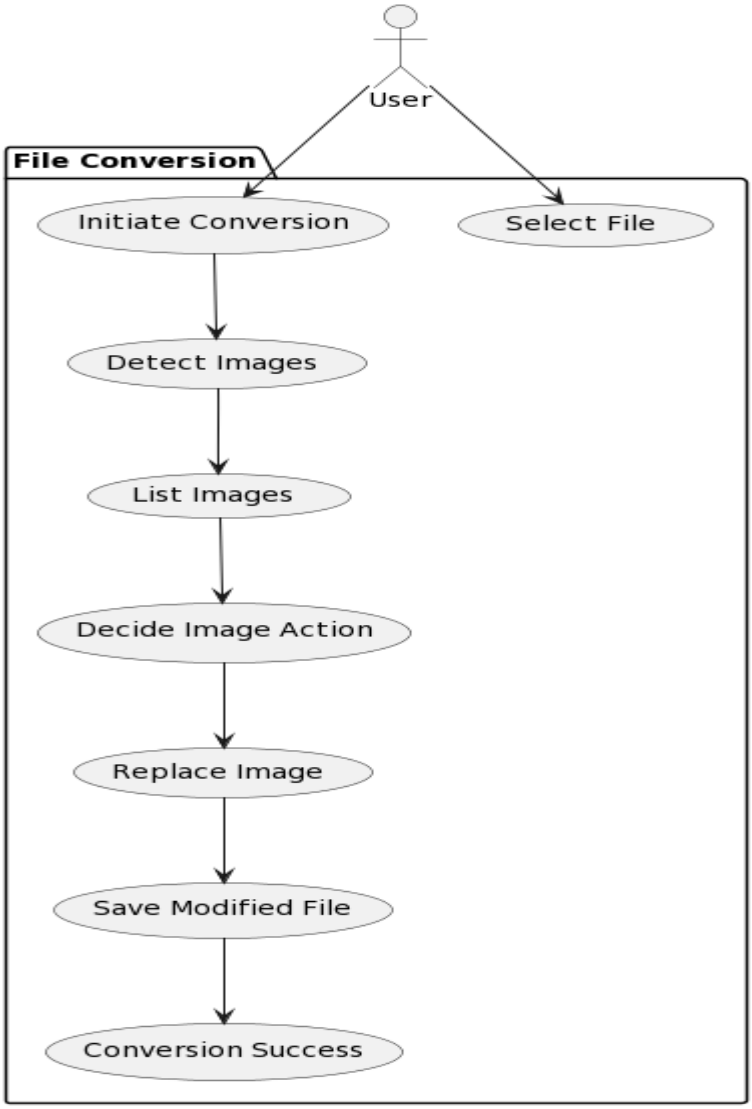


Figure 17. File Conversion Use Case Diagram

Use Case 2: Image Replacement Decision

Description: Validate the decision-making process for image replacements as shown in Figure 18 below.

1. User reviews the listed images.
2. User decides to replace, mark, or exclude an image.
3. Confirm that the tool allows searching for suitable replacements in Creative Commons datasets.
4. Ensure the replacement process is carried out correctly.

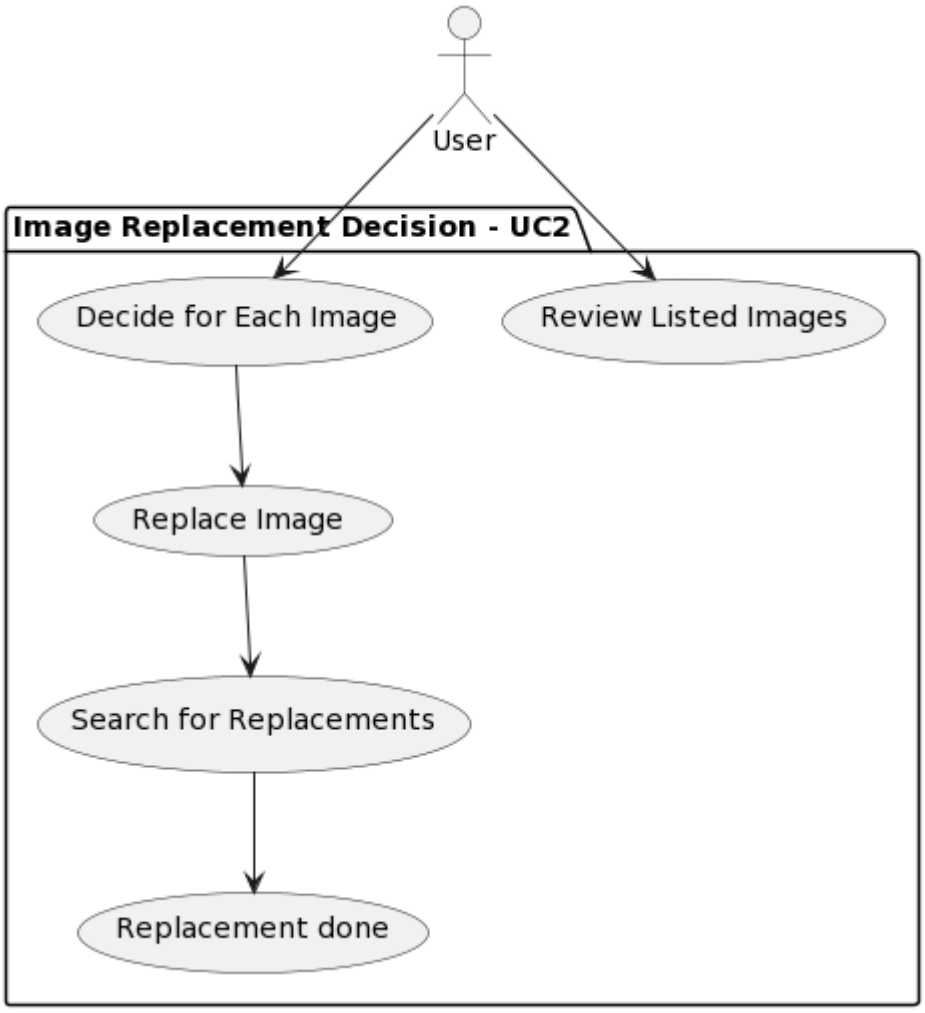


Figure 18. Image Replacement Use Case Diagram

Use Case 3: License Selection

Description: Ensure proper selection and handling of licenses as shown in Figure 19 below.

1. User enters their name and selects a target license from the dropdown.
2. User edits the generated disclaimer.
3. Confirm that the selected license is applied to the document.
4. Verify the accuracy of the disclaimer.

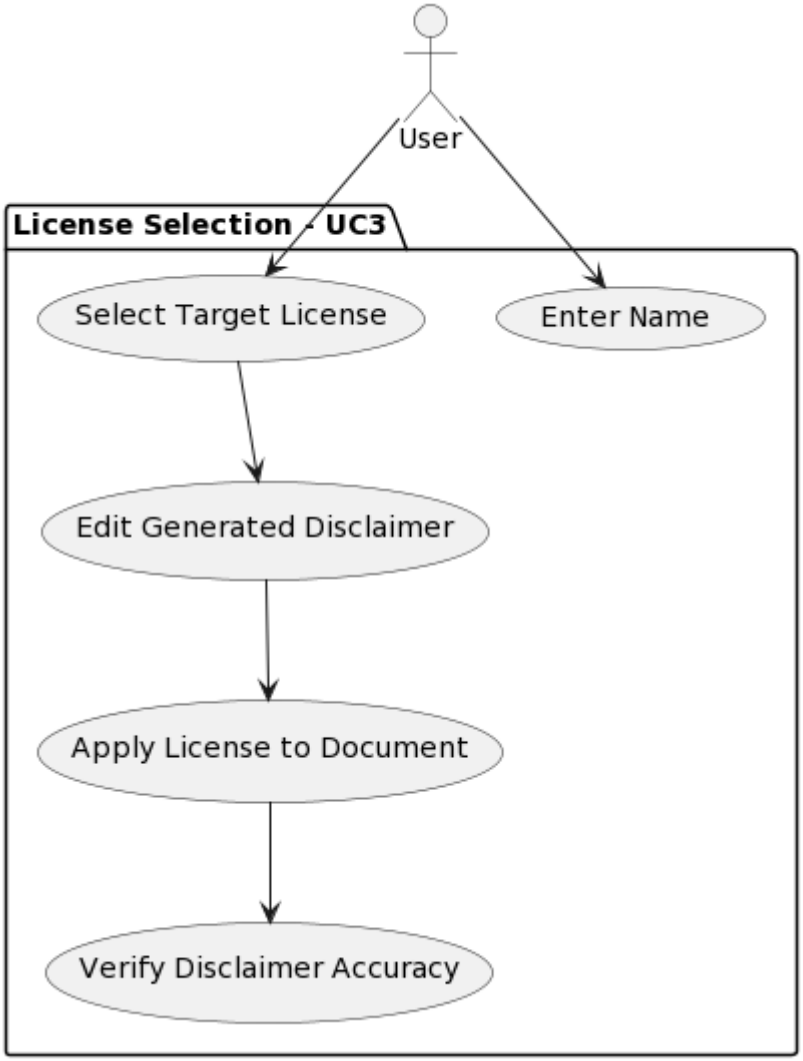


Figure 19. License Selection Use Case Diagram

Use Case 4: Image Recognition

Description: Validate the automatic recognition of images as illustrated in Figure 20 below.

1. User enables auto-recognition or manually selects an image.
2. Confirm that the tool correctly recognizes images and processes them accordingly.
3. Ensure the user's decision (keep, publish, search) is accurately reflected.

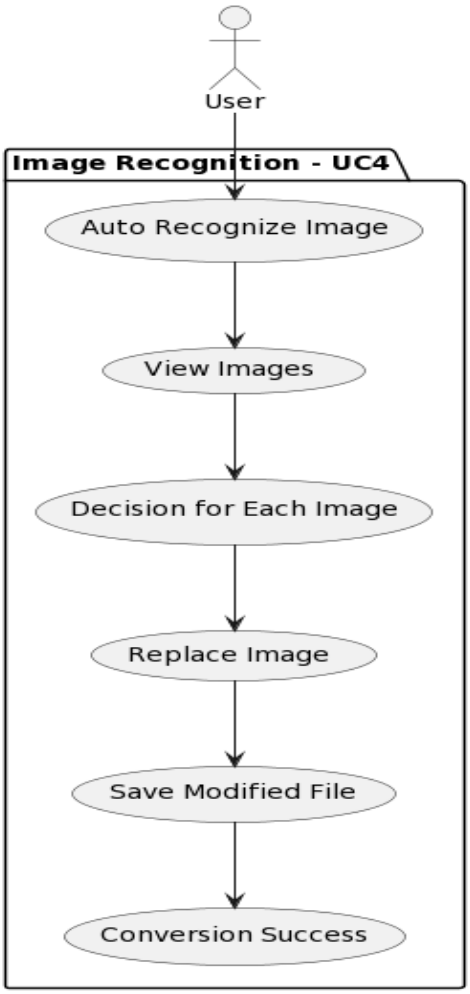


Figure 20. Image Recognition Use Case Diagram

Use Case 5: Conversion Success

Description: Verify the successful completion of the conversion process as shown in Figure 21 below.

1. Ensure that the modified document with all image sources and license notice is saved.
2. Confirm that the tool communicates the successful conversion to the user.
3. Validate that the document can be published as OER.

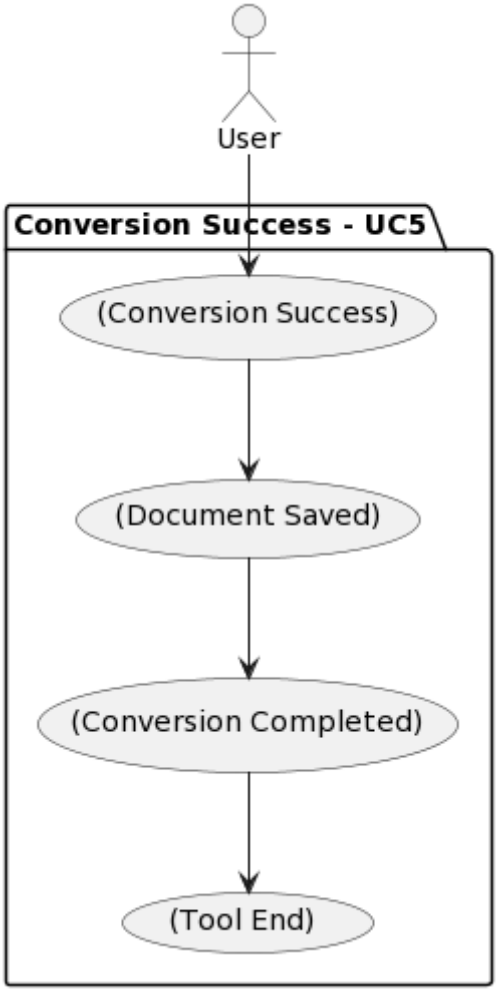


Figure 21. Conversion Success Use Case Diagram

Use case testing allows us to systematically assess convOERter's functionality, ensuring it meets specified requirements and user expectations. This approach ensures smooth operation across diverse user scenarios and alignment with its intended design and interactions. Usecase testing is found as a basis for number of testing techniques which include system testing, regression testing, acceptance testing, integration level testing, and requirements based testing which ensures that user's requirements are completely fulfilled *(Qazi et al., 2016)*.

5.1.5 Combinatorial Testing

Combinatorial testing is a testing approach in blackbox testing that utilizes various combinations of input parameters to perform testing for a software application. It ensures that software application produce same output for each combination of input *(Wickramasinghe, 2023)* . Combinatorial testing reduce the testing cost and increase the effectiveness of software testing because the inputs are split into combinations and one element from the each combination is testing rather than going through each input parameter *(Kuhn & Bryce, 2015)*.

Combinatorial testing cannot be applied to the convOERter tool due to its dynamic user interactions, diverse functionalities, and varied input dependencies.

Inapplicability to convOERter due to following reasons:

Dynamic Interactions:

Challenge: convOERter involves dynamic and context-dependent user interactions during document conversion and image replacement.

Impact: Combinatorial testing relies on predefined input combinations, making it less adaptable to the varied user inputs of convOERter. It performs well where we have numerical input or the data which can be divided into combinations depending on similarities.

Varied Functionalities:

Challenge: convOERter performs different operations, such as image recognition, license selection, and disclaimer editing.

Impact: The diverse functionalities of convOERter create a complex and interconnected feature space, making it challenging to combine all relevant possibilities within the structured constraints of combinatorial testing.

Varied Input Dependencies:

Challenge: Inputs in convOERter are dependable on file uploaded

Impact: Combinatorial testing assumes that inputs are predefined, whereas in our case, they depend on the images present in the uploaded file and whether the user go for automatic replacement or manual replacement.

5.1.6 Equivalence Class Testing

Equivalence Class Testing is a software testing technique that divides the input domain of a system into classes or groups of valid and invalid inputs, where each class is expected to exhibit similar behavior. Test cases are then designed to represent each class, ensuring comprehensive coverage of input scenarios (Ira*wan et al., 2018)*. Equivalence Class Testing[20] is more effective when used as a hybrid approach with boundary value testing.

[20] Lichter, H. (n.d.). Equivalence class testing. Published at RWTH Aachen University for the course Software Construction. Available at: Google Drive file (Accessed: 6 November 2024).

Inapplicability to convOERter due to following reasons:

Limited Input Diversity:

Challenge: convOERter has limited user inputs as compared to systems with diverse user interactions.

Impact: Equivalence class testing relies on distinct input classes; however, the nature of convOERter results in fewer distinguishable classes.

Homogeneous Inputs:

Challenge: convOERter primarily deals with homogeneous input types, like specific file formats, licenses, or textual content.

Impact: The uniformity of inputs reduces the effectiveness of equivalence class testing, as there might be fewer distinct classes within the constrained input space.

5.2 Quality Metrics Testing Techniques

To evaluate convOERter's quality effectively, I will implement the Goal-Question-Metric (GQM) approach, which links specific goals to measurable metrics. This technique provides a structured framework to assess functionality, usability, and reliability, ensuring the tool meets defined quality standards (*Goal Question Metric Approach in Software Quality*, 2020).

5.2.1 Goal Question Metric Technique

The GQM (Goal-Question-Metric) model is an approach used for defining and structuring measurement processes in software development as shown in Figure 22.

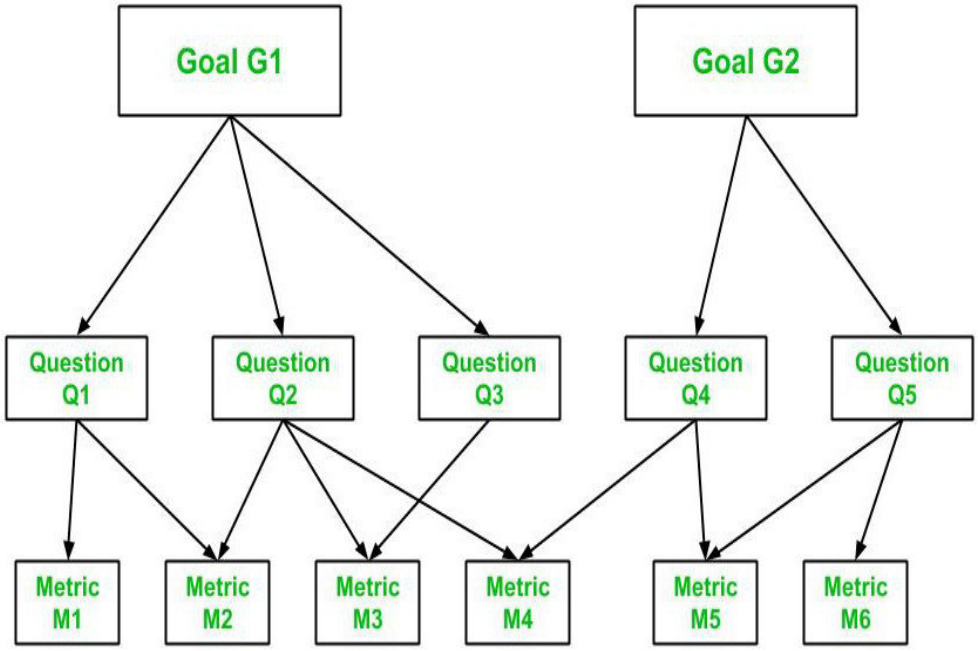


Figure 22. GQM Diagram taken from geeksforgeeks

To apply GQM to the convOERter, we can start by defining goals, formulating questions related to those goals, and then deriving metrics to measure the success of achieving those goals. Some of the questions are listed as follows:

Goal: Improve the Efficiency of OER Conversion Process

i. **Question:** What is the average time taken to convert a non-OER compliant resource to OER using the convOERter tool?
   Metric: Average Conversion Time (measured in minutes/seconds)

ii. **Question:** How many resources are successfully converted on a monthly basis?
   Metric: Monthly Conversion Rate (number of resources)

iii. **Question**: What percentage of converted resources pass OER compliance standards?

iv. **Metric:** Compliance Success Rate (percentage)

Goal: Enhance User Satisfaction and Accessibility

i. **Question:** How satisfied are users with the usability of the convOERter tool?
Metric: User Experience Rate (measurable through feedback **"**This tool was helpful**"**)

ii. **Question:** What is the level of satisfaction regarding the quality of the user interface of the convOERter tool?
Metric: User Interface Satisfaction Score (measurable through feedback **"**The user interface was good**"**)

iii. **Question:** What is the average number of support requests received per month?
Metric: Monthly Support Request Rate (measurable through feedback **"**Did you face any problem?**"**

iv. **Question:** What is the accessibility rating of the convOERter tool according to industry standards?
Metric: Accessibility Rating (using Auto. Accessibility Testing Tools i.e, Axe, Lighthouse etc)

Goal: Ensure Data Security and Integrity

i. **Question:** How often are security audits conducted on the convOERter tool?
Metric: Frequency of Security Audits (number of time per year)

ii. **Question:** What is the percentage of successfully resolved security vulnerabilities in the tool?
Metric: Security Vulnerability Resolution Rate (percentage)

iii. **Question:** How many instances of data corruption or loss have occurred over the past year?
Metric: Data Integrity Incidents (number of incidents)

Goal: Assess User Adoption and Future Use

i. **Question:** Would users consider using the convOERter tool in the future?
Metric: Future Use Intent (measurable through feedback "Would you use this tool?")

ii. **Question:** What factors would influence users' decision to continue using the convOERter tool in the future?
Metric: Factors Influencing Future Use (measured through a qualitative analysis of user responses to open-ended questions about influencing factors)

iii. **Question:** How likely are users to recommend the convOERter tool to others for similar tasks?
Metric: Likelihood to Recommend (measured through a quantitative scale, e.g., on a scale of 1 to 5)

iv. **Question:** What features or improvements would encourage users to continue using the convOERter tool in the future?
Metric: Feature Improvement Suggestions

The hierarchical representation of Goal-Question-Metric is below in Figure 23 where first level contains a goal, second and third contain questions and sub-questions respectively whereas third level contains derived metrics to be measured.

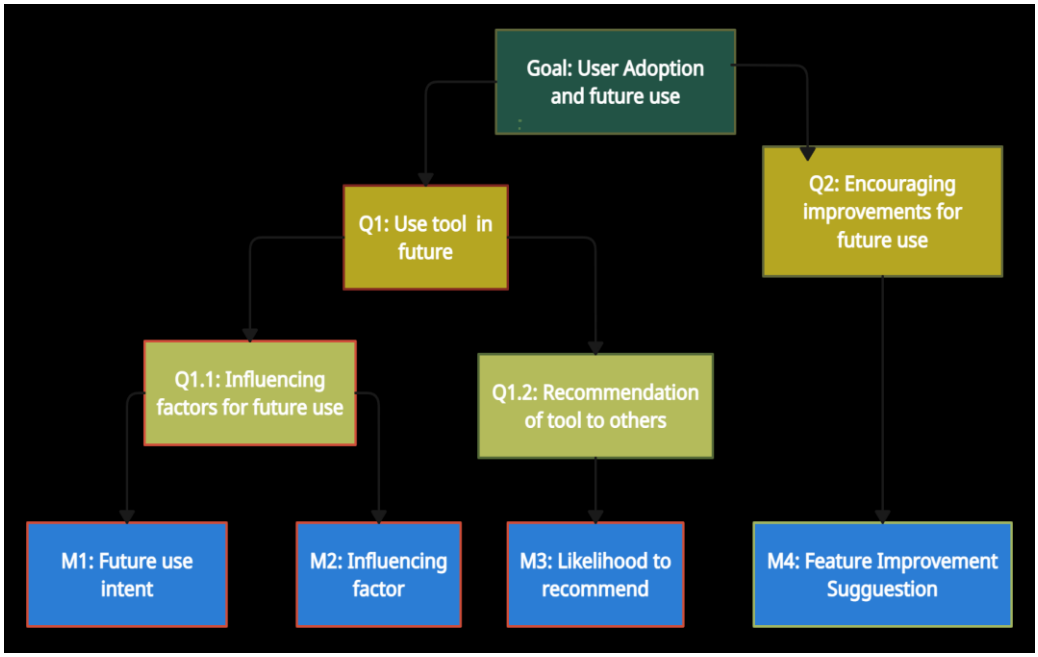


Figure 23. GQM Diagram for User Adoption and Future User

Goal: Identify and Address Major Problems in Tool Usage

i. **Question:** Have users encountered any major problems while using the convOERter tool?

   Metric: Incidence of Major Problems (measurable through feedback "Did you run into any major problems?")

ii. **Question:** What types of major problems were most frequently reported by users?

   Metric: Common Types of Major Problems (measured through a categorization and counting of reported issues)

Goal: Enhance Maintainability of the convOERter

i. **Question:** To what extent can components or modules of the convOERter be reused?

   Metric: Reusability[21]

   Calculation: *ReusabilityIndex = Total number of components / Number of reusable components*

ii. **Question:** To what extent can the convOERter be easily modified or extended to accommodate changes or new features?

   Metric: Modifiability

   Calculation: *Modification Cost = (Total development time / Time spent on modifications) ×100*

iii. **Question:** How well can the tool be tested for various scenarios, including edge cases?

   Metric: Testability

   Calculation: *Test Coverage Percentage = (Total number of components / Number of tested components) ×100*

The Goal Question Metric technique is found very useful for measuring the quality metrics of web application and it helped in deriving question for designing feedback survey where most of the metrics were measured on the basis of rating from 1 – 5 and feedback analysis was done using data collected from feedback from.

[21] Opsera. (n.d.). 13 code quality metrics that you must track. Available at: https://www.opsera.io/blog/13-code-quality-metrics-that-you-must-track#:~:text=Reusability%20metric%20measures%20whether%20the,number%20of%20interdependencies%20it%20has (Accessed: 6 November 2024).

5.3 Performance and Accessibility Testing using Automated Tools

For performance and accessibility testing of convOERter, automated tools such as Lighthouse, AXE, and WebPageTest were used. These tools evaluate the performance, responsiveness, and accessibility features, identifying any issues that might prevent users with special requirements from using the application effectively. Considering blackbox testing approach, these automated testing tools focuses purely on how convOERter performs externally, without accessing the internal code, ensuring that it meets accessibility standards from a user's perspective.

5.3.1 Lighthouse (Opensource automated tool by Google)

Lighthouse is a tool developed by Google to improve the quality of web pages on by analyzing the metrics set. The report generated after scanning convOERter which gives information about the where the app performed well and where it needs to be improved.

The Lighthouse report provides an analysis of convOERter's performance, accessibility[22] (practice for making webpage available to as many people as possible), best practices, and SEO which is a process to improve the ranting of a webpage on search engines like Google, Bing etc (Barbar & Ismail, 2019). Below is a breakdown of the key metrics, suggested improvements, and definitions to better understand each area:

5.3.1.1 Performance Metrics

The performance score for convOERter is relatively low, marked at 53 (shown in Figure 24), indicating that several aspects of the tool could be optimized for faster loading and smoother user experience. Key performance metrics identified include:

First Contentful Paint (FCP): FCP measures the time it takes for the first element (like text or an image) to load and become visible to the user. ConvOERter's FCP is currently at 7.4 seconds as depicted in Figure 24, suggesting a delay in

[22] MDN Web Docs. (n.d.). What is accessibility? Available at: https://developer.mozilla.org/en-US/docs/Learn_web_development/Core/Accessibility/What_is_accessibility (Accessed: 6 November 2024).

displaying initial content. Ideally, FCP should be below 2 seconds to provide users with a responsive experience. Improving FCP involves optimizing resources that block rendering, such as large images or unused JavaScript.

Largest Concertful Paint (LCP): LCP represents the time it takes for the main content of the page (often a large image or block of text) to load and appear on the screen. With an LCP of 7.8 seconds (shown in Figure 24), convOERter may feel slow to users, as they have to wait longer for all content to be displayed. A target LCP under 2.5 seconds[23] is considered as good. Strategies to improve LCP include reducing server response times, enabling text compression, and optimizing image size.

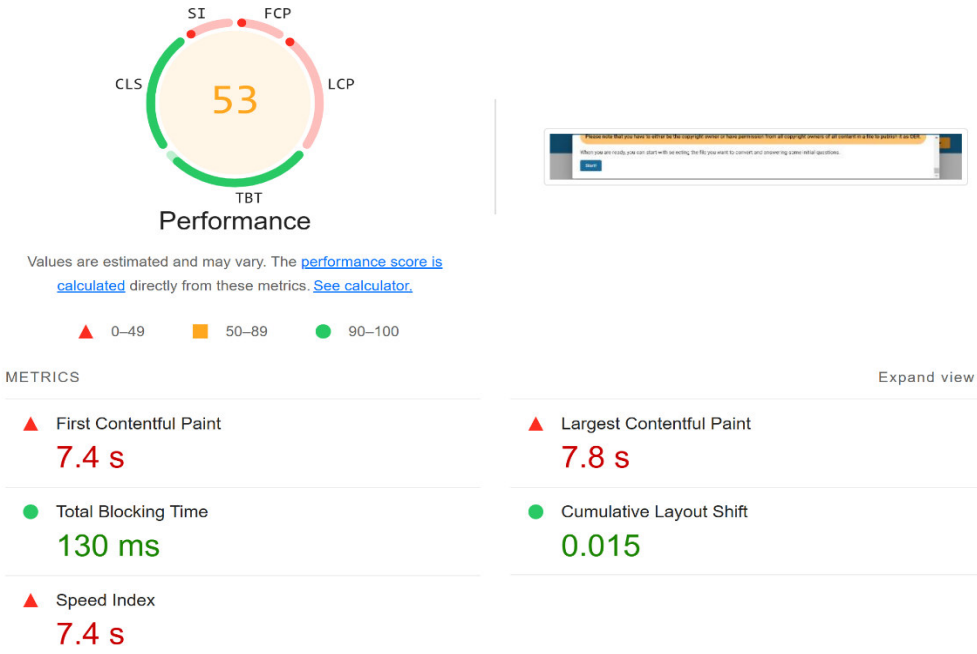


Figure 24. Lighthouse Performance Score

The Lighthouse calculates the performance score on the weightage set for metrics of web application which contribute the overall performance of an application. The score calculator along with respective weightage is shown in Figure 25 below:

[23] Google Developers. (n.d.). Largest Contentful Paint (LCP). Available at: https://web.dev/articles/lcp (Accessed: 6 November 2024).

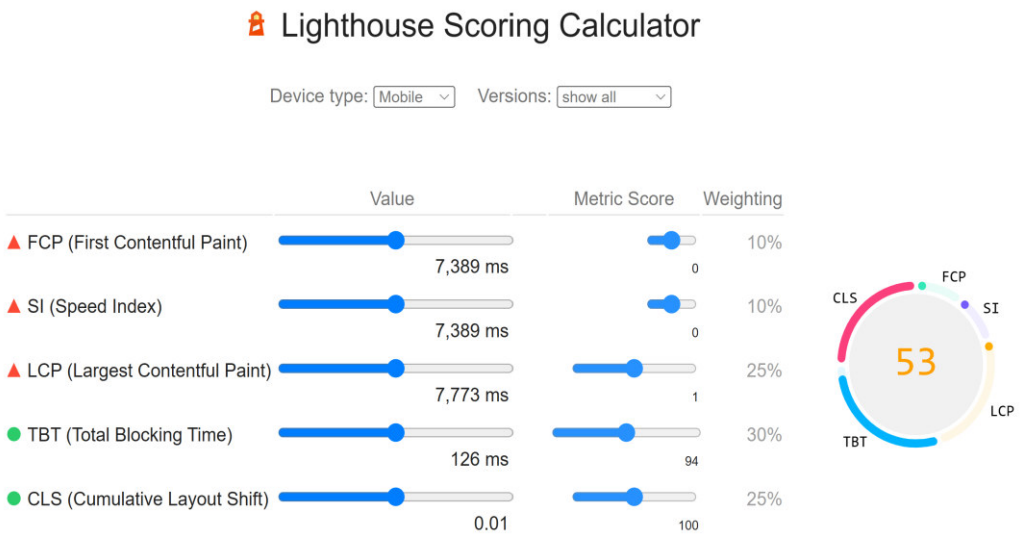


Figure 25. Lighthouse Score Calculator

Performance Improvements:

Text Compression: Compressing text resources (HTML, CSS, JavaScript) reduces file size and speeds up loading times, helping to improve both FCP and LCP. This involves using methods like GZIP compression to minimize data transfer.

Reducing Unused JavaScript: Unused JavaScript can slow down the rendering process, causing delays in loading. By eliminating or deferring JavaScript that isn't needed immediately, convOERter can reduce initial loading times and improve performance metrics.

Eliminating Render-Blocking Resources: Render-blocking resources, such as large CSS files, prevent the page from displaying content quickly.

Optimizing Image Formats: Using modern image formats like WebP instead of JPEG or PNG can reduce file sizes without compromising quality, leading to quicker image loading and better LCP scores.

5.3.1.2 Accessibility Metrics

The accessibility score of 100 (shown in Figure 26) indicates strong compliance with accessibility standards, which is important for convOERter's users. High accessibility means the application is usable by people with disabilities or special requirements. Key accessibility features include proper alt text for images, sufficient color contrast, and intuitive navigation.

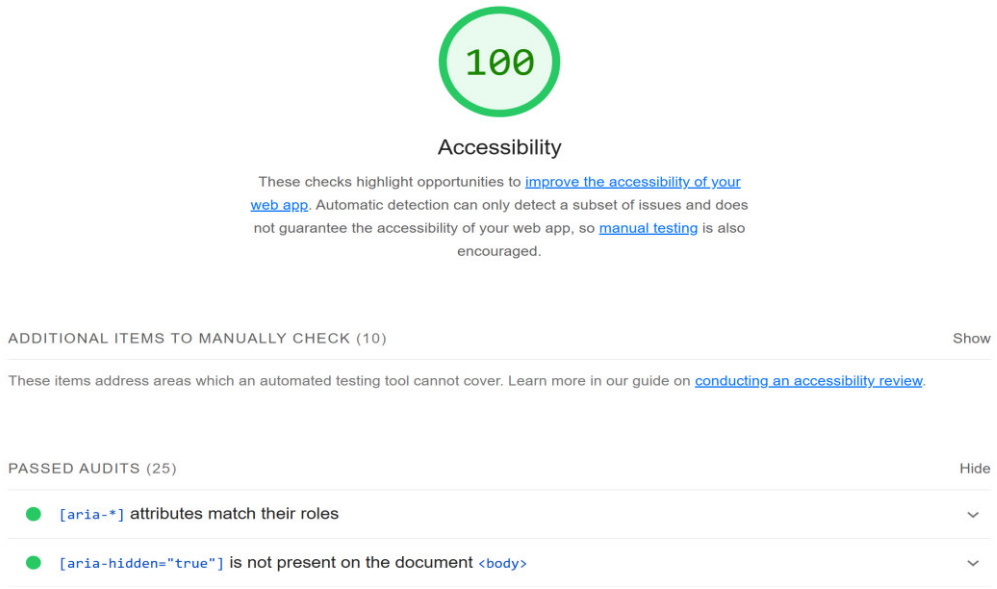


Figure 26. Lighthouse Accessibility Score

5.3.1.3 SEO and Best Practices

The web application scores well in best practices (96) and SEO (91) *(Barbar & Ismail, 2019)* shown in Figure 27 and Figure 28 respectively . This suggests that the tool follows security guidelines, modern coding practices, and SEO-friendly design, which contribute to a secure and user-friendly experience. Strong SEO performance ensures that the tool is discoverable by search engines, making it easier for educators to find and access the tool.

The detailed reports for SEO and Best Practices are available below:

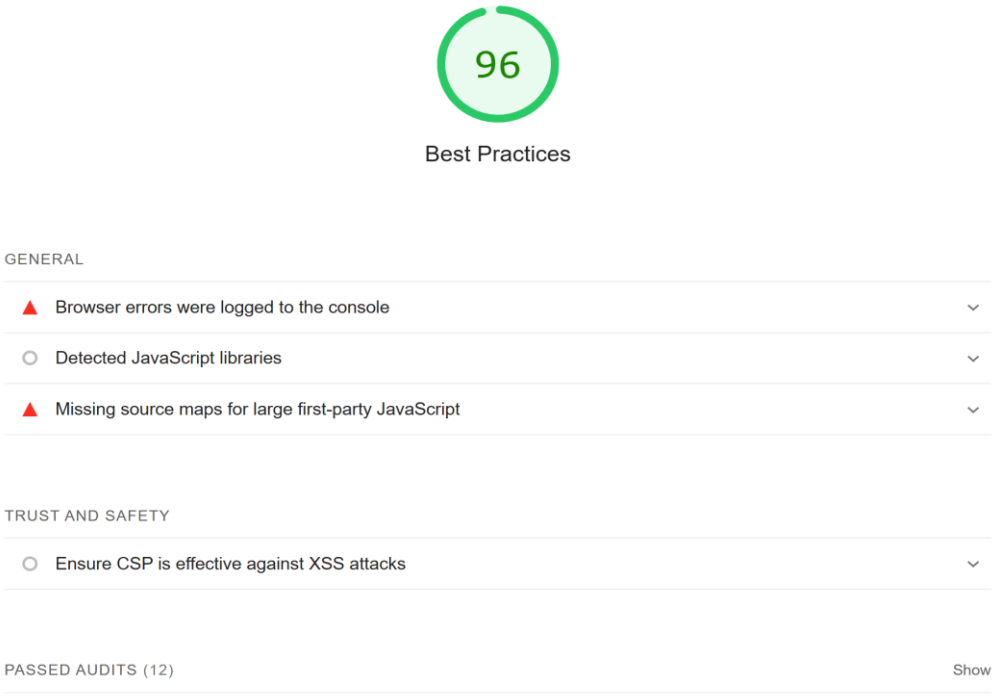


Figure 27. Best Practices Score

ConvOERter meets several best practices that enhance security, usability, and overall functionality:

Uses HTTPS: Ensures secure data transmission, protecting user information during interactions.

Avoids Deprecated APIs: Maintains compatibility and reliability by using up-to-date APIs.

Avoids Third-Party Cookies: Enhances privacy by not tracking users through external cookies.

Allows Pasting into Input Fields: Improves user convenience, making data entry easier and faster.

Avoids Geolocation and Notification Permissions on Load: Reduces user disruption by not requesting location or notification access immediately upon loading.

Displays Images with Correct Aspect Ratio: Maintains image quality and layout integrity by preventing distorted visuals.

Serves Images with Appropriate Resolution: Optimizes performance and display quality by using images sized appropriately for different devices.

These best practices contribute to a secure, user-friendly, and visually consistent experience in convOERter.

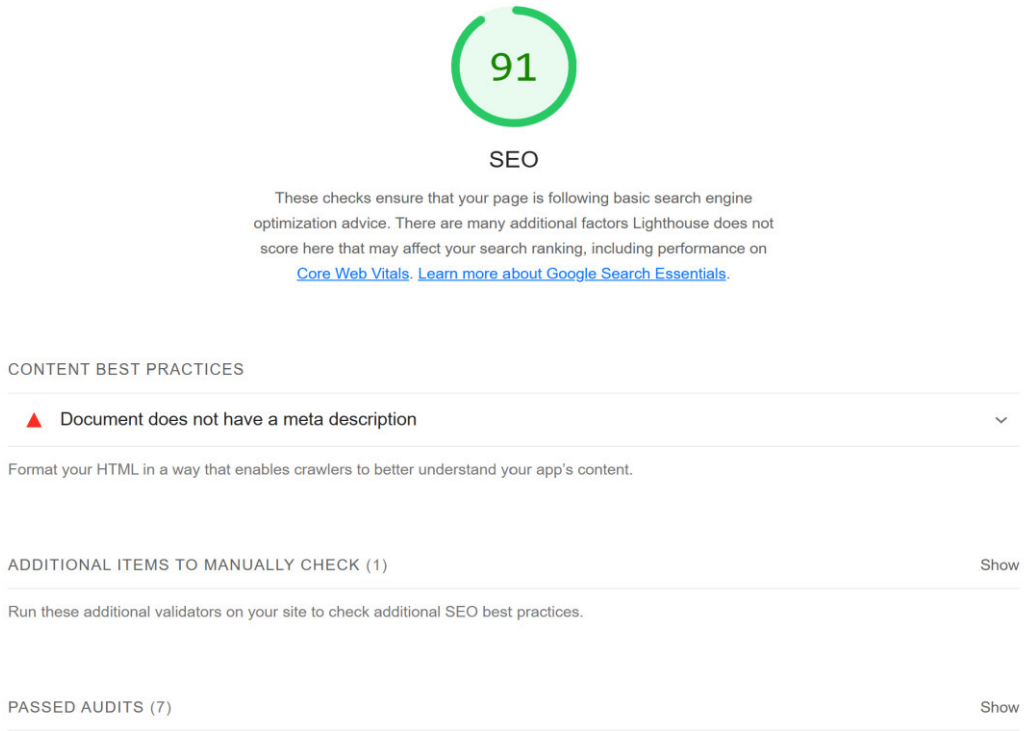


Figure 28. Best Practices Score

ConvOERter successfully meets key SEO audit criteria as shown by the score in Figure 28, ensuring optimal search engine visibility. The page is indexable, includes a <title> element, and returns a successful HTTP status code. Additionally, all links have descriptive text, images contain [alt] attributes, and the document uses valid hreflang, supporting accessibility and crawlability.

Further details can be in Lighthouse Test Report[24].

[24] Lighthouse Test Report. (n.d.). Available at: Google Drive folder (Accessed: 6 November 2024).

5.3.2 WebPageTest

For performance evaluation of the convOERter tool, *WebPageTest (Viscomi et al., 2015)*, an automated tool specifically designed to assess web application performance is used. WebPageTest provides detailed insights into load times, resource usage, and potential bottlenecks, allowing for a thorough test of how efficiently the tool operates. This testing approach highlights areas where optimization can enhance user experience by improving speed and reducing load delays.

The WebPageTest report (Figure 29) for convOERter mentions areas for improvement with respect to software quality:

Zero Render-Blocking CSS Files: The application avoids render-blocking CSS files, which helps the browser load and display content faster, improving the overall user experience.

Lazy Loading for Critical Images: The report generated by WebPageTest shows that convOERter uses lazy loading effectively, particularly for large images, ensuring they load only when necessary and reducing initial load time.

No Third-Party Fonts: By using fonts hosted within its own domain, the application avoids potential delays caused by loading external resources, enhancing performance reliability.

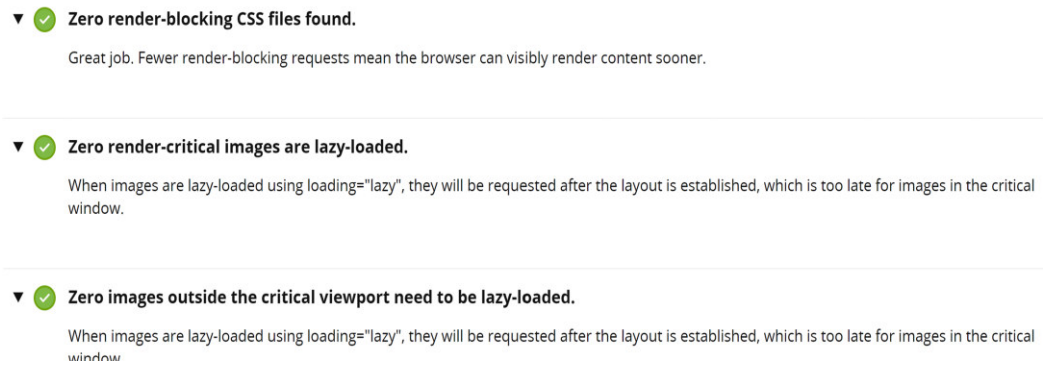


Figure 29. WebPageTest report

Issues detected by WebPageTest:

Font Loading Settings: Some fonts are loaded with settings that hide text while loading, which can lead to delays in displaying content. Adjusting font-display settings could provide a smoother experience.

Lack of CDN for Resource Files: Ten files are hosted without using a Content Delivery Network (CDN), which could increase latency for users further from the server location. Implementing a CDN would improve load times globally.

Document Object Model [25](DOM) Size Expansion: The report notes that the final HTML Document Object Model (DOM) size is slightly larger than the initial HTML size, indicating potential reliance on JavaScript to generate content, which could slow down performance.

This analysis highlights the strengths in convOERter's design while identifying areas like font loading, CDN usage, and DOM optimization for enhancement.

▼ **Several fonts are loaded with settings that hide text while they are loading.**

When fonts are loaded with default display settings, like font-display="block", browsers will hide text entirely for several seconds instead of showing text with a fallback font.

The details can be found in WebPageTest Report[26].

5.3.3 AXE

Another automated tool applied to perform accessibility testing is AXE, which is automated testing tool for identifying code accessibility issues. It works based on the rules defined in WCAG[27] (Web Content Accessibility Guidelines) but could not find any issues by crawling the web URL (Unified Resource Locator) to convOERter as shown in the Figure 30.

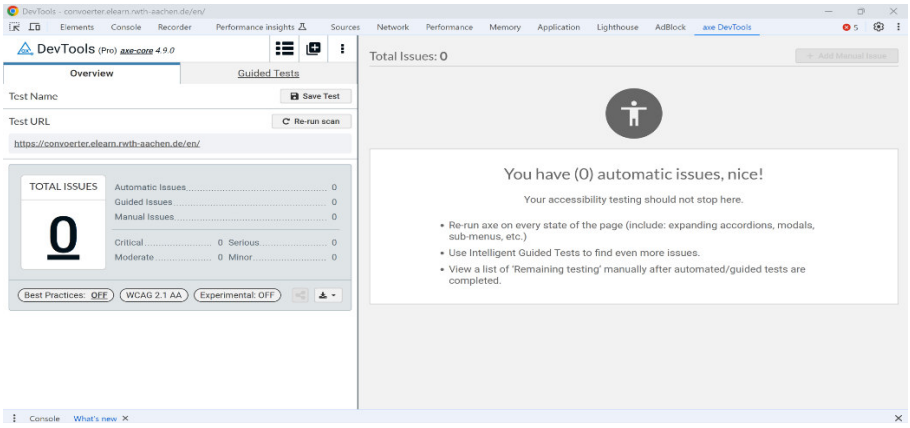


Figure 30. AXE Test report

[25] MDN Web Docs. (n.d.). Document Object Model (DOM). Available at: https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model (Accessed: 6 November 2024).
[26] WebPageTest Report. (n.d.). Available at: Google Drive file (Accessed: 6 November 2024).
[27] W3C. (n.d.). Web Content Accessibility Guidelines (WCAG). Available at: https://www.w3.org/WAI/standards-guidelines/wcag/ (Accessed: 6 November 2024).

5.4 Vulnerability Testing

For vulnerability testing, ZAP[28] (Zed Attack Proxy) automated testing is performed in accordance with OWASP guidelines. This automated testing aligns with the blackbox approach, in which the tester does not need to know the internal structure of the code. The ZAP Vulnerability Testing Report [29] is shown in Figure 31.

The ZAP vulnerability assessment for convOERter reveals multiple vulnerability risks categorized across various risk levels, from medium to informational. Addressing these vulnerabilities will significantly enhance the security of convOERter.

[28] OWASP. (n.d.). ZAP (Zed Attack Proxy). Available at: https://www.zaproxy.org/ (Accessed: 6 November 2024).

[29] ZAP Vulnerability Testing Report. (n.d.). Available at: Google Drive folder (Accessed: 6 November 2024).

Alert type	Risk	Count
Content Security Policy (CSP) Header Not Set	Medium	6 (60.0%)
Missing Anti-clickjacking Header	Medium	4 (40.0%)
Server Leaks Version Information via "Server" HTTP Response Header Field	Low	32 (320.0%)
Strict-Transport-Security Header Not Set	Low	31 (310.0%)
Timestamp Disclosure - Unix	Low	39 (390.0%)
X-Content-Type-Options Header Missing	Low	29 (290.0%)
Information Disclosure - Sensitive Information in URL	Informational	1 (10.0%)
Information Disclosure - Suspicious Comments	Informational	13 (130.0%)
Modern Web Application	Informational	4 (40.0%)
Re-examine Cache-control Directives	Informational	17 (170.0%)
Total		10

Figure 31. ZAP Test report

Medium-Level Risks

Content Security Policy [30] (CSP) Header Not Set

One of the primary medium-level risks identified is the absence of a Content Security Policy (CSP) header. CSP helps protect against a wide range of attacks, including Cross-Site Scripting (XSS) and data injection attacks, by restricting the resources that the browser is allowed to load.

[30] MDN Web Docs. (n.d.). Content Security Policy (CSP). Available at: https://developer.mozilla.org/en-US/docs/Web/HTTP/CSP (Accessed: 6 November 2024).

Anti-clickjacking Header Missing

Another medium-level risk involves the missing anti-clickjacking headers. Clickjacking attacks trick users into clicking on something different from what they perceive, leading them to mistakenly perform unwanted actions.

Low-Level Risks

Server Leaks Version Information via HTTP Headers

The report[32] indicates that the server leaks version information through the "Server" HTTP response header. Showing server information, such as version and software, can be useful for attackers, who may exploit known vulnerabilities associated with specific server versions.

X-Content-Type-Options Header Missing

The missing X-Content-Type-Options header is also highlighted as a low-risk issue. This header instructs browsers not to interpret files as a different MIME type, which can prevent MIME-type sniffing attacks.

Informational Risks

Information Disclosure - Sensitive Information in URLs

An informational risk was identified related to sensitive information disclosed in URL parameters.

Suspicious Comments in JavaScript Files

Suspicious comments within JavaScript files are flagged as another informational risk. These comments may contain debugging information, outdated code, or hints about the tool's inner workings.

Cache-Control Directives[31]

The report[32] suggests re-examining cache-control directives, as improper settings could lead to privacy and security concerns. Ensuring that sensitive

[31] MDN Web Docs. (n.d.). Cache-Control. Available at: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Cache-Control (Accessed: 6 November 2024).

information is not cached, or setting appropriate cache expiry directives, helps control what data is stored by the browser and limits exposure.

ZAP generates as detailed report which describes the risk level as well as confidence level associated with each risk detect as shown in the Figure 32 below.

		Confidence				
		User Confirmed	High	Medium	Low	Total
Risk	High	0 (0.0%)	0 (0.0%)	0 (0.0%)	0 (0.0%)	0 (0.0%)
	Medium	0 (0.0%)	1 (10.0%)	1 (10.0%)	0 (0.0%)	2 (20.0%)
	Low	0 (0.0%)	2 (20.0%)	1 (10.0%)	1 (10.0%)	4 (40.0%)
	Informational	0 (0.0%)	0 (0.0%)	2 (20.0%)	2 (20.0%)	4 (40.0%)
	Total	0 (0.0%)	3 (30.0%)	4 (40.0%)	3 (30.0%)	10 (100%)

Figure 32. ZAP Test report

The comprehensive details can be found in ZAP Vulnerability Testing Report[32].

[32] ZAP Vulnerability Testing Report. (n.d.). Available at: Google Drive folder (Accessed: 6 November 2024).

Chapter 6 Quality Model Evaluation

This chapter evaluates the quality assurance model developed for convOERter, focusing on user feedback, satisfaction metrics, and system performance. The evaluation provides insights into user experiences and highlights areas of improvement. The chapter outlines the study design, participant feedback collection process, and analysis of the results.

6.1 Study Design

The evaluation process for convOERter's quality assurance model involves structured feedback collection from participants using an extended feedback form. The form captures user satisfaction, usability, and functionality ratings ranging from *Strongly Disagree* to *Strongly Agree* (0 - 4) and last two questions has binary answers (Yes/No). Questions were derived from metrics developed using the Goal-Question-Metric (GQM) model, ensuring that feedback aligns with specific quality goals for the tool. This section details the participant selection, survey design, and feedback analysis methods.

6.1.1 Participants for Evaluation

According to Virzi, "80% of usability problems are detected with four or five subjects, as additional subjects are less likely to reveal new information" (Virzi, 1992). This suggests that a 4 - 5 sample size, is typically sufficient for uncovering significant usability issues in a system evaluation. A total of nine participants were contacted to participate in a structured feedback session. Out of them, six were agreed to participate in the feedback session for the usability of convOERter. The participants including educators and potential end-users were contacted via email with a brief introduction to convOERter and the purpose of the study. They were requested to prepare a sample academic file (in supported formats) containing images, as this would allow for a real-time interaction with the tool during the session. An online meeting was scheduled with each participant based on their availability.

6.1.2 Feedback Form Design

The feedback form was designed to gather quantitative data regarding the user experience. To assess usability and satisfaction, the form includes scaled questions based on a scale (ranging from 0 to 4 for "strongly disagree" to "strongly agree") and two binary questions. This extended feedback form aims to cover various aspects of the user experience, including image quality, interface satisfaction, and tool usability. The feedback form contains the following questions as shown in Figure 33, Figure 34 and Figure 35:

1. Time taken to convert an educational resource(seconds)
2. How do you rate the relevance of the alternative image provided by tool compared to original image?
3. How satisfied are you with the content of the alternative images provided by the tool?
4. How satisfied are you with the resolution of the alternative images provided by the tool?
5. How likely are you to recommend the tool to others?
6. This tool was helpful.
7. The user interface was good.
8. Would you use this tool?
9. Did you run into any major problems?

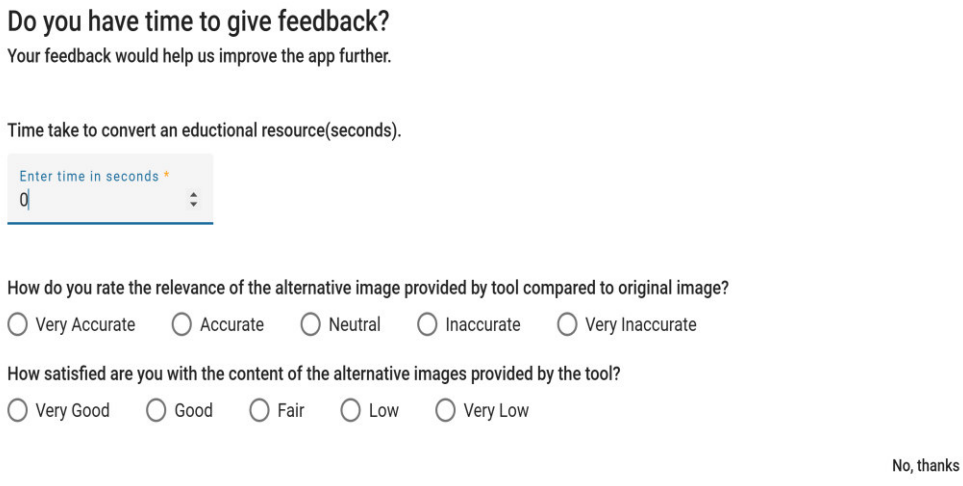


Figure 33. Feedback form convOERter

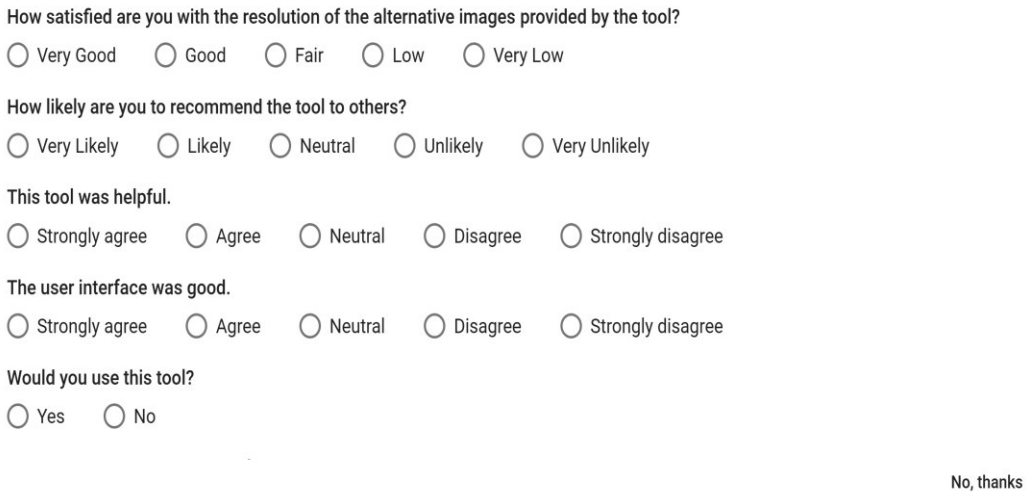


Figure 34. Feedback form convOERter

The feedback submission is voluntary for the user as there is an option of "No, thanks" also available to close without submitting feedback.

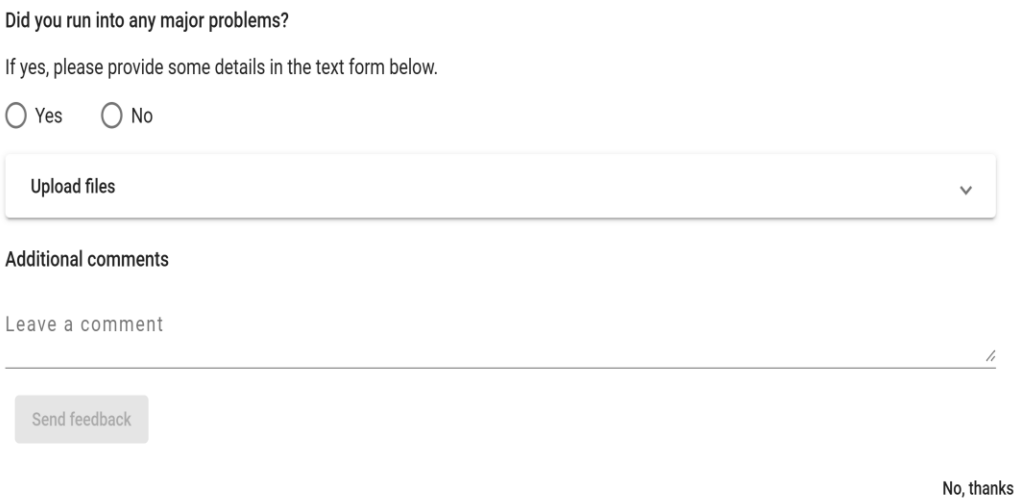


Figure 35. Feedback form convOERter

The figures from the feedback form show that it is not mandatory for the user to give the feedback but it will help the development team to make improvements on basis of collected feedback. Moreover, the user has given free text field if they want to add some other suggestions or problems faced. Furthermore, there

is a possibility to upload a file for which they face problem during conversion or add screenshot of the error faced during the conversion process.

The feedback form targeted particularly the following areas of potential improvements:

i. Satisfaction with alternative images provided by the tool.
ii. Rating of image resolution and relevance compared to the original.
iii. User interface and overall satisfaction.
iv. Likelihood of recommending the tool to others.
v. Functionality and accessibility of key tool features.

6.2 Feedback Session

Each participant attended a 30-minute Zoom session where they interacted with convOERter and completed the feedback form at the end of conversion process. At the beginning of the session, the participants were informed about functionality of the tool and instructed to perform a sample OER conversion as all of the participants were not frequent user of the tool. Then, the participant was given control access to perform the conversion process and the QA app was running in local environment using containerization. The extended feedback form was completed afterward, and participants were encouraged to discuss their experience regarding user interface, functionality, and performance.

During the session, participants tested convOERter's functionalities, particularly focusing on image replacement accuracy, ease of use, and overall effectiveness in converting educational resources. Each participant's responses and comments were documented for analysis.

6.3 Results

This section presents the analysis of the user feedback data, including quantitative ratings and insights.

6.3.1 Feedback Result Analysis

The responses from the feedback form depict different level of user satisfaction as shown in Figures 36, 37 and 38. The following metrics summarize the key findings:

Image Relevance and Quality: Participants rated the relevance and quality of alternative images provided by convOERter. Most users rated image relevance as "Agree" or "Strongly Agree," and resolution as "Good" or "Very Good." This suggests that convOERter's image replacement functionality aligns well with user expectations.

User Interface Satisfaction: The majority of participants rated the user interface as "Agree" or "Strongly Agree" regarding its usability and design. This positive feedback highlights that convOERter's interface is generally user-friendly and accessible, enhancing the user experience.

Tool Usability and Recommendation: Participants expressed a likelihood of recommending the tool, indicating positive user engagement. Ratings on the question of whether the tool was helpful showed high satisfaction, reflecting the tool's value in the OER conversion process.

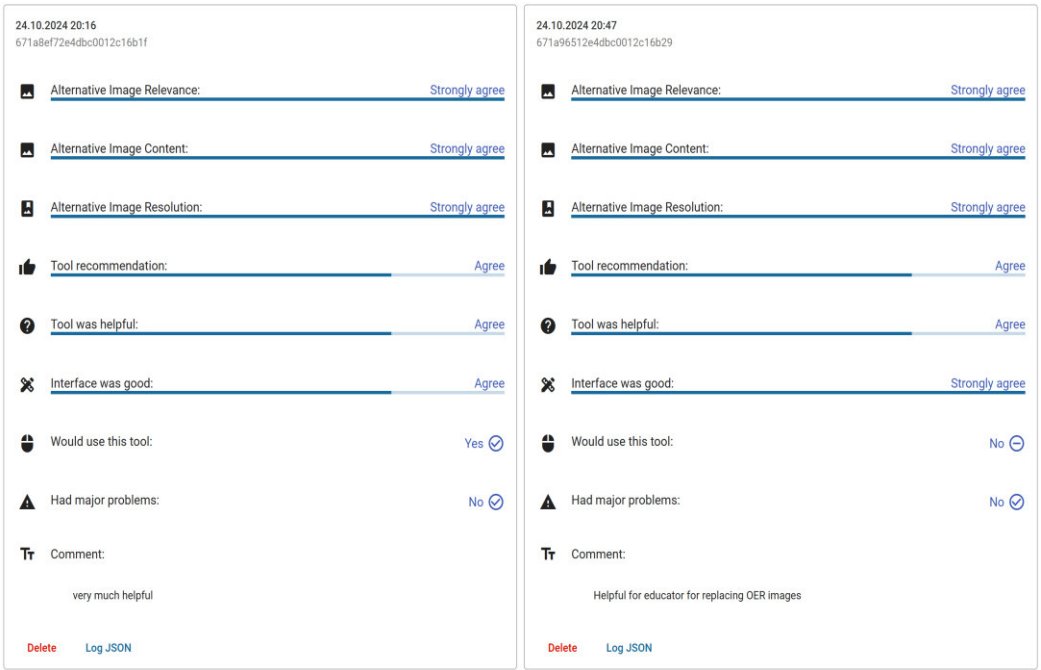


Figure 36. Feedback result (Participants 1 & 2)

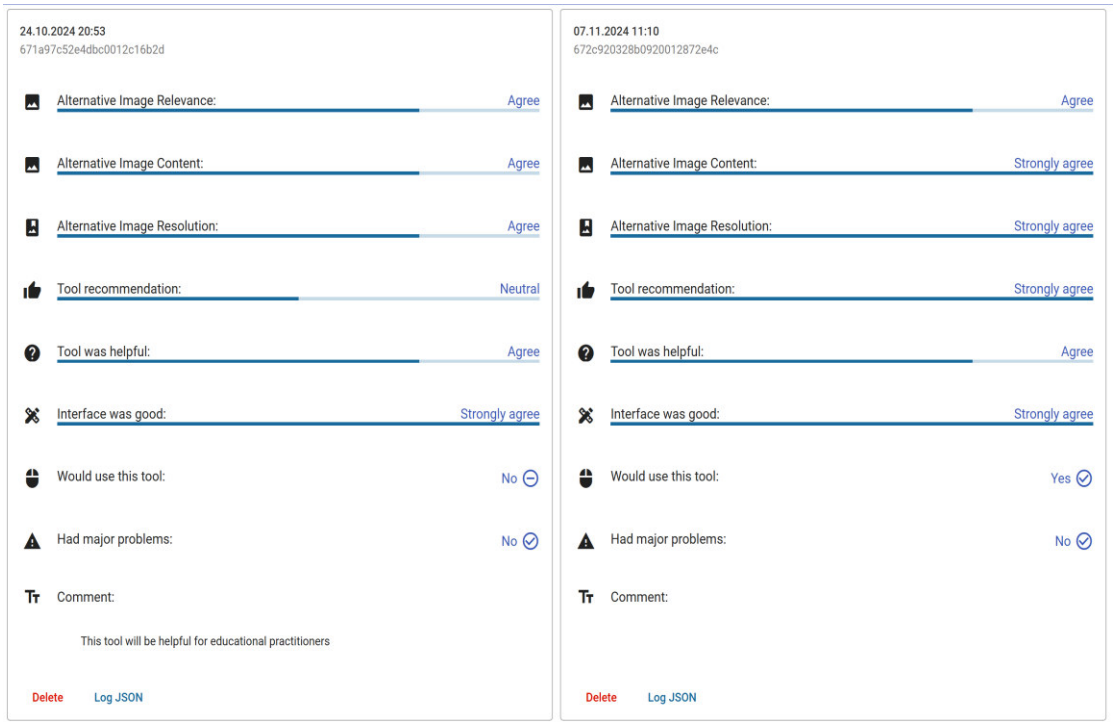


Figure 37. Feedback result (Participants 3 & 4)

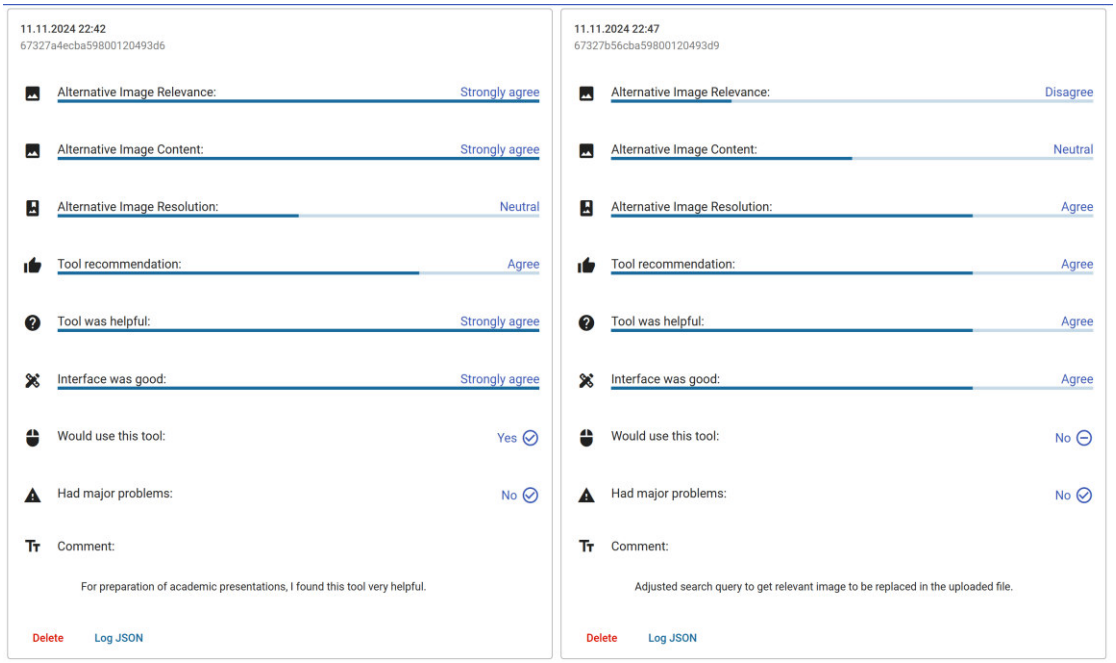


Figure 38. Feedback result (Participants. 4 & 5)

The participants for the feedback session are selected with mix of expertise, one participant was educator who frequently creates OER resources, two participants were from data science who worked on learning analytics and have basic knowledge of OER, one participant is frequent user of convOERter in regard of improvements and further development, one participant is from project management domain who usually creates resources for academic purpose but has no experience of using convOERter or any other OER tools. The last participant is from data analytics background who was interested in quality of software and presentation of data collected through feedback form.

6.3.2 Feedback Analysis Chart

To visualize feedback distribution, satisfaction levels for each question were represented in feedback analysis charts. The charts display percentages of responses across categories, ranging from "Strongly Disagree" to "Strongly Agree," as well as "Very Likely" to "Very Unlikely" for recommendation questions. This quantitative breakdown provides clear insights into user preferences and areas improvement illustrated in Figure 39.

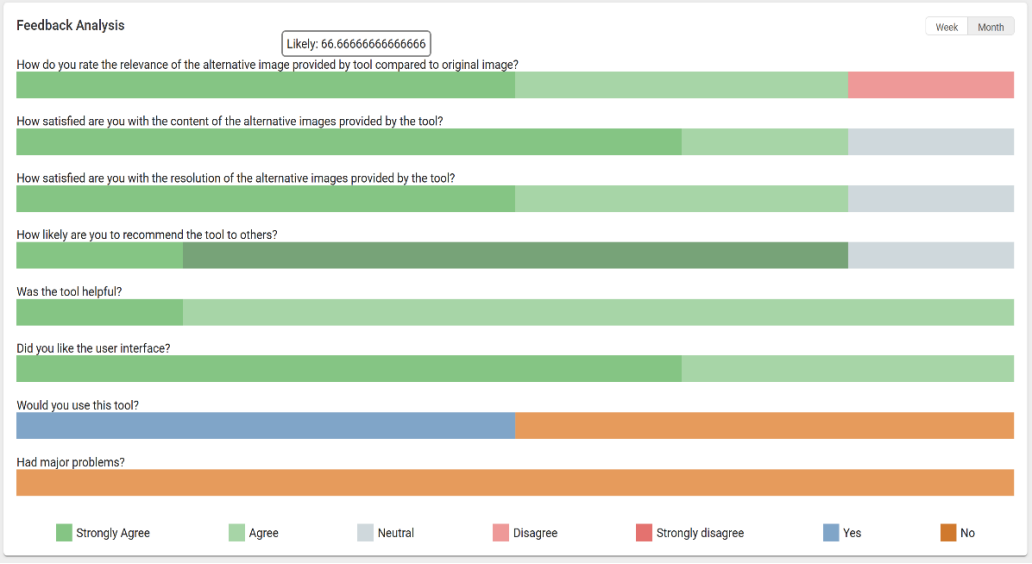


Figure 39. Feedback analysis chart

Each bar in the chart represents the distribution of user responses, with ratings like "Strongly Agree," "Agree," "Neutral," indicated by color, as shown in the legend at the bottom. By hovering over each section, the percentage of users

selecting each rating can be viewed, offering detailed insights into user sentiments across different aspects of the tool.

Feedback analysis is developed in admin panel as shown by code snippets in Figure 40:

```
const wasHelpfulPercentages = wasHelpfulSubgroups.map((subgroup, index) => {
  const count = wasHelpfulCounts[index + 1];
  return (count / totalFeedbacks) * 100;
});      You, 2 weeks ago • feedback and analysis changes
```

```
const wasHelpfulResponse = [{
  label: "wasHelpful",
  [wasHelpfulSubgroups[0]]: wasHelpfulPercentages[0],
  [wasHelpfulSubgroups[1]]: wasHelpfulPercentages[1],
  [wasHelpfulSubgroups[2]]: wasHelpfulPercentages[2],
  [wasHelpfulSubgroups[3]]: wasHelpfulPercentages[3],
  [wasHelpfulSubgroups[4]]: wasHelpfulPercentages[4],
}];
```

```
  D3Wrapper.setupHorizontalStackedBarChart("#wasHelpful-freq-chart-svg",
      wasHelpfulSubgroups, wasHelpfulResponse);       You, 3 weeks ago • U
```

Figure 40. Feedback analysis code snippet

6.4 Key Findings

6.4.1 User Satisfaction Insights

Participants generally found the tool interactive, with minor usability issues reported. The GQM-based metrics effectively gathered insights on user satisfaction, helping to assess whether the tool meets the needs of educators.

6.4.2 Improvement Areas

Some participants indicated potential areas for improvement. These included:

Image Quality and Selection: A few participants found that images suggested by the tool are not relevant due to which they need to manually change the search query to find new images.

Interface Clarity: Minor improvements in guidance of license information using tooltips and process of manual uploading of image will make the tool more useful.

6.5 Summary of Evaluation

The evaluation of convOERter's quality assurance model reflects positive user feedback on functionality, usability, and image quality. The use of GQM metrics helped in quantitative assessment, revealing that most users are satisfied with the tool's performance and willing to recommend it. The insights gained from participant feedback will be used for future updates of convOERter, further refining its quality and usability for educators in the OER domain.

Chapter 7 Conclusion and Future Work

7.1 Conclusion

Open Educational Resources (OER) are important for modern education, providing open access to educational materials. As, convOERter supports this by enabling educators to convert existing resources into OER-compliant formats by replacing non-OER images with OER images. This thesis developed a quality assurance model for convOERter, focusing on usability, performance, and security.

Black-box testing techniques like boundary value analysis and use-case testing etc were used to assess the tool's response to user inputs without internal code knowledge. Automated accessibility and performance tools such as Lighthouse analyzed convOERter's accessibility, while WebPageTest identified areas for performance improvement. GQM model helped in defining measurable metrics for user satisfaction, functionality. Vulnerability testing with ZAP detected security issues, like missing headers, that could expose the tool to security threats and suggested preventive measures. The evaluation showed user satisfaction with convOERter's suggested images and interface usability.

The study was limited by a small number of participants, suggesting a need for expanded testing. Security updates and continuous performance improvements are recommended to maintain a high-quality of user experience.

7.2 Future Work

Based on the insights gained, several recommendations for future work have been identified:

Enhanced Testing Techniques: Introducing grey-box and white-box testing could improve code-level analysis. As grey-box testing is a hybrid approach in which the tester has partial knowledge of the internal structure of the code but tests it from external perspective. It will help testers to understand how convOERter works internally (i.e, image processing algorithms and user interface workflows) without having complete access to the source code. This will enable tester to design the test cases to explore the edge cases which were not covered in blackbox testing.

For instance, users upload academic files with images to convOERter that needs to be replaced with OER-images. Greybox testing could help to identify the issues caused by large image sizes, network latency and unusual file formats.

The tester should know how the system will respond if the image will not be found in the integrated open source image platforms. This will help to ensure that the system will respond accordingly without giving error or crashing. It can be achieved by API level testing using POSTMAN[33] etc.

Integration Testing and API Evaluation:

The functionalities like image replacement are dependent on external services like Image Search API in convOERter. Greybox testing allows us to assess how well these external services interact with core functionality of the tool. A tester can examine how the API calls for image replacement are handled which helps to examine data flow from external services and internal mechanisms (such as image processing or storage). By performing integration testing, a tester can uncover the problems of timeouts, incorrect responses and data mappings.

Improve Image Search: Integration of new image search API (e.g. Google vision, Bing Visual Search) will enhance user satisfaction related to image relevance and quality. Future improvements could involve expanding the database of OER-compliant images and implementing adaptive learning algorithms to refine search results dynamically based on user behavior, creating a more personalized experience.

Security Measures: Implementing regular audits could enhance security. Cross-Site Scripting (XSS) and security flaws in image upload to the server needs to be checked properly so that attacker cannot breach the security of the system. Incorporating automated scanning tools such as ZAP, which will run before each production build and setting a threshold for the production build to pass will improve the overall security of the tool.

Feedback System: An interactive feedback mechanism could provide more actionable insights for developers. Integrating feedback form with sentiment analysis system could help developers to prioritize the issues and features. An

[33] Postman. (n.d.). Postman. Available at: https://www.postman.com/ (Accessed: 6 November 2024).

interactive feedback method i.e, in-app pop-ups will help to gather real time responses.

As software quality assurance is continuous improvement process so it always require enhancements and stay updated with the latest security threats. It is involved at every stage of the software like planning, development and deployment. By taking into consideration, the quality standards set by the World Wide Web Consortium[34] (W3C), we can ensure that the web application ranks well over the web.

[34] W3C. (n.d.). World Wide Web Consortium (W3C). Available at: https://www.w3.org/ (Accessed: 6 November 2024).

Bibliography

2010_Hodgkinson-Williams_Benefits-Challenges-of-OER-for-HE-Institutions.pdf. (n.d.).

About CC Licenses. (n.d.). Creative Commons. Retrieved November 3, 2024, from https://creativecommons.org/share-your-work/cclicenses/

Barbar, A., & Ismail, A. (2019). Search Engine Optimization (SEO) for Websites. *Proceedings of the 2019 5th International Conference on Computer and Technology Applications*, 51–55. https://doi.org/10.1145/3323933.3324072

Bates, A. W. (Tony), & Bates, A. W. (2015). *10.2 Open educational resources (OER)*. https://opentextbc.ca/teachinginadigitalage/chapter/oer/

Caldiera, G., Rombach, H. D., & Basili, V. R. (n.d.). The goal question metric approach. In *Encyclopedia of software engineering* (pp. 528–532).

D. Rajya Lakshmi, & S. Suguna Mallika. (2017). A Review on Web Application Testing and its Current Research Directions. *International Journal of Electrical and Computer Engineering (IJECE)*. https://doi.org/10.11591/ijece.v7i4.pp2132-2141

Di Lucca, G. A., & Fasolino, A. R. (2006). Web Application Testing. In E. Mendes & N. Mosley (Eds.), *Web Engineering* (pp. 219–260). Springer Berlin Heidelberg. https://doi.org/10.1007/3-540-28218-1_7

Difference between Black Box and White and Grey Box Testing. (2021, October 15). GeeksforGeeks. https://www.geeksforgeeks.org/difference-between-black-box-vs-white-vs-grey-box-testing/

Dobslaw, F., de Oliveira Neto, F., & Feldt, R. (2020). *Boundary Value Exploration for Software Analysis*. https://doi.org/10.48550/arXiv.2001.06652

Dobslaw, F., Feldt, R., & Gomes De Oliveira Neto, F. (2023). Automated black-box boundary value detection. *PeerJ Computer Science*, *9*, e1625. https://doi.org/10.7717/peerj-cs.1625

Duggal, G., & Suri, B. (2008). *Understanding regression testing techniques. 1*, 8.

Education | German Commission for UNESCO. (n.d.). Retrieved November 3, 2024, from https://www.unesco.de/en/education/open-educational-resources

Ehmer Khan, Mohd. (2011). Different Approaches To Black box Testing Technique For Finding Errors. *International Journal of Software Engineering & Applications*, *2*(4), 31–40. https://doi.org/10.5121/ijsea.2011.2404

El-Rayyes, E. K., & Abu-Zaid, I. M. (2012). New Model to Achieve Software Quality Assurance (SQA) in Web Application. *International Journal of Science and Technology*, *2(7)*.

Goal Question Metric Approach in Software Quality. (2020, August 14). GeeksforGeeks. https://www.geeksforgeeks.org/goal-question-metric-approach-in-software-quality/

Hussain, T., & Singh, S. (2015). A comparative study of software testing techniques viz. White box testing black box testing and grey box testing. *International Journal of Allied Practice, Research and Review*.

Hylen, J. (2006). Open educational resources: Opportunities and challenges. *Proceedings of Open Education*, 49–63.

Irawan, Y., Muzid, S., Susanti, N., & Setiawan, R. (2018). System Testing using Black Box Testing Equivalence Partitioning (Case Study at Garbage Bank Management Information System on Karya Sentosa). *Proceedings of the The 1st International Conference on Computer Science and Engineering Technology Universitas Muria Kudus*. The 1st International Conference on Computer Science and Engineering Technology Universitas Muria Kudus, Kudus, Indonesia. https://doi.org/10.4108/eai.24-10-2018.2280526

Jain, S. (2024). *Software Testing Techniques: Explained with Examples*. BrowserStack. https://browserstack.wpengine.com/guide/software-testing-techniques/

Jason Bau, Elie Bursztein, Divij Gupta, & John C. Mitchell. (2009). Automated Black-Box Web Application Vulnerability Testing. *31st IEEE Symposium on Security and Privacy, S&P 2010, 16-19 May 2010, Berleley/Oakland, California, USA*. https://doi.org/10.1109/SP.2010.27

Kasowaki, L., & Eymen, M. (2023). Quality Assurance Metrics and Measurement in Software Engineering. *EasyChair Preprint 11359*.

Khalid, M. W., Ali, L., & Schroeder, U. (2024). Developing a Quality Assurance Model for OER Conversion Tool (CONVOERTER). *21st International Conference on Cognition and Exploratory Learning in the Digital Age (CELDA 2024)*, 409.

Khan, Mohd. E., & Khan, F. (2012). A Comparative Study of White Box, Black Box and Grey Box Testing Techniques. *(IJACSA) International Journal of Advanced Computer Science and Applications*, *3*.

Krupalija, E., Cogo, E., Becirovic, S., Prazina, I., Hodzic, K., & Besic, I. (2022). Forward-Propagation Approach for Generating Feasible and Minimum Test Case Suites From Cause-Effect Graph Specifications. *IEEE Access*, *10*, 124545–124562. https://doi.org/10.1109/ACCESS.2022.3225097

Kuhn, D. R., & Bryce, R. (2015). Combinatorial testing: Theory and practice. *Advances in Computers*, *99*, 1–66.

L. T. Giang, D. Kang, & D. -H. Bae. (n.d.). Software Fault Prediction Models for Web Applications. *2010 IEEE 34th Annual Computer Software and Applications Conference Workshops, Seoul, Korea (South)*. https://doi.org/10.1109/COMPSACW.2010.19

Li, H., & Meissner, J. (2009). *Improving quality in business process outsourcing through technology (No. MRG/0009)*. *MRG/0009*. www.meiss.com/download/BPO-Li-Meissner.pdf

Ming-Chang Lee. (2014). Software Quality Factors and Software Quality Metrics to Enhance Software Quality Assurance. *British Journal of Applied Science & Technology*, *4*. https://doi.org/10.9734/BJAST/2014/10548

Nidhra, S., & Dondeti, J. (2012). Black box and white box testing techniques-a literature review. *International Journal of Embedded Systems and Applications (IJESA)*, *2(2)*, 29–50.

OECD. (2007). *Giving Knowledge for Free: The Emergence of Open Educational Resources*. Organisation for Economic Co-operation and Development. https://www.oecd-ilibrary.org/education/giving-knowledge-for-free_9789264032125-en

OER Benefits. (n.d.). Retrieved November 3, 2024, from https://www.lsu.edu/hss/wllc/oer/oerbenefits.php

OER principles. (n.d.). Retrieved November 3, 2024, from https://www.lsu.edu/hss/wllc/oer/oerprinciples.php

Open Education. (n.d.). *Hewlett Foundation*. Retrieved November 3, 2024, from https://hewlett.org/strategy/open-education/

Open Educational Resources reviewing initiatives and issues.pdf. (n.d.).

PDF. (n.d.).

Qazi, A. M., Rauf, A., & Minhas, N. M. (2016). A Systematic Review of Use Cases based Software Testing Techniques. *International Journal of Software Engineering and Its Applications*, *10*(11), 337–360. https://doi.org/10.14257/ijseia.2016.10.11.28

Shetty, D. R. (2022). Design and implementation of an evaluation system for OER conversion tool (convOERter). *Masterarbeit*, *RWTH Aachen University*, pages 1 Online-Ressource : Illustrationen, Diagramme. https://doi.org/10.18154/RWTH-2023-00418

Software Testing—Use Case Testing. (2019, May 8). GeeksforGeeks. https://www.geeksforgeeks.org/software-testing-use-case-testing/

State Transition Testing Techniques in Software Testing. (n.d.). Retrieved November 10, 2024, from https://testsigma.com/blog/state-transition-testing/

Sunghun Kim. (2012). Defect prediction. *Proceedings of the 8th International Conference on Predictive Models in Software Engineering (PROMISE '12). Association for Computing Machinery, New York, NY, USA,*. https://doi.org/10.1145/2365324.2365325

Suryaningrat, A., & Henderi. (2019). Software Testing as Quality Assurance on Web Application. *Advances in Social Science, Education and Humanities Research, Volume 410*, 199–203.

Virzi, R. A. (1992). Refining the Test Phase of Usability Evaluation: How Many Subjects Is Enough? *Human Factors*. https://doi.org/10.1177/001872089203400407

Viscomi, R., Davies, A., & Duran, M. (2015). *Using WebPageTest: Web performance testing for novices and power users.* O'Reilly Media, Inc.

What an OER is not. (n.d.). Retrieved November 12, 2024, from https://www.lsu.edu/hss/wllc/oer/oerwhatisnot.php

White box Testing—Software Engineering. (2018, July 12). GeeksforGeeks. https://www.geeksforgeeks.org/software-engineering-white-box-testing/

Wickramasinghe, S. (2023). *Combinatorial Testing | What it is, How to Perform & Tools*. Testsigma. https://testsigma.com/blog/combinatorial-testing/

Yoon, I.-C., Sussman, A., Memon, A., & Porter, A. (2008). Effective and scalable software compatibility testing. *Proceedings of the 2008 International Symposium on Software Testing and Analysis*, 63–74. https://doi.org/10.1145/1390630.1390640

Eidesstattliche Versicherung

Khalid, Muhammad Waseem — 403545

Name, Vorname — Matrikelnummer (freiwillige Angabe)

Ich versichere hiermit an Eides Statt, dass ich die vorliegende Arbeit/Bachelorarbeit/ Masterarbeit* mit dem Titel

Entwicklung eines Qualitätssicherungsmodells für das OER Conversion Tool (convOERter

selbstständig und ohne unzulässige fremde Hilfe erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt. Für den Fall, dass die Arbeit zusätzlich auf einem Datenträger eingereicht wird, erkläre ich, dass die schriftliche und die elektronische Form vollständig übereinstimmen. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Aachen, 12.11.2024 — _____

Ort, Datum — Unterschrift

*Nichtzutreffendes bitte streichen

Belehrung:

§ 156 StGB: Falsche Versicherung an Eides Statt

Wer vor einer zur Abnahme einer Versicherung an Eides Statt zuständigen Behörde eine solche Versicherung falsch abgibt oder unter Berufung auf eine solche Versicherung falsch aussagt, wird mit Freiheitsstrafe bis zu drei Jahren oder mit Geldstrafe bestraft.

§ 161 StGB: Fahrlässiger Falscheid; fahrlässige falsche Versicherung an Eides Statt

(1) Wenn eine der in den §§ 154 bis 156 bezeichneten Handlungen aus Fahrlässigkeit begangen worden ist, so tritt Freiheitsstrafe bis zu einem Jahr oder Geldstrafe ein.

(2) Straflosigkeit tritt ein, wenn der Täter die falsche Angabe rechtzeitig berichtigt. Die Vorschriften des § 158 Abs. 2 und 3 gelten entsprechend.

Die vorstehende Belehrung habe ich zur Kenntnis genommen:

Aachen, 12.11.2024 — _____

Ort, Datum — Unterschrift